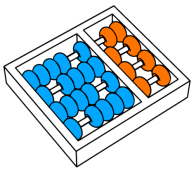
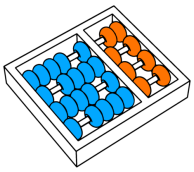


Pedro Cipriano Feijão

"On genome rearrangement models"

"*Sobre modelos de rearranjo de genomas*"

CAMPINAS
2012



University of Campinas
Institute of Computing

Universidade Estadual de Campinas
Instituto de Computação

Pedro Cipriano Feijão

"On genome rearrangement models"

Supervisor:
Orientador(a): **João Meidanis**

"Sobre modelos de rearranjo de genomas"

PhD Thesis presented to the Post Graduate Program of the Institute of Computing of the University of Campinas to obtain a PhD degree in Computer Science.

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Doutor em Ciência da Computação.

This volume corresponds to the final version of the Thesis defended by Pedro Cipriano Feijão, under the supervision of João Meidanis.

Este exemplar corresponde à versão final da Tese defendida por Pedro Cipriano Feijão, sob orientação de João Meidanis.

Supervisor's signature / *Assinatura do Orientador(a)*

CAMPINAS
2012

FICHA CATALOGRÁFICA ELABORADA POR
MARIA FABIANA BEZERRA MULLER - CRB8/6162
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA - UNICAMP

Feijão, Pedro Cipriano, 1975-
F324s Sobre modelos de rearranjo de genomas / Pedro Cipriano Feijão. – Campinas, SP : [s.n.], 2012.

Orientador: João Meidanis.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Bioinformática. 2. Biologia computacional. 3. Genomas. 4. Evolução molecular - Simulação por computador. 5. Grupos de permutação. I. Meidanis, João,1960-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: On genome rearrangement models
Palavras-chave em inglês:
Bioinformatics
Computational biology
Genomes
Molecular evolution - Computer simulation
Permutation groups
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
João Meidanis [Orientador]
David Sankoff
Celina Miraglia Herrera de Figueiredo
Zanoni Dias
João Frederico da Costa Azevedo Meyer
Data de defesa: 16-10-2012
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 16 de outubro de 2012, pela Banca examinadora composta pelos Professores Doutores:

Profª. Drª. Celina Miraglia Herrera de Figueiredo
COPPE / UFRJ

Prof. Dr. David Sankoff
University of Ottawa

Prof. Dr. João Frederico da Costa Azevedo Meyer
IMECC - UNICAMP

Prof. Dr. Zanoni Dias
IC / UNICAMP

Prof. Dr. João Meidanis
IC - UNICAMP

Institute of Computing /*Instituto de Computação*
University of Campinas /*Universidade Estadual de Campinas*

On genome rearrangement models

Pedro Cipriano Feijão[1]

October 16, 2012

Examiner Board/*Banca Examinadora*:

- João Meidanis (Orientador)
- David Sankoff
  Department of Mathematics and Statistics – University of Ottawa
- Celina Miraglia Herrera de Figueiredo
  PESC – COPPE – Universidade Federal do Rio de Janeiro
- Zanoni Dias
  IC – Universidade Estadual de Campinas
- João Frederico da Costa Azevedo Meyer
  IMECC – Universidade Estadual de Campinas

[1]Suporte financeiro de: Bolsa CAPES 06/2007–07/2009 e Bolsa do CNPq (processo 142871/2009-5) 08/2009–01/2010

Abstract

Genome rearrangements are events where large blocks of DNA exchange places during evolution. With the growing availability of whole genome data, the analysis of these events can be a very important and promising tool for understanding evolutionary genomics.

Several mathematical models of genome rearrangement have been proposed in the last 20 years. In this thesis, we propose two new rearrangement models. The first was introduced as an alternative definition of the breakpoint distance. The breakpoint distance is one of the most straightforward genome comparison measures, but when it comes to defining it precisely for multichromosomal genomes, there is more than one way to go about it. Pevzner and Tesler gave a definition in a 2003 paper, and Tannier et al. defined it differently in 2008. In this thesis we provide yet another alternative, calling it single-cut-or-join (SCJ). We show that several genome rearrangement problems, such as genome median, genome halving and small parsimony, become easy for SCJ, and provide polynomial time algorithms for them.

The second model we introduce is the Adjacency Algebraic Theory, an extension of the Algebraic Formalism proposed by Meidanis and Dias that allows the modeling of linear chromosomes, the main limitation of the original formalism, which could deal with circular chromosomes only. We believe that the algebraic formalism is an interesting alternative for solving rearrangement problems, with a different perspective that could complement the more commonly used combinatorial graph-theoretic approach. We present polynomial time algorithms to compute the algebraic distance and find rearrangement scenarios between two genomes. We show how to compute the rearrangement distance from the adjacency graph, for an easier comparison with other rearrangement distances. Finally, we show how all classic rearrangement operations can be modeled using the algebraic theory.

Resumo

Rearranjo de genomas é o nome dado a eventos onde grandes blocos de DNA trocam de posição durante o processo evolutivo. Com a crescente disponibilidade de sequências completas de DNA, a análise desse tipo de eventos pode ser uma importante ferramenta para o entendimento da genômica evolutiva.

Vários modelos matemáticos de rearranjo de genomas foram propostos ao longo dos últimos vinte anos. Nesta tese, desenvolvemos dois novos modelos. O primeiro foi proposto como uma definição alternativa ao conceito de distância de *breakpoint*. Essa distância é uma das mais simples medidas de rearranjo, mas ainda não há um consenso quanto à sua definição para o caso de genomas multi-cromossomais. Pevzner e Tesler deram uma definição em 2003 e Tannier et al. a definiram de forma diferente em 2008. Nesta tese, nós desenvolvemos uma outra alternativa, chamada de *single-cut-or-join* (SCJ). Nós mostramos que, no modelo SCJ, além da distância, vários problemas clássicos de rearranjo, como a mediana de rearranjo, *genome halving* e pequena parcimônia são fáceis, e apresentamos algoritmos polinomiais para eles.

O segundo modelo que apresentamos é o *formalismo algébrico por adjacências*, uma extensão do formalismo algébrico proposto por Meidanis e Dias, que permite a modelagem de cromossomos lineares. Esta era a principal limitação do formalismo original, que só tratava de cromossomos circulares. Apresentamos algoritmos polinomiais para o cálculo da distância algébrica e também para encontrar cenários de rearranjo entre dois genomas. Também mostramos como calcular a distância algébrica através do grafo de adjacências, para facilitar a comparação com outras distâncias de rearranjo. Por fim, mostramos como modelar todas as operações clássicas de rearranjo de genomas utilizando o formalismo algébrico.

Dedico este trabalho aos meus pais, Moisés e Mizé, à minha esposa Giovana e ao meu filho Nuno.

Agradecimentos

É inegável que um trabalho desta magnitude, apesar de ser assinado por um autor único, depende de um grande número de pessoas para ser concretizado. Devo a todas as pessoas que participaram neste processo, direta ou indiretamente, os meus mais sinceros agradecimentos, ainda que não seja possível citar todos aqui.

Gostaria de agradecer em primeiro lugar aos meus pais, que sempre me apoiaram muito e torcem muito por mim. Desde que vim para Campinas, em 1993, nós não passamos mais tanto tempo juntos, ficando às vezes mais de 6 meses ou 1 ano sem nos encontrar. Ainda assim, sei que eles estão sempre comigo, mais ainda depois que me tornei pai. Vovó Mizé e vovô Moisesito, muito obrigado.

À minha linda esposa Giovana, por tudo que passamos juntos nos últimos 15 anos, e em especial o nosso filho Nuno. Sem eles, tenho certeza que nunca chegaria onde cheguei. Sei que tenho muita sorte de tê-la encontrado, não consigo imaginar mais ninguém que consiga me aguentar por todo esse tempo e ainda ser feliz no processo. Gatinha, te amo.

Agradeço também ao meu orientador e amigo, João Meidanis, pela confiança e total liberdade que ele sempre me deu. Aprendi muito em todas as excelentes conversas e discussões, e com o seu senso de humor peculiar.

Gostaria também de agradecer ao professor Cid de Souza, que além de ser um excelente professor, me ajudou no momento da minha inscrição no doutorado do IC. Sem a sua ajuda eu não teria sido aceito no programa de pós-graduação, e por isso sou muito grato.

Agradeço também a todos os funcionários do Instituto de Computação, por sempre estarem à disposição pra ajudar em tudo, documentos e afins, mesmo os alunos relapsos e que deixam tudo pra última hora.

Por fim, agradeço à CAPES, CNPq e FAEPEX pela ajuda financeira durante o doutorado, na forma de bolsas de estudo e auxílio a viagem.

Contents

Chapter 1

Introduction

In the last decades, large-scale genome mapping and sequencing allowed a finer understanding of the evolutionary processes occurring at the molecular level. Besides point mutations, movements of larger genome blocks are key contributors to changing the genetic footprint of living organisms.

With literally thousands of genomes now available at the sequence level, whole genome comparisons rises in importance. Recent years have seen an increasing interest in developing methods and algorithms to perform such comparisons, focusing on the larger DNA genome movements, known as *genome rearrangements*.

The history of genome rearrangements in Brazil can be traced back to 1996, when there was a DIMACS Challenge event involving both fragment assembly and genome rearrangement problems. Professor João Meidanis, my PhD advisor, was there because of his work in fragment assembly, but he soon got interested in rearrangements. At that time, Maria Emília Walter was his PhD student at the University of Campinas, and Zanoni Dias was also part of this initial group that started the study of rearrangement problems in Brazil. From this period are the first conference articles published by the group in the subject, a notable example being the seminal paper introducing the *Algebraic Formalism*, that was later studied by several of Meidanis' students, myself included, with an entire chapter devoted to it in this thesis. A short paper in JCB also dates from his period [75].

Walter graduated in 2000 and took back to her hometown, Brasilia, the genome rearrangement spark. Several of her students worked on the problem. Dias graduated in 2002, on the same topic, and, after spending some time in a private enterprise, is now a faculty member at the University of Campinas, where many of his students also picked up the topic.

More recently, Celina Figueiredo, a faculty member in Rio de Janeiro, with a long and successful career in graph theory, started a genome rearrangement line of research with several of her students, working mainly with the transposition diameter, which is an open

problem [32].

Very recently, Marilia Dias, a former MSc student of Meidanis, obtained her PhD degree in France under Marie-France Sagot on genome rearrangements [21]. She later spent some time in the University of Bielefeld with Jens Stoye, joining the Brazilian community as a researcher in Rio de Janeiro in 2011.

In the University of Campinas, other MSc and PhD that worked on genome rearrangement problems under the supervision of João Meidanis include Vinicius Fortuna (MSc 2005 [45]), Andre Almeida (MSc 2007 [7]), Cleber Mira (PhD 2007 [76]), Patrícia Côgo (MSc 2008 [29]), Karina Zupo de Oliveira (MSc 2010 [33]), and now myself.

This thesis is focused on two mathematical models for genome rearrangements that were introduced during the course of my doctorate studies.

This text is organized as follows: in Chapter 2, we give an overview of the current research on the field of genome rearrangements. Then, theoretical background on three different mathematical models of rearrangement is presented.

In Chapter 3, a new rearrangement model, called Single-Cut-or-Join, published by Feijão and Meidanis [40], is presented. Another new model is presented in Chapter 4, the Adjacency Algebraic Model [41], which is an extension of the Algebraic Formalism proposed by Meidanis and Dias [74]. Finally, in Chapter 5, we give concluding remarks about both of the presented models and some possibilities for future investigation.

Chapter 2

Genome Rearrangements

Genome rearrangements are evolutionary events where large, continuous pieces of the genome shuffle around, and have been studied since shortly after the very advent of genetics [73, 82, 94]. With the increased availability of whole genome sequences, gene order data have been used to estimate the evolutionary distance between present-day genomes, and to reconstruct the gene order of ancestral genomes. The inference of evolutionary scenarios based on gene order is a hard problem, with its simplest version being the pairwise genome rearrangement problem: given two genomes, represented as sequences of conserved segments called *syntenic blocks*, find the most parsimonious sequence of rearrangement events that transforms one genome into the other. The number of events of such a sequence is the *distance* between the two genomes. If fact, in a more general setting, the distance is the *sum of the weights* of all the events in the sequence, since in models where multiple types of operations are allowed, each operation can have a different weight.

When more than two genomes are considered, we have the more challenging problem of rearrangement-based phylogeny reconstruction, when we want to find an evolutive rearrangement scenario for the input genomes; in a parsimonious sense, this means finding a tree that minimizes the total number of rearrangement events (or, more generally, the sum of the event weights). Two problems are cornerstones used to find the gene order of ancient genomes in rearrangement-based phylogeny reconstruction: the median problem and the halving problem. As we will see, these problems are NP-hard in many cases, even under the simplest distances.

In the next section we will present a very brief overview of research related to genome rearrangement theory. This is not an extensive review by any means, but a highlight of important papers in this field.

2.1 Background

Several rearrangement events, or operations, have been proposed. In Figure 2.1 the most commonly studied operations are summarized. The *unsigned reversal* inverts the order in a block of genes, without changing orientation; a *signed reversal* is similar, but also changes the orientation of the genes within the inverted block; in a *transposition*, two adjacent blocks of genes exchange position within the same chromosome; a *translocation* is the operation where two different chromosomes exchange genes in their extremities; in a *block-interchange* (also called generalized transposition) two blocks of genes, not necessarily adjacent, exchange position within the same chromosome; a *fission* divides one circular chromosome into two, and a *fusion* is the inverse operation.

2.1.1 Rearrangement Distance Between Two Genomes

When analyzing two genomes in the context of rearrangement, there are two main questions to be answered: (i) what is the evolutionary distance between them? and (ii) can we find a sequence of rearrangement events that transforms one genome into the other, giving a possible evolutive scenario between the two?

One of the first attempts to answer these questions was the notion of *breakpoints*, introduced by Sankoff and Blanchette [90]. A breakpoint between two genomes occurs when a pair of genes is adjacent in one genome but not adjacent in the other. The *breakpoint distance* is then defined as the number of breakpoints between two genomes. This rearrangement distance measure is trivial to compute and, although very simple, it may be as realistic as others, more sophisticated distances, in many situations [90].

When we are interested in inferring a possible rearrangement scenario between two genomes, usually the assumption of the parsimony criteria is considered, that is, we are looking for scenarios with the smallest number of events. The problem of *pairwise rearrangement distance* can be stated as follows: given two genomes π and σ, find a *shortest* sequence of rearrangement operations that transforms π into σ. The length of such a sequence is called the *distance* between π and σ.

This problem is sometimes called *genome sorting*, because the target genome σ is usually the so-called *identity genome*, where all genes are in an ascending order. Therefore, to transform π into σ would be equivalent to sort it. Note that any given unichromosomal genome can be transformed into the identity, with a gene label change. Although the term "sort" makes perfect sense in the unichromosomal domain, it is somewhat awkward when multichromosomal genomes are considered, since there is no way to define an identity that is equivalent to any genome apart from a gene label change. Nevertheless, the term *sorting* is still used to specify that one is interested not only in the distance, but also in the rearrangement events, and we will use it in this sense throughout this text.

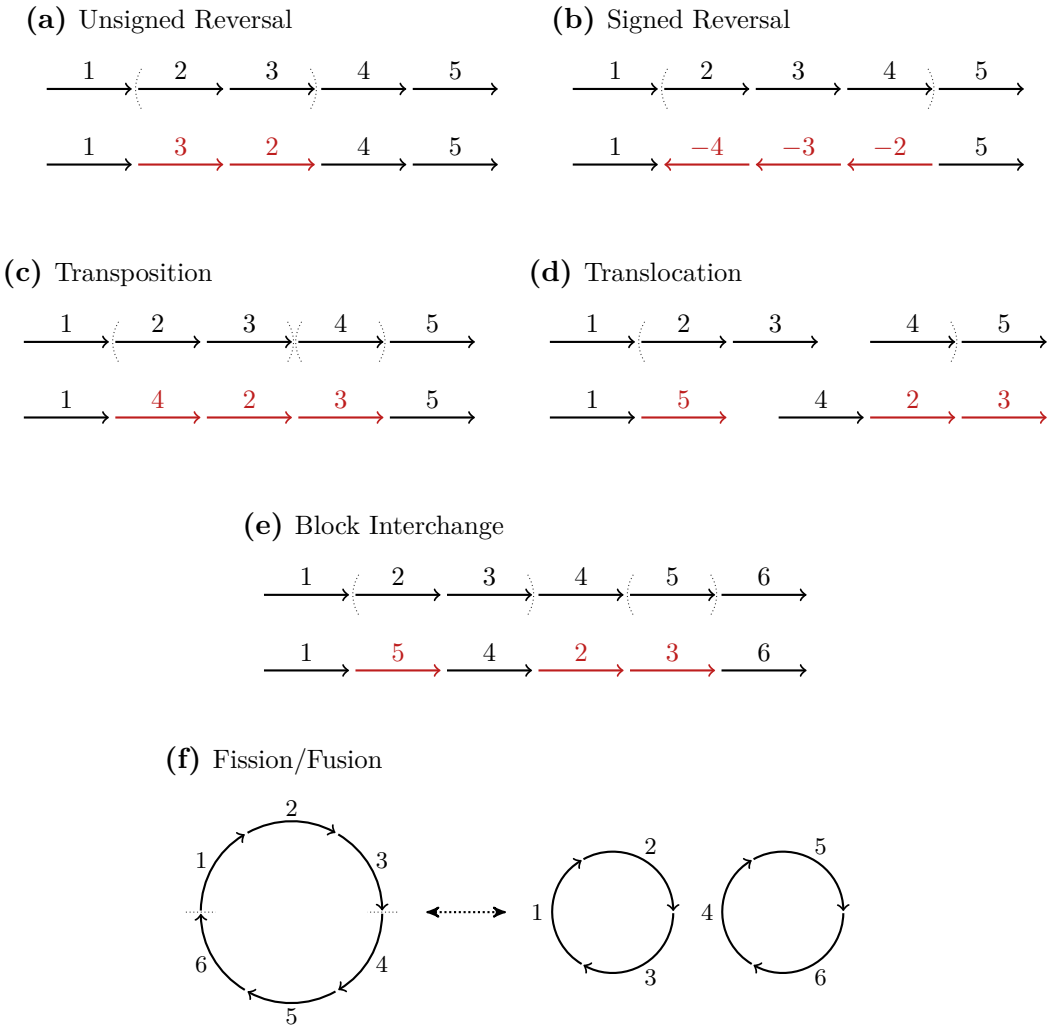


Figure 2.1: An overview of the most commonly studied rearrangement operations. Genes are represented by arrows and labeled with integers; arrow direction and corresponding integer sign indicate gene orientation; red arrows represent genes being affected by the operations, and dotted lines show where gene adjacencies are being cut. (a) *Unsigned reversal*: reversal of the order in a block of genes, without changing orientation; (b) *Signed reversal*: reversal of the order in a block of genes, also changing the orientation; (c) *Transposition*: two adjacent blocks of genes exchange positions within the same chromosome; (d) *Translocation*: two different chromosomes exchange gene blocks in their extremities; (e) *Block-interchange* (also called generalized transposition): two blocks of genes, not necessarily adjacent, exchange positions within the same chromosome; (f) *Fission*: fission of one circular chromosome into two; *Fusion*: the inverse operation.

Early approaches to solving the sorting problem considered the case where just one operation is allowed. One of the first studied operations was the unsigned reversal, introduced by Watterson et al. [106] on circular permutations. Kececioglu and Sankoff developed 2-approximation algorithms [57], later improved to a factor of $\frac{7}{4}$ by Bafna and Pevzner [10]. It was then shown to be NP-hard by Caprara [25]. More recently, Berman et al. reduced the approximation factor even further, to $\frac{11}{8}$, the best known to date [17].

The signed reversal problem has drawn much more attention, since it seems to be more biologically relevant, taking into consideration the orientation of the genes. Also, fast algorithms are currently available, as opposed to the NP-hardness of the unsigned case. From now on, when the word "reversal" is used on its own, it refers to the signed case. The reversal sorting problem was introduced by Bafna and Pevzner [10]. The first polynomial solution was developed by Hannenhalli and Pevzner [51], using the *breakpoint graph*, a structure introduced by Bafna and Pevzner [10] and largely used on several sorting algorithms (see Section 2.2.1 for the formal definition and an example of this graph). Several faster implementations followed [16, 55] and the reversal distance can now be calculated in linear time [8], but the best *sorting* algorithm — which also finds the optimal sequence of reversals — runs in $O(n^{3/2}\sqrt{\log n})$ [97].

Another largely studied rearrangement operator is the *transposition*, where two adjacent blocks are exchanged, without changing gene orientation. Since its introduction by Bafna and Pevzner [11], it was studied in several papers. Transposition sorting was recently shown to be NP-hard [24]. Current approaches strive to find the best approximation algorithms. The best algorithm so far is by Elias and Hartman [38], running in $O(n^2)$ with an approximation factor of $\frac{11}{8}$. Recently, Dias and Dias developed an algorithm with the same theoretical factor but with better experimental results [34].

A related operation is the *block-interchange*, which is a generalization of the transposition, where the exchanged blocks are not necessarily consecutive. It was introduced by Christie [28], and the block-interchange distance can be found in $O(n)$. If the distance is δ for a given instance, then the sorting problem can be solved in $O(\delta n)$ [67]. Alternatively, the sorting problem can be solved in $O(n \log n)$ using a data structure by Feng and Zhu [42].

When multichromosomal genomes are considered, additional rearrangement operations are relevant. The first studied multichromosomal operation was the *translocation*, defined by Kececioglu and Ravi [56], where two chromosomes exchange blocks at their ends. It was solved polynomially by Hannenhalli [49], with further corrections by Bergeron et al. [14]. Computing distance alone can be done in linear time (Li et al. [63]) and the sorting problem is solvable in $O(n^{3/2}\sqrt{\log n})$ [84].

Other multichromosomal operations are *fissions* and *fusions*. In the fission operation, a chromosome is split forming two new chromosomes. The fusion is the inverse operation, joining two different chromosomes into one. Algorithms that use these operations usually

include a third operation to move genes within a chromosome. Dias and Meidanis solved the problem using fissions, fusions and transpositions in $O(n^2)$ [35]. It was the first polynomial algorithm for a rearrangement problem involving transpositions. Using fissions, fusions and block-interchanges, Lu et al. [70] developed an $O(n^2)$ algorithm.

Ideally, we would like a rearrangement problem combining several types of operations in the same model, preferably assigning weights that correspond to their probability occurrence. The two papers just cited in the previous paragraph are examples of combined operation models, combining fissions and fusions with another rearrangement operation. A very well-known model was introduced by Hannenhalli and Pevzner, allowing reversals and translocations in linear genomes, sometimes called the *RT-distance* [50]. The best known algorithms for this distance run in $O(n^2)$ [15, 99]. Another notable example is a model combining reversals and block-interchanges [68, 77].

A model that combined several operations was proposed by Yancopoulos et al. [110] as a unifying operation for genomic distance, called Double Cut-and-Join (DCJ), and it has become very popular. It consists of cutting the genome in two points and rejoining the resulting four unconnected extremities. Depending on the selected edges and the possible ways of rejoining the vertices, this single operation models reversals, translocations, fissions and fusions. The authors also developed an $O(n)$ sorting algorithm using the DCJ. Bergeron, Mixtacki and Stoye [13] improved the DCJ theory introducing a structure called *adjacency graph*, a simple graph that is a union of paths and cycles and can be used to model genomes. Using this structure they developed a new distance equation and an optimal greedy sorting algorithm that runs in $O(n)$.

Although several rearrangement models were proposed, each having their precise definitions, there were no formal definitions of what is a rearrangement model. To expand in this direction, Bergeron et al. [12] proposed a formal definition of rearrangement models and operations, using it to compare various versions of the DCJ model with unichromosomal and linear/circular restrictions. They also proposed a single cut-and-join model, presenting a linear time algorithm for computing its distance.

Another single-cut model was proposed by Feijão and Meidanis [40], the Single Cut-or-Join (SCJ), as an attempt to unify the definition of the breakpoint distance, where only operations like a single cut or a single join are allowed. The SCJ is very similar to previously proposed multichromosomal breakpoint distances [86, 98], but several rearrangement problems have polynomial time solutions, where in other distances the same problems are NP-hard.

In the search for a unifying theory there is also the approach of Meidanis and Dias [74], in which they use permutation groups to develop a rearrangement theory with a strong algebraic formalism. With this formalism many different operations can be easily modeled, without the need for graphs that are often used in rearrangement algorithms. Some of

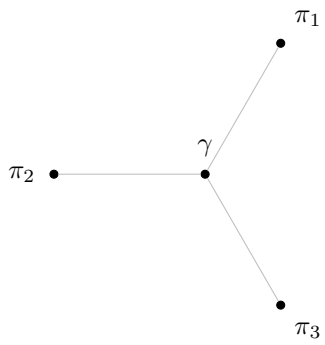


Figure 2.2: The median of three input genomes π_1, π_2 and π_3 is any genome γ that minimizes the median score $d(\pi_1, \gamma) + d(\pi_2, \gamma) + d(\pi_3, \gamma)$.

the papers already cited in this review applied this theory to develop their algorithms (for instance [35, 77]). The resulting genomic distance is equal to DCJ but is limited to circular genomes only. Recently, Feijão and Meidanis [41] extended this theory to allow linear genomes also, resulting in a model very similar to DCJ but slightly different. In fact, this extended model is an important part of this thesis. The model will be introduced in Section 2.2.2 and detailed in Chapter 4.

2.1.2 Multiple Genome Rearrangement

A natural generalization of the genome rearrangement problems is to infer evolutionary scenarios with more than two genomes, leading to the inference of ancestral genomes and rearrangement-based phylogenies.

When the number of genomes increases from two to three or more, most of the resulting problems become NP-hard, even when the problem with two genomes is trivial.

The best studied multiple genome rearrangement problem is the *genome median problem*, in which we are given three genomes and need to find a fourth one minimizing the sum of its distances to each of the other three. Such a genome is called a *median*, and the sum of the distances is the *median score*. Figure 2.2 shows a graphical representation of the median. It was proposed by Sankoff and Blanchette [90] and Blanchette et al. [19], using breakpoints as the genomic distance, and was later studied under different distances. This problem can be used to infer ancestral genomes [20] or as a subproblem for a more general multiple genome problem, as we will see later in this section.

When genomes are unichromosomal, this problem is NP-hard under the breakpoint [23, 85], reversal, and DCJ distances [27], despite the fact that the pairwise problem is polynomial in these distances. In the multichromosomal general case, when there are no restrictions as to whether the genomes are linear or circular, Tannier et al. [98] showed that the problem is still NP-hard under the DCJ distance, but it becomes polynomial under the

breakpoint distance (BP), the first polynomial time result for the median problem. Feijão and Meidanis [40] also showed that the median problem is linear under their breakpoint-like SCJ model.

Despite its NP-completeness, there have been numerous studies on exact [92, 107, 108] and heuristic [48, 62, 87] solutions for the DCJ median problem, but they are limited to small genomes.

Genome median problems aim at inferring ancestral genomes by finding a genome that is, in a sense, as close as possible to all the input genomes. Another ancestor reconstructing problem is the so-called *genome halving problem*, where it is assumed that a genome has suffered *whole genome duplication* followed by rearrangement events, and one needs to find the ancestral genome right before the duplication event.

The whole genome duplication event was postulated by Susumu Ohno in 1970 [83] and has been very controversial over the years. Recently, evidence in its favor has been found on different species, with particularly convincing examples occurring in yeast species [58, 61].

In order to understand problems with duplications, additional definitions are needed. A *duplicated gene* g is a pair of genes g_1 and g_2 identified as homologous. A *duplicated genome* is any genome defined on a set of duplicated genes. For instance, Figure 2.3 (a) shows a duplicated genome with duplicated genes 1, 2, 3 and 4, forming the gene set $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2\}$. This definition should not be confused with the concept of *doubled genome*: given a genome π with gene set $\mathcal{G}$, without duplicated genes (called an *ordinary genome*), a *doubled genome* is a genome in the set of duplicated genes from $\mathcal{G}$ where each chromosome of π is duplicated, with an arbitrary assignment of labels 1 and 2 (indicating the homologous genes). The set of all possible doubled genomes from π is denoted by $\pi \oplus \pi$, and it has an exponential number of genomes, namely 2^n, where n is the number of adjacencies in π.

Therefore, while both doubled and duplicated genomes are defined on a duplicated gene set, a doubled genome represents a genome that has just suffered a whole genome duplication, while a duplicated genome has also suffered rearrangements after the whole duplication event.

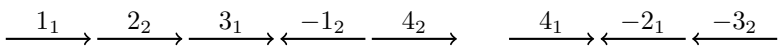
A problem that helps in the definition of the halving problem in the *double distance problem*, proposed by Alekseyev and Pevzner [4]. Given a duplicated genome γ, an ordinary genome π, and a genomic distance d, the *double distance* between π and γ is

$$dd(\gamma, \pi) = \min_{\alpha \in \pi \oplus \pi} d(\gamma, \alpha). \tag{2.1}$$

Tannier et al. showed how to solve this problem in $O(n^3)$ for the breakpoint distance [98], later improved to $O(n \log n)$ by Kováč [59]. It is NP-hard for the DCJ distance [98].

With the double distance definition, we have an easy definition for the genome halving problem: given a duplicated genome γ and a distance d between genomes, the *genome*

(a) A duplicated genome.



(b) A doubled genome.

(c) The ordinary genome that generated the doubled genome in (b).

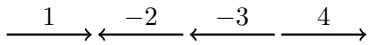


Figure 2.3: Examples of a (a) duplicated genome and a (b) doubled genome both with duplicated genes 1, 2, 3 and 4, in the gene set $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2\}$. The doubled genome (b) was obtained from the genome shown in (c).

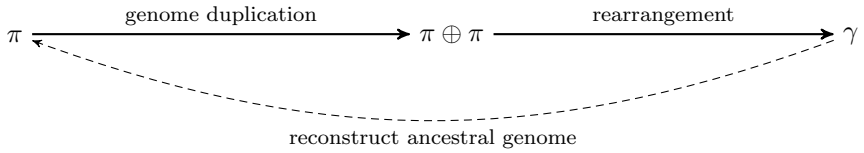


Figure 2.4: The genome halving problem consists in finding the ancestral genome π right before the whole genome duplication, given genome γ with duplicated genes that also suffered rearrangement events.

halving problem consists in finding an ordinary genome π that minimizes $dd(\gamma, \pi)$. This problem is summarized in Figure 2.4.

The genome halving problem was introduced by El-Mabrouk [36]. El-Mabrouk and Sankoff devised a linear time solution for this problem under the reversal and translocation distance [37]. Alekseyev and Sankoff solved it in $O(n^2)$ for the reversal and DCJ distances on unichromosomal circular genomes [5]. For the DCJ distance on multichromosomal genomes, there are solutions by Mixtacki [78] and also by Warren and Sankoff [104]. Kováč, Warren and Braga developed a linear time algorithm for the case where only linear genomes are allowed (the so-called restricted DCJ model) [60].

Warren and Sankoff recently proposed a generalization of the halving problem, the *genome aliquoting problem* [103]. While in the halving problem the input genome has exactly two copies of each gene, in the aliquoting problem it has p copies of each gene, with $p > 2$. The biological motivation of this problem is the polyploidation event, common in plants. In their original paper, the authors presented a heuristic algorithm for the problem under the DCJ distance. In a follow-up paper, they showed that this problem can be solved in linear time with the breakpoint distance, and proposed a 2-approximation

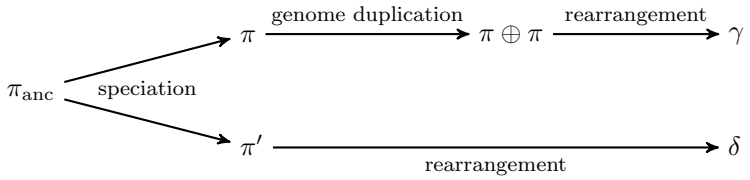


Figure 2.5: The guided halving is a variant of the halving problem where an outgroup genome δ, assumed to share a common ancestor with γ before the duplication event, is also given to guide in the reconstruction.

algorithm for the DCJ distance [105].

Seoighe and Wolfe [93] observed that the genome halving problem can yield a very large number of solutions, suggesting that this problem can be reduced by using as an outgroup a second species that diverged from the duplicated genome right before the duplication event, as shown in Figure 2.5. This problem was later proposed by Zheng et al. [114], and is called the *guided halving problem*. Formally, given a duplicated genome γ, an ordinary genome δ, and a genomic distance d, find an ordinary genome π that minimizes $dd(\gamma, \pi) + d(\delta, \pi)$. The guided halving problem under the breakpoint distance is polynomial on general multichromosomal genomes [98], but NP-hard on multichromosomal linear genomes [98] and unichromosomal genomes [59]. It is also NP-hard for the DCJ distance on multichromosomal genomes, and open in the other variants for DCJ [98].

As we will see in detail on Chapter 3, the genome halving and guided halving problems are polynomial for the SCJ distance. If fact, both problems are solved in the more general formulations, the genome aliquoting and guided aliquoting problems.

When more than three genomes are involved, we have the *multiple genome rearrangement problem*, also called the *large parsimony problem*, when we search the most parsimonious phylogenetic tree, where the given genomes (usually extant genomes) are leaves, and the inferred ancestral genomes are internal nodes. This problem can be seen as a particular case of the more general *Steiner tree problem*. We define the weight of a tree as the sum of the weights of its edges, where the weight of an edge is the weight between the vertices of the edge. The weight function is required to be a metric, and in the context of genome rearrangements it is usually a rearrangement distance, such as breakpoint or reversal distance. The Steiner tree problem can be defined as: given a graph $G(V, E)$ with weight function w and a subset of vertices $\Pi \subseteq V$, find a tree $\mathcal{T}$ with minimum weight spanning all vertices in Π.

The tree $\mathcal{T}$ is called a *Steiner tree* with *Steiner set* Π. When G is a complete graph, Π is a set of genomes, and the weight function is a rearrangement distance, the Steiner tree problem seeks a phylogenetic tree explaining the evolution of the genomes in Π under the maximum parsimony criterion. However, since the genomes in Π are usually extant genomes, from a phylogenetic point of view an additional constraint is needed; that the

genomes in Π are leaves in $\mathcal{T}$. But this is the exact definition of a *full Steiner tree*: $\mathcal{T}$ is a *full* (or *terminal*) Steiner tree when the elements in Π label the leaves of $\mathcal{T}$. With this, we can formally define the large parsimony problem: given a set of genomes $\Pi = \{\pi_1, \ldots, \pi_n\}$ from a set V and a distance d between genomes, find a full Steiner tree $\mathcal{T}$ with Steiner set Π.

The full Steiner tree is an APX-hard problem [64], admitting a 2.52-approximation algorithm [72]. However, although formally the large parsimony problem is a particular case of the full Steiner tree problem, it is not feasible to approximate the former using algorithms for the latter, because of the size of the vertex set. In the large parsimony problem, V is the set of all possible genomes, which can be huge. Specifically, for signed genomes with n genes, there are $2^n n!$ combinations.

The first approaches of solving the large parsimony problem used distances that could be easily computed in the pairwise case, such as breakpoint and reversal distances. Blanchette et al. [19] and Sankoff and Blanchette [91] proposed the *breakpoint phylogeny*: solving the large parsimony under the breakpoint distance. They proposed an algorithm, called BPAnalysis, with an approach consisting in two parts: generating all tree topologies with the input genomes as leaves, and then, for each such tree, labeling the internal nodes with genomes in order to minimize the weight of the tree. This second step is usually called the *small parsimony problem*. It is NP-hard for any distance in which the genome median problem is already NP-hard. Even for the breakpoint distance, where the multichromosomal median is polynomial, the small parsimony with four or more genomes is NP-hard [59]. The only rearrangement distance for which a polynomial algorithm for the small parsimony was introduced is the SCJ distance [40], and its solution will be shown on Chapter 3.

In the breakpoint phylogeny paper, Blanchette et al. [19] proposed an iterative heuristic — sometimes called the *Steinerization algorithm* — for the small parsimony problem that solves a genome median problem for the three neighbors of each internal node of the tree, replacing the internal node with the median, repeating until convergence is reached. The downside of this algorithm is that the median problem is NP-hard for most distances, including the breakpoint distance for unichromosomal genomes, used in the original paper.

Later, Moret et al. developed a faster alternative method called GRAPPA [79], based on BPAnalysis, that improved the speed by several orders of magnitude. Also, with the availability of a linear algorithm for reversal distance [8], the *reversal phylogeny* was included, with better results [80]. Since all possible trees must be computed, both BPAnalysis and GRAPPA are limited to trees of 15 genomes. To overcome this limitation, Tang and Moret proposed using the Disk-Covering method [54] to allow GRAPPA to scale to up to about a thousand genomes [96], trading accuracy for scalability.

Another approach to the *reversal phylogeny problem* was presented by Bourque and

Pevzner in their MGR program [20]. MGR does not search all possible trees, but instead uses a sequential addition heuristic to grow a tree one genome at a time, therefore handling a larger number of genomes than the previous approaches.

Heuristics for other rearrangement distances were proposed to solve the large parsimony problem, using an approach similar to BPAnalysis and GRAPPA. Adam and Sankoff developed a heuristic for the median problem with the Double Cut-and-Join distance and used it in their Steinerization algorithm [1]. In other example, Bader et al. [9] presented a heuristic using weighted reversals and transpositions, with different weight ratios.

Different approaches to tackling the MGRP have also been proposed in the past years. Ma et al. [71] proposed an algorithm (inferCARs) for inferring contiguous ancestral regions (CARs) of genomes at the internal nodes of a given rooted weighted tree with extant genomes at the leaves. They use a variation of Fitch's small parsimony algorithm [44], using gene adjacencies as the discrete character, to infer which adjacencies will be present in ancestral genomes. Zhao and Bourque [111, 112] presented EMRAE, an algorithm that focuses on predicting highly reliable rearrangement events, by analyzing which gene adjacencies are conserved along the edges of the phylogenetic tree. Another recent algorithm is MGRA, proposed by Alekseyev and Pevzner [6], where they formulated the MGRP as finding a shortest series of DCJ operations needed to transform a multiple breakpoint graph [26] into an identity breakpoint graph. MGRA has the advantage of not needing a phylogeny tree as input, unlike inferCARs and EMRAE.

Another possibility is building a phylogeny tree using distance-based methods, such as Neighbor-Joining [89], which runs in polynomial in the number of genomes and genes. Therefore, this approach has the advantage of being very fast, although only a phylogenetic tree is returned, without ancestral genomes in its internal nodes.

When using distance-based methods, it is important to try to estimate the *true distance* between the genomes. Since the rearrangement distance is defined as the minimum number of rearrangements, it is usually an underestimate of the true distance, and some form of statistical correction is usually applied. This kind of correction has long been used for sequence (DNA) data [95].

A number of rearrangement distance corrections were proposed, using the breakpoint distance [100, 101], inversion distance [81, 102], and DCJ distance [65]. Lin et al. later refined this DCJ distance correction including gene duplication and loss events [66]. The trees generated with the distance corrections are usually better than the uncorrected distances [102]. Distance-based methods can be a valuable method for rearrangement-based phylogenetic reconstruction, specially for their fast speed, when one in interested only in the inferred tree topology, without ancestor reconstruction.

In this text, we will show some initial results of our experiments in solving the large parsimony problem with the Single Cut-or-Join distance, in Section 3.6, that were recently

presented by Biller in her Master's thesis [18]. The main advantage of this approach is that the small parsimony is polynomial under the Single Cut-or-Join distance.

2.2 Genome Rearrangement Theory

In this section we will present different theories that were used to model genomes and rearrangement events, in the chronological order that they were introduced.

The first theory, that we call here *Classic Genome Rearrangement Theory*, is based on representing genomes as permutations, and is used in seminal works such as the first polynomial algorithm for the reversal distance, by Hannenhalli and Pevzner [51]. Later, Meidanis and Dias [74] introduced the *Algebraic Formalism*, based on representing genomes and operations using permutation group theory. More recently, a different view of genomes as a set of adjacencies (connections between adjacent genes) is becoming increasingly more popular, specially in papers using the DCJ distance (for instance, [13]). In this document we will call this approach the *Adjacency Set Theory*.

In the following three subsections we will show basic concepts of each of these theories, focusing on the last two, since they will be important for the understanding of Chapter 3, where the Adjacency Set Theory is used to define a new rearrangement model, and Chapter 4, were the Algebraic Formalism Theory will be extended to allow the modeling of linear chromosomes.

2.2.1 Classic Genome Rearrangement Theory

The first way to represent genomes in the context of rearrangement problems was by using permutations. Most of this section will be based on the excellent presentation found in the 2009 book by Fertin el al. [43].

Basic Concepts

A *permutation* π is a bijection over the set $\{1, 2, \ldots, n\}$. The image of $i \in \{1, 2, \ldots, n\}$ is denoted by $\pi(i)$ or usually in a more simplified way, as π_i. A classical notation in combinatorics to denote a permutation π is using the *two-row notation*

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{pmatrix}$$

but in the genome rearrangement literature a more compact version is used, where only the second line is kept, that is,

$$\pi = (\pi_1\ \pi_2\ \cdots\ \pi_n)$$

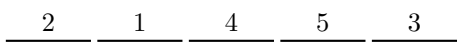


Figure 2.6: Unsigned genome with five genes that can be represented by the permutation $\pi = (2\ 1\ 4\ 5\ 3)$.

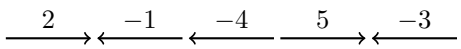


Figure 2.7: Signed genome with five genes that can be represented by the permutation $\pi = (2\ {-1}\ {-4}\ \ 5\ {-3})$.

For instance, the permutation π satisfying $\pi(1) = 2$, $\pi(2) = 1$, $\pi(3) = 4$, $\pi(4) = 5$, and $\pi(5) = 3$ is denoted by $\pi = (2\ 1\ 4\ 5\ 3)$. Figure 2.6 shows a genome that can be represented by this permutation.

To model genomes more realistically, it is more adequate to use signed permutations, because they take into account gene orientation, that corresponds to the direction in which genes are transcribed. A *signed permutation* π on $\{1, 2, \ldots, n\}$ is a permutation of the set $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ that satisfies $\pi_{-i} = -\pi_i$. The one-row notation used for unsigned permutations can also be used for signed permutations. For instance, the permutation

$$\pi = \begin{pmatrix} -4 & -3 & -2 & -1 & 1 & 2 & 3 & 4 \\ 3 & -1 & 4 & 2 & -2 & -4 & 1 & -3 \end{pmatrix}$$

can be represented as

$$\pi = (-2\ {-4}\quad 1\ {-3})$$

in which the mapping of negative elements is dropped because it is redundant. Figure 2.7 shows an example genome with the corresponding signed permutation.

The operation of *multiplication* or *composition* is applied in genomes the same way as in permutations, from right to left; for instance, the permutation $\pi \circ \sigma$ is obtained by applying σ and then π, resulting in the permutation $(\pi_{\sigma_1}\ \pi_{\sigma_2}\ \cdots\ \pi_{\sigma_n})$. For example, if $\pi = (3\ 1\ 4\ 2)$ and $\sigma = (4\ 1\ 3\ 2)$, then $\pi \circ \sigma = (2\ 3\ 4\ 1)$.

In fact, using the composition operation, we see that genome rearrangements can also be modeled as permutations. For instance, the reversal of a segment $\pi_i, \ldots, \pi_j$ $(i < j)$ from a permutation π, is modeled by the permutation

$$\rho = (1\ \cdots\ i-1\ \ j\ \ j-1\ \cdots\ i+1\ \ i\ \ j+1\ \cdots\ n)$$

in the unsigned case, and in the signed case (where the gene orientation is also reversed),

$$\rho = (1\ \cdots\ i-1\ \ -j\ \ -(j-1)\ \cdots\ -(i+1)\ \ -i\ \ j+1\ \cdots\ n)$$

and

$$\pi \circ \rho = (\pi_1 \ \cdots \ \pi_{i-1} \ \ \pi_j \ \ \pi_{j-1} \ \cdots \ \pi_{i+1} \ \ \pi_i \ \ \pi_{j+1} \ \cdots \ \pi_n)$$

and in the signed case

$$\pi \circ \rho = (\pi_1 \ \cdots \ \pi_{i-1} \ \ -\pi_j \ \ -\pi_{j-1} \ \cdots \ -\pi_{i+1} \ \ -\pi_i \ \ \pi_{j+1} \ \cdots \ \pi_n)$$

The *identity permutation* $\imath = (1 \ 2 \ \cdots \ n)$ is the neutral element for permutation composition.

The problem of transforming a genome π into another genome σ is then equivalent to finding a sequence $\rho_1, \ldots, \rho_k$ of permutations such that $\pi \circ \rho_1 \circ \cdots \circ \rho_k = \sigma$. The problem of generating a single permutation by composing a series of permutations had been studied long before its use in genome rearrangements. For a general set of allowed permutations, this problem is known to be NP-hard [39]. However, when the set is fixed beforehand, there might be an easy solution. Notable examples include the "bubble sort", when the allowed permutations are exchanges between adjacent elements; also, in the context of genome rearrangements, this problem has a polynomial time solution for some operations, for instance signed reversals, as we saw in Section 2.1.

During the development of the classical rearrangement theory, some important graph structures were proposed to help solving the sorting problems. These structures are usually based on what is called the *linear extension* of a permutation, obtained by adding two dummy genes on the extremity of the corresponding linear chromosome. Since permutations are defined in the set $\{1, \ldots, n\}$, the dummy genes are commonly labeled 0 and $n + 1$. Formally, the *linear extension* of a (signed or unsigned) permutation π of $\{1, 2, \ldots, n\}$ is the permutation of $\{0, 1, \ldots, n, n + 1\}$ defined by $\pi^l = (0 \ \ \pi_1 \ \cdots \ \pi_n \ \ n + 1)$.

The first graph structure was introduced in the context of sorting by unsigned reversals, and it was proposed by Bafna and Pevzner [10]. It is called the *breakpoint graph*, with its vertices representing genes and two types of edges, called *reality edges* (representing the input genome) and *desire edges* (representing the genome we need to get to, usually the identity). Another very similar structure is the *cycle graph*, proposed by Bafna and Pevzner later, this time in the context of sorting by transpositions [11]. The only difference between the cycle and breakpoint graphs is that the former is directed, while the latter is undirected. In fact, sometimes the terms cycle and breakpoint graph are used interchangeably.

The formal definitions are as follows: the *cycle graph* of a permutation π of $\{1, 2, \ldots, n\}$ is the directed graph $G(\pi)$ with vertex set $\{0, 1, \ldots, n, n + 1\}$ and two types of edges: (i) *black* (or *reality*) edges $(\pi^l_i, \pi^l_{(i-1)})$ for $1 \leq i \leq n + 1$; (ii) *gray* (or *desire*) edges, $(i, (i + 1))$ for $0 \leq i \leq n$.

As we saw, reality edges represent the input permutation π, and desire edges indicate what we want to obtain (the permutation $\imath$). Figure 2.8 shows the cycle graph of a permutation.

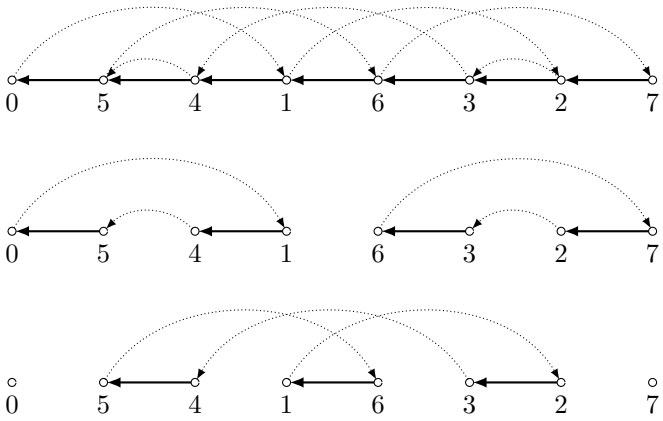


Figure 2.8: Cycle graph of the permutation $\pi = (5\ 4\ 1\ 6\ 3\ 2)$ and its decomposition into three cycles. Adapted from Fertin et al. [43, pp. 28]

The *breakpoint graph* of a permutation π, denoted $BG(\pi)$, is the undirected version of the cycle graph. An example is shown in Figure 2.9.

For signed permutations, the breakpoint graph is slightly different, to take into account the gene orientation. The *breakpoint graph* of a signed permutation π is the graph $BG(\pi) = (V, E)$ where the vertex set V contains, for $1 \leq g \leq n$, two vertices g_t and g_h, called the *tail* and the *head* of the gene, plus two vertices denoted 0_h and $(n+1)_t$. The edge set E of $BG(\pi)$ has two types of edges: (i) *desire edges* $(g_h, (g+1)_t)$ for $0 \leq g \leq n$, and (ii) *reality edges* from π_{i_h} if π_i is nonnegative, and from π_{i_t} otherwise, to $\pi_{(i+1)_t}$ if π_{i+1} is nonnegative, and to $\pi_{(i+1)_h}$ otherwise, for $0 \leq i \leq n$. We see an example of the breakpoint graph in Figure 2.10.

An important property of the breakpoint graph is that it can be decomposed in cycles that alternate reality and desire edges, and the number of cycles has a direct relationship with the distance of the input genome to the identity. Specifically, when both genomes are equal, the number of cycles is maximum, and equal to n (or $n+1$ in the signed case). Therefore, when studying a particular type of rearrangement operations, analyzing how these operations change the number of cycles in the breakpoint graph leads to distance bounds and possibly to polynomial algorithms for the sorting problem. For instance, if we define a set of rearrangement operations and are able to prove that any operation of this type can change the number of cycles in $BG(\pi)$ by h, then $d(\pi) \geq \frac{n-c(\pi)}{h}$, where $d(\pi)$ is the distance from π to the identity, and $c(\pi)$ is the number of cycles in $BG(\pi)$.

Although genomes and rearrangements are modeled by permutations, most results in the classic rearrangement theory are based on graphs, such as the breakpoint graph. In the next section, we will see a rearrangement theory that aims to use permutation group theory directly to derive its results.

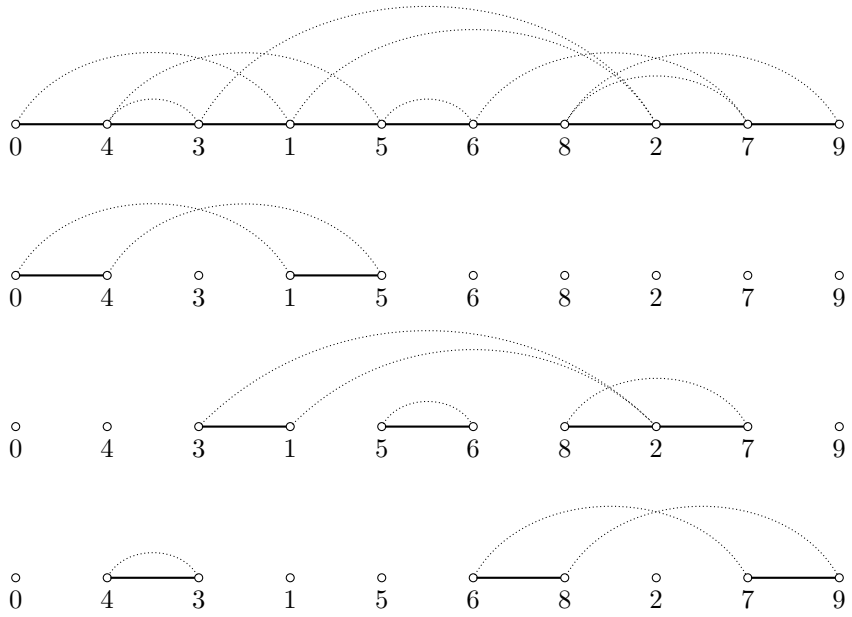


Figure 2.9: Breakpoint graph of the permutation $\pi = (4\ 3\ 1\ 5\ 6\ 8\ 2\ 7)$ and a maximal cycle decomposition into five alternating cycles. Adapted from Fertin et al. [43, pp. 42]

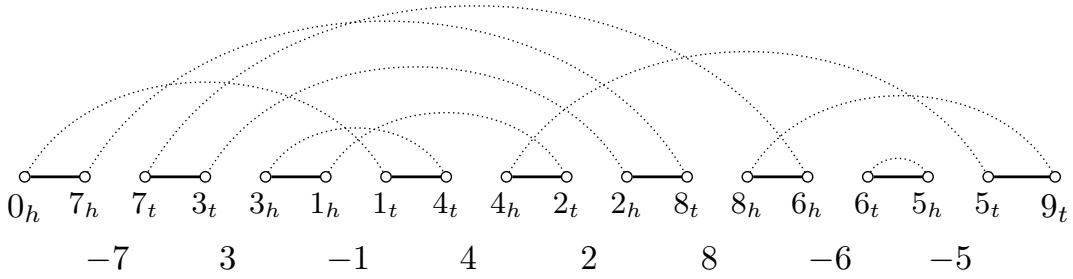


Figure 2.10: Breakpoint graph of the signed permutation $\pi = (-7\ \ 3\ -1\ \ 4\ \ 2\ \ 8\ -6\ -5\ )$. Adapted from Fertin et al. [43, pp. 65]

2.2.2 Algebraic Formalism Theory

Meidanis and Dias [74] introduced a model where permutation group theory was used to model genome rearrangement problems, and called it the *algebraic rearrangement theory*. The main motivation was to prove in an algebraic way some previous results that were proved using graph-theoretic arguments. In its original form, it is limited to circular chromosomes only. Some new problems were solved using this theory, such as sorting by fusions, fissions, and transpositions [35]. Also, some new algorithms for problems that were already solved under the classic theory, such as sorting by block-interchanges [52, 67] and sorting circular genomes with Double Cut-and-Join (2-break) operations [70, 77], gained new proofs.

Since the algebraic formalism theory is limited to circular chromosomes only, adaptations are needed in order to use it for linear chromosomes, usually in the form of dummy genes, sometimes called *caps*. For unichromosomal genomes, the linear and circular problems are usually equivalent, but this is not usually the case for multichromosomal genomes. Recently, Huang and Lu solved the problem of sorting by reversals, generalized transpositions, and translocations using permutation groups, allowing linear chromosomes [53]. This was the first model to use algebraic theory with multichromosomal linear genomes, and it is essentially the same model as the DCJ, under a different perspective. However, while DCJ is a very simple and straightforward model, the one by Huang and Lu is more complex, involving a pre-processing step for the addition of caps.

Another approach allowing linear chromosomes in the algebraic formalism theory was proposed by Feijão and Meidanis [41], in which they introduce an alternative genome representation scheme, expanding the theory but at the same time maintaining most of its important results. This new model, called *adjacency algebraic theory*, is also one of the main results of the present text and will be explained in more detail on Chapter 4.

Basic Concepts

Given a set E, a *permutation* α is a map from E onto itself, that is, $\alpha : E \to E$. The set E can be composed by any type of element, and in the original formulation by Meidanis and Dias usually lowercase roman letters are used. In this text we will use integer numbers from 1 to n, to use a closer notation to the other rearrangement theories that also usually denote genes with integers.

Permutations are represented with each element followed by its image. For instance, with $E = \{1, 2, 3\}$, $\alpha = (1\ 2\ 3)$ is the permutation where 1 is mapped to 2, which is mapped to 3, which in turn is mapped back to 1. In other words, $\alpha(1) = 2$, $\alpha(2) = 3$ and $\alpha(3) = 1$. This representation is not unique; (2 3 1) and (3 1 2) are equivalent. Figure 2.11 gives a graphical representation of the permutation (1 2 3 4 5)(6 7 8).

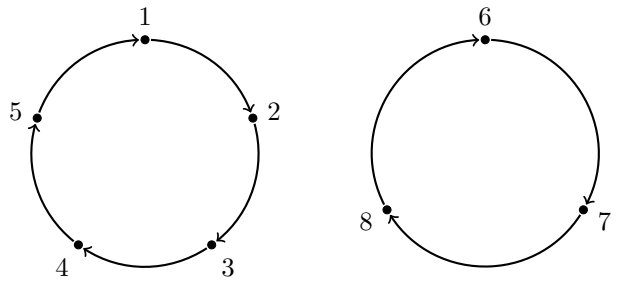


Figure 2.11: Graphic representation of the permutation (1 2 3 4 5)(6 7 8), composed by one 5-cycle and one 3-cycle.

Permutations are composed of one or more *cycles*. For instance, the permutation $\alpha = (1\ 2\ 3)(4\ 5)(6)$ on the set $E = \{1, 2, 3, 4, 5, 6\}$ 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. For instance, in the permutation α shown in this paragraph, we have the fixed element $\alpha(6) = 6$, and α is usually represented as $\alpha = (1\ 2\ 3)(4\ 5)$. Figure 2.11 shows an example of a permutation composed by two cycles.

The *support* of a permutation is the set of its non-fixed elements. In the previous example, $Supp(\alpha) = \{1, 2, 3, 4, 5\}$.

The *product* or *composition* of two permutations α, β 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 $E = \{1, 2, 3, 4, 5, 6\}$, $\alpha = (2\ 4\ 5)$ and $\beta = (3\ 1\ 5\ 2\ 6\ 4)$, we have $\alpha\beta = (3\ 1\ 2\ 6\ 5\ 4)$.

In general $\alpha\beta \neq \beta\alpha$, but when α and β are disjoint cycles, that is, when their supports do not have any element in common, they commute: $\alpha\beta = \beta\alpha$. Every permutation can be written in a 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 α has an *inverse* α^{-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: (3 2 1) is the inverse of (1 2 3).

The *conjugation* of β by α, denoted by $\alpha \cdot \beta$, is the permutation $\alpha\beta\alpha^{-1}$. This results in a permutation with the same structure as β, but with α 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 αb_i is a simpler notation for $\alpha(b_i)$.

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

For any permutations α and β we have the following known results on the norm [77]:

- $\|\alpha\| = 0$ if and only if $\alpha = \mathbf{1}$
- $\|\alpha^{-1}\| = \|\alpha\|$
- $\|\beta \cdot \alpha\| = \|\alpha\|$
- $\|\alpha\beta\| = \|\beta\alpha\|$
- $\|\alpha\beta\| \leq \|\alpha\| + \|\beta\|$

Another important concept in the algebraic theory is *divisibility*. A permutation α *divides* another permutation β if $\|\beta\alpha^{-1}\| = \|\beta\| - \|\alpha\|$. This is denoted by $\alpha|\beta$. The divisibility is *transitive*, that is, if $\alpha|\beta$ and $\beta|\gamma$, then $\alpha|\gamma$.

Modeling Genomes and Rearrangement Operations

Let $E_n = \{-1, +1, -2, +2, \ldots, -n, +n\}$, where n is the number of genes, be the base set to model genomes as permutations, representing all genes in both orientations. We also define the very important permutation Γ, as the permutation that maps each gene into its reverse complement, that is,

$$\Gamma = (-1\ +1)(-2\ +2)\ldots(-n\ +n)$$

We can think of Γ as the equivalent of a "minus sign" in the classical rearrangement theory, that is, it inverts the sign (and therefore the orientation) of a gene.

A *circular chromosome* is the product of two cycles α and $\Gamma \cdot \alpha^{-1}$, representing the strands of the chromosome. A *circular genome* is the product of disjoint circular chromosomes.

A necessary and sufficient condition for a permutation π to represent a valid genome is: (1) $\Gamma\pi\Gamma = \pi^{-1}$ and (2) no cycle in π contains both $-i$ and $+i$ for any gene i.

For instance, the circular genome depicted in Figure 2.12 is modeled by $\pi = (+1\ +2\ +3\ +4\ +5\ +6)(-6\ -5\ -4\ -3\ -2\ -1)$. Notice that $(+1\ +2\ +3\ +4\ +5\ +6) = \Gamma \cdot (-6\ -5\ -4\ -3\ -2\ -1)$, making it a valid product of cycles representing a circular chromosome, and also guaranteeing $\Gamma\pi\Gamma = \pi^{-1}$.

A *rearrangement operation* ρ applicable to a genome π is defined as a permutation ρ for which $\pi' = \rho\pi$ is a valid genome. On problems where more than one type of rearrangement operation is allowed, we need a definition of the *weight* of each operation. Usually, in the algebraic theory, the *weight* of an operation ρ is defined as $\|\rho\|/2$ (for instance [77]). With this definition, reversals, circular fusions and circular fissions are modeled with permutations formed with two 2-cycles, and have therefore weight 1, whereas transpositions are modeled with two 3-cycles, and block interchanges with four 2-cycles,

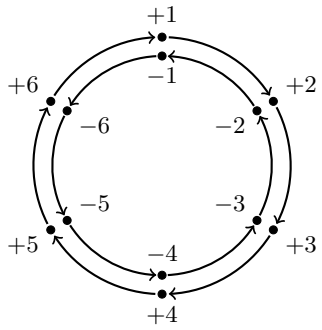


Figure 2.12: A circular genome represented by the permutation $\pi = (+1\ +2\ +3\ +4\ +5\ +6)\ (-6\ -5\ -4\ -3\ -2\ -1)$, with two 6-cycles, one for each strand.

both with a resulting weight of 2 [77]. This weight definition agrees with usual weights, for instance, the Double Cut-and-Join model for operations on circular genomes [110].

With this background, we can formulate the *algebraic rearrangement problem* as finding permutations $\rho_1, \rho_2, \ldots, \rho_n$ that minimally transform π into σ, that is, $\rho_n \ldots \rho_2 \rho_1 \pi = \sigma$, with $\rho_n \rho_{n-1} \cdots \rho_i \cdots \rho_1 \pi$ representing a valid genome for every $i = 1, \ldots, n$, and and the sum of the weight of the operations, $\sum_{i=1}^{n} \|\rho_i\|/2$, is minimum.

As we mentioned earlier, in this original algebraic theory only circular chromosomes are allowed. Recently, Feijão and Meidanis [41] extended this model to allow linear chromosomes as well by introducing an additional genome representation scheme, expanding the theory but at the same time maintaining most of its important results. This new model, called *adjacency algebraic theory*, is also one of the main results of the present text and will be explained in more detail on Chapter 4.

In the next section we will show another theory used for modeling genomes in the context of rearrangement problems.

2.2.3 Adjacency Set Theory

This theory is based on representing genomes as sets of adjacencies, the connections between consecutive genes in a genome. Several representative papers of this approach are found in more recent literature [13, 40, 98].

Basic Definitions

A *gene* is an oriented sequence of DNA that starts with a tail and ends with a head, called the *extremities* of the gene. The tail of a gene a is denoted by a_t, and its head by a_h. Given a set of genes $\mathcal{G}$, the extremity set is $\mathcal{E} = \{a_t : a \in \mathcal{G}\} \cup \{a_h : a \in \mathcal{G}\}$. An *adjacency* is an unordered pair of two extremities, which represents the linkage between two consecutive genes in a certain orientation on a chromosome, for instance $2_t 3_h$ in Figure 2.13. An

Figure 2.13: A genome $\pi = \{1_h2_h, 2_t3_h, 4_h6_h, 6_t5_t\}$ with two linear chromosomes, represented by the graph G_π above. Black directed edges represent genes, while gray edges link consecutive extremities.

extremity that is not adjacent to any other extremity is called a *telomere*, for instance, 1_t in Figure 2.13.

A *genome* is represented by a set of adjacencies where the tail and head of each gene appear at most once. Telomeres will be omitted in our representation, since they are uniquely determined by the set of adjacencies and the extremity set $\mathcal{E}$. Two adjacencies are said to be *conflicting* when they have at least one extremity in common. Thus, a genome can be characterized as a set of mutually non-conflicting adjacencies.

The *graph representation* of a genome π is a graph G_π in which the vertices are the extremities of π and there is a gray edge connecting the extremities x and y when xy is an adjacency of π or a directed black edge from x to y when they are tail and head of the same gene, respectively. A connected component in G_π is a *chromosome* of π, and it is *linear* if it is a path, and *circular* if it is a cycle. A *circular genome* is a genome composed by circular chromosomes only, and a *linear genome* is a genome with linear chromosomes only.

A *string representation* of a genome π, denoted by π_S, is a set of strings corresponding to the genes of π in the order they appear on each chromosome, with a bar over the gene if it is read from head to tail and no bar otherwise. Notice that the string representation is not unique: each chromosome can be replaced by its reverse complement.

For instance, given the set $\mathcal{G} = \{1, 2, 3, 4, 5, 6\}$, and the genome $\pi = \{1_h2_h, 2_t3_h, 4_h6_h, 6_t5_t\}$, the graph G_π is given in Figure 2.13. Notice that telomeres 1_t, 3_t, 4_t, and 5_h are omitted from the set representation without any ambiguity.

A string representation of this genome is $\pi_S = (1\ \overline{2}\ \overline{3}, 4\ \overline{6}\ 5)$.

In problems where gene duplicates are allowed, a gene can have any number of homologous copies within a genome. Each copy of a gene is called a *duplicated gene* and is identified by its tail and head with the addition of an integer superscript identifying the copy. For instance, a gene g with three copies has extremities g_h^1, g_t^1, g_h^2, g_t^2, g_h^3, and g_t^3. An *n-duplicate genome* is a genome where each gene has exactly n copies. An *ordinary genome* is a genome with a single copy of each gene. We can obtain n-duplicate genomes from an ordinary genome with the following operation [103]: for an ordinary genome π on a set $\mathcal{G}$, $n\pi \equiv \pi \oplus \pi \oplus \ldots \oplus \pi$ represents a set of n-duplicate genomes on $n\mathcal{G} = \{a^1, a^2, \ldots, a^n : a \in \mathcal{G}\}$ such that if the adjacency xy belongs to π, n adjacencies of the form x^iy^j belong to any genome in $n\pi$. The assignment of labels i and j to the duplicated adjacencies is arbitrary, with the restriction that each extremity copy has to

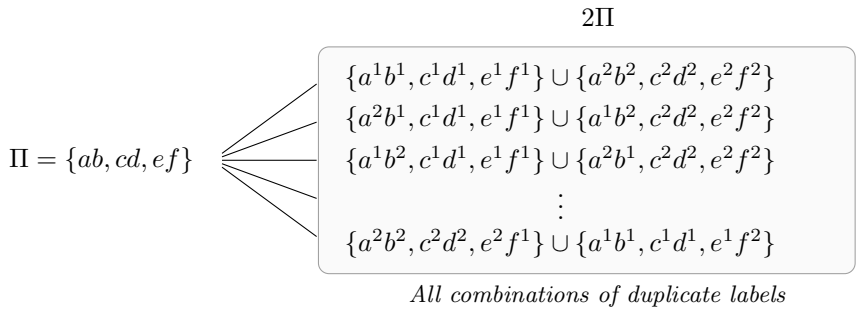


Figure 2.14: Example of the n-duplicate operation, with $n = 2$ and $\pi = \{ab, cd, ef\}$.

appear exactly once. For instance, for $n = 3$ a valid choice could be $\{x^1y^2, x^2y^1, x^3y^3\}$. Since for each adjacency in π we have $n!$ possible choices for adjacencies in $n\pi$, the number of genomes in the set $n\pi$ is $n!^{|\pi|}$, where $|\pi|$ is the number of adjacencies of π.

The theory presented in this section will be used in Chapter 3, where the rearrangement model called Single-Cut-or-Join, introduced by Feijão and Meidanis [40], will be explained in more detail.

Chapter 3

The *Single-Cut-or-Join* - a New Breakpoint-like Operation

In this chapter we present the Single-Cut-or-Join operation (SCJ), introduced by Feijão and Meidanis [40].

The motivation for this new operation is to introduce a very simple model, closely related to the breakpoint distance, but with the advantage of simplifying problems that are NP-hard under other models, such as the genome median and small parsimony, for which solutions can be found in polynomial time under the SCJ model.

This model is based on the Adjacency Set Theory, presented in Section 2.2.3. The sections in this chapter are organized as follows: in Section 3.1, the SCJ operation is defined; in the following sections, several rearrangement problems are considered under this model, such as pairwise distance (Section 3.2), genome median (Section 3.3), genome halving and aliquoting (Section 3.4), and multiple genome rearrangement (Section 3.5). In Section 3.6, some preliminary experimental results of using SCJ for phylogenetic reconstruction are shown.

3.1 Defining the SCJ Operation

First, we will define two simple operations applied directly on the adjacencies and telomeres of a genome. A *cut* is an operation that breaks an adjacency in two telomeres (namely, its extremities), and a *join* is the reverse operation, pairing two telomeres into an adjacency. These operations have been used to define other models, for instance the well-known Double-Cut-and-Join (DCJ) model [110], where any operation that is composed by two cuts followed by two joins on the extremities that where cut is considered a DCJ.

In our model any operation that is a single cut *or* a single join applied to a genome will be called a *Single-Cut-or-Join* (SCJ) operation. In this chapter we will show how to

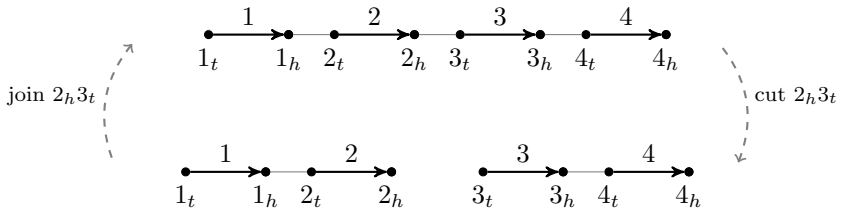


Figure 3.1: Example of cut and join operations.

solve several rearrangement problems under the SCJ model. Figure 3.1 shows an example of a cut and a join applied to a genome.

3.1.1 Rearrangement Operations Modeled as Set Operations

Since each genome is a set of adjacencies, standard set operations such as union, intersection and set difference can be applied to two (or more) genomes. In particular, we note that a cut is equivalent to the removal of an adjacency in the set that models a genome, or, more formally, a set difference between a genome π and a singleton set with the removed adjacency xy, denoted by $\pi - \{xy\}$. Equivalently, a join is the opposite operation, namely, the inclusion of an adjacency in a genome set, or the union between the genome and a singleton set with the included adjacency, denoted by $\pi \cup \{xy\}$.

When applying set operations to genomes, we must be careful not to generate an invalid genome. It is easy to see that intersection and set difference are closed operations in the genome space, that is, when both operands are genomes, the resulting set is also a genome. On the other hand, the set resulting from a union operation between genomes might not represent a genome, since the same extremity could be present in adjacencies of distinct genomes, that is, the union may have conflicting adjacencies. As we will use these set operations throughout this chapter in the presented algorithms, whenever a union is applied, we will prove that the resulting set represents a valid genome.

3.2 Pairwise Distance and Sorting

In this section, we are interested in SCJ sorting, that is, solving the pairwise distance problem using only SCJ operations. The SCJ distance is the length of a minimal sorting sequence between genomes π and σ, and is denoted as $d_{\text{SCJ}}(\pi, \sigma)$. Since the only possible operations are to remove (cut) or include (join) an adjacency in a genome, the obvious way of transforming π into σ is to remove all adjacencies that belong to π and not to σ, and then include all adjacencies that belong to σ and not to π.

Lemma 3.1. *Consider the genomes π and σ, and let $\gamma = \pi - \sigma$ and $\lambda = \sigma - \pi$. Then, γ*

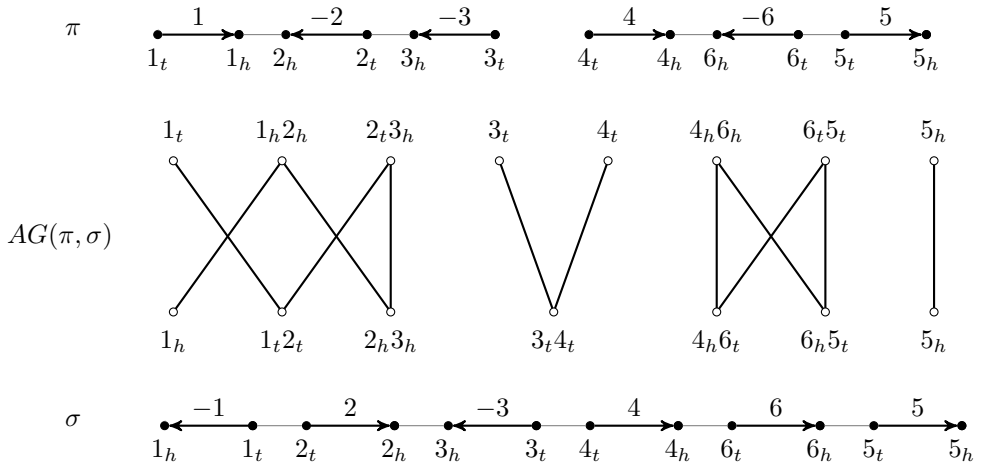


Figure 3.2: The adjacency graph $AG(\pi, \sigma)$ between genomes $\pi = \{1_t, 1_h2_h, 2_t3_h, 3_t, 4_t, 4_h6_h, 6_t5_t, 5_h\}$ and $\sigma = \{1_h, 1_t2_t, 2_h3_h, 3_t4_t, 4_h6_t, 6_h5_t, 5_h\}$, with the graph representation of π and σ shown above and below the adjacency graph, respectively.

and λ can be found in linear time and they define a minimum set of SCJ operations that transform π into σ, where adjacencies in γ define cuts and adjacencies in λ define joins. Consequently, $d_{SCJ}(\pi, \sigma) = |\pi - \sigma| + |\sigma - \pi|$.

Proof. Considering the effect an arbitrary cut or join on π can have on the quantity $f_\sigma(\pi) = |\pi - \sigma| + |\sigma - \pi|$, it is straightforward to verify that $f_\sigma(\pi)$ can increase or decrease by at most 1. Hence, the original value is a lower bound on the distance. Given that the sequence of operations proposed in the statement leads from π to σ along valid genomes in that number of steps, we have our lemma. □

3.2.1 SCJ Distance with the Adjacency Graph

The Adjacency Graph, introduced by Bergeron et al. [13], was used to find an easy equation for the DCJ distance. The adjacency graph $AG(\pi, \sigma)$ is a bipartite graph whose vertices are the adjacencies and telomeres of the genomes π and σ and whose edges connect two vertices that have a common extremity. Therefore, vertices representing adjacencies will have degree two and telomeres will have degree one, and this graph will be a union of paths and cycles.

A formula for the SCJ distance based on the cycles and paths of $AG(\pi, \sigma)$ can be easily found, as we will see in the next lemma. We will use the following notation: C and P represent the number of cycles and paths of $AG(\pi, \sigma)$, respectively, optionally followed by a subscript to indicate the number of edges (the *length*) of the cycle or path, or if the length is odd or even. For instance, P_2 is the number of paths of length two, $C_{\geq 4}$ is the

number of cycles with length four or more and P_{odd} is the number of paths with an odd length.

Lemma 3.2. *Consider two genomes π and σ with a common set of genes $\mathcal{G}$. We have*

$$d_{SCJ}(\pi, \sigma) = 2N - 2C_2 - P, \tag{3.1}$$

where N is the number of genes, C_2 is the number of cycles of length two, and P the number of paths in $AG(\pi, \sigma)$.

Proof. We know from the definition of SCJ distance and basic set theory that

$$d_{\text{SCJ}}(\pi, \sigma) = |\pi - \sigma| + |\sigma - \pi| = |\sigma| + |\pi| - 2|\sigma \cap \pi|.$$

Since the number of cycles of length two in $AG(\pi, \sigma)$ is the number of common adjacencies of π and σ, we have $|\sigma \cap \pi| = C_2$. For any genome π, we know that $|\pi| = N - t_\pi/2$, where t_π is the number of telomeres of π. Since each path in $AG(\pi, \sigma)$ has exactly two vertices corresponding to telomeres of either π or σ, the total number of paths in $AG(\pi, \sigma)$, denoted by P, is given by $P = (t_\pi + t_\sigma)/2$. Therefore,

$$\begin{aligned} d_{\text{SCJ}}(\pi, \sigma) &= |\sigma| + |\pi| - 2|\sigma \cap \pi| \\ &= 2N - (t_\pi + t_\sigma)/2 - 2C_2 \\ &= 2N - 2C_2 - P. \end{aligned}$$

□

3.2.2 Comparing the SCJ Distance to Other Distances

Based on the adjacency graph, we have the following equation for the DCJ distance [13]:

$$d_{\text{DCJ}}(\pi, \sigma) = N - C - P_{odd}/2, \tag{3.2}$$

where N is the number of genes, C is the number of cycles, and P_{odd} is the number of odd paths in $AG(\pi, \sigma)$.

For the BP distance defined by Tannier et al. [98], we have

$$d_{BP}(\pi, \sigma) = N - C_2 - P_1/2, \tag{3.3}$$

where C_2 is the number of cycles with length two and P_1 is the number of paths with length one in $AG(\pi, \sigma)$.

The breakpoint distance *br* of Pevzner and Tesler [86] is defined in a study involving linear chromosomes only (human and mouse genomes). The number of breakpoints $br(\pi, \sigma)$ is given by

$$\begin{aligned} br(\pi, \sigma) &= i(\pi, \sigma) + e(\pi, \sigma) \\ &= i(\pi, \sigma) + e_1(\pi, \sigma) + e_2(\pi, \sigma), \end{aligned}$$

where $i(\pi, \sigma)$ is the number of internal breakpoints of (π, σ), while the term $e(\pi, \sigma)$ counts the external breakpoints of (π, σ), which in turn are classified into external breakpoints of the first or second kind, and their quantities indicated by $e_1(\pi, \sigma)$ and $e_2(\pi, \sigma)$, respectively. Although Pevzner and Tesler do not explicitly state it, it is reasonable to assume that circular chromosomes do not generate external breakpoints of either kind, which permits us to extend their distance to any type of chromosome and relate it to our notation and to parameters of the adjacency graph as follows:

$$\begin{aligned} i(\pi, \sigma) &= |\pi - \sigma|, \\ e_1(\pi, \sigma) &= t_\pi - P_1, \\ e_2(\pi, \sigma) &= \max(0, \nu_\pi - \nu_\sigma), \end{aligned}$$

where $t_\pi = 2(N - |\pi|)$ is the number of telomeres of π, the term P_1 is the number of paths of length 1 in the adjacency graph, and ν_π is the number of null chromosomes in π. The null chromosomes are a device created to deal with genomes with different numbers of linear chromosomes. One needs to add as many null chromosomes to the genome with smaller number of linear chromosomes as needed to leave the number of linear chromosomes the same in both genomes.

The *br* distance is symmetric. Adding $br(\pi, \sigma)$ to $br(\sigma, \pi)$, dividing by 2, and observing that $|\nu_\pi - \nu_\sigma|$ is exactly the difference $|t_\pi/2 - t_\sigma/2|$ between the numbers of linear chromosomes in each genome, we have the following formula:

$$br(\pi, \sigma) = N - C_2 - P_1 + P/2 + |t_\pi - t_\sigma|/4. \tag{3.4}$$

Although the SCJ, DCJ, *br* and BP distances were derived from different contexts, their distance formulae show a surprisingly similar form.

With these equations we can find relationships between SCJ, *br*, BP, and DCJ distances. First, for the BP distance, we have

$$d_{\mathrm{SCJ}}(\pi, \sigma) = 2d_{BP}(\pi, \sigma) - P_{\geq 2}.$$

As expected, the SCJ distance is related to the BP distance, differing only by a factor of 2 and the term $P_{\geq 2}$, the number of paths with two or more edges. For circular genomes,

$P = 0$ and the SCJ distance is exactly twice the BP distance. For the general case, the following sandwich formula holds:

$$d_{BP}(\pi, \sigma) \leq d_{\text{SCJ}}(\pi, \sigma) \leq 2d_{BP}(\pi, \sigma). \tag{3.5}$$

The proof is outlined in Section 3.2.3. The inequalities are tight, as witnessed by genomes of the form $(a_h b_h, c_h d_h, e_h f_h, \ldots)$ and $(a_t b_t, c_t d_t, e_t f_t, \ldots)$ for the lower bound, and by circular genomes for the upper bound.

For the *br* distance, the sandwich formula is:

$$\frac{1}{2} br(\pi, \sigma) \leq d_{\text{SCJ}}(\pi, \sigma) \leq 2\, br(\pi, \sigma). \tag{3.6}$$

Again, the proof is outlined in Section 3.2.3, and circular genomes provide equality with the upper bound. Taking π to be any linear unichromosomal genome with k genes and $\sigma = \emptyset$, that is, each of the k genes alone in a linear chromosome in σ, we see that the lower bound is also tight.

For the DCJ distance, a reasonable guess is that it would be one fourth of the SCJ distance, since a DCJ operation, being formed by two cuts and two joins, should correspond to four SCJ operations. This is not always true, however, for two reasons.

To begin with, a DCJ operation may correspond to four, two, or even one SCJ operation. Examples of these three cases are shown in Figure 3.3, with *caps* represented by the symbol $\circ$. In each case the target genome is $\pi = \{1_h 2_t, 2_h 3_t, 3_h 4_t\}$, or $\pi_S = (1\ 2\ 3\ 4)$ in the string representation. The figure shows a reversal, an affix reversal, and a linear fusion, all of which have the same weight under the DCJ model, but different SCJ distances. Incidentally, they have different BP distances as well.

The second reason is that, when consecutive DCJ operations use common spots, the SCJ model is able to "cancel" operations, resulting in a shorter sequence. Both arguments show SCJ saving steps, which still leaves four times DCJ distance as an upper bound on SCJ distance. The full sandwich result is

$$d_{\text{DCJ}}(\pi, \sigma) \leq d_{\text{SCJ}}(\pi, \sigma) \leq 4d_{\text{DCJ}}(\pi, \sigma). \tag{3.7}$$

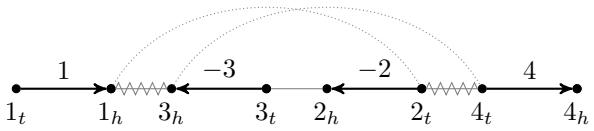
The proof is outlined in Section 3.2.3. Here again the inequalities are tight, as witnessed adjacency graphs consisting entirely of paths of length 2, for the lower bound, and by adjacency graphs consisting entirely of cycles of length 4, for the upper bound.

Another interesting relationship is

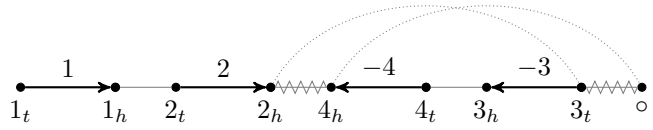
$$d_{\text{SCJ}}(\pi, \sigma) = 2d_{\text{DCJ}}(\pi, \sigma) + (2C_{\geq 4} - P_{even}),$$

coming directly from Equations 3.1 and 3.2.

(a) DCJ operation corresponding to a reversal.



(b) DCJ operation corresponding to an *affix* reversal, a reversal including a telomere.



(c) DCJ operation corresponding to a linear fusion.

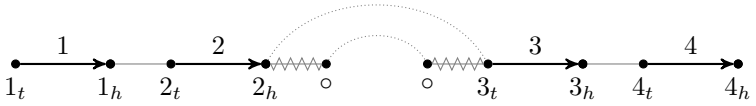


Figure 3.3: Three types of single DCJ operations transforming each genome into $\pi = \{1_h2_t, 2_h3_t, 3_h4_t\}$ (or $\pi_S = (1\ 2\ 3\ 4)$ in the string representation) that correspond to different quantities of SCJ operations. (a) Reversal – four SCJ operations. (b) *Affix* Reversal – two SCJ operations. (c) Linear Fusion – one SCJ operation. Caps are represented by ∘.

3.2.3 Alternative Distance Equations

The proofs of the sandwich results rely on the following formula, in which the number $2N$ of edges of $AG(\pi, \sigma)$ is expressed in terms of the numbers of paths and cycles of different lengths:

$$2N = \sum_{i=1}^{\infty} iC_i + \sum_{i=1}^{\infty} iP_i. \tag{3.8}$$

Using Equation 3.8 to eliminate N from Equations 3.1, 3.2, 3.3, and 3.4, we have:

$$
\begin{aligned}
d_{\mathrm{DCJ}}(\pi,\sigma) &= \sum_{i=4}^{N}\left(\frac{i}{2}-1\right)C_i + \sum_{i=2}^{N}\left\lfloor\frac{i}{2}\right\rfloor P_i \\
d_{BP}(\pi,\sigma) &= \sum_{i=4}^{N}\frac{i}{2}\,C_i + \sum_{i=2}^{N}\frac{i}{2}\,P_i \\
br(\pi,\sigma) &= \sum_{i=4}^{N}\frac{i}{2}\,C_i + \sum_{i=2}^{N}\frac{i+1}{2}\,P_i + \frac{1}{4}\,|t_\pi - t_\sigma| \\
d_{\mathrm{SCJ}}(\pi,\sigma) &= \sum_{i=4}^{N} i\,C_i + \sum_{i=2}^{N}(i-1)P_i
\end{aligned}
$$

The sandwich inequalities easily follow from these formulae, by comparing the summations term by term. One exception is the lower bound for *br*, in which we need the fact that

$$
\begin{aligned}
|t_\pi - t_\sigma| &= |(t_\pi - P_1) - (t_\sigma - P_1)| \\
&\leq |t_\pi - P_1| + |t_\sigma - P_1| \\
&\leq \sum_{i=2}^{N} P_i,
\end{aligned}
$$

because paths of length strictly greater than 1 will have their extremities in the set of non common telomeres (P_1 being the number of common telomeres). The tightness results also follow easily, for instance, since $d_{\mathrm{DCJ}} = d_{\mathrm{SCJ}}$ when just P_2 is different from zero.

Although the theoretical bounds do not seem very tight, with ratios of 2:1 between SCJ and BP (Eq. 3.5) and up to 4:1 between SCJ and *br* (Eq. 3.6) and DCJ (Eq. 3.7), we found empirically that the ratios are actually much tighter, as we can see in the scatter plots of Figure 3.4. From an initial genome with 5 linear chromosomes with 200 genes each, we generated 1000 other genomes simulating rearrangement evolution by applying between 1 and 250 random reversals, translocations and transpositions, with ratio 17:2:1, and calculating the distances between the original genome and the simulated ones.

As expected, SCJ is very close to twice the value of BP and *br* distances, differing by 1 or less in 93% of the cases for BP and 76% for *br*. As for DCJ, when the distance is small, SCJ distance is four times the DCJ distance, with this ratio decreasing as the distance increases.

3.3 The Genome Median Problem

The *Genome Median Problem* (GMP) is an important tool for phylogenetic reconstruction of trees with ancestral genomes based on rearrangement events. When genomes are

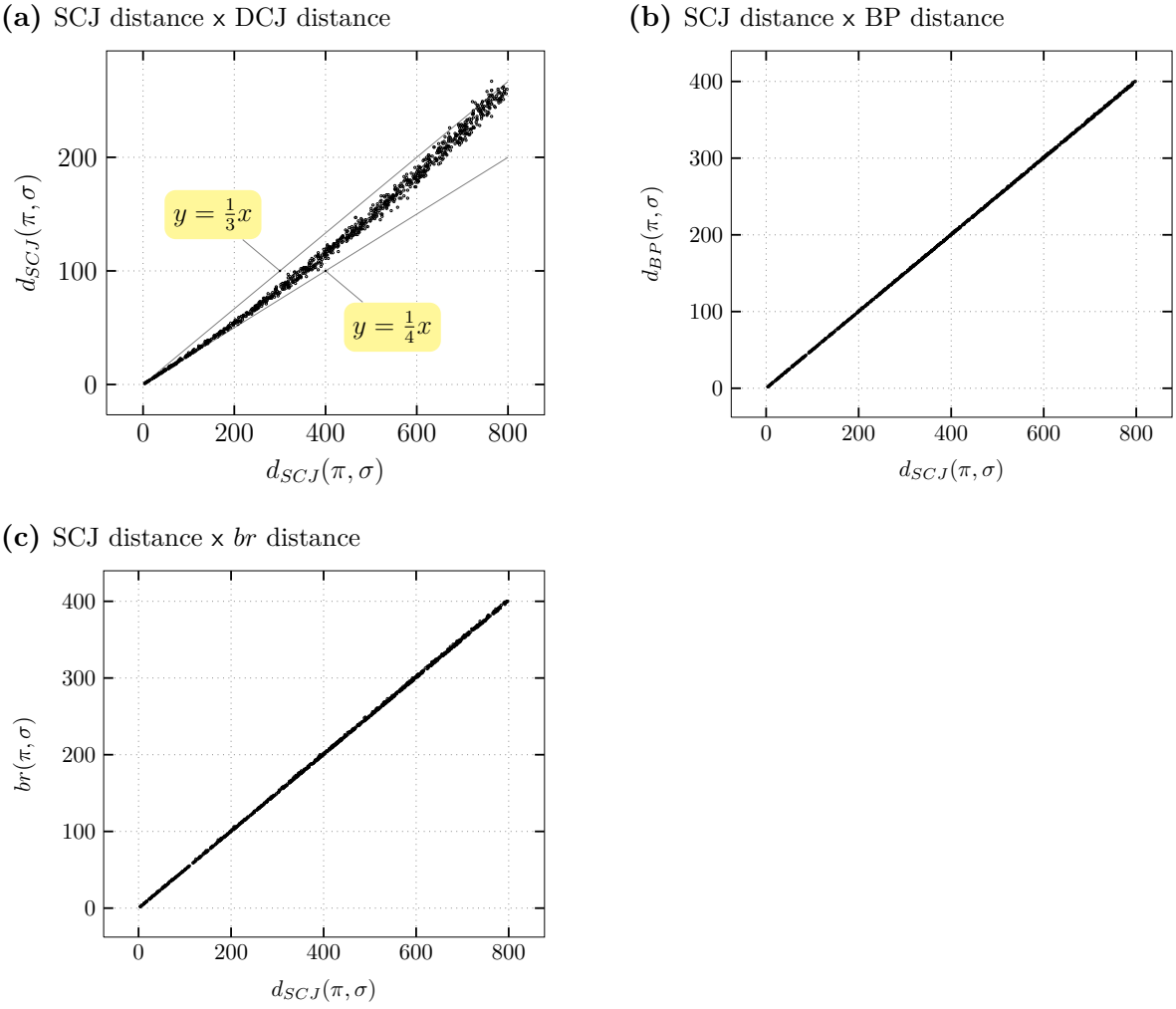


Figure 3.4: Scatter plots comparing SCJ with (a) DCJ, (b) BP, and (c) *br* distances between randomly evolved genomes.

unichromosomal this problem is NP-hard under the breakpoint [23, 85], reversal and DCJ distances [27]. In the multichromosomal general case, when there are no restrictions as to whether the genomes are linear or circular, Tannier et al. [98] recently showed that under the DCJ distance the problem is still NP-hard, but it becomes polynomial under the breakpoint distance (BP), the first polynomial time result for the median problem. The problem can be solved in linear time for SCJ, our version of the breakpoint distance.

We show this by proposing a more general problem, *Weighted Multichromosomal Genome Median Problem* (WMGMP), where we find the genome median among any number of genomes with weights for the genomes. We will give a straightforward algorithm for this problem under the SCJ distance in the general case, from which the special case of GMP follows with unique solution, and then proceed to solve it with the additional restrictions of allowing only linear or only circular chromosomes.

3.3.1 Weighted Multichromosomal Median

The WMGMP is stated as follows: Given n genomes $\pi_1, \ldots, \pi_n$ with a common set of genes $\mathcal{G}$, and nonnegative weights $w_1, \ldots, w_n$, we want to find a genome γ that minimizes the median score m_γ defined as

$$m_\gamma = \sum_{i=1}^{n} w_i \, d(\pi_i, \gamma).$$

For the SCJ distance, we know that

$$\sum_{i=1}^{n} w_i \, d_{\mathrm{SCJ}}(\pi_i, \gamma) = \sum_{i=1}^{n} w_i |\pi_i| + \sum_{i=1}^{n} w_i |\gamma| - 2 \sum_{i=1}^{n} w_i |\gamma \cap \pi_i|$$

and, since $\sum_{i=1}^{n} w_i |\pi_i|$ does not depend on γ, we want to minimize

$$f(\gamma) = \sum_{i=1}^{n} w_i |\gamma| - 2 \sum_{i=1}^{n} w_i |\gamma \cap \pi_i|. \tag{3.9}$$

To simplify the notation, we will write $f(\alpha)$ for $f(\{\alpha\})$. Then we have

$$\begin{aligned} f(\alpha) &= \sum_{i=1}^{n} w_i |\{\alpha\}| - 2 \sum_{i=1}^{n} w_i |\{\alpha\} \cap \pi_i| \\ &= \sum_{i=1}^{n} w_i - 2 \sum_{\alpha \in \pi_i} w_i = \sum_{\alpha \notin \pi_i} w_i - \sum_{\alpha \in \pi_i} w_i \end{aligned}$$

and it is easy to see that for any genome γ we have

$$f(\gamma) = \sum_{\alpha \in \gamma} f(\alpha). \tag{3.10}$$

Since we want to minimize $f(\gamma)$, a sensible approach would be to choose γ as the genome with all adjacencies α such as $f(\alpha) < 0$. As we will see from the next two lemmas, this strategy is optimal. Lemma 3.3 below shows that if an adjacency α has $f(\alpha) < 0$, then any adjacency conflicting with α has a positive value for f.

Lemma 3.3. *Given n genomes $\pi_1, \ldots, \pi_n$ and nonnegative weights $w_1, \ldots, w_n$, consider the function*

$$f(\alpha) = \sum_{\alpha \notin \pi_i} w_i - \sum_{\alpha \in \pi_i} w_i.$$

Let xy be an adjacency such that $f(xy) < 0$. Then, for any extremity z with $z \neq y$ and $z \neq x$, we have $f(xz) > 0$.

Proof. If $xy \in \pi_i$ then $xz \notin \pi_i$, and if $xz \in \pi_i$ then $xy \notin \pi_i$. Therefore,

$$\begin{aligned} f(xz) &= \sum_{xz \notin \pi_i} w_i - \sum_{xz \in \pi_i} w_i \\ &\geq \sum_{xy \in \pi_i} w_i - \sum_{xy \notin \pi_i} w_i = -f(xy) > 0. \end{aligned}$$

□

With that result, we show in the next lemma that an optimal solution can be found by selecting all adjacencies for which f is negative.

Lemma 3.4. *Given n genomes $\pi_1, \ldots, \pi_n$ and nonnegative weights $w_1, \ldots, w_n$, the genome $\gamma = \{\alpha : f(\alpha) < 0\}$ minimizes $\sum_{i=1}^{n} w_i \, d_{SCJ}(\pi_i, \gamma)$. Furthermore, if there is no adjacency $\alpha \in \pi_i$ for which $f(\alpha) = 0$, then γ is a unique solution.*

Proof. From Lemma 3.3 we know that all adjacencies α with $f(\alpha) < 0$ do not have extremities in common. Therefore, it is then possible to add all those adjacencies to form a valid genome γ, minimizing $f(\gamma)$ and consequently $\sum_{i=1}^{n} w_i \, d_{\mathrm{SCJ}}(\pi_i, \gamma)$.

To prove the uniqueness of the solution, suppose there is no adjacency α such that $f(\alpha) = 0$. Since any adjacency α belonging to γ satisfies $f(\alpha) < 0$ and any other adjacency α' satisfies $f(\alpha') > 0$, for any genome $\gamma' \neq \gamma$ we have $f(\gamma') > f(\gamma)$, confirming that γ is a unique solution. If there is α with $f(\alpha) = 0$, then $\gamma' = (\gamma \cup \alpha)$ is a valid genome (that is, the extremities of α are telomeres in γ), which is also a solution, and uniqueness does not hold. □

One corollary of this lemma is that the GMP has a unique solution under the SCJ distance, because when the number of genomes is odd and the weights are unitary there is no adjacency α with $f(\alpha) = 0$.

After solving the general case, we will restrict the problem to circular or linear genomes in the next two sections.

3.3.2 Weighted Multichromosomal Circular Median

In this section we will solve the WMGMP under SCJ restricted to circular genomes: given n *circular* genomes $\pi_1, \ldots, \pi_n$ with a common set of genes $\mathcal{G}$, and nonnegative weights $w_1, \ldots, w_n$, we want to find a *circular* genome γ which minimizes $\sum_{i=1}^{n} w_i\, d_{\mathrm{SCJ}}(\pi_i, \gamma)$.

It is easy to see that a genome is circular if and only if it has N adjacencies, where N is the number of genes. Basically we want to minimize the same function f defined in equation (3.9) with the additional constraint $|\gamma| = N$. To solve this problem, let G be a complete graph where every extremity in the set $\mathcal{E}$ is a vertex and the weight of an edge connecting vertices x and y is $f(xy)$. Then, a perfect matching on this graph corresponds to a circular genome γ and the total weight of this matching is $f(\gamma)$. A minimum weight perfect matching can be found in polynomial time [69] and it is an optimum solution to the weighted circular median problem.

3.3.3 Weighted Multichromosomal Linear Median

The solution of this problem under SCJ is found easily beginning with the same strategy as in the WMGMP. The procedure is as follows. First, construct γ as defined in Lemma 3.4, including only adjacencies for which $f < 0$. If γ is linear, this is an optimum solution. If γ has circular chromosomes, a linear median γ' can be obtained by removing, in each circular chromosome of γ, an adjacency α with maximum $f(\alpha)$. To see that this is indeed a valid solution, consider any linear genome σ minimizing f. Without loss of generality, we may assume that $\sigma \subseteq \gamma$, since adjacencies α with $f(\alpha) \geq 0$ can be deleted keeping the genome linear and without increasing $f(\sigma)$. The minimality of σ implies that it must contain all linear chromosomes of γ, and all but one adjacency from each circular chromosome of γ. In addition, the missing adjacencies α must have maximum $f(\alpha)$ in their γ chromosome. It follows that σ and γ' have the same value under f, and we conclude that γ' is indeed an optimum solution.

This is the first polynomial time result for this problem.

3.4 Genome Halving and Aliquoting

The *Genome Halving Problem* (GHP) is motivated by whole genome duplication events in molecular evolution, postulated by Susumu Ohno in 1970 [83]. Whole genome duplication has been very controversial over the years, but recently, very strong evidence in its favor was discovered in yeast species [58, 61]. The goal of a halving analysis is to reconstruct the ancestor of a 2-duplicate genome at the time of the doubling event.

The GHP is stated as follows: given a 2-duplicate genome Δ, find an ordinary genome

γ that minimizes $d(\Delta, 2\gamma)$, where

$$d(\Delta, 2\gamma) = \min_{\sigma \in 2\gamma} d(\Delta, \sigma). \tag{3.11}$$

If both Δ and γ are given, computing the right hand side of Equation (3.11) is known as the *Double Distance* problem, which has a polynomial time solution under the breakpoint distance but is NP-hard under the DCJ distance [98].

The genome halving problem was introduced by El-Mabrouk [36]. El-Mabrouk and Sankoff devised a linear time solution for this problem under the reversal and translocation distance [37]. Alekseyev and Pevzner solved it in $O(n^2)$ for the reversal and DCJ distances on unichromosomal circular genomes [5]. For the DCJ distance on multichromosomal genomes, there are solutions by Mixtacki [78] and also by Warren and Sankoff [104].

Warren and Sankoff recently proposed a generalization of the halving problem, the *Genome Aliquoting Problem* (GAP) [103]: Given an n-duplicate genome Δ, find an ordinary genome γ that minimizes

$$d(\Delta, n\gamma) = \min_{\sigma \in n\gamma} d(\Delta, \sigma).$$

In their paper, they use the DCJ distance and develop heuristics for this problem, but a polynomial time exact solution remains open. In a follow-up paper, they showed that this problem can be solved in linear time with the breakpoint distance, and proposed a 2-approximation algorithm for the DCJ distance [105]. We will show that under the SCJ distance this problem has a polynomial time solution in the general multichromosomal case.

For a n-duplicate genome Δ and an adjacency xy, let Δ_{xy} be the set of all adjacencies of the form $x^i y^j$ in Δ, that is,

$$\Delta_{xy} = \Delta \cap \left(\bigcup_{i,j=1}^{n} \{x^i y^j\} \right).$$

Let us consider the effect the presence or absence of an adjacency α in γ has on the distance $d_{\mathrm{SCJ}}(\Delta, n\gamma)$. If α is not in γ, then in any genome $\gamma' \in n\gamma$ we will have to apply exactly $|\Delta_\alpha|$ joins to include in γ' the necessary adjacencies. Conversely, if $\alpha = xy$ is present in γ, then the best course of action is to include in γ' the $|\Delta_{xy}|$ adjacencies of Δ plus $n - |\Delta_{xy}|$ arbitrarily superscripted adjacencies $x^i y^j$, provided they do not conflict with the ones already added or among themselves. In this case we still need to apply $n - |\Delta_{xy}|$ cuts, to remove the arbitrarily superscripted adjacencies $x^i y^j$ that are not present in Δ. Another factor in the distance $d_{\mathrm{SCJ}}(\Delta, n\gamma)$ is the presence of adjacencies of the form $x^i x^j$ in Δ. These adjacencies cannot occur in any genome from $n\gamma$, and will always result in

extra cuts. Adding the contribution of all adjacencies, and using the indicator function $\delta_\alpha(\pi)$ defined as

$$\delta_\alpha(\pi) = \begin{cases} 1, & \alpha \in \pi \\ 0, & \alpha \notin \pi, \end{cases} \tag{3.12}$$

we have

$$\begin{aligned} d_{\text{SCJ}}(\Delta, n\gamma) &= \sum_{\alpha \notin \gamma} |\Delta_\alpha| + \sum_{\alpha \in \gamma} (n - |\Delta_\alpha|) + \sum_{x \in \mathcal{E}} |\Delta_{xx}| \\ &= \sum_{\alpha \in A} [(1 - \delta_\alpha(\gamma))|\Delta_\alpha| + \delta_\alpha(\gamma)(n - |\Delta_\alpha|)] + \sum_{x \in \mathcal{E}} |\Delta_{xx}| \\ &= \sum_{\alpha \in A} [|\Delta_\alpha| + \delta_\alpha(\gamma)(n - 2|\Delta_\alpha|)] + \sum_{x \in \mathcal{E}} |\Delta_{xx}| \\ &= |\Delta| + \sum_{\alpha \in A} \delta_\alpha(\gamma)(n - 2|\Delta_\alpha|) \end{aligned}$$

where A is the set of all adjacencies xy with $x \neq y$.

Since we want to minimize $d_{\text{SCJ}}(\Delta, n\gamma)$, a sensible approach would be to choose γ as the genome with all adjacencies α such as $n - 2|\Delta_\alpha| < 0$. As we will see from the next two lemmas, this strategy is optimal.

Lemma 3.5. *Given an n-duplicate genome Δ, let xy be an adjacency such that $n - 2|\Delta_{xy}| < 0$. Then, for any extremity z with $z \neq y$ and $z \neq x$, we have $n - 2|\Delta_{xz}| > 0$.*

Proof. We have that $|\Delta_{xy}| + |\Delta_{xz}| \leq n$, because at most n copies of extremity x can belong to Δ. Using this equation and $n - 2|\Delta_{xy}| < 0$, we have our result. □

Lemma 3.6. *Given an n-duplicate genome Δ, the genome $\gamma = \{\alpha : n - 2|\Delta_\alpha| < 0\}$ minimizes $d_{SCJ}(\Delta, n\gamma)$. Furthermore, if there is no adjacency α for which $n - 2|\Delta_\alpha| = 0$, then γ is a unique solution.*

Proof. From Lemma 3.5 we know that all adjacencies α with $n - 2|\Delta_\alpha| < 0$ do not have extremities in common. Therefore, it is possible to add all these adjacencies to form a valid genome γ, minimizing $\sum_{\alpha \in A} \delta_\alpha(\gamma)(n - 2|\Delta_\alpha|)$ and consequently $d_{\text{SCJ}}(\Delta, n\gamma)$.

The unicity can be proved using the same arguments as in Lemma 3.3. □

Both linear and circular constrained versions of the GHP can be solved polynomially using the same procedures as in Sections 3.3.2 and 3.3.3.

3.4.1 Guided Genome Halving

Seoighe and Wolfe [93] observed that the genome halving problem can yield a very large number of solutions, suggesting that this problem can be reduced by using as an outgroup a second species that diverged from the duplicated genome right before the duplication event. This problem was later proposed by Zheng et al. [114], and is called the *Guided Genome Halving* (GGH). Formally, it can be stated as follows: given a 2-duplicate genome Δ and an ordinary genome γ, find an ordinary genome π that minimizes $d(\Delta, 2\pi) + d(\gamma, \pi)$. The guided genome halving problem under the breakpoint distance is polynomial on general multichromosomal genomes [98], but NP-hard on multichromosomal linear genomes [98] and unichromosomal genomes [59]. It is also NP-hard for the DCJ distance on multichromosomal genomes, and open in the other variants for DCJ [98].

As in the Halving Problem, here we will solve a generalization of GGH, the *Guided Genome Aliquoting* problem: given an n-duplicate genome Δ and an ordinary genome γ, find an ordinary genome π that minimizes $d(\Delta, n\pi) + d(\gamma, \pi)$. It turns out that the version with an n-duplicate genome Δ as input is very similar to the "unguided" version with an $(n+1)$-duplicate genome.

Lemma 3.7. *Given an n-duplicate genome Δ and an ordinary genome γ, let Δ' be the $(n+1)$-duplicate genome defined as $\Delta' = \Delta \cup \{x^{n+1}y^{n+1} : xy \in \gamma\}$. Then, for any genome π, we have $d_{SCJ}(\Delta', (n+1)\pi) = d_{SCJ}(\Delta, n\pi) + d_{SCJ}(\gamma, \pi)$.*

Proof. We have that

$$\min_{\sigma' \in (n+1)\pi} d_{\mathrm{SCJ}}(\Delta', \sigma') = |\Delta'| + (n+1)|\pi| - 2 \max_{\sigma' \in (n+1)\pi} |\Delta' \cap \sigma'|.$$

We notice that Δ' does not have adjacencies mixing superscript $n+1$ with superscripts up to n. Then, to maximize $\Delta' \cap \sigma'$, we must include in σ' all $|\gamma \cap \pi|$ common adjacencies between γ and π with both superscripts equal to $n+1$, add the remaining π adjacencies also with both superscripts equal to $n+1$, and maximize over the part with superscripts up to n, yielding

$$\max_{\sigma' \in \pi} |\Delta' \cap \sigma'| = |\gamma \cap \pi| + \max_{\sigma \in n\pi} |\Delta \cap \sigma|.$$

On the other hand,

$$\begin{aligned} \min_{\sigma \in n\pi} d_{\mathrm{SCJ}}(\Delta, \sigma) &= \min_{\sigma \in n\pi} (|\Delta| + |\sigma| - 2|\Delta \cap \sigma|) \\ &= |\Delta| + n|\pi| - 2 \max_{\sigma \in n\pi} |\Delta \cap \sigma|. \end{aligned}$$

Since $|\Delta'| = |\Delta| + |\gamma|$, and $d_{\mathrm{SCJ}}(\gamma, \pi) = |\gamma| + |\pi| - 2|\gamma \cap \pi|$, we have our result. □

The last lemma implies that GGH is a special case of GAP, therefore the constrained linear or circular versions are also polynomial for GGH in the SCJ model.

3.5 Multiple Genome Rearrangement

In the multiple genome rearrangement problem, given more than three genomes, we search the most parsimonious phylogenetic tree where the given genomes (usually extant genomes) are leaves, and the inferred ancestral genomes are internal nodes. In the parsimonious sense, this corresponds to finding a tree with minimum weight, where the weight of a tree is the sum of the weight of all its edges, and in turn the weight of an edge is the rearrangement distance between the genomes at its vertices.

This problem is also called the *large parsimony problem*, in contrast with the *small parsimony problem*, where a tree is given, and the problem is to find only internal nodes that minimize the weight of the tree.

In the next sections, the small and large parsimony problems will be considered under the SCJ distance.

3.5.1 The Small Parsimony Problem

Given a tree T in which each leaf corresponds to a genome defined over a common set of genes $\mathcal{G}$, the *small parsimony problem* (SPP) consists in finding an ancestral genome γ_v for each internal node v of T such that the total branch length of T (the sum of the weight of each edge, defined as the distance between the genomes of its incident vertices) is minimized. Formally, we want to find

$$M = \min_{G} \sum_{uv \in E(T)} d(\gamma_u, \gamma_v), \tag{3.13}$$

where $E(T)$ is the set of edges of T and G is the mapping from v to γ_v.

A traditional way to solve this problem is the approach proposed by Blanchette et al. [19], where one iterates over each internal node of T, solving a GMP until convergence to a local minimum is achieved. This method has been used with some rearrangement distances, such as BP [19, 20], reversal [80] and DCJ [1] distances. One difficulty with this technique is that the GMP is NP-hard for most rearrangement distances, notable exceptions being the SCJ distance [40] and the BP distance in some specific cases [98]. Since the GMP is easy under the SCJ distance, we could use the same approach, but we will show the stronger result that the SPP has a polynomial time solution under the SCJ distance, the first (and, to the best of our knowledge, the only known) polynomial time result for this problem under any proposed rearrangement distance.

In the SCJ model we can think of each adjacency as a binary character, and charge a unitary cost for including or removing an adjacency — changing a character. Using this analogy, we can solve the SPP running Fitch's algorithm for small parsimony with binary characters [44] for every adjacency, deciding which ancestral genomes contain the

adjacency. When the characters are independent, inference of ancestral genomes is just a matter of running Fitch's algorithm for each character and joining the results. In our case, however, the characters are not independent, since conflicting adjacencies cannot belong simultaneously to the same genome. Nevertheless, with a little care, this very strategy generates valid ancestral genomes in polynomial time and is indeed optimal, as we will see in Theorem 3.1.

Fitch's algorithm runs on rooted trees, whereas SPP has an unrooted tree as its input. The usual procedure to transform an unrooted problem into a rooted one, which works here as well, is to choose any edge, remove it, and make its two extremes the children of a newly created root. Afterwards, the root is simply removed and its two children linked by an edge. The total branch length is conserved in this procedure. If we have an outgroup in our genome set, the edge leading to it is the natural candidate. Otherwise, any edge will do.

We need some additional definitions related to Fitch's algorithm before proceeding. In the first part of the algorithm, every internal node is assigned a set depending on the sets of its children, in a bottom-up way. We will call these sets *bottom-up sets*. Furthermore, given an adjacency α and a rooted tree T with leaves corresponding to the genomes being analyzed, consider the result of running Fitch's algorithm on tree T, using as binary characteristic the presence of adjacency of α, and initializing the root with zero (absence) when its bottom-up set is $\{0, 1\}$. We denote by $B(\alpha, v)$ the bottom-up set generated for node v of T in the first (bottom-up) pass of the algorithm, and by $F(\alpha, v)$ the final character assignment to node v.

It is important to notice that in the original Fitch's algorithm, whenever the bottom-up set of the root contains more than one element, we are free to choose any element for the root's state. However, in our case, since we need to avoid conflicts, the safest bet is to *not* include the adjacency. Therefore, when the bottom-up set of the root is $\{0, 1\}$, we will always initialize it with 0.

Before we prove the main theorem, we will need the results of two preliminary lemmas.

Lemma 3.8. *Given two conflicting adjacencies α and β, we have the following result: for each node v of T, if $B(\alpha, v) = \{1\}$, then $B(\beta, v) = \{0\}$.*

Proof. We will prove the result by strong induction on h, the height of an internal node v, defined as the maximum length of a path from v to any of its descendant leaves.

When $h = 0$, the node is a leaf, and both $B(\alpha, v)$ and $B(\beta, v)$ are singletons. If $B(\alpha, v) = \{1\}$, the corresponding genome contains adjacency α, and therefore does not contain adjacency β, since they are conflicting, giving us $B(\beta, v) = \{0\}$. The property is therefore valid for nodes with $h = 0$.

For an internal node v with height $h \geq 1$, assume by induction that any node with height $k < h$ satisfies the property. By definition of height, both children of v have heights

strictly smaller than h, and therefore satisfy the property. To prove the property for v, it can be observed that there are two ways for $B(\alpha, v)$ to be $\{1\}$, shown in Figure 3.5. In case (a), both children have sets $\{1\}$ with respect to α. Therefore, by the induction hypothesis, both children have sets $\{0\}$ with respect to β, implying that $B(\beta, v) = \{0\}$. In case (b), one child has set $\{1\}$ and the other has set $\{0, 1\}$ with respect to α. Then, with respect to β, one child has set $\{0\}$ and the other cannot have set $\{1\}$ (otherwise the corresponding set with respect to α would be $\{0\}$, by the induction hypothesis), again implying that $B(\beta, v) = \{0\}$. □

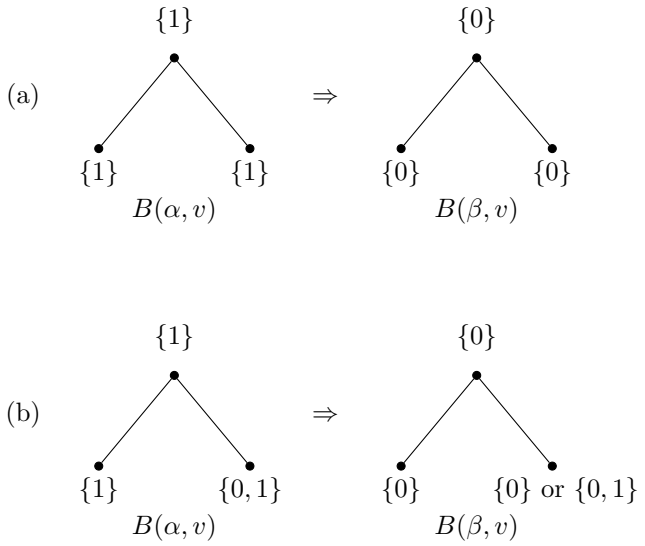


Figure 3.5: Possible cases for an internal node with bottom-up set $\{1\}$, where α and β are conflicting adjacencies.

Lemma 3.9. *Given two conflicting adjacencies α and β, for every node v of T, if $F(\alpha, v) = 1$ then $F(\beta, v) = 0$.*

Proof. The proof is by contradiction. Suppose that there are internal nodes with value 1 with respect to both α and β. Choose such a node with minimum depth (distance from root to node) and call it v. Since $F(\alpha, v) = 1$, we have that $B(\alpha, v)$ is either $\{1\}$ or $\{0, 1\}$. It cannot be $\{1\}$, otherwise Lemma 3.8 would imply that $B(\beta, v) = \{0\}$, contradicting the fact that $F(\beta, v) = 1$. Therefore, $B(\alpha, v) = \{0, 1\}$. The same reasoning applies exchanging α and β, implying that $B(\beta, v) = \{0, 1\}$ as well.

We now notice that v cannot be the root of the tree, otherwise $F(\alpha, v) = F(\beta, v) = 0$ by the way we set up Fitch's algorithm to decide ties at the root. Since v is not the root, its parent node p must have $F(\alpha, p) = 1$, because this is the only explanation for

having $F(\alpha, v) = 1$ given that $B(\alpha, v) = \{0, 1\}$. Similarly, we must have $F(\beta, p) = 1$, contradicting the minimality of v's depth, and concluding the proof. □

Theorem 3.1. *Consider a rooted tree T with leaves corresponding to genomes over a common set of genes $\mathcal{G}$. Then the sets $\gamma_v = \{\alpha : F(\alpha, v) = 1\}$, where v is an internal node of T, are valid genomes and assigning γ_v to node v minimizes the total SCJ branch length of the tree T.*

Proof. From the definition of SCJ distance, we derive a distance equation where the contribution of each adjacency is independent:

$$\begin{aligned} d_{\text{SCJ}}(\pi_1, \pi_2) &= |\pi_1| + |\pi_1| - 2|\pi_1 \cap \pi_2| \\ &= \sum_{\alpha \in A} \Big(\delta_\alpha(\pi_1) + \delta_\alpha(\pi_2) - 2\delta_\alpha(\pi_1 \cap \pi_2)\Big), \end{aligned}$$

where A is the set of all adjacencies and δ_α is the indicator function defined in Equation (3.12).

We want to minimize the total branch length of the tree, which can be written as:

$$\begin{aligned} M &= \min_G \sum_{uv \in E(T)} d_{\text{SCJ}}(\gamma_u, \gamma_v) \\ &= \min_G \sum_{uv \in E(T)} \sum_{\alpha \in A} \Big(\delta_\alpha(\gamma_u) + \delta_\alpha(\gamma_v) - 2\delta_\alpha(\gamma_u \cap \gamma_v)\Big) \\ &= \min_G \sum_{\alpha \in A} \sum_{uv \in E(T)} \Big(\delta_\alpha(\gamma_u) + \delta_\alpha(\gamma_v) - 2\delta_\alpha(\gamma_u \cap \gamma_v)\Big) \\ &= \sum_{\alpha \in A} \min_G \sum_{uv \in E(T)} \Big(\delta_\alpha(\gamma_u) + \delta_\alpha(\gamma_v) - 2\delta_\alpha(\gamma_u \cap \gamma_v)\Big). \end{aligned}$$

Fitch's algorithm guarantees that, given an adjacency α, the minimum

$$\min_G \sum_{uv \in E(T)} \Big(\delta_\alpha(\gamma_u) + \delta_\alpha(\gamma_v) - 2\delta_\alpha(\gamma_u \cap \gamma_v)\Big)$$

is achieved including the adjacency α in every node v where $F(\alpha, v) = 1$. Repeating this procedure for each adjacency, we build the genomes γ_v, which are valid because no pair of conflicting adjacencies can belong to the same genome, as a result of Lemma 3.9. Therefore, assigning genome γ_v to node v for all v in T minimizes the total sum of the branch lengths. □

3.5.2 The Large Parsimony Problem

The large parsimony problem under SCJ can be stated as follows. Given n genomes $\pi_1, \ldots, \pi_n$ defined on a common set of genes $\mathcal{G}$, find a tree T whose leaves are in one-to-one correspondence with the genomes $\pi_1, \ldots, \pi_n$, and find an ancestral genome γ_v for each

internal node v of T so that the total branch length of T (the sum of the weights of all edges, defined as the distance between the genomes of its vertices) is minimized.

To prove that the large parsimony problem under the SCJ distance is NP-hard, we will use a reduction from the Steiner tree problem in $\{0,1\}^N$ under the Hamming distance, which is NP-hard [46].

Again, we can think of each adjacency as a binary character, and charge a unitary cost for including or removing an adjacency — changing a character. Using this analogy, we can easily see that the Hamming distance between two vectors in $\{0,1\}^N$ is the same as the SCJ distance between two genomes if we choose N appropriate adjacencies, assuming that 0 and 1 correspond to absence and presence of the adjacencies in the genomes, respectively. We must choose the N adjacencies carefully though, otherwise conflicting adjacencies might arise.

It is actually very simple to code any Steiner tree problem in $\{0,1\}^N$ as a SCJ large parsimony problem while avoiding conflicting adjacencies. Given a vector $v = (v_1, v_2, \ldots, v_N)$ of size N, consider the set of genes $\mathcal{G} = \{g^1, g^2, ..., g^N\}$ and code v as a genome $\pi(v)$ having adjacencies:

$$\pi(v) = \{g_h^i g_t^{i+1} : 1 \leq i \leq N \text{ and } v_i = 1\},$$

where the sum $i+1$ "wraps around", that is, when it becomes $N+1$ we assume the value is in fact 1.

Given a set V of n binary vectors of size N, think about the instance of the large parsimony problem under SCJ formed by the genomes $\pi(v)$ for $v \in V$. With no loss of generality, there is a solution to this problem containing only adjacencies present in at least one of the input genomes, since any other adjacency can be safely removed throughout without increasing the total distance. Such a solution can be translated back to $\{0,1\}^N$ vectors, that is, for each genome π at an internal node there will be a unique vector v such that $\pi(v) = \pi$. This yields a solution to the original Steiner tree problem, because the coding preserves distances (Hamming distance in the origin, SCJ distance in the destination). Therefore, the SCJ large parsimony problem is NP-hard.

Table 3.1 summarizes the complexity of the problems we treated in this chapter under the Breakpoint, DCJ and SCJ models.

3.6 Experimental Results

To verify the accuracy of the SCJ model as a tool for phylogenetic reconstruction, a series of initial experiments was performed, with the collaboration of the student Priscila Biller. The results of these experiments are thoroughly explained in Biller's Master's thesis [18]. In this section, we will give a brief overview of the most important results.

	Problems				
Models	Distance	Median	Halving	Guided Halving	SPP
Breakpoint					
Unichromosomal	P	NP [23]	open	open	NP [23]
Multichr., general	P [98]	P [98]	P [98]	P [98]	NP [59]
Multichr., linear	P [98]	NP [98]	open	NP [113]	NP [98]
DCJ					
Unichromosomal	P [13, 110]	NP [27]	P [4]	open	NP [27]
Multichr., general	P [13, 110]	NP [98]	P [78, 104]	NP [98]	NP [98]
Multichr., linear	P [60, 110]	NP [60]	P [60]	open	NP [60]
SCJ					
Unichromosomal	P	open	open	open	open
Multichr., general	P	P	P	P	P
Multichr., linear	P	P	P	P	open

Table 3.1: Complexity of several rearrangement problems under different distances. Aliquoting and Guided Aliquoting Problems are not listed, but are polynomial for SCJ and open for the other distances.

3.6.1 Experiment Setup

We investigated two main questions: first, how accurate are the reconstructed topologies? Second, how accurate are the reconstructed ancestral genomes?

To answer these questions, we developed two experiments. In the first experiment, we solved the large parsimony problem for simulated genome trees, giving as input only the leaf nodes, and comparing the inferred trees with the simulated ones using the Robinson-Foulds (RF) distance [88]. Because we used trees from different sizes, we measured what we will call the *RF error*, the actual RF distance divided by the maximum RF distance for a given tree size.

In the second experiment, where the objective was to measure the accuracy of the reconstructed ancestral genomes, we solved the small parsimony problem for simulated trees and analyzed the percentage of reconstructed adjacencies that were present on the original simulated genomes, also checking for false-positives and false-negatives.

There were two types of data: real datasets and simulated datasets. We used two real datasets; the *Campanulaceae* family [31], with 13 genomes and 105 chloroplast DNA markers, and another with the mitochondrial DNA of 66 *Protostome* species, with 36 genes each [47].

For the simulated datasets, we first generated several random trees using the *beta-splitting* model, proposed by Aldous [2], for which parameter β, that varies from -2 to ∞, alters the balancing of the simulated trees. We used $\beta = -1$, which, according to Aldous, is the value of β that better simulates the balancing observed in real trees. We generated trees with 12, 32, 64, 128, and 200 leaves.

Given a random tree, we simulated an evolutionary scenario by creating an arbitrary genome at the tree's root, then applying a random number of rearrangement events on each tree edge until all internal nodes and leaves were generated. Several parameters were used in this simulated evolution, but in this summary we will focus on the most influential parameters: (a) the number of events per edge, also called *rearrangement rate* (varying from 5% to 25% of the number of genes in the initial genome) and (b) the number of tree leaves of the simulated tree. The genome at the tree root was initialized with 2000 genes in 5 linear chromosomes, and the random rearrangements were reversals and translocations, with 9:1 ratio.

The simulated evolutionary scenario was then used as an input for both experiments. In the case of the large parsimony problem, only the leaf nodes were given as input, whereas in the small parsimony problem, the tree topology was also given.

In our current implementation, the large parsimony problem was solved exactly using a branch-and-bound method for trees with up to 12 leaves. For larger trees, we used a sequential addition-based heuristic, where at each step we choose a genome to include in the current tree and solve a number of genome median problems to greedily find the most suitable position for this new genome. This strategy is similar to previously used methods (for instance, [43, pp. 216]).

For the small parsimony problem, the polynomial algorithm shown in Section 3.5.1 was used.

3.6.2 Results

Large Parsimony Problem - topology reconstruction

For the simulated datasets, when varying the number of genomes, the average RF distance ranged from 10% of the maximum RF distance, for trees with 12 genomes, up to 30%, for trees with 200 genomes, using a fixed rearrangement rate of 20%. When varying the rearrangement rate, using trees with 64 genomes, RF distances varied from 20% (with a 5% rearrangement rate) up to 35% (with a 25% rearrangement rate).

For the real datasets, the resulting trees were compared to trees proposed in similar studies, and in both datasets the resulting trees were mostly in agreement with previously proposed trees. For instance, in Figure 3.6, we see that the SCJ tree is very similar to the other trees.

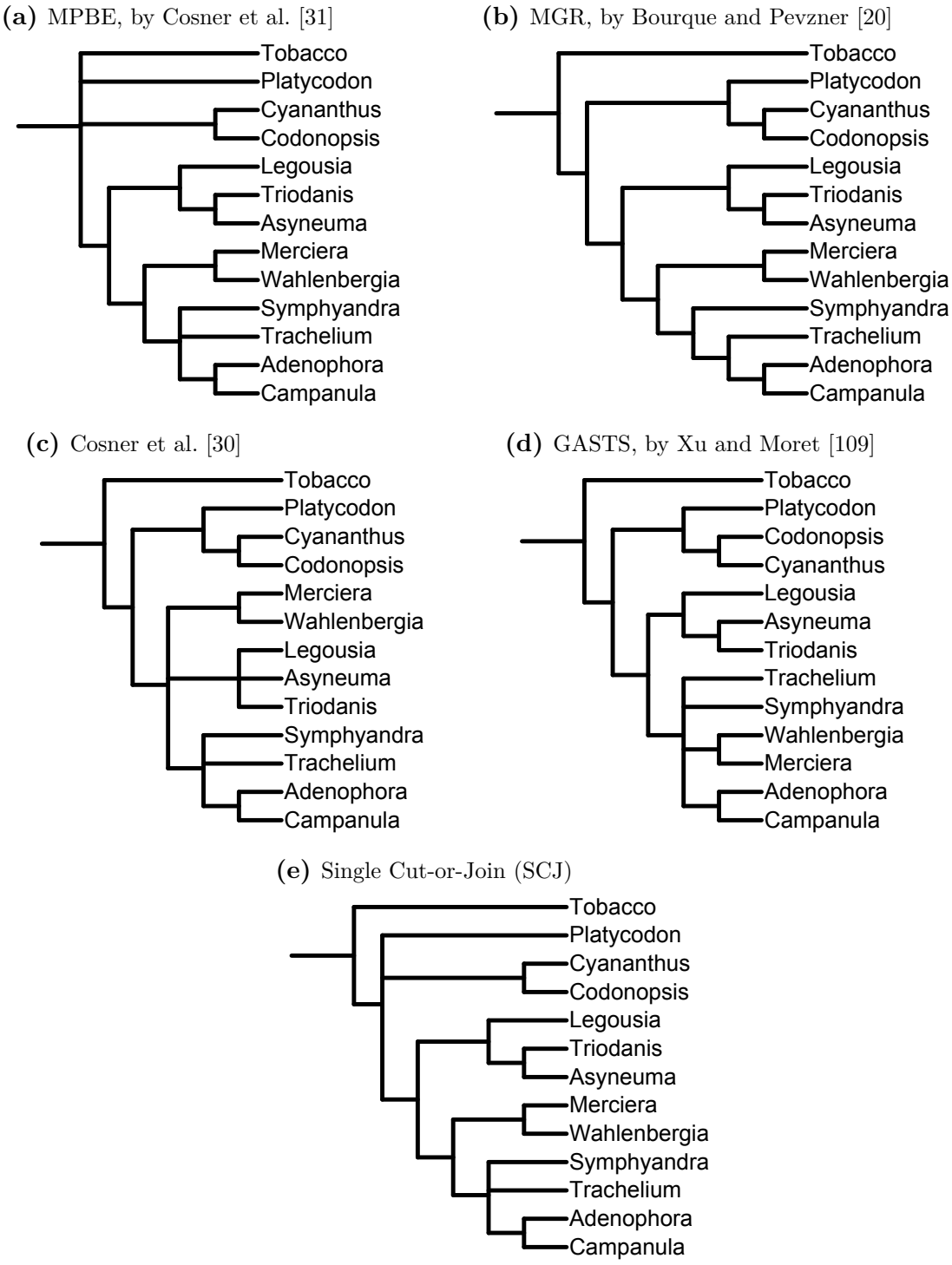


Figure 3.6: Campanulaceae topologies obtained using different methods.

Small Parsimony Problem - ancestor reconstruction

In the small parsimony problem, we noticed that the SCJ is very conservative in choosing which adjacencies are included in the ancestral genomes. Therefore, the number of false-positive adjacencies (adjacencies that are present in the reconstructed ancestral but not in the simulated one) is very low, less than 3% on average. On the other hand, the proportion of correctly reconstructed adjacencies is very high when the rearrangement rate is small, about 95% of reconstruction with 5% rearrangement rate. Increasing the rearrangement rate lowers this figure to about 70% reconstruction when the rearrangement rate is 25%.

The proportion of correctly reconstructed adjacencies also depends on the position of the ancestral genome in the tree. The farther it is from the leaves, the lower the number of correctly reconstructed adjacencies. On average, with 25% of rearrangement rate, 78% of the adjacencies were reconstructed in the bottom third of the tree, while 62% and 57% of the adjacencies were reconstructed in the middle and top thirds of the tree, respectively.

Chapter 4

Adjacency Algebraic Theory

In this chapter we will present an extension to the algebraic theory of Meidanis and Dias [74] that was proposed by Feijão and Meidanis [41]. Its main advantage is the possibility of representing linear chromosomes, which it was believed to be not possible in the original algebraic theory.

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

4.1 Adjacency Algebraic Theory

The formulation of the adjacency algebraic theory is based on the set representation of a genome [13, 40], that was explained in Section 2.2. 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, with $-a = a_h$ 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, with telomeres possibly omitted when the gene set is given, and where each extremity appears at most once. For instance, the genome in Fig. 4.1 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 represented by permutations, just

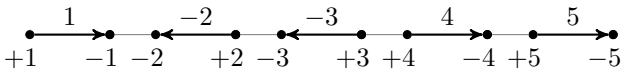


Figure 4.1: A genome composed of just a linear chromosome, with gene extremities labeled according to the adjacency set genome representation, also used in our adjacency algebraic theory.

as in the chromosomal algebraic theory, but a *genome* is a product of 2-cycles, where each 2-cycle corresponds to an adjacency. Therefore, the genome in Fig. 4.1 is represented as $\pi_{\text{adj}} = (-1\ -2)(+2\ -3)(+3\ +4)(-4\ +5)$. 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 the adjacency representation is that there is a direct relationship with the chromosomal representation. Let us recall that Γ is the permutation that changes the sign of each gene, that is,

$$\Gamma = (-1\ +1)(-2\ +2)\cdots(-n\ +n).$$

Then, if π_{chr} is a permutation representing a circular genome in the chromosomal algebraic formulation, $\pi_{\text{adj}} = \pi_{\text{chr}}\Gamma$ represents the same genome in the adjacency algebraic theory. Since $\Gamma = \Gamma^{-1}$, multiplying again by Γ returns to the original formulation, that is, $\pi_{\text{chr}} = \pi_{\text{adj}}\Gamma$. Therefore, multiplying by Γ always switches the representation of a genome between adjacency and chromosomal representations.

For instance, for the genome in Fig. 4.2, we have

$$\pi_{\text{chr}} = (+1\ +2\ +3\ +4\ +5\ +6)(-6\ -5\ -4\ -3\ -2\ -1)$$

and

$$\pi_{\text{adj}} = \pi_{\text{chr}}\Gamma = (-1\ +2)(-2\ +3)(-3\ +4)(-4\ +5)(-5\ +6)(-6\ +1).$$

Another property is that the rearrangement events are represented by the *same permutations* in both formulations. If π_{chr} and σ_{chr} are genomes in the chromosomal formulation such that $\rho\pi_{\text{chr}} = \sigma_{\text{chr}}$, then by multiplying by Γ on the right we easily get $\rho\pi_{\text{adj}} = \sigma_{\text{adj}}$, where π_{adj} and σ_{adj} represent the same genomes in the adjacency formulation. Also, we have that

$$\sigma_{\text{adj}}\pi_{\text{adj}}^{-1} = \sigma_{\text{chr}}\Gamma(\pi_{\text{chr}}\Gamma)^{-1} = \sigma_{\text{chr}}\Gamma\Gamma^{-1}\pi_{\text{chr}}^{-1} = \sigma_{\text{chr}}\pi_{\text{chr}}^{-1}.$$

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.

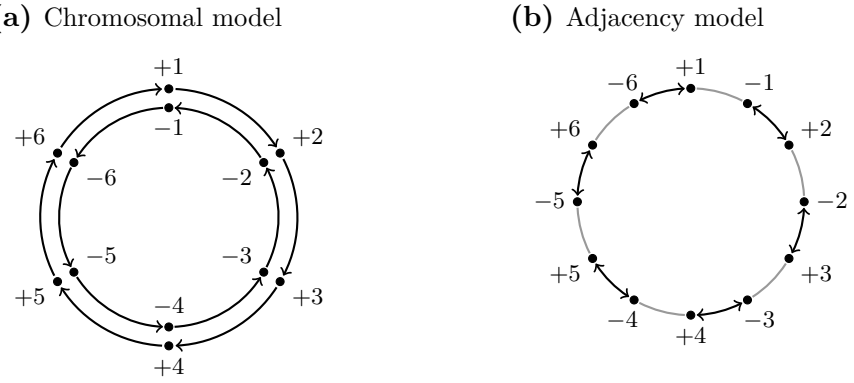


Figure 4.2: The same genome modeled using both chromosomal and adjacency formulations. In (a), the genome is modeled by the permutation $\pi_{\text{chr}} = (+1\ +2\ +3\ +4\ +5\ +6)(-6\ -5\ -4\ -3\ -2\ -1)$, where each cycle corresponds to a chromosome strand. In (b), the permutation is $\pi_{\text{adj}} = \pi_{\text{chr}}\Gamma = (-1\ +2)(-2\ +3)(-3\ +4)(-4\ +5)(-5\ +6)(-6\ +1)$, each cycle corresponding to an adjacency between two genes.

4.2 Sorting by Algebraic Operations

In this section we are interested in solving the sorting problem under the adjacency algebraic theory. Let us recall some basic definitions that were introduced in Section 2.2.

A *rearrangement operation* ρ applicable to a genome π is defined as a permutation ρ for which $\pi' = \rho\pi$ is a valid genome. The *weight* of an operation ρ is defined as $\|\rho\|/2$. With this definition, reversals, translocations, circular fusions and circular fissions are modeled with permutations formed with two 2-cycles, and have therefore weight 1, whereas transpositions are modeled with two 3-cycles, and block interchanges with four 2-cycles, both with a resulting weight of 2 [77]. This weight definition agrees with usual weights, for instance, the Double Cut-and-Join model [110]. We will use this weight definition in this chapter, but it is important to notice that it is possible to change the weight of rearrangement operations, for instance, by changing the norm used in its definition, potentially modifying the sorting problem and its resulting complexity. This seems like an interesting area for further investigation.

In the context of the algebraic theory, we can formulate the *algebraic rearrangement problem* as finding permutations $\rho_1, \rho_2, \ldots, \rho_n$ that minimally transform π into σ, that is, $\rho_i \ldots \rho_2\rho_1\pi$ is a valid genome for every i, $\rho_n \ldots \rho_2\rho_1\pi = \sigma$, and $\sum_{i=1}^{n} \|\rho_i\|/2$ is minimum. This minimum value is called the *algebraic distance* between π and σ, and will be 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, as follows.

If $\rho_1, \rho_2, \ldots, \rho_n$ minimally transform π into σ, we have:

$$\frac{\|\sigma\pi^{-1}\|}{2} = \frac{\|\rho_n \ldots \rho_2\rho_1\|}{2} \leq \frac{\sum_{i=1}^{n} \|\rho_i\|}{2} = d(\pi, \sigma). \tag{4.1}$$

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.

The classical approach restricts the available operations to small weight permutations, such as just reversals, translocations, fusions, fissions, etc. We will show in Section 4.2.2 that one can achieve the same distance using only operations of weight 1 or less.

A permutation ρ is called a *sorting operation* from π to σ if $\rho\pi$ is a valid genome and

$$d(\rho\pi, \sigma) = d(\pi, \sigma) - \|\rho\|/2, \tag{4.2}$$

that is, applying ρ in π decreases the distance between π and σ by the weight of operation ρ, that we already defined as being half the norm. But we see that, if ρ is such a sorting operation,

$$\begin{aligned} \|(\sigma\pi^{-1})\rho^{-1}\| &= \|\sigma(\rho\pi)^{-1}\| \\ &= 2d(\rho\pi, \sigma) \\ &= 2d(\pi, \sigma) - \|\rho\| \\ &= \|\sigma\pi^{-1}\| - \|\rho\|. \end{aligned} \tag{4.3}$$

Therefore, equation (4.3) shows that if ρ decreases the distance by its weight, then ρ divides $\sigma\pi^{-1}$. In other words, a sorting operation from π to σ is an applicable operation on π that divides $\sigma\pi^{-1}$.

We will now give some results on applicable operations and divisibility in the following lemmas. Let us briefly recap some important algebraic properties that will be used in the proofs of these lemmas. For two permutations α and β, we have that $(\alpha\beta)^{-1} = \beta^{-1}\alpha^{-1}$. The *conjugation* of β by α, denoted by $\alpha \cdot \beta$, is the permutation $\alpha\beta\alpha^{-1}$. For any permutations α and β we have the following known results on the norm: $\|\alpha^{-1}\| = \|\alpha\|$, $\|\beta \cdot \alpha\| = \|\alpha\|$, and $\|\alpha\beta\| = \|\beta\alpha\|$.

Lemma 4.1 (Applicable operations)**.** *Given a genome π, a permutation ρ is applicable to π, that is, $\rho\pi$ is a valid genome, if and only if $\pi \cdot \rho = \rho^{-1}$.*

Proof. A permutation π 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 ρ. 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 4.1. *Any permutation ρ in the format $\rho = \mu(\pi \cdot \mu^{-1})$ is applicable to π.*

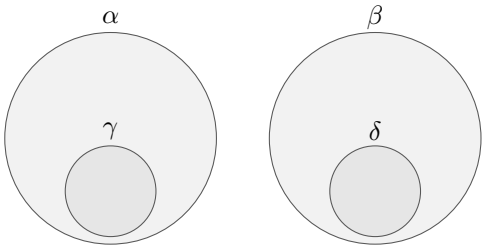


Figure 4.3: Permutations represented as sets, as an intuitive way of thinking about division and constructiveness. Disjoint sets represent constructive permutations, and subsets represent divisors. In this example, $\gamma|\alpha$, $\delta|\beta$, and α and β are constructive.

Proof. If $\rho = \mu(\pi \cdot \mu^{-1})$, then

$$\pi \cdot \rho = \pi \cdot \left[\mu(\pi \cdot \mu^{-1})\right] = (\pi \cdot \mu)\mu^{-1} = \left[\mu(\pi \cdot \mu^{-1})\right]^{-1} = \rho^{-1}$$

and since $\pi \cdot \rho = \rho^{-1}$, by Lemma 4.1 ρ is applicable to π. □

This corollary is important because permutations of 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.

For the next results, we will need a new definition: two permutations α and β are said to be *constructive* if $\|\alpha\beta\| = \|\alpha\| + \|\beta\|$.

An intuitive way of thinking about the some of the concepts in algebraic theory is to relate them to set theory concepts. For instance, permutation product can be related to set union. Therefore, when $\|\alpha\beta\| = \|\alpha\| + \|\beta\|$, meaning that permutations α and β are constructive, we can think of a similar set-theoretic equation $|A \cup B| = |A| + |B|$, implying that sets A and B are disjoint. Permutation divisibility, on the other hand, can be seen as set difference. Then, $\|\alpha\beta^{-1}\| = \|\alpha\| - \|\beta\|$, meaning that $\beta|\alpha$, in set theory would roughly correspond to a similar relation $|A \setminus B| = |A| - |B|$, implying that B is a subset of A. Therefore, constructive permutations can be seen as disjoint sets, and a permutation dividing another permutation can be seen as a set contained in another set.

This intuition will be useful for understanding some permutation properties. For instance, given permutations α, β, γ, and δ such that $\gamma|\alpha$, $\delta|\beta$, and α and β are constructive, we can represent these relationships as sets, shown in Fig. 4.3. In this particular case, it is intuitive to see that γ and δ are also constructive, and $\gamma\delta|\alpha\beta$, as we will prove formally in Lemmas 4.2 and 4.3.

Lemma 4.2. *Given permutations α, β, and γ, if α and β are constructive and $\gamma|\alpha$, then β and γ are constructive.*

Proof. From the norm properties we know that $\|\beta\gamma\| \leq \|\beta\| + \|\gamma\|$, and

$$\begin{aligned} \|\alpha\| + \|\beta\| &= \|\beta\alpha\| = \|\beta\gamma\gamma^{-1}\alpha\| \leq \|\beta\gamma\| + \|\gamma^{-1}\alpha\| = \\ &= \|\beta\gamma\| + \|\alpha\| - \|\gamma\| \end{aligned}$$

which means that $\|\beta\gamma\| \geq \|\beta\| + \|\gamma\|$. Therefore $\|\beta\gamma\| = \|\beta\| + \|\gamma\|$, so β and γ are constructive.

□

Lemma 4.3. *Given permutations α, β, γ, and δ, if $\gamma|\alpha$, $\delta|\beta$, and α and β are constructive, then $\gamma\delta|\alpha\beta$.*

Proof. Since $\delta|\beta$, and α and β are constructive, by Lemma 4.2 we have that δ and α are constructive. Similarly, since $\gamma|\alpha$, and δ and α are constructive, γ and δ are also constructive.

Now, we will show that $\alpha\delta|\alpha\beta$:

$$\|\alpha\beta(\alpha\delta)^{-1}\| = \|\alpha\beta\delta^{-1}\alpha^{-1}\| = \|\beta\delta^{-1}\| = \|\beta\| - \|\delta\|$$

and since $\|\alpha\beta\| = \|\alpha\| + \|\beta\|$ and $\|\delta\alpha\| = \|\delta\| + \|\alpha\|$ we get

$$\|\beta\| - \|\delta\| = (\|\alpha\beta\| - \|\alpha\|) - (\|\delta\alpha\| - \|\alpha\|) = \|\alpha\beta\| - \|\delta\alpha\|,$$

and indeed $\alpha\delta|\alpha\beta$. Also, we will show that $\gamma\delta|\alpha\delta$:

$$\|\alpha\delta(\gamma\delta)^{-1})\| = \|\alpha\delta\delta^{-1}\gamma^{-1}\| = \|\alpha\gamma^{-1}\| = \|\alpha\| - \|\gamma\|.$$

Since $\|\alpha\delta\| = \|\delta\| + \|\alpha\|$ and $\|\gamma\delta\| = \|\gamma\| + \|\delta\|$ we have

$$\|\alpha\| - \|\gamma\| = (\|\alpha\delta\| - \|\delta\|) - (\|\gamma\delta\| - \|\delta\|) = \|\alpha\delta\| - \|\gamma\delta\|,$$

and we have that $\gamma\delta|\alpha\delta$. Since $\alpha\delta|\alpha\beta$, with the transitivity of the division, we have that $\gamma\delta|\alpha\beta$. □

Next we will show an important result about permutation divisibility, stating that applying the reverse conjugation maintains divisibility:

Lemma 4.4. *Given a genome π and a permutation α, for any permutation μ 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:

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

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

Using these previous lemmas, we will prove a theorem that will be the base for finding sorting operations from $\sigma\pi^{-1}$.

Theorem 4.1. *Given a genome π and a permutation $\tau = \alpha(\pi \cdot \alpha^{-1})$ with $\|\tau\| = \|\alpha\| + \|\pi \cdot \alpha^{-1}\|$, then for any permutation μ for which $\mu|\alpha$ we have that the permutation $\rho = \mu(\pi \cdot \mu^{-1})$ divides τ, and ρ is an applicable operation on π.*

Proof. Since $\mu|\alpha$, $\pi \cdot \mu^{-1}|\pi \cdot \alpha^{-1}$ (from Lemma 4.4), and $\|\alpha(\pi \cdot \alpha^{-1})\| = \|\alpha\| + \|\pi \cdot \alpha^{-1}\|$, from Lemma 4.3 we have that $\mu(\pi \cdot \mu^{-1})|\alpha(\pi \cdot \alpha^{-1})$, therefore, $\rho = \mu(\pi \cdot \mu^{-1})$ divides τ. Also, from Corollary 4.1 we know that $\rho = \mu(\pi \cdot \mu^{-1})$ is an applicable operation on π. □

It should be noted that finding all cycles $\mu = (e_1 \ \ldots \ e_k)$ that divide a cycle α is easy: it suffices to choose $e_1, \ldots, e_k$ as a subset of elements of α in the cyclic order that they appear in α. For instance, if $\alpha = (1\ 2\ 3\ 4\ 5)$, then $\mu = (1\ 3\ 4)$ divides α, but $\mu' = (1\ 4\ 3)$ does not. A formal proof of this fact can be found in a paper by Huang and Lu [53, Lemma 2.7].

A last lemma exposes the relationship between these $\mu(\pi \cdot \mu^{-1})$ operations and the k-break operation, introduced by Alekseyev and Pevzner [3]. A *k-break* is an operation that cuts k adjacencies in π and then joins k new ones with the same extremities.

Lemma 4.5. *A permutation $\rho = \mu(\pi \cdot \mu^{-1})$ where μ and $\pi \cdot \mu^{-1}$ are disjoint cycles and $||\mu|| = k - 1$, is a k-break operation on π.*

Proof. Let $\mu = (e_1\ e_2\ \cdots\ e_k)$, so $\|\mu\| = k - 1$, and let $\rho = \mu(\pi \cdot \mu^{-1})$, that is, $\rho = (e_1\ e_2\ \cdots\ e_k)(\pi e_k\ \cdots\ \pi e_2\ \pi e_1)$. If μ and $\pi \cdot \mu^{-1}$ are disjoint, no e_i is fixed in π, so π must have the adjacencies $(e_1\ \pi e_1)\ldots(e_k\ \pi e_k)$ and can be written as $\pi = (e_1\ \pi e_k)\cdots(e_k\ \pi e_{k-1})\pi'$, where π' and ρ are disjoint. Applying ρ in π we have

$$\begin{aligned}
\rho\pi &= \mu(\pi \cdot \mu^{-1})\pi \\
&= (e_1\ e_2\ \cdots\ e_k)(\pi e_k\ \cdots\ \pi e_2\ \pi e_1)\pi \\
&= (e_1\ e_2\ \cdots\ e_k)(\pi e_k\ \cdots\ \pi e_2\ \pi e_1)(e_1\ \pi e_1)\cdots(e_k\ \pi e_k)\pi' \\
&= (e_1\ \pi e_k)(e_2\ \pi e_1)(e_3\ \pi e_2)\cdots(e_k\ \pi e_{k-1})\pi'
\end{aligned}$$

and $\rho\pi$ will have the adjacencies $(e_1\ \pi e_k)(e_2\ \pi e_1)\ldots(e_k\ \pi e_{k-1})$. Therefore, k adjacencies of π were removed and changed to k new ones, and ρ is a k-break. □

In the next section we will use Theorem 4.1 to find sorting operations, that is, applicable operations that divide $\sigma\pi^{-1}$.

4.2.1 Characterizing $\sigma\pi^{-1}$ and Finding Sorting Operations

In this section we will use the *Adjacency Graph* between two genomes π and σ, defined by Bergeron, Mixtacki, and Stoye [13]. In this graph, denoted as $AG(\pi, \sigma)$, the vertices are the adjacencies and telomeres of π and σ, 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 cycles in $\sigma\pi^{-1}$ and then determine sorting operations on these permutations.

i) 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 π and alternating vertices of π and σ:

$$\underbrace{(e_1, e_2)}_{\pi_1}, \underbrace{(e_2, e_3)}_{\sigma_1}, \ldots, \underbrace{(e_{2k-1}, e_{2k})}_{\pi_k}, \underbrace{(e_{2k}, e_{2k+1})}_{\sigma_k}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi_{n/2}}, \underbrace{(e_n, e_1)}_{\sigma_{n/2}}$$

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

In the product $\sigma\pi^{-1}$, the adjacencies in the $AG(\pi, \sigma)$ cycle will be 2-cycles and disjoint from the rest of the adjacencies in π and σ. Therefore, they 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 τ of $\sigma\pi^{-1}$ to the $AG(\pi, \sigma)$ cycle (recalling that $\pi_i^{-1} = \pi_i$ for each i):

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

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 π-conjugation of the other. Figure 4.4(a) shows an example with $n = 8$.

To extract sorting operations in this case, we see that τ satisfies Theorem 4.1, therefore any μ that divides α generates a sorting operation $\rho = \mu(\pi \cdot \mu^{-1})$, with weight $w = \|\rho\|/2 = \|\mu\|$.

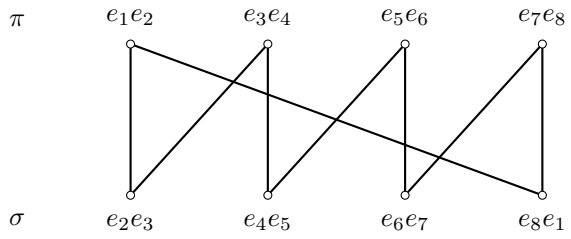
An example showing the relationship between $AG(\pi, \sigma)$ and $\sigma\pi^{-1}$ and how to extract sorting operations from $\sigma\pi^{-1}$ is given in Figure 4.5.

ii) Odd Paths in $AG(\pi, \sigma)$

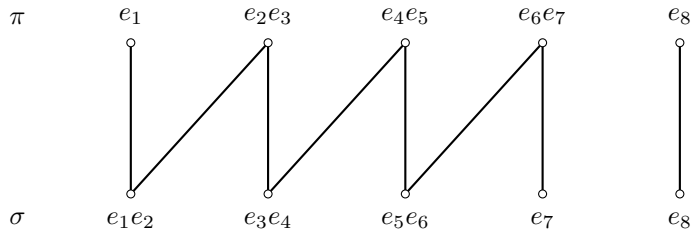
An odd path of size n in $AG(\pi, \sigma)$, that is, a path with n edges where n is odd, starting with a telomere e_1 in π and ending at a telomere e_n in σ, has vertices

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

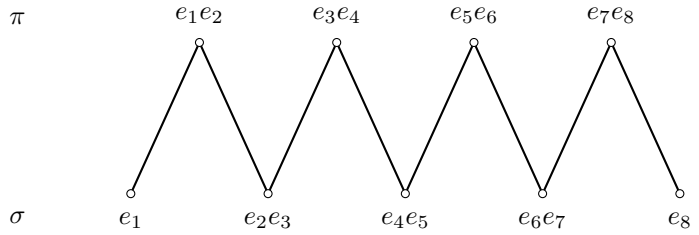
(a) Cycle of length $n = 8$ in $AG(\pi, \sigma)$. In this case, $\pi = (e_1\ e_2)(e_3\ e_4)(e_5\ e_6)(e_7\ e_8)$, $\sigma = (e_2\ e_3)(e_4\ e_5)(e_6\ e_7)(e_8\ e_1)$, and $\sigma\pi^{-1} = (e_8\ e_6\ e_4\ e_2)(e_1\ e_3\ e_5\ e_7)$. Notice that $(e_8\ e_6\ e_4\ e_2) = \pi \cdot (e_1\ e_3\ e_5\ e_7)^{-1}$.



(b) Odd path of length $n = 7$ in $AG(\pi, \sigma)$. In this case, $\pi = (e_1)(e_2\ e_3)(e_4\ e_5)(e_6\ e_7)(e_8)$, $\sigma = (e_1\ e_2)(e_3\ e_4)(e_5\ e_6)(e_7)(e_8)$, and $\sigma\pi^{-1} = (e_7\ e_5\ e_3\ e_1\ e_2\ e_4\ e_6)$, which can be rewritten as $\sigma\pi^{-1} = (e_7\ e_5\ e_3\ e_1)(e_1\ e_2\ e_4\ e_6)$, with $(e_7\ e_5\ e_3\ e_1) = \pi \cdot (e_1\ e_2\ e_4\ e_6)^{-1}$.



(c) Even path of length $n = 8$, with both telomeres in σ. In this case, $\pi = (e_1\ e_2)(e_3\ e_4)(e_5\ e_6)(e_7\ e_8)$, $\sigma = (e_1)(e_2\ e_3)(e_4\ e_5)(e_6\ e_7)(e_8)$, and $\sigma\pi^{-1} = (e_8\ e_6\ e_4\ e_2\ e_1\ e_3\ e_5\ e_7)$, which can be rewritten as $\sigma\pi^{-1} = (e_1\ e_8)(e_8\ e_6\ e_4\ e_2)(e_1\ e_3\ e_5\ e_7)$, with $(e_8\ e_6\ e_4\ e_2) = \pi \cdot (e_1\ e_3\ e_5\ e_7)^{-1}$.



(d) Even path of length $n = 8$, with both telomeres in π. In this case, $\pi = (e_1)(e_2\ e_3)(e_4\ e_5)(e_6\ e_7)(e_8)$, $\sigma = (e_1\ e_2)(e_3\ e_4)(e_5\ e_6)(e_7\ e_8)$, and $\sigma\pi^{-1} = (e_7\ e_5\ e_3\ e_1\ e_2\ e_4\ e_6\ e_8)$, which can be rewritten as $\sigma\pi^{-1} = (e_7\ e_8)(e_7\ e_5\ e_3\ e_1)(e_1\ e_2\ e_4\ e_6)$, with $(e_7\ e_5\ e_3\ e_1) = \pi \cdot (e_1\ e_2\ e_4\ e_6)^{-1}$.

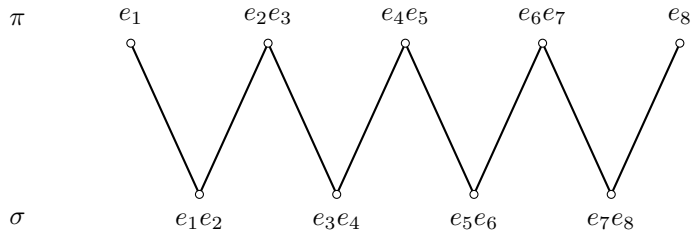
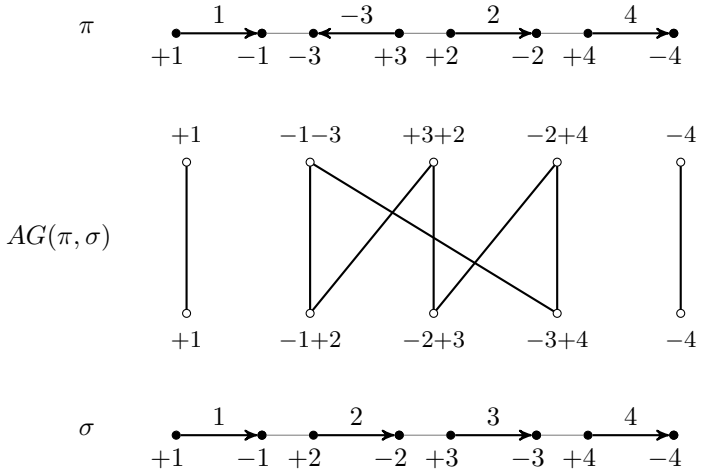
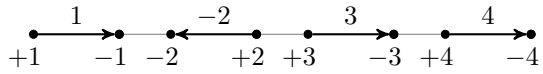


Figure 4.4: Cycles and paths in $AG(\pi, \sigma)$ and their relationship with $\sigma\pi^{-1}$

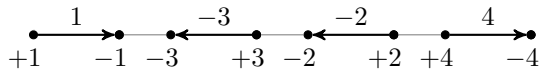
(a) Genomes $\pi = (-1\ -3)(+2\ +3)(-2\ +4)$ and $\sigma = (-1\ +2)(-2\ +3)(-3\ +4)$ and the adjacency graph $AG(\pi, \sigma)$.



(b) Genome $\sigma_1 = \rho_1\pi$, obtained by applying the sorting operation $\rho_1 = (-1\ +4)(-2\ -3)$ to π. The permutation ρ_1 models a reversal on the block containing genes 3 and 2.



(c) Genome $\sigma_2 = \rho_2\pi$, obtained by applying the sorting operation $\rho_2 = (+4\ +3)(+2\ -2)$ to π. The permutation ρ_2 models a reversal on the gene 2.



(d) Genome $\sigma_3 = \rho_3\pi$, obtained by applying the sorting operation $\rho_3 = (-1\ +3)(+2\ -3)$ to π. The permutation ρ_3 models the circular excision of gene 3.

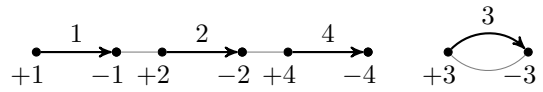


Figure 4.5: Example of obtaining sorting operations from the permutation $\sigma\pi^{-1}$, with $\pi = (-1\ -3)(+2\ +3)(-2\ +4)$ and $\sigma = (-1\ +2)\ (-2\ +3)(-3\ +4)$. In this case, $\sigma\pi^{-1} = (-1\ +4\ +3)(+2\ -2\ -3)$. These two 3-cycles correspond to one cycle of length 6 in the adjacency graph $AG(\pi, \sigma)$ shown in (a). The permutation $\sigma\pi^{-1}$ itself is a sorting operation, but if we want smaller operations, we can notice that the cycle $(-1\ +4\ +3)$ has three possible 2-cycles dividing it: $\mu_1 = (-1\ +4)$, $\mu_2 = (+4\ +3)$, and $\mu_3 = (-1\ +3)$. Each one generates a sorting operation of the form $\rho_i = \mu_i(\pi \cdot \mu_i^{-1})$, for $i = 1, 2, 3$, namely, $\rho_1 = (-1\ +4)(-2\ -3)$, $\rho_2 = (+4\ +3)(+2\ -2)$, and $\rho_3 = (-1\ +3)(+2\ -3)$. The results of applying each of these operations is shown in Figures (b),(c) and (d).

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

$$\begin{aligned}\tau &= (e_1\ e_2)\dots(e_{2k-1}\ e_{2k})\dots(e_n)(e_1)(e_2\ e_3)\dots(e_{2k}\ e_{2k+1})\dots(e_{n-1}\ e_n)\\ \tau &= (e_n\ e_{n-2}\ \dots\ e_3\ e_1\ e_2\ e_4\ \dots\ e_{n-3}\ e_{n-1}).\end{aligned}$$

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 (non disjoint) reversed π-conjugates:

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

where $\alpha = (e_n\ e_{n-2}\ \dots\ e_3\ e_1)$, then τ satisfies Theorem 4.1, and with any μ such that $\mu|\alpha$ we derive a sorting operation $\rho = \mu(\pi\cdot\mu^{-1})$, with weight $w = \|\rho\|/2 = \|\mu\|$.

iii) 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 π or in σ.

iii.1) Both telomeres in σ. If both telomeres are in σ, the vertices are of the form

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

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

$$\begin{aligned}\tau &= (e_1)(e_2\ e_3)\dots(e_{2k}\ e_{2k+1})\dots(e_n)(e_1\ e_2)\dots(e_{2k-1}\ e_{2k})\dots(e_{n-1}\ e_n)\\ \tau &= (e_n\ e_{n-2}\ \dots\ e_4\ e_2\ e_1\ e_3\ \dots\ e_{n-3}\ e_{n_1}),\end{aligned}$$

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

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

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

iii.2) Both telomeres in π. If both telomeres are in π, 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 τ of $\sigma\pi^{-1}$ will be

$$\begin{aligned} \tau &= (e_1\ e_2)\ldots(e_{2k-1}\ e_{2k})\ldots(e_{n-1}\ e_n)(e_1)(e_2\ e_3)\ldots(e_{2k}\ e_{2k+1})\ldots(e_{n-2}\ e_{n_1})(e_n) \\ \tau &= (e_{n-1}\ e_{n-3}\ \ldots\ e_3\ e_1\ e_2\ \ldots\ e_{n-4}\ e_{n-2}\ e_n), \end{aligned}$$

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

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

and again we get to a product of a 2-cycle with a permutation in the format $\alpha(\pi\cdot\alpha^{-1})$, where $\alpha = (e_{n-1}\ e_{n-3}\ \ldots\ e_3\ e_1)$.

In both even path cases, the permutation $\alpha(\pi\cdot\alpha^{-1})$ divides τ, and then any permutation $\rho = \mu(\pi\cdot\mu^{-1})$, where μ divides α is a sorting operation with weight $w = \|\rho\|/2 = \|\mu\|$.

iii.3) Special cases. In both types of even paths there is one special case, specifically when $n = 2$, where permutation α becomes the identity and τ is reduced to a 2-cycle; in both cases, $\tau = (e_1\ e_2)$. In the first case, both telomeres are in σ and τ is a *cut* in π, splitting the adjacency $(e_1\ e_2)$ of π into two telomeres. We have that τ is a sorting operation, since it divides $\sigma\pi^{-1}$. On the second case, both telomeres are in π, and τ is a *join* in π, joining telomeres e_1 and e_2 into an adjacency. Again, τ 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 that there is a direct relationship between the adjacency graph $AG(\pi,\sigma)$ and the permutation $\sigma\pi^{-1}$ and how to derive sorting operations from $\sigma\pi^{-1}$. Sorting operations are usually in the format $\rho = \mu(\pi\cdot\mu^{-1})$, where μ divides a cycle of $\sigma\pi^{-1}$, and these operations are known as k-breaks, where $k = \|\mu\| + 1$. There are also special cases of single 2-cycle operations such as cuts and joins.

4.2.2 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 rearrangement 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 fissions (generalized transpositions are also possible by applying two specific operations of weight 1).

Using the results from Section 4.2.1 we can see that it is always possible to find sorting 2-breaks. In all cases where the sorting operation is of the form $\rho = \mu(\pi \cdot \mu^{-1})$, if μ is a 2-cycle, then the operation weight is $\|\rho\|/2 = \|\mu\| = 1$ and ρ is a 2-break. There are also the special cases of cuts or joins, and in these 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 that 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 [13] are also sorting operations under the algebraic theory, which means that algebraic sorting by 2-breaks can be achieved in linear time.

4.2.3 Comparing the Algebraic Distance with the DCJ Distance

To compare the algebraic and DCJ distances, we will use the graph $AG(\pi, \sigma)$ again. From Section 4.2.1, 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:

$$d_{\text{alg}}(\pi, \sigma) = \sum_{k=1}^{n} (k-1)C_{2k} + \sum_{k=1}^{n} \frac{k-1}{2} P_k, \tag{4.4}$$

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 $4k$ extremities, and each path of size k has $k + 1$ vertices with a total of $2k$ extremities, we have

$$N = \sum_{k=1}^{n} kC_{2k} + \sum_{k=1}^{n} \frac{k}{2} P_k. \tag{4.5}$$

Using (4.4) and (4.5) we have

$$d_{\text{alg}}(\pi,\sigma) = N - \sum_{k=1}^{n} C_{2k} - \sum_{k=1}^{n} \frac{1}{2} P_k = N - (C + \frac{P}{2}), \tag{4.6}$$

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 the DCJ distance [13] is given by

$$d_{\text{DCJ}}(\pi,\sigma) = N - (C + P_{odd}/2), \tag{4.7}$$

we have

$$d_{\text{alg}}(\pi,\sigma) = d_{\text{DCJ}}(\pi,\sigma) - \frac{P_{even}}{2}, \tag{4.8}$$

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 modeled 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 [22]. We can see in the distance equations (4.6) and (4.7) 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 comparison, we also compared algebraic and DCJ distances with a scatter plot 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 scatter plot shown in Figure 4.6.

4.3 Modeling Linear Genomes with the Chromosomal Algebraic Theory

We saw in this chapter that using the adjacency representation, linear chromosomes can be modeled in the algebraic theory. But, in the chromosomal algebraic theory, it was believed that only circular chromosomes could be modeled, due to the cyclical nature of the permutations not allowing to model the extremities of linear chromosomes. But we also saw that there is a direct relationship between genomes modeled using the adjacency theory and the chromosomal theory, that is, a right multiplication by the permutation Γ.

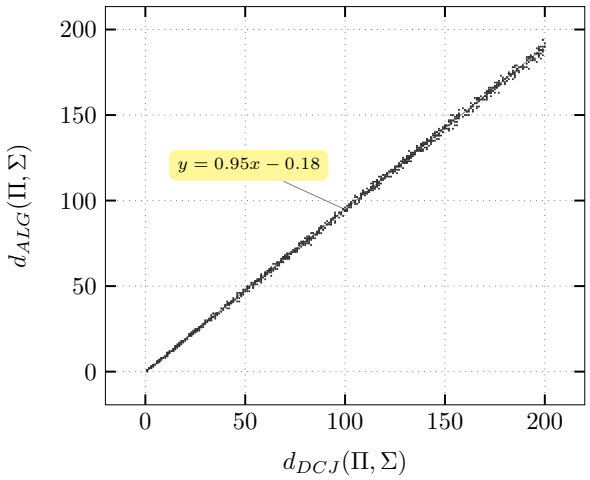


Figure 4.6: Scatterplot comparing algebraic and DCJ distances between randomly evolved genomes. Linear regression on the points resulted in the equation $y = 0.95x - 0.18$.

So the following question arises: what happens if we transform a linear genome from the algebraic representation to the chromosomal representation, multiplying it by Γ?

Take for instance the linear genome $\pi_{\text{adj}} = (-1\ -2)(+2\ -3)(+3\ +4)(-4\ +5)$, represented using the adjacency algebraic theory. To obtain the same genome, represented by π_{chr}, in the chromosomal algebraic theory, we have

$$\pi_{\text{chr}} = \pi_{\text{adj}}\Gamma = (+1\ -2\ -3\ +4\ +5\ -5\ -4\ +3\ +2\ -1).$$

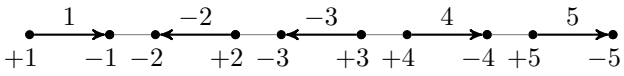
Both representations are shown in Figure 4.7.

Indeed, a linear chromosome can be represented in the chromosomal algebraic theory; instead of each strand being modeled by a different cycle, in a linear chromosome both strands are contained inside the same cycle, where *if an extremity u is a telomere, then u and Γu are consecutive*. In the example of Figure 4.7(b), $+1$ and -5 are telomeres, since -1 and $+1$ are consecutive, as well as $+5$ and -5. More formally, if an extremity u satisfies $u = \pi\Gamma u$, then u is a *telomere*.

Note that the cycle $\pi = (+1\ -2\ -3\ +4\ +5\ -5\ -4\ +3\ +2\ -1)$ satisfies the equation $\Gamma\pi\Gamma = \pi^{-1}$, which is a necessary condition for a permutation to represent a genome, as we saw in Section 2.2.2, page 21. However, in that same section, another condition was given, namely that no strand in π contains both $-i$ and $+i$ for any gene i. This second condition is necessary only for circular chromosomes. Then, new definitions are needed to accommodate linear chromosomes and genomes.

Any cycle π that satisfies $\Gamma\pi\Gamma = \pi^{-1}$ is a *linear chromosome*. If a pair of disjoint cycles π_1 and π_2 satisfy $\Gamma\pi_1\Gamma = \pi_2^{-1}$, then $\pi = \pi_1\pi_2$ is a *circular chromosome*, where π_1 and π_2 represent each strand. Note that $\pi = \pi_1\pi_2$ also satisfies $\Gamma\pi\Gamma = \pi^{-1}$. A *genome* is the product of any number of disjoint chromosomes.

(a) The genome $\pi_{\text{adj}} = (-1\ -2)(+2\ -3)(+3\ +4)(-4\ +5)$, modeled with the adjacency algebraic theory.



(b) The same genome modeled with the chromosomal algebraic theory, by the permutation $\pi_{\text{chr}} = (+1\ -2\ -3\ +4\ +5\ -5\ -4\ +3\ +2\ -1)$

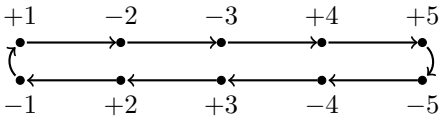


Figure 4.7: The same linear genome represented by the adjacency and chromosomal algebraic theories.

Therefore, *linear genomes could always be modeled in the chromosomal algebraic theory*, but until now no one had realized that dropping the condition that no cycle in π contains both $-i$ and $+i$ for any gene i was all that was necessary.

In the next section, we will show how all the usual rearrangement operations can be modeled using the classic algebraic theory, and how all of the operations share the same general formula.

4.3.1 Classic Rearrangement Operations

In this section we will model several classic rearrangement operations using permutations with small norm. In this whole section we will use the *chromosomal* representation, that is, $\pi = \pi_{\text{chr}}$.

Some of these permutations modeling rearrangement operations were already proposed in earlier studies, since they are applicable in circular genomes [70, 77]. Along with these known permutations, we will introduce new ones that model operations on linear chromosomes, namely translocations, operations involving telomeres, and single cut or join operations, highlighting the fact that all these permutation have a common format.

For instance, all 2-break operations (signed reversals, translocations, fissions and fusions) on a genome π are modeled by the same permutation composed by two 2-cycles, $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, with $||\rho|| = 2$, by choosing appropriate elements u and v. Other classic operations such as transpositions and block-interchanges can also be modeled by permutations composed by two 3-cycles and four 2-cycles, respectively, as we will see.

i) Reversals

When u and v belong to different strands in a chromosome, the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ is a signed reversal. For instance, we can take the genome

$$\pi = (\cdots\ \pi^{-1}u\ \underbrace{\boxed{u\ \cdots\ \Gamma v}}_{A_1}\ \pi\Gamma v\ \cdots)(\cdots\ \pi^{-1}v\ \underbrace{\boxed{v\ \cdots\ \Gamma u}}_{A_2}\ \pi\Gamma u\ \cdots)$$

where u and v are in different strands (notice that u and v might be in the same cycle in a linear chromosome, but in different cycles in a circular chromosome, as in this example). Applying the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ generates the genome

$$\sigma = \rho\pi = (\cdots\ \pi^{-1}u\ \underbrace{\boxed{v\ \cdots\ \Gamma u}}_{A_2}\ \pi\Gamma v\ \cdots)(\cdots\ \pi^{-1}v\ \underbrace{\boxed{u\ \cdots\ \Gamma v}}_{A_1}\ \pi\Gamma u\ \cdots)$$

where blocks A_1 and A_2 were exchanged between strands, effectively applying a signed reversal in the segment from gene u to Γv, as depicted in Figure 4.8.

An example might help. Consider the genome

$$\pi = (+1\ +2\ +3\ +4\ -4\ -3\ -2\ -1),$$

shown in Figure 4.9. To apply a reversal in the block of genes $(+2\ +3)$, we can choose $u = +2$ and $v = -3$ (the reverse complement of 3) in the template 2-break permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, originating the operation $\rho = (+2\ -3)(+4\ -1)$. The resulting genome is then

$$\sigma = \rho\pi = (+1\ -3\ -2\ +4\ -4\ +2\ +3\ -1)$$

after the application of the reversal, with the chosen block reversed as expected.

ii) Translocations

When u and v belong to two different linear chromosomes, $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ is a translocation. For instance, we can take the genome

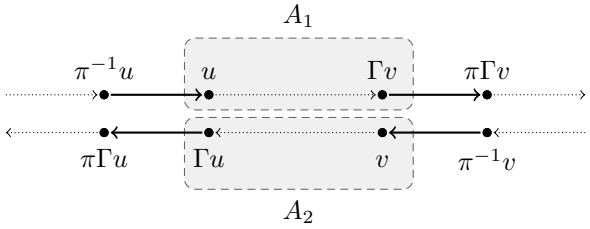
$$\pi = (\cdots\ \pi^{-1}u\ \underbrace{\boxed{u\ \cdots\ \Gamma u}}_{A}\ \pi\Gamma u\ \cdots)(\cdots\ \pi^{-1}v\ \underbrace{\boxed{v\ \cdots\ \Gamma v}}_{B}\ \pi\Gamma v\ \cdots)$$

where u and v are in two different linear chromosomes. We know that each of these cycles represents a linear chromosome because u and Γu are in the same cycle, as are v and Γv. Applying the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ generates the genome

$$\sigma = \rho\pi = (\cdots\ \pi^{-1}u\ \underbrace{\boxed{v\ \cdots\ \Gamma v}}_{B}\ \pi\Gamma u\ \cdots)(\cdots\ \pi^{-1}v\ \underbrace{\boxed{u\ \cdots\ \Gamma u}}_{A}\ \pi\Gamma v\ \cdots)$$

where the blocks A and B, located at the extremities of the chromosomes, were exchanged. This operation is depicted in Figure 4.10.

(a) Genome π with elements u and v in different strands.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, corresponding to a reversal.

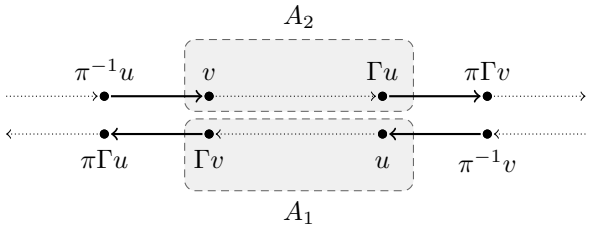


Figure 4.8: Example of the application of a reversal, modeled by the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when u and v belong to different strands of a chromosome. Blocks A_1 and A_2, which are reverse complementary, exchange places, resulting in a reversal.

iii) Fissions and Fusions

If u and v belong to the same strand of a circular chromosome, $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ is a fission, breaking the chromosome in two.

For instance, if we take the genome

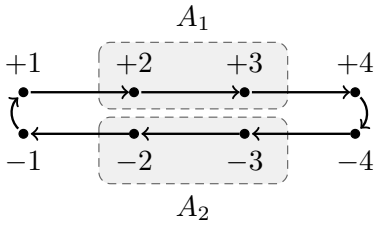
$$\pi = (\underbrace{\cdots\ \pi^{-1}u}_{B_1}\ \underbrace{u\ \cdots\ \pi^{-1}v}_{A_1}\ \underbrace{v\ \cdots}_{B_1})(\underbrace{\cdots\ \Gamma v}_{B_2}\ \underbrace{\pi\Gamma v\ \cdots\ \Gamma u}_{A_2}\ \underbrace{\pi\Gamma u\ \cdots}_{B_2})$$

in which u and v are in the same circular chromosome, where each strand is modeled by one cycle. Blocks A_1 and A_2 are reverse complementary, as are B_1 and B_2. Applying the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ generates the genome

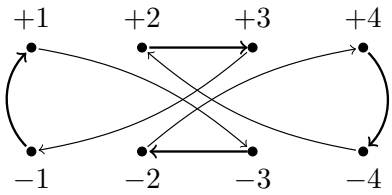
$$\sigma = \rho\pi = (\underbrace{\cdots\ \pi^{-1}v\ \ u\ \cdots}_{A_1})(\underbrace{\cdots\ \Gamma u\ \ \pi\Gamma v\ \cdots}_{A_2})(\underbrace{\cdots\ \pi^{-1}u\ \ v\ \cdots}_{B_1})(\underbrace{\cdots\ \Gamma v\ \ \pi\Gamma u\ \cdots}_{B_2})$$

where blocks A_1 and B_1 were separated, each into its own cycle, and the same happened to A_2 and B_2. This has the effect of dividing the circular chromosome in two, as shown in Figure 4.11, resulting in a fission. Applying the same operation again, this time in σ, results in the inverse operation, the fusion of two circular chromosomes into one. This is because whenever u and v belong to two different circular chromosomes, $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ models a fusion, merging two chromosomes into one.

(a) Linear genome $\pi = (+1\ +2\ +3\ +4\ -4\ -3\ -2\ -1)$.



(b) Applying the operation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, with $u = 2$ and $v = -3$, changes the connections between genes, modeling a rearrangement operation.



(c) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, with $u = 2$ and $v = -3$, corresponding to a reversal.

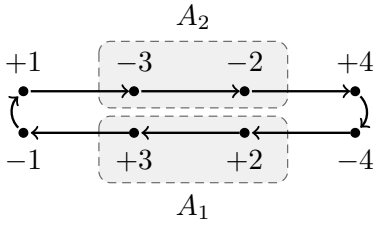
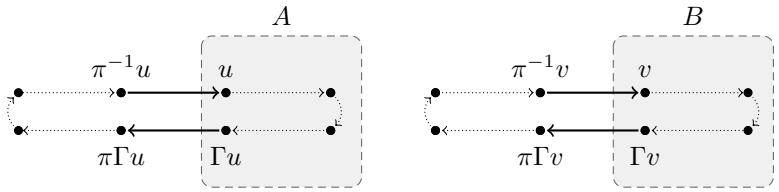


Figure 4.9: In genome $\pi = (+1\ +2\ +3\ +4\ -4\ -3\ -2\ -1)$, shown in (a), to apply a reversal in the block $(+2\ +3)$, we choose $u = +2$ and $v = -3$ (the reverse complement of 3) in the template 2-break $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, resulting in $\rho = (+2\ -3)(+4\ -1)$. Applying this operation results in the genome $\sigma = \rho\pi = (+1\ -3\ -2\ +4\ -4\ +2\ +3\ -1)$, shown in (b) and also (c).

(a) Genome where elements u and v belong to different linear chromosomes. Blocks A and B represent the tail of each chromosome starting with elements u and v, respectively.



(b) Applying the operation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ results in a translocation, exchanging blocks A and B between the chromosomes.

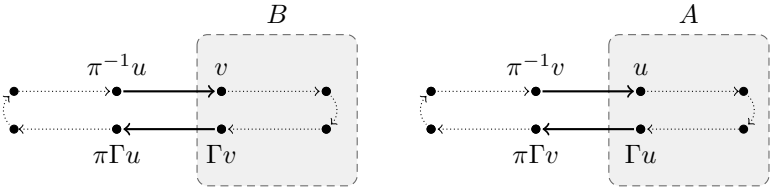


Figure 4.10: Example of the application of a translocation, modeled by the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when u and v belong to different linear chromosomes, exchanging the blocks A and B between chromosomes.

If u and v are in the same *linear* chromosome, a particular case of fission called *circular excision* happens. In this operation, a block is removed from a linear chromosome to form a new circular chromosome. For instance, the genome

$$\pi = (\underbrace{\cdots\ \pi^{-1}u}_{A_1}\ \underbrace{u\ \cdots\ \pi^{-1}v}_{B_1}\ \underbrace{v\ \cdots\ \Gamma v}_{C}\ \underbrace{\pi\Gamma v\ \cdots\ \Gamma u}_{B_2}\ \underbrace{\pi\Gamma u\ \cdots}_{A_2})$$

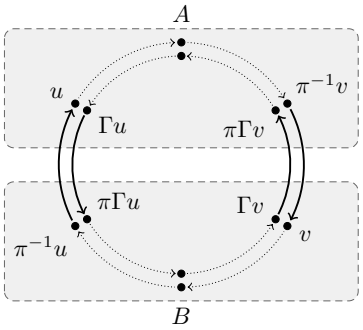
has u and v are in the same linear chromosome, therefore in the same cycle, as are Γu and Γv. Applying the $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ results in the genome

$$\sigma = \rho\pi = (\underbrace{\cdots\ \pi^{-1}v\ \ u\ \cdots}_{B_1})(\underbrace{\cdots\ \Gamma u\ \ \pi\Gamma v\ \cdots}_{B_2})(\underbrace{\cdots\ \pi^{-1}u}_{A_1}\ \underbrace{v\ \cdots\ \Gamma v}_{C}\ \underbrace{\pi\Gamma u\ \cdots}_{A_2})$$

that is composed by the circular chromosome $(\cdots\ \pi^{-1}v\ \ u\ \cdots)(\cdots\ \Gamma u\ \ \pi\Gamma v\ \cdots)$, formed by blocks B_1 and B_2, and the linear chromosome $(\cdots\ \pi^{-1}u\ \ v\ \cdots\ \Gamma v\ \ \pi\Gamma u\ \cdots)$, with blocks A_1, C, and A_2, as seen in Figure 4.12.

Circular excisions were discussed for the first time in the context of the DCJ model [110], where the block-interchange operation is modeled by two consecutive DCJ operations. First, a circular excision is applied, creating what is called a *circular intermediate*, followed by the reabsorption of the circular intermediate into the original linear chromosome, with the net effect of a block-interchange. The reabsorption operation is a particular case of the fusion operation, and can also be modeled with the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when one of u, v belongs to a linear chromosome and the other to a circular chromosome.

(a) Genome π with elements u and v in the same strand of a circular chromosome.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, corresponding to a fission.

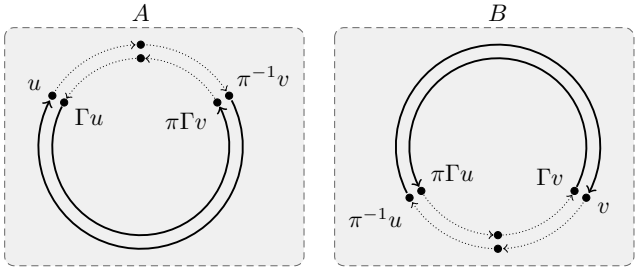


Figure 4.11: Example of the application of a fission operation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when u and v belong to the same strand in a circular chromosome. The same operation also models the inverse fusion operation, if applied to σ, transforming it back to π.

As we will see in the subsection on block-interchanges, a similar concept occurs in the algebraic theory, since a block-interchange is modeled by the product of two permutations of the form $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$.

iv) Transpositions

Transpositions on a genome π are modeled by the permutation $\rho = (u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$, if u, v, and w belong to the same chromosome and appear in this order (or more formally, $(u\ v\ w)$ divides π). For instance, we can take the genome

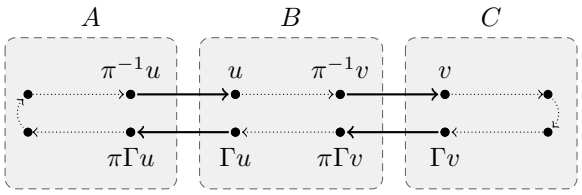
$$\pi = (\cdots\ \pi^{-1}v\ \underbrace{v\ \cdots\ \pi^{-1}w}_{A_1}\ \underbrace{w\ \cdots\pi^{-1}u}_{B_1}\ u\ \cdots)(\cdots\ \Gamma u\ \underbrace{\pi\Gamma u\ \cdots\ \Gamma w}_{B_2}\ \underbrace{\pi\Gamma w\ \cdots\Gamma v}_{A_2}\ \pi\Gamma v\ \cdots)$$

where blocks A_1 and A_2 are reverse complementary, as are B_1 and B_2. Applying $\rho = (u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$ we get the genome $\sigma = \rho\pi$,

$$\sigma = (\cdots\ \pi^{-1}v\ \underbrace{w\ \cdots\ \pi^{-1}u}_{B_1}\ \underbrace{v\ \cdots\pi^{-1}w}_{A_1}\ u\ \cdots)(\cdots\ \Gamma u\ \underbrace{\pi\Gamma w\ \cdots\Gamma v}_{A_2}\ \underbrace{\pi\Gamma u\ \cdots\ \Gamma w}_{B_2}\ \pi\Gamma v\ \cdots)$$

where blocks A_1 and B_1 where exchanged, and also A_2 and B_2, resulting in a transposition, where two consecutive blocks exchange positions, as shown in Figure 4.13.

(a) Genome π with elements u and v in the same strand of a linear chromosome.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, corresponding to the excision of block B from the linear chromosome into a new circular chromosome.

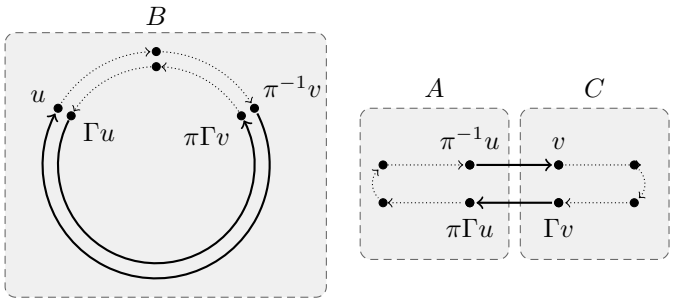
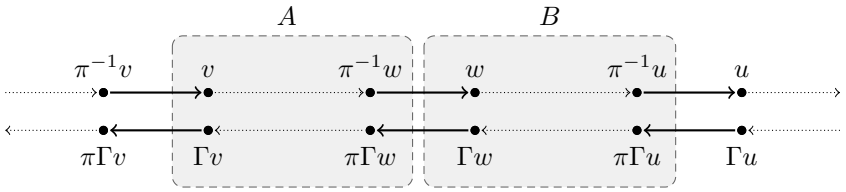


Figure 4.12: Example of the application of a circular excision operation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when u and v belong to the same strand in a linear chromosome.

(a) Genome π with elements u, v and w on the same strand, in this order.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$, resulting in a transposition of blocks A and B.

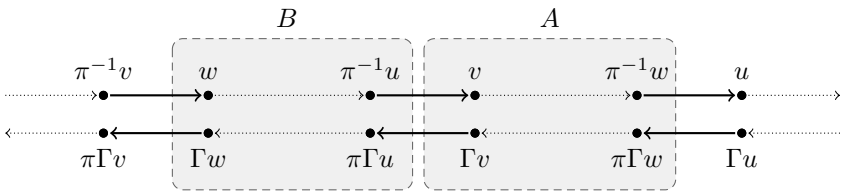


Figure 4.13: Example of the application of a transposition, modeled by the permutation $\rho = (u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$, when u,v, and w are in the same strand and in this order.

v) Block-interchanges

The permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)(w\ x)(\pi\Gamma x\ \pi\Gamma w)$ defines a block-interchange in π, given that elements x, u, w and v are on the same strand and in this order in π. For instance, in the genome

$$\begin{aligned}\pi \;=\; (\cdots\ \pi^{-1}x\ \underbrace{\boxed{x\ \cdots\ \pi^{-1}u}}_{A_1}\ u\ \cdots\ \pi^{-1}w\ \underbrace{\boxed{w\ \cdots\ \pi^{-1}v}}_{B_1}\ v\ \cdots)\\ (\cdots\ \Gamma v\ \underbrace{\boxed{\pi\Gamma v\ \cdots\ \Gamma w}}_{B_2}\ \pi\Gamma w\ \cdots\ \Gamma u\ \underbrace{\boxed{\pi\Gamma u\ \cdots\ \Gamma x}}_{A_2}\ \pi\Gamma x\ \cdots)\end{aligned}$$

where blocks A_1 and A_2 are reverse complementary, as are B_1 and B_2, applying $\rho = (u\ w)(\pi\Gamma w\ \pi\Gamma u)(v\ x)(\pi\Gamma x\ \pi\Gamma v)$ we get the genome

$$\begin{aligned}\sigma = \rho\pi \;=\; (\cdots\ \pi^{-1}x\ \underbrace{\boxed{w\ \cdots\ \pi^{-1}v}}_{B_1}\ u\ \cdots\ \ \pi^{-1}w\ \underbrace{\boxed{x\ \cdots\ \pi^{-1}u}}_{A_1}\ v\ \cdots)\\ (\cdots\ \Gamma v\ \underbrace{\boxed{\pi\Gamma u\ \cdots\ \Gamma x}}_{A_2}\ \pi\Gamma w\ \cdots\ \Gamma u\ \underbrace{\boxed{\pi\Gamma v\ \cdots\ \Gamma w}}_{B_2}\ \pi\Gamma x\ \cdots)\end{aligned}$$

resulting in a block-interchange, swapping blocks A and B, as shown in Figure 4.14.

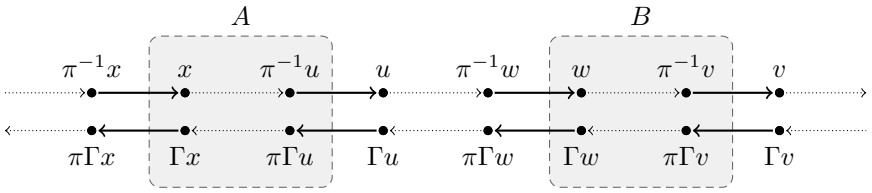
Notice that, as we said in the section about fusion and fissions, the block-interchange operation can be seen as a combination of two 2-break operations, just as in the DCJ model [110]. If the affected chromosome is circular, then the operations will be a fission followed by a fusion; if the chromosome is linear, then we will have a circular excision, creating a circular intermediate chromosome, followed by a circular reabsorption. In fact, we can see that the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)(w\ x)(\pi\Gamma x\ \pi\Gamma w)$ is the product of two 2-break operations, modeled by the permutations $(u\ v)(\pi\Gamma v\ \pi\Gamma u)$ and $(w\ x)(\pi\Gamma x\ \pi\Gamma w)$. If we apply them separately, say $(u\ v)(\pi\Gamma v\ \pi\Gamma u)$ first, the result is a fission (or circular excision), since the elements u and v are on the same strand of the chromosome. After applying this first operation, the element w will be on one of the resulting chromosomes and x in the other. Therefore, the permutation $(w\ x)(\pi\Gamma x\ \pi\Gamma w)$ will merge these two chromosomes with a fusion (or circular reabsorption).

vi) Operations on Telomeres

When a 2-break operation involves telomeres, the resulting permutation is slightly modified. Intuitively, this happens because the telomere is not connected to another extremity, and therefore we "save" a cut and a later join, that is, we do not need two cut-and-joins, a single cut-and-join will do. For instance, the reversal of a block of genes in the tail of a linear chromosome can be achieved by cutting the start of the block and then joining the telomere of the block back in the cut position.

In the context of the chromosomal algebraic theory, we saw in the beginning of Section 4.3 that if an extremity v of a genome π satisfies $v = \pi\Gamma v$, then v is a telomere. If

(a) Genome π with elements x, u, w and v on the same strand, in this order.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)(w\ x)(\pi\Gamma x\ \pi\Gamma w)$, resulting in a block-interchange of blocks A and B.

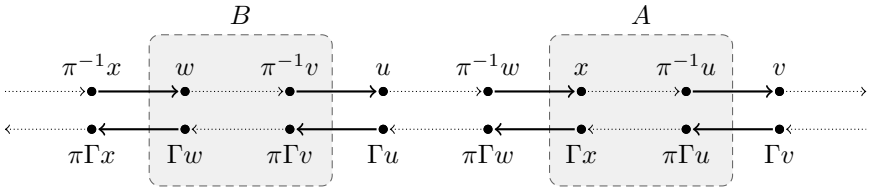


Figure 4.14: Example of the application of a block-interchange, modeled by the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)(w\ x)(\pi\Gamma x\ \pi\Gamma w)$, when x,u, w, and v are in the same strand and in this order.

we build a 2-break permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ with such a v, the permutation can be simplified, since

$$\rho = (u\ \ v)(\pi\Gamma v\ \ \pi\Gamma u) = (u\ \ v)(v\ \ \pi\Gamma u) = (u\ \ v\ \ \pi\Gamma u),$$

meaning that this particular case of 2-break operation is modeled by a 3-cycle, instead of the usual product of two 2-cycles. Notice that even though the permutation is slightly simpler, its norm remains the same, since the norm of a 3-cycle is also two.

We have two types of 2-break operations involving chromosome extremities. One is the *affix reversal*, that is, the reversal of a block of genes in a tail of a linear chromosome. For instance, we can take the genome

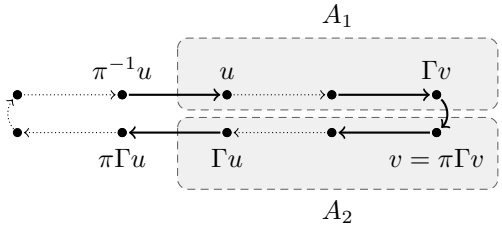
$$\pi = (\cdots\ \pi^{-1}u\ \underbrace{u\ \cdots\ \Gamma v}_{A_1}\ \underbrace{v\ \cdots\ \Gamma u}_{A_2}\ \pi\Gamma u\ \cdots)$$

where u and v are in different strands of a linear chromosome and v is a telomere (since v and Γv are consecutive), and blocks A_1 and A_2 are reverse complementary. Applying $\rho = (u\ \ v)(\pi\Gamma v\ \ \pi\Gamma u) = (u\ \ v\ \ \pi\Gamma u)$ results in

$$\sigma = \rho\pi = (\cdots\ \pi^{-1}u\ \underbrace{v\ \cdots\ \Gamma u}_{A_2}\ \underbrace{u\ \cdots\ \Gamma v}_{A_1}\ \pi\Gamma u\ \cdots)$$

swapping A_1 and A_2, reversing the tail, and causing u to be the new telomere, as shown in Figure 4.15.

(a) Genome π with elements u and v in different strands, and v a telomere.



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u) = (u\ \ v\ \ \pi\Gamma u)$ and v is a telomere, corresponding to an affix reversal.

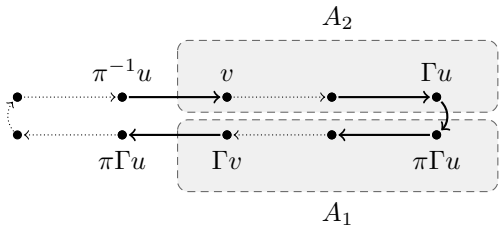


Figure 4.15: Example of the application of an affix reversal, modeled by the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u) = (u\ \ v\ \ \pi\Gamma u)$, when u and v belong to different strands of a chromosome and v is a telomere. Blocks A_1 and A_2, which are reverse complementary, exchange places, resulting in this reversal.

The other type of operation on linear chromosomes extremities is the *non-reciprocal translocation*. This operation is similar to a translocation, but instead of two linear chromosomes exchanging tails, only one tail of a chromosome is moved to the other chromosome. For instance, taking the genome

$$\pi = (\cdots\ \Gamma u\ \underbrace{\pi\Gamma u\ \cdots\ \pi^{-1}u}_{A}\ \ u\ \cdots)(\cdots\ \Gamma v\ \ v\ \cdots),$$

where u and v are in different linear chromosomes we observe that v is a telomere. We know that the block A is a tail of the genome π, since it is surrounded by complementary extremities u and Γu. Applying $\rho = (u\ \ v)(\pi\Gamma v\ \ \pi\Gamma u)$ we get

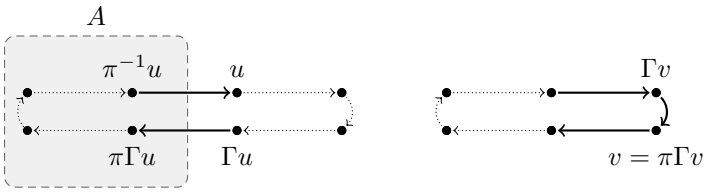
$$\sigma = \rho\pi = (\cdots\ \Gamma u\ \ u\ \cdots)(\cdots\ \Gamma v\ \underbrace{\pi\Gamma u\ \cdots\ \pi^{-1}u}_{A}\ v\ \cdots),$$

that is, the block A is removed from π and inserted σ, resulting in a non-reciprocal translocation, as shown in Figure 4.16. Notice that, in σ, the extremity u becomes a telomere and the gene with extremity v becomes an internal gene.

vii) Single Cut-or-Join Operations

This last subsection deals with *single cut-or-join* rearrangement operations, where a single cut or join is applied. They are modeled using a single 2-cycle, as we will see.

(a) Genome π with elements u and v in different linear chromosomes, and v a telomere



(b) Genome $\sigma = \rho\pi$, where $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$ and v is a telomere, corresponding to a non-reciprocal translocation.

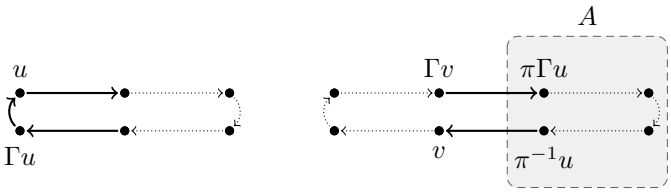


Figure 4.16: Example of the application of a non-reciprocal translocation, modeled by the permutation $\rho = (u\ v)(\pi\Gamma v\ \pi\Gamma u)$, when u and v belong to different linear chromosomes and v is a telomere. The block A is removed from π and included in σ, making u a telomere.

A *cut* between two adjacent extremities u and v breaks the connection between their two genes. Each gene will then have a free extremity, which will become a telomere. The opposite operation, *join*, links two telomeres creating a new adjacency.

Considering the genome

$$\pi = (\cdots\ u\ \ v\ \cdots)(\cdots\ \Gamma v\ \ \Gamma u\ \cdots),$$

where u and v are adjacent, that is, $\pi u = v$, suppose we want to make a cut between u and v. After applying this cut, in the new genome the successor of u will be Γu, and the predecessor of v will be Γv. Formally, if ρ is the permutation modeling the cut, then

$$\rho\pi(u) = \Gamma u \tag{4.9}$$

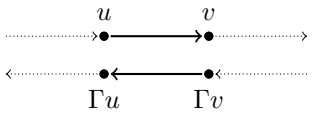
and

$$\rho\pi\Gamma(v) = v. \tag{4.10}$$

From Equation (4.9) we have that $\rho(v) = \Gamma u$, since $\pi u = v$. Also, since π is a valid genome, it satisfies $\pi = \Gamma\pi^{-1}\Gamma$, which implies that $\pi\Gamma = \Gamma\pi^{-1}$. Using this in Equation (4.10), we get $\rho\Gamma\pi^{-1}(\pi u) = v$ implying that $\rho(\Gamma u) = v$. Therefore, we have that $\rho = (\Gamma u\ v)$. Indeed, applying ρ to the genome π we get

$$\sigma = \rho\pi = (\cdots\ u\ \ \Gamma u\ \cdots)(\cdots\ \Gamma v\ \ v\ \cdots),$$

(a) Genome π with consecutive elements u and v on the same strand.



(b) Genome $\sigma = \rho\pi$, where $\rho = (\Gamma u\ v)$, resulting in a cut between u and v.

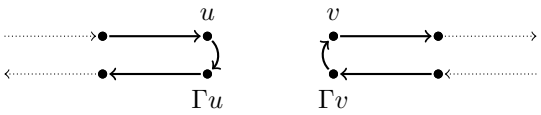


Figure 4.17: Example of the application of a cut operation, modeled by the permutation $\rho = (\Gamma u\ v)$, when u and v are in the same strand and are consecutive, that is, $\pi u = v$.

cutting the connection between u and v, causing Γu and v to become telomeres in σ. This operation is shown in Figure 4.17.

If the chromosome where u and v are is linear, a cut can be called a *linear fission*, an operation that breaks a linear chromosome in two. If the chromosome is circular, the cut is a *linearization*, transforming the circular chromosome into a linear chromosome.

Since $\rho = \rho^{-1}$, applying the same permutation ρ to the genome σ results in the inverse operation, a *join*. In this case, Γu and v are telomeres, so $\sigma u = \Gamma u$ and $\sigma\Gamma v = v$. If u and v belong to the same linear chromosome, the operation is a *circularization*. If u and v belong to different linear chromosomes, it is a *linear fusion*.

So, to summarize, a *cut* in a genome π between consecutive elements u and v, such that $\pi u = v$, is modeled by a permutation $\rho = (\Gamma u\ v)$. A *join* in a genome π between telomeres Γu and v, such that $\pi u = \Gamma u$ and $\pi\Gamma v = v$, is also modeled by a permutation $\rho = (\Gamma u\ v)$.

Table 4.1 shows a summary of the classic rearrangement operations and how they are modeled with algebraic permutations.

Permutation	Operation	Conditions
$(\Gamma u\ v)$	Cut	u and v are consecutive, satisfying $\pi u = v$.
	Join	Γu and v are telomeres in π, with $\pi u = \Gamma u$ and $\pi\Gamma v = v$.
$(u\ v)(\pi\Gamma v\ \pi\Gamma u)$	Signed reversal	u and v in different strands of the same chromosome.
	Fission	u and v on the same strand of a circular chromosome.
	Fusion	u and v in different circular chromosomes.
	Translocation	u and v in different linear chromosomes.
	Circular excision	u and v on the same strand of a linear chromosome.
	Circular reabsorption	u and v in different circular and linear chromosomes.
	Translocation	u and v in different linear chromosomes.
$(u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$	Transposition	u, v and w in same strand and appear in this order.
$(u\ v)(\pi\Gamma v\ \pi\Gamma u)(w\ x)(\pi\Gamma x\ \pi\Gamma w)$	Block-interchange	x, u, w and v in same strand and appear in this order.

Table 4.1: Permutations modeling classic rearrangement operations.

Chapter 5

Conclusions

In this thesis, we presented two new genome rearrangement models, that have been published in the last three years.

In Chapter 3, we presented an alternative measure of the breakpoint distance, based on the *Single-Cut-or-Join* (SCJ) operation, that allows linear- and polynomial-time solutions to some rearrangement problems that are NP-hard under the breakpoint distance. This is the case, for instance, for the multichromosomal linear versions of the genome halving, guided halving and genome median problems. In addition, the SCJ approach is able to produce a rearrangement scenario between genomes, not only a distance. Also, the multiple genome rearrangement problem under the SCJ distance is a much easier problem than in any other rearrangement distance. In fact, it is the only proposed distance for which the small parsimony problem has a known polynomial time algorithm.

From a biological point of view, we can think of a rearrangement event as an accepted mutation, that is, a mutational event involving large, continuous genome segments that was accepted by natural selection, and therefore became fixed in a population. SCJ may model the mutation part well, but a model for the acceptance part is missing. For instance, while the mutational effort of doing a fission seems to be less than that of an inversion, the latter is more frequent as a rearrangement event, probably because it has a better chance of being accepted. This may have to do with the location and movement of origins of replication, since any free segment will need one to become fixed.

Other considerations, such as the length of segments, hotspots, presence of flanking repeats, etc. are likely to play a role in genome rearrangements, and need to be taken into account in a comprehensive model.

Although crude from the standpoint of evolutionary genomics, this distance may serve as a fast, first-order approximation for other, better founded genomic rearrangement distances, and also for reconstructed phylogenies.

In Chapter 4, the *Adjacency Algebraic Theory*, which is an expansion to the Alge-

braic Theory of Meidanis and Dias [74], was presented, allowing the modeling of linear chromosomes using the algebraic theory of genome rearrangements.

With the possibility of modeling multichromosomal linear and circular genomes, we believe that the algebraic theory is a interesting alternative for solving rearrangement problems, with a different perspective that might complement the usual combinatorial, graph-theoretical approach.

Since the algebraic rearrangement distance is different from other proposed models, such as Double Cut-and-Join, several rearrangement problems are still open, such as the genome median and genome halving problems. This opens up a vast area of investigation of the potential for the algebraic theory to solve these problems in different ways.

We are particularly interested in the genome median, since it is a fundamental problem in rearrangement-based phylogenetic reconstruction. Not only in trying to solve it in its original formulation, but also investigating new metrics for defining what is a good genome median. Instead of the usual definition, minimizing the sum of its distances to the three input genomes, alternative metrics such as minimizing the maximum distance could be used. The algebraic theory can be an important tools in solving this and several other open problems.

Bibliography

[1] Z. Adam and D. Sankoff. The ABCs of MGR with DCJ. *Evolutionary Bioinformatics Online*, 4:69, 2008.

[2] D. Aldous. Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. *Statistical Science*, 16(1):23–34, 2001.

[3] M. Alekseyev and P. Pevzner. Multi-break rearrangements and chromosomal evolution. *Theoretical Computer Science*, 395(2-3):193–202, 2008.

[4] M. A. Alekseyev and P. A. Pevzner. Colored de Bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):98–107, 2007.

[5] M. A. Alekseyev and P. A. Pevzner. Whole genome duplications, multi-break rearrangements, and genome halving problem. In *Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 07, pages 665–679. Society for Industrial and Applied Mathematics, 2007.

[6] M. A. Alekseyev and P. A. Pevzner. Breakpoint graphs and ancestral genome reconstructions. *Genome Research*, 19(5):943–57, 2009.

[7] A. A. M. Almeida. Comparação algébrica de genomas: o caso da distância de reversão. Master's thesis, Institute of Computing, University of Campinas, 2007.

[8] D. A. Bader, B. M. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[9] M. Bader, M. I. Abouelhoda, and E. Ohlebusch. A fast algorithm for the multiple genome rearrangement problem with weighted reversals and transpositions. *BMC Bioinformatics*, 9:516, 2008.

[10] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[11] V. Bafna and P. P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998.

[12] A. Bergeron, P. Medvedev, and J. Stoye. Rearrangement models and single-cut operations. *Journal of Computational Biology*, 17(9):1213–1225, 2010.

[13] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. *Lecture Notes in Computer Science*, 4175:163–173, 2006.

[14] A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. *Journal of Computational Biology*, 13(2):567–578, 2006.

[15] A. Bergeron, J. Mixtacki, and J. Stoye. HP distance via double cut and join distance. In *Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching*, volume 5029 of *Lecture Notes in Computer Science*, pages 56–68, 2008.

[16] P. Berman and S. Hannenhalli. Fast sorting by reversals. In *Proceedings of the 7th Annual Symposium in Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185, 1996.

[17] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-Approximation Algorithm for Sorting by Reversals. In R. H. Möhring and R. Raman, editors, *Proceedings of the 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210, 2002.

[18] P. Biller. *Experimentos em rearranjo de genomas com a operação Single-Cut-or-Join*. Master's thesis, Institute of Computing, University of Campinas, 2012.

[19] M. Blanchette, G. Bourque, and D. Sankoff. Breakpoint phylogenies. *Genome Informatics*, 8:25–34, 1997.

[20] G. Bourque and P. P. A. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Research*, 12(1):26, 2002.

[21] M. D. V. Braga. *Exploring the solution space of sorting by reversals when analyzing genome rearrangements*. PhD thesis, Université Lyon, 2009.

[22] M. D. V. Braga and J. Stoye. The solution space of sorting by DCJ. *Journal of Computational Biology*, 17(9):1145–65, 2010.

[23] D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM-2579, Centre de recherches mathematiques, Universite de Montreal, 1998.

[24] L. Bulteau, G. Fertin, and I. Rusu. Sorting by Transpositions Is Difficult. *SIAM Journal on Discrete Mathematics*, 26(3):1148–1180, 2012.

[25] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the first annual international conference on computational molecular biology*, RECOMB '97, pages 75–83, 1997.

[26] A. Caprara. On the tightness of the alternating-cycle lower bound for sorting by reversals. *Journal of Combinatorial Optimization*, 3(2):149–182, 1999.

[27] A. Caprara. The reversal median problem. *INFORMS Journal on Computing*, 15:93–113, 2003.

[28] D. A. Christie. Sorting permutations by block-interchanges. *Information Processing Letters*, 60(4):165–169, 1996.

[29] P. P. Côgo. *Comparação de Genomas Completos de Espécies da Família Vibrionacea Empregando Rearranjos de Genomas*. Master's thesis, Institute of Computing, University of Campinas, 2008.

[30] M. Cosner. Chloroplast DNA rearrangements in Campanulaceae: phylogenetic utility of highly rearranged genomes. *BMC Evolutionary Biology*, 4:27, 2004.

[31] M. Cosner, R. Jansen, B. Moret, L. Raubeson, L. Wang, T. Warnow, and S. Wyman. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 99–121. Kluwer Academic Publishers, Dordrecht, 2000.

[32] L. F. I. Cunha, L. A. B. Kowada, R. de A. Hausen, and C. M. H. de Figueiredo. Transposition Diameter and Lonely Permutations. In *Proceedings of the 7th Brazilian Symposium on Bioinformatics, BSB 2012*, volume 7409 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2012.

[33] K. Z. de Oliveira. *Construção de Filogenias Baseadas em Genomas Completos*. Master's thesis, Institute of Computing, University of Campinas, 2010.

[34] U. Dias and Z. Dias. An improved 1.375-approximation algorithm for the transposition distance problem. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, BCB '10, pages 334–337. ACM, 2010.

[35] Z. Dias and J. Meidanis. Genome rearrangements distance by fusion, fission, and transposition is easy. In *Proceedings of the Eighth International Symposium on String Processing and Information Retrieval*, SPIRE 2001, pages 250–253, 2001.

[36] N. El-Mabrouk, Joseph H. Nadeau, and D. Sankoff. Genome halving. In *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching*, volume 1448 of *Lecture Notes in Computer Science*, pages 235–250. Springer-Verlag, 1998.

[37] N. El-Mabrouk and D. Sankoff. The reconstruction of doubled genomes. *SIAM Journal on Computing*, 32(3):754–792, 2003.

[38] I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):369–379, 2006.

[39] S. Even and O. Goldreich. The minimum-length generator sequence problem is NP-Hard. *Journal of Algorithms*, 2(3):311–313, 1981.

[40] P. Feijao and J. Meidanis. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8:1318–1329, 2011.

[41] P. Feijao and J. Meidanis. Extending the algebraic formalism for genome rearrangements to include linear chromosomes. In *Proceedings of the 7th Brazilian Symposium on Bioinformatics, BSB 2012*, volume 7409 of *Lecture Notes on Computer Science*, pages 13–24, Heidelberg, 2012. Springer-Verlag Berlin.

[42] J. Feng and D. Zhu. Faster algorithms for sorting by transpositions and sorting by block interchanges. *ACM Transaction on Algorithms*, 3(3), 2007.

[43] G. Fertin, A. Labarre, and I. Rusu. *Combinatorics of genome rearrangements*. The MIT Press, 2009.

[44] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[45] V. Fortuna. *Distâncias de Transposição entre Genomas*. Master's thesis, Institute of Computing, University of Campinas, 2005.

[46] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[47] G. Fritzsch, M. Schlegel, and P. P. F. Stadler. Alignments of mitochondrial genome arrangements: applications to metazoan phylogeny. *Journal of Theoretical Biology*, 240(4):511–520, 2006.

[48] A. Goeffon, M. Nikolski, and D. J. Sherman. An efficient probabilistic population-based descent for the median genome problem. *Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO 08*, pages 315–322, 2008.

[49] S. Hannenhalli. Polynomial-time Algorithm for Computing Translocation Distance between Genomes. *Discrete Applied Mathematics*, 71(1–3):137–151, 1996.

[50] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS '95*, pages 581–592. IEEE Computer Society, 1995.

[51] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):200–223, 1999.

[52] Y.-L. Huang, C.-C. Huang, C. Y. Tang, and C. L. Lu. An improved algorithm for sorting by block-interchanges based on permutation groups. *Information Processing Letters*, 110(8-9):345–350, 2010.

[53] Y.-L. Huang and C. L. Lu. Sorting by reversals, generalized transpositions, and translocations using permutation groups. *Journal of Computational Biology*, 17(5):685–705, 2010.

[54] D. H. Huson, S. M. Nettles, and T. J. Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *Journal of Computational Biology*, 6(3-4):369–386, 1999.

[55] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29(3):880–892, 1999.

[56] J. D. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the sixth ACM-SIAM Symposium on Discrete Algorithms, SODA 95*, pages 604–613. Society of Industrial and Applied Mathematics, 1995.

[57] J. D. Kececioglu and D. Sankoff. Exact and Approximation Algorithms for Sorting by Reversals, with Application to Genome Rearrangement. *Algorithmica*, 13(1-2):180–210, 1995.

[58] M. Kellis, B. W. Birren, and E. S. Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast Saccharomyces cerevisiae. *Nature*, 428(6983):617–624, 2004.

[59] J. Kováč. On the complexity of rearrangement problems under the breakpoint distance. *Arxiv preprint arXiv:1112.2172*, pages 1–15, 2011.

[60] J. Kováč, R. Warren, M. D. V. Braga, and J. Stoye. Restricted DCJ model: rearrangement problems with chromosome reincorporation. *Journal of computational biology : a journal of computational molecular cell biology*, 18(9):1231–41, Sept. 2011.

[61] R. B. Langkjaer, P. F. Cliften, M. Johnston, and J. Piskur. Yeast genome duplication was followed by asynchronous differentiation of duplicated genes. *Nature*, 421(6925):848–852, 2003.

[62] R. Lenne, C. Solnon, T. Stützle, E. Tannier, and M. Birattari. Effective stochastic local search algorithms for the genomic median problem. In *Doctoral Symposium on Engineering: Stochastic Local Search Algorithms (SLS-DS)*, IRIDIA Technical Report Series, pages 1–5. Springer-Verlag, 2007.

[63] G. Li, X. Qi, X. Wang, and B. Zhu. A linear-time algorithm for computing translocation distance between signed genomes. *Combinatorial Pattern Matching*, pages 323–332, 2004.

[64] G.-H. Lin and G. Xue. On the terminal Steiner tree problem. *Information Processing Letters*, 84(2):103–107, 2002.

[65] Y. Lin and B. Moret. Estimating true evolutionary distances under the DCJ model. *Bioinformatics*, 24(13):i114–i122, 2008.

[66] Y. Lin, V. Rajan, K. M. Swenson, and B. M. E. Moret. Estimating true evolutionary distances under rearrangements, duplications, and losses. *BMC bioinformatics*, 11 Suppl 1:S54, 2010.

[67] Y. C. Lin, C. L. Lu, H.-Y. Chang, and C. Y. Tang. An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species. *Journal of Computational Biology*, 12(1):102–112, 2005.

[68] Y. C. Lin, C. L. Lu, Y.-C. Liu, and C. Y. Tang. SPRING: a tool for the analysis of genome rearrangement using reversals and block-interchanges. *Nucleic Acids Research*, 34(Web Server issue):W696–W699, 2006.

[69] L. Lovász and M. D. Plummer. Matching theory. In *Annals of Discrete Mathematics*, volume 29. North-Holland, 1986.

[70] C. L. Lu, Y. L. Huang, T. C. Wang, and H.-T. Chiu. Analysis of circular genome rearrangement by fusions, fissions and block-interchanges. *BMC Bioinformatics*, 7:295, 2006.

[71] J. Ma, L. Zhang, B. B. Suh, B. J. B. Raney, R. C. Burhans, W. J. Kent, M. Blanchette, D. Haussler, and W. Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Research*, 16(12):1557–1565, 2006.

[72] F. V. Martinez, J. C. de Pina, and J. Soares. Algorithms for terminal Steiner trees. *Theoretical Computer Science*, 389(1-2):133–142, 2007.

[73] B. McClintock. The origin and behavior of mutable loci in maize. *Proceedings of the National Academy of Sciences of the United States of America*, 36(6):344, 1950.

[74] J. Meidanis and Z. Dias. An alternative algebraic formalism for genome rearrangements. In *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*, pages 213–223. Kluwer Academic Publishers, 2000.

[75] J. Meidanis, M. M. T. Walter, and Z. Dias. A lower bound on the reversal and transposition diameter. *Journal of computational biology*, 9(5):743–5, 2002.

[76] C. Mira. *Análise Algébrica de Problemas de Rearranjo em Genomas: Algoritmos e Complexidade*. PhD thesis, Institute of Computing, University of Campinas, 2008.

[77] C. Mira and J. Meidanis. Sorting by block-interchanges and signed reversals. In *Proceedings of the Fourth International Conference on Information Technology (ITNG'07)*, pages 670–676. IEEE Computer Society, 2007.

[78] J. Mixtacki. Genome halving under DCJ revisited. In *Proceedings of the The 14th Annual International Computing and Combinatorics Conference (COCOON 2008)*, volume 5092 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 2008.

[79] B. M. Moret, L. S. Wang, T. Warnow, and S. K. Wyman. New approaches for reconstructing phylogenies from gene order data. *Bioinformatics*, 17 Suppl 1(suppl 1):S165–S173, 2001.

[80] B. M. E. Moret, A. C. Siepel, J. Tang, T. Liu1, and T. Liu. Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In *Proceedings of the 2nd International Workshop of Algorithms in Bioinformatics (WABI 2002)*, volume 2452 of *Lecture Notes in Computer Science*, pages 521–536. Springer Berlin / Heidelberg, 2002.

[81] B. M. E. Moret, J. Tang, L.-S. Wang, and T. Warnow. Steps toward accurate reconstructions of phylogenies from gene-order data. *Journal of Computer and System Sciences*, 65(3):508–525, Nov. 2002.

[82] J. Nadeau and B. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences*, 81(3):814, 1984.

[83] S. Ohno. *Evolution by gene duplication*. Springer-Verlag, 1970.

[84] M. Ozery-Flato and R. Shamir. An $O(n^{3/2}\sqrt{\log(n)})$ algorithm for sorting by reciprocal translocations. *The Journal of Heredity*, 4009(2):258, 2006.

[85] I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity*, 71(5), 1998.

[86] P. Pevzner and G. Tessler. Transforming men into mice: the Nadeau-Taylor chromosomal breakage model revisited. In *Proceedings of the 7th Annual International Conference on Computational Biology (RECOMB 2003)*, pages 247–256. ACM Press, 2003.

[87] V. Rajan, A. W. Xu, Y. Lin, K. M. Swenson, and B. M. Moret. Heuristics for the inversion median problem. *BMC Bioinformatics*, 11(Suppl 1):S30, 2010.

[88] D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1–2):131–147, 1981.

[89] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[90] D. Sankoff and M. Blanchette. The median problem for breakpoints in comparative genomics. In *Proceedings of the Third Annual International Conference on Computing and Combinatorics (COCOON 97)*, volume 1276 of *Lecture Notes in Computer Science*, pages 251–263. Springer Berlin / Heidelberg, 1997.

[91] D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *Journal of Computational Biology*, 5(3):555–570, 1998.

[92] D. Sankoff and A. W. Xu. Decompositions of multiple breakpoint graphs and rapid exact solutions. In *Proceedings of the 8th international workshop of Algorithms in bioinformatics (WABI 2008)*, volume 5251 of *Lecture Notes in Computer Science*, page 25. Springer Verlag, 2008.

[93] C. Seoighe and K. H. Wolfe. Extent of genomic rearrangement after genome duplication in yeast. *Proceedings of the National Academy of Sciences of the United States of America*, 95(8):4447–52, May 1998.

[94] A. H. Sturtevant and T. Dobzhansky. Inversions in the third chromosome of wild races of *Drosophila pseudoobscura*, and their use in the study of the history of the species. *Proceedings of the National Academy of Sciences*, 22(7):448–450, 1936.

[95] D. Swofford, G. Olsen, P. Waddell, and D. Hillis. Phylogenetic inference. In D. Hiillis, D. Moritz, and B. Mable, editors, *Molecular Systematics*, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.

[96] J. Tang and B. M. E. Moret. Scaling up accurate phylogenetic reconstruction from gene-order data. *Bioinformatics*, 19 Suppl 1(90001):i305–i312, 2003.

[97] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching, CPM 2004*, volume 3109 of *Lecture Notes in Computer Science*, pages 1–13s. Springer Berlin / Heidelberg, 2004.

[98] E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC bioinformatics*, 10:120, 2009.

[99] G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *Journal of Computer and System Sciences*, 65(3):587–609, 2002.

[100] L.-S. Wang. Exact-IEBP: a new technique for estimating evolutionary distances between whole genomes. In *Proceedings of the First International Workshop on Algorithms in Bioinformatics (WABI 2001)*, pages 175–188. Springer-Verlag, 2001.

[101] L.-S. Wang and T. Warnow. Estimating true evolutionary distances between genomes. In *Proceedings of the 33rd annual ACM symposium on Theory of computing (STOC '01)*, pages 637–646. ACM Press, 2001.

[102] L.-S. Wang, T. Warnow, B. M. E. Moret, R. K. Jansen, and L. A. Raubeson. Distance-based genome rearrangement phylogeny. *Journal of Molecular Evolution*, 63(4):473–483, Oct. 2006.

[103] R. Warren and D. Sankoff. Genome aliquoting with double cut and join. *BMC bioinformatics*, 10 Suppl 1:S2, 2009.

[104] R. Warren and D. Sankoff. Genome halving with double cut and join. *Journal of Bioinformatics and Computational Biology*, 7(2):357–71, 2009.

[105] R. Warren and D. Sankoff. Genome aliquoting revisited. *Journal of Computational Biology*, 18(9):1065–75, 2011.

[106] G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99:1–7, 1982.

[107] A. W. Xu. A fast and exact algorithm for the median of three problem: a graph decomposition approach. *Journal of Computational Biology*, 16(10):1369–81, 2009.

[108] A. W. Xu. The median problems on linear multichromosomal genomes: graph representation and fast exact solutions. *Journal of Computational Biology*, 17(9):1195–211, 2010.

[109] W. Xu and B. Moret. GASTS: Parsimony scoring under rearrangements. In *Proceedings of the 11th international workshop on Algorithms in Bioinformatics (WABI 2011)*, pages 351–363, 2011.

[110] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–6, 2005.

[111] H. Zhao and G. Bourque. Recovering true rearrangement events on phylogenetic trees. In G. Tesler and D. Durand, editors, *Comparative Genomics*, volume 4751 of *Lecture Notes in Computer Science*, pages 149–161. Springer Berlin / Heidelberg, 2007.

[112] H. Zhao and G. Bourque. Recovering genome rearrangements in the mammalian phylogeny. *Genome Research*, 19(5):934–942, 2009.

[113] C. Zheng, Q. Zhu, Z. Adam, and D. Sankoff. Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes. *Bioinformatics*, 24(13):i96–104, 2008.

[114] C. Zheng, Q. Zhu, and D. Sankoff. Genome halving with an outgroup. *Evolutionary bioinformatics online*, 2:295, 2006.