UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Pseudo-Boolean Approaches to Comparative Genomics

João Benigno Delgado

Dissertação para obtenção do Grau de Mestre em

Matemática e Aplicações

Júri

Presidente:	Doutora Maria Cristina Sales Viana Serôdio Sernadas
Orientador:	Doutor Vasco Miguel Gomes Nunes Manquinho
Co-orientador:	Doutora Maria Inês Camarate de Campos Lynce de Faria
Vogal:	Doutor João Filipe Quintas dos Santos Rasga

Junho, 2009

Resumo

A genomica comparativa é o estudo de relações funcionais e estruturais entre genomas. A comparação de genomas em termos de evolução genética teve como ponto de partida a comparação de genes comuns a dois genomas. Contudo, o aumento do número de genomas totalmente sequenciados abriu portas ao estudo de rearranjos genómicos, ou seja, à comparação de dois genomas em termos da evolução da ordem dos seus genes.

Foram feitas várias abordagens a este tópico, ora simples de mais para serem aplicáveis a genomas complexos, ora demasiado complexas para serem resolvidas de forma eficiente. Recentemente procurou-se encontrar uma abordagem que pudesse lidar com genes duplicados. Deste desafio nasceram os modelos de emparelhamento: exemplar [33] e máximo [13]; e várias medidas de semelhança, tais como: o número de pontos de quebra [28], o número de intervalos comuns [39] e a soma de adjacências interrompidas (SAD) [35]. A ideia consiste em estabelecer um emparelhamento, ou seja, uma correspondência entre genes homólogos, por forma a desambiguar os dados de genes duplicados e poder calcular uma medida de semelhança. O objectivo é encontrar o emparelhamento que melhor preserva a ordem dos genes em dois genomas, sendo a preservação de ordem avaliada por uma medida de semelhança.

Esta dissertação apresenta formalismos claros para as abordagens feitas em rearranjos genómicos e estende o trabalho existente para a resolução dos problemas do número de intervalos comuns [2] e do número de pontos de quebra [1] usando Optimização Linear Inteira $0-1$ ($\{0,1\}-$ILO). Por fim, propõe-se uma codificação nova usando $\{0,1\}-$ILO para resolver o problema SAD.

Palavras-chave

Genomica Comparativa, Medidas de Semelhança, Modelos de emparelhamento, Optimização Linear Inteira $0-1$, Rearranjos Genómicos, Soma de Adjacências Interrompidas.

Abstract

Comparative genomics is the study of structural and functional relationships between genomes. Primarily, the comparison of genomes in terms of genetic evolution was based on gene comparison. However, the increasing number of fully sequenced genomes has led to the study of genome rearrangements, i.e., to the comparison of two genomes in terms of gene order evolution.

Several approaches have been made to this topic, all of them being either too complex to be solved efficiently or too simple to be applied to the genomes of complex organisms. The latest challenge has been to overcome the problem of having genomes with duplicate genes. This led to the definition of two matching models: exemplar [33] and maximum [13]; and of several similarity measures, such as: the number of breakpoints [28], the number of common intervals [39] and the Summed Adjacency Disruption number (SAD) [35]. The idea is to find a matching, i.e., a correspondence between homologous genes in two genomes, in order to disambiguate the data of duplicate genes and a calculate a similarity measure. The problem becomes that of finding a matching that best preserves the order of genes in two genomes, where gene order is evaluated by a chosen similarity measure.

This dissertation presents clearer formalisms of the approaches made on genome rearrangements and extends previous work using Pseudo-Boolean Optimization (PBO) to encode the problems of finding the number of common intervals [2] and the number of breakpoints [1]. Moreover, a new PBO encoding of the SAD problem is presented.

Keywords

Comparative Genomics, Genome Rearrangements, Matching Models, Pseudo-Boolean Optimization, Similarity Measures, Summed Adjacency Disruption Number.

Contents

List of Figures

List of Tables

1

Introduction

Comparative genomics is the study of structural, functional and evolutionary relationships between genomes. Primarily, genome evolutionary studies were based on gene comparison via sequence alignment tools [12, 3, 36]. Gene comparison is still an important step in genome evolutionary studies, however, the increasing number of fully sequenced genomes led to the recent study of genome rearrangements, where attention is shifted from gene level comparison to chromosomal level comparison [22, 17, 4, 32, 36, 40].

Genomes evolve in two ways: through local point mutations in DNA, that may concern at most one gene, and through wider operations that may affect the order of several genes. Traditional methods for comparing genomes in terms of genetic evolution are based in the comparison of homologous versions of genes in two genomes. These methods capture point mutations in the sequence of genes, but are limited in disregarding difference in the order by which genes occur in the considered genomes [36]. Indeed, there are cases of genomes that have many genes nearly identical but differ dramatically in gene order. Such a case was illustrated by Jeffrey Palmer and Laura Herbon when comparing the mitochondrial genomes of *Brassica oleracea* (cabbage) and *Brassica campestris* (turnip) [29]. Their study, among others, made clear that gene order rearranging operations represent a common mode of molecular evolution [31]. Later, it became clear that the application of gene comparison methods in order to measure evolutionary relations between highly rearranged genomes (such as herpes virus or plant mitochondrial DNA) may lead to contradicting results in phylogeny studies [3]. These results proved the study of genome rearrangements not only to be an important complementary study to follow gene comparison in genome evolution studies, but also to be the method of choice for comparing genomes that evolve very slowly in terms of point mutations [31].

1.1 Genome rearrangements

In genome rearrangements the objective is to estimate the most parsimonious scenario of gene order evolution between genomes of two species. The idea is to find the smallest number of genome rearrangement events necessary to transform the order of genes in one genome into the order of genes in

another genome. Assuming that evolution proceeded only through the considered type of rearrangement events, finding such a scenario provides a lower bound on the actual number of rearrangement events that must have occurred since the considered genomes began to diverge [19]. Such information can be used, for instance, to estimate how long ago the two genomes began to diverge from each other or, when comparing several genomes, to compute phylogenetic trees [17, 29, 36].

Several approaches have been made to *genome rearrangements*, all of them being either too complex to be solved efficiently or too simple to be applied to the genomes of more complex organisms. The latest challenge has been to overcome a limitation shared by all these approaches: the incapability of dealing with the presence of duplicate genes in genomes. All proposed approaches take for input two different orders over a same set of genes, assuming that no gene is present more than once in each genome being compared. However, it is known that the presence of duplicate genes is being common in genomes and that the duplication of a genomic segment is a process that can occur in the evolution of species [34, 26].

1.2 Matching models and similarity measures

One of the proposed approaches to overcome the problem of having duplicate genes in genomes has been the use of *matching models* and *similarity measures*, an idea first proposed by David Sankoff [33]. A matching is a one-to-one correspondence between equal (or more precisely, homologous) genes in two genomes. Such a correspondence allows to see the order of genes in one genome as a permutation of the order of genes in the other genome, this is, each matching induces a new order of genes in both genomes. The idea of this approach is to find a matching that best preserves the order of genes in two considered genomes, where order preservation is evaluated by a chosen similarity measure that takes for input the two new orders of genes induced by each matching.

Two matching models have been considered: the exemplar model [33] and the maximum model [13]. Several similarity measures have been proposed: *number of reversals* [40], *number of breakpoints* [28], *number of common intervals* [39], *number of conserved intervals* [5], *maximum adjacency disruption number* and *summed adjacency disruption number* [35], among others. Each choice of a matching model and similarity measure results in a different optimization problem. Unfortunately, most of these problems have been shown to be NP-hard in the presence of duplicate genes in genomes [10, 6, 13, 8].

Due to the hardness of these results, several heuristics have been proposed [38, 9, 8, 6]. Approximate results given by these heuristics have been used to compute phylogenetic trees for sets of genomes [38, 7] and also to identify ancestral homologs [9] or better identify orthologs [14] in the gene set of two genomes. However, there is still a lack of efficient tools to provide optimal results for these problems, without which one can not validate the accuracy of heuristics.

Recently, there have been proposed some Pseudo-Boolean approaches to compute exact solutions

for the number of breakpoints [1] and the number of common intervals [2] for both exemplar and maximum matchings. In this thesis, we further develop the work on these models and propose a new pseudo-Boolean optimization encoding for computing the summed adjacency disruption number for both exemplar and maximum matchings.

1.3 Organization

This thesis starts by introducing some basic concepts of molecular biology and bioinformatics that are required to better understand the biological problem being studied and its applications. In chapter 3, a detailed overview is presented on the approaches made on comparative genomics. Chapter 4, gives the definition of the problem of finding an exemplar (or maximum) matching optimizing a given measure, the definition of each considered measure and an example to better illustrate this approach. The Pseudo-Boolean Optimization models are presented in chapter 5. Experimental results and their analysis are presented in chapter 6 and the thesis concludes in chapter 7 where future research is also discussed.

2

Biological concepts

This chapter has the purpose of presenting concepts that are necessary to understand the applications in biology of the work done in this dissertation. However, these concepts are not necessary to understand the models developed here.

The chapter begins with a brief description of DNA, genes and the roles these have in organisms. Flowingly, the known methods for attaining the input for a gene order evolution study are introduced and the rearrangement scenario between the genomes of cabbage and turnip, provided by Jeffrey Palmer and Laura Herbon [29], is illustrated. Finally, the known applications of matching models and similarity measures are presented.

2.1 Organisms, proteins and DNA

Organisms are composed of individual compartments called cells and each cell of an individual contains an exact copy of the same DNA molecule (or genome). There are two types of organisms: those whose cells encapsulate their DNA in a nucleus (*eukaryotes*) and those that do not (*prokaryotes*). All multicellular organisms (like humans) are eukaryotes, while most unicellular organisms (organisms composed of only one cell, like bacteria) are prokaryotes. The genome of prokaryotes typically has a linear form, while the genome of eukaryotes is usually separated in blocks called *chromosomes* [31].

Life in organisms depends mainly on three types of molecules: DNA, RNA (*ribonucleic acid*) and proteins. Cells produce proteins and these are of utmost importance to the functioning of organisms: they are responsible for forming *enzymes* (that preform biochemical reactions), sending signals to other cells, forming *antibodies* for the *immune system*, transporting other molecules to different locations in an organism (such as, in the case of vertebrates, taking oxygen from the lungs to other organs), forming the body's major components (such as the *keratin* in our skin) and otherwise preforming the actual work in cells [20]. However, all the information on which proteins can be produced by the cells of an individual is contained in that individual's DNA molecule.

DNA, or *deoxyribonucleic acid*, is a long molecule consisting of four types of nitrogenous bases: adenine (A), thymine (T), guanine (G) and cytosine (C). This molecule has a double helical structure

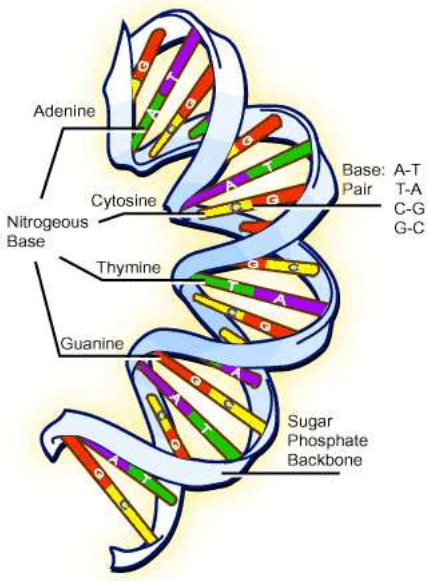


Figure 2.1: The double helical structure of DNA (figure taken from `http://www.scq.ubc.ca/wp-content/dna.gif`).

consisting of two strands (or polymers). Both strands are linear chains of nitrogenous bases, held together in a sugar phosphate backbone. Each base A, T, G and C in one strand, is held together by hydrogen bonds to its complementer T, A, C and G (respectively) in the other strand, according to the specific base pairings A↔T and C↔G. For example, one segment of DNA could be the one presented in figure 2.1.

The production of a protein in a cell is a complex process in which DNA and RNA play a fundamental role. This process is referred to as the central dogma of molecular biology and consists of two steps: the transcription of a DNA segment into a messenger RNA molecule (mRNA), followed by the translation of that mRNA molecule to a protein.

2.2 Genes, transcription and translation

Proteins are linear chains of amino acids and genes are segments of DNA that encode proteins. In DNA, each sequence of three bases (or codon) encodes an amino acid, but not all segments of DNA have the function of encoding proteins. In a simplified perspective, what distinguishes genes from non-coding DNA segments is the presence of two specific codons in a gene's beginning and end: a start codon (TAC, that also encodes the amino acid Methionine) indicating where the protein encoding starts and a stop codon (ATT, ATC, or ACT) indicating where the protein encoding ends. Moreover, for each gene, this encoding takes place in only one of the DNA strands. The two strands of DNA are denoted as $5' \rightarrow 3'$ and $3' \rightarrow 5'$, and have opposite reading directions in what concerns the transcription of a

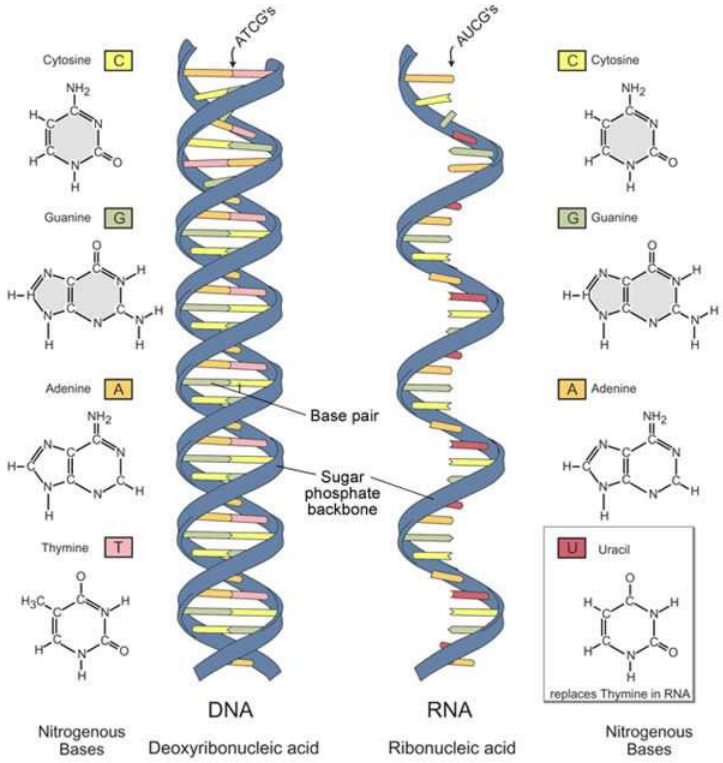


Figure 2.2: RNA structure compared to the structure of DNA (figure taken from `http://www.algosobre.com.br/images/stories/biologia/dna_rna.jpg`).

gene. This confers to genes a transcriptional reading direction in order to specify which strand is read in the process of transcription. A gene whose encoded protein lies in the strand $5' \rightarrow 3'$ is said to have a positive transcriptional reading direction. Otherwise, it is said to have a negative transcriptional reading direction.

RNA is very similar to DNA, differing mainly in being a single stranded molecule, composed by the nitrogenous bases A, C, G and uracil (U), rather than thymine (T). Figure 2.2 compares the structure of RNA to the structure of DNA. The transcription of a gene is done by a molecular complex, named RNA *polymerase*, that reads the gene in its transcriptional reading direction, and produces, base by base, a mRNA molecule (messenger RNA). This mRNA molecule is a template of the original DNA segment in the sense that it follows the pairing rules of T→A, A→U, C→G and G→C. However, this is only verified in the case of prokariotes; for that in eukariotes the RNA template of the gene further undergoes the process of splicing to form the mRNA molecule [20].

The constructed mRNA molecule then undergoes the process of translation, where it is attracted by large molecular complexes known as ribosomes that read its consecutive codons to determine which amino acid is added at each step. As each codon of the mRNA molecule is being read, a different type of RNA molecules, called transfer RNA molecules (tRNA), are used to identify the amino acid encoded by that codon (according to the mapping presented in table 2.1, also know as the genetic code) and

codon	protein	codon	protein	codon	protein	codon	protein
UUU	F	UCU	S	UAU	Y	UGU	C
UUC	F	UCC	S	UAC	Y	UGC	C
UUA	L	UCA	S	UAA	Stop	UGA	Stop
UUG	L	UCG	S	UAG	Stop	UGG	W
CUU	L	CCU	P	CAU	H	CGU	R
CUC	L	CCC	P	CAC	H	CGC	R
CUA	L	CCA	P	CAA	Q	CGA	R
CUG	L	CCG	P	CAG	Q	CGG	R
AUU	I	ACU	T	AAU	N	AGU	S
AUC	I	ACC	T	AAC	N	AGC	S
AUA	I	ACA	T	AAA	K	AGA	R
AUG	M,Start	ACG	T	AAG	K	AGG	R
GUU	V	GCU	A	GAU	D	GGU	G
GUC	V	GCC	A	GAC	D	GGC	G
GUA	V	GCA	A	GAA	E	GGA	G
GUG	V	GCG	A	GAG	E	GGG	G

A = Alanine	Q = Glutamine	L = Leucine	S = Serine
R = Arginine	E = Glutamate	K = Lysine	T = Threonine
N = Asparagine	G = Glycine	M = Methionine	W = Tryptophan
D = Aspartate	H = Histidine	F = Phenylalanine	Y = Tyrosine
C = Cysteine	I = Isoleucine	P = Proline	V = Valine

Table 2.1: The genetic code.

make it available for the ribosomes to continue translation. The produced protein will always start with the amino acid Methionine (encoded by the start codon AUG) and translation ends once a stop codon (UAA, UAG or UGA) is read. Figure 2.3 illustrates the transcription and translation of a (fictitious) gene with a positive reading direction.

2.3 Genome evolution and gene order representation

There are several processes by which genomes may evolve. Some are responsible for the evolution of individual genes and concern only point mutations in the DNA sequence, such as the insertion of a base somewhere in the DNA sequence, the deletion of a base or the substitution of a given base by another one. Others, known as rearrangement events, act more globally and affect the order by which genes occur in one or more chromosomes. Also, there is the case of the duplication of a genomic segment.

The most common rearrangement event in the evolution of unichromosomal genomes are reversals. A reversal acts on a segment of DNA by inverting the order by which genes occur in that segment and by conferring to each affected gene an opposite transcriptional reading direction.

In order to proceed with a gene order evolution study between two genomes, it is necessary one of two things. One possibility is to know the locations of genes in each genome and have some idea on which genes in both genomes can be considered equal (more precisely, homologous). Another possibility

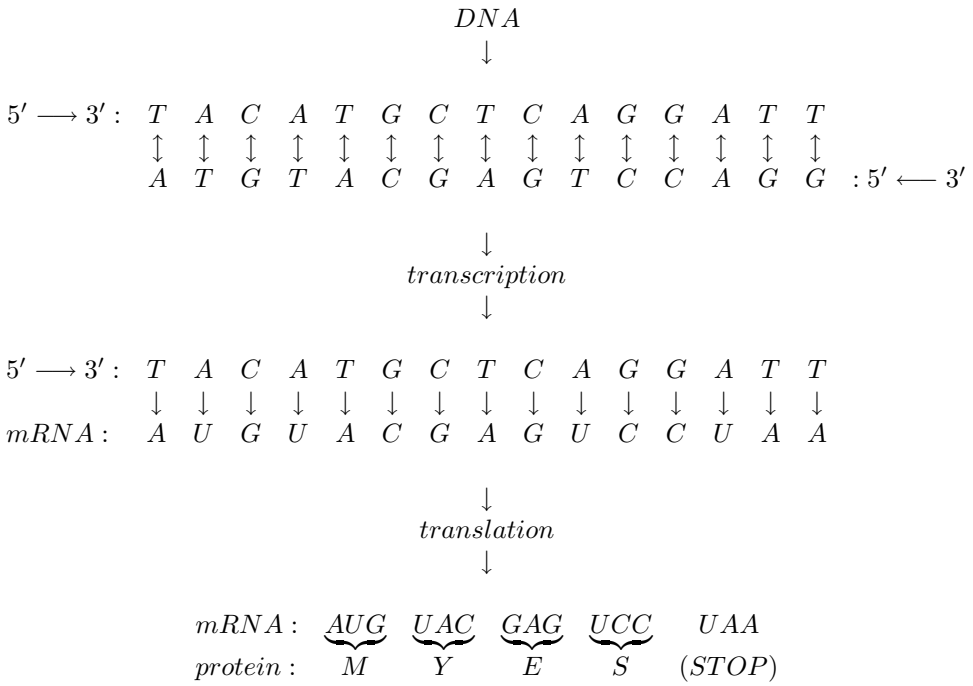


Figure 2.3: Transcription and translation of a gene with a positive transcriptional reading direction.

is to at least be able to see both genomes as a different order of conserved blocks, in which case a genome rearrangement study can take place [9, 31].

In the first gene order evolution studies, information on gene order in two genomes was derived from comparative physical maps [29, 31]. Comparative physical maps are obtained with biological laboratory methods and establish a correspondence between conserved segments of DNA in two genomes [31]. This information allows to represent two genomes as a different order of directed conserved blocks, which are groups of consecutive DNA segments that occur in the same, or reverse, order in two genomes: if the segments in a conserved block appear in the same order in the two genomes, both blocks will have the same direction; if the segments appear in the reverse order in one of the genomes, the two blocks will have opposite directions. In this representation, a reversal is modeled as an event that operates on groups of consecutive conserved blocks by reverting the order by which those blocks occur and by reverting the direction of each affected block: a reversal that affects a conserved block will revert the order by which the segments in that block occur and consequently that block will have its direction inverted as well.

When Jeffrey Palmer and Laura Herbon compared the mitochondrial genomes of cabbage and turnip [29], they used this representation for the two genomes and proceeded to a pen and pencil analysis to find the shortest number of reversals necessary to transform one genome into the other. The rearrangement scenario they presented is shown in figure 2.4.

In recent studies [38, 9, 7, 24, 25, 14, 2, 1], the information on gene order in two genomes has been derived with the use of sequence alignment tools. Sequence alignment seeks to estimate how similar

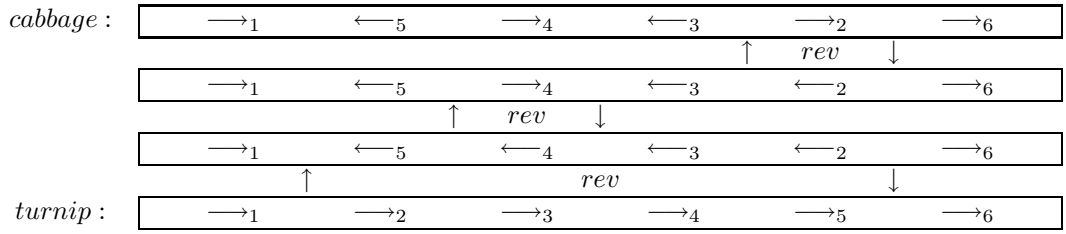


Figure 2.4: A sequence of three reversals transforming the order of six directed conserved blocks in the mitochondrial genome of cabbage into the order of these in the mitochondrial genome of turnip.

two genes are when regarded as sequences [20]. The idea is that a similarity between the sequences of nitrogenous bases that compose two genes (or between the sequences of amino acids that compose the proteins they encode) will represent a similarity in their function, i.e., a similarity in the function of the proteins they encode. If the similarity in the sequences is high enough, one might infer that the two genes are *homologous*, i.e., that they both have evolved from a common ancestor gene.

Methods have been devised that use sequence alignment tools to derive homologies between the whole set of genes contained in two or more genomes [7, 24, 25]. The idea is to partition the original set of genes into gene families, where each family is supposed to represent a group of homologous genes. What is obtained is a representation of genomes as signed sequences over an alphabet of gene families: a label is assigned to each gene family. Every gene is represented with the label assigned to the gene family it belongs to and is given a + or − sign that stands for its transcriptional reading direction. In this representation, genes that belong to the same gene family may differ its in sequence but are considered equal.

These methods, in contrast to the ones used to obtain comparative physical maps and represent genomes in terms of conserved blocks [31], are purely computational and require no laboratory work. They do require the location of genes in both genomes to be known. However, this is not a problem when dealing with two fully sequenced genomes because gene finding in a fully sequenced genome is a well known problem in bioinformatics that can be efficiently solved [31, 20].

2.4 Applications of matching models and similarity measures

The type of gene order output provided by sequence alignment tools can be analyzed with the use of matching models and similarity measures. This approach provides gene order evolutionary distances between genomes and it has been used, likewise the approaches on genome rearrangements, to compute phylogenetic trees for sets of genomes [38, 7]. These trees represent possible evolutionary scenarios between genomes that are believed to share a common ancestor genome: nodes represent genomes and each node with descendants represents the most recent common ancestor genome of its descendants. Another application of this approach has been the identification of orthologs [14] and of ancestral homologs [33] in two genomes containing duplicate genes, i.e, several homologous genes.

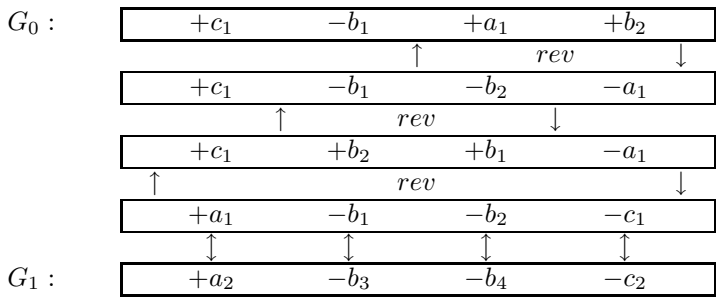


Figure 2.5: A scenario of gene order evolution, in terms of reversals, between genomes $G_0 = +c_1 - b_1 + a_1 + b_2$ and $G_1 = +a_2 - b_3 - b_4 - c_2$.

Homologous genes in two genomes can be divided into two types: orthologs, which are genes that evolved (through point mutations) from the same gene in the last common ancestor genome of the two genomes, and paralogs, which are genes that were duplicated from a single gene in the same genome (due, for instance, to the duplication of a genomic segment) [14, 9]. Distinguishing orthologous genes from paralogous genes and finding which pairs of homologous genes in two genomes can be considered orthologous has become a fundamental problem in computational biology [14]. Also, it is sometimes appropriate to search a gene family for its true exemplar gene (or ancestral homolog gene) in each genome, i.e., the gene that best reflects the original position of the ancestral gene that gave birth to that gene family [33].

When a new gene is found (for instance, as a result of the sequencing of a new genome), there is usually no idea about its function. A common approach to inferring the function of a new gene is to search for similarities between genes of known functions in other genomes, using sequence alignment tools. The idea is the following: if among the genes of a well studied genome exactly one gene shows a very high similarity with the new gene, then the function of that gene will be a good indicator of the function of the new gene [20, 31]. However, in the presence of duplicate genes in a genome (i.e., when a genome has several homologous genes), there can be the case when homology search in that genome, using sequence alignment tools, will point out several highly similar genes and will not be able to further differentiate between them [14]. In such cases, what is desired to gain knowledge on the new gene's function is to identify which of those genes is the ortholog pair of the new gene. Orthologous genes are typically functional counterparts in different species [14], and a gene order evolution study can help in the establishment of ortholog pairs in two genomes.

For example consider that the genes in a new genome G_0 and in an annotated genome G_1 have been divided into three gene families $a = \{a_1, a_2\}$, $b = \{b_1, b_2, b_3, b_4\}$ and $c = \{c_1, c_2\}$ and that these genes occur in the two genomes in the following order: $G_0 = +c_1 - b_1 + a_1 + b_2$ and $G_1 = +a_2 - b_3 - b_4 - c_2$. Any scenario of gene order evolution (in terms of reversals) between these genomes requires at least three reversals. The scenario illustrated in figure 2.5 consists of three reversals and suggests that b_1 and b_3, and also b_2 and b_4 are orthologous pairs. Moreover, it could be the case that b_1 and b_4 have more similar sequences than b_1 and b_3 to b_4, but that this similarity does not reflect an evolutionary relation

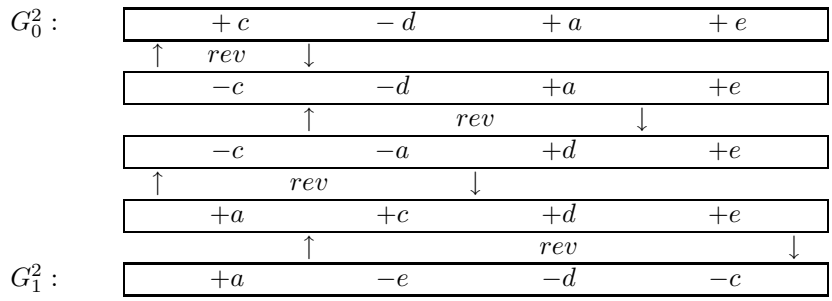


Figure 2.6: A scenario of gene order evolution, in terms of reversals, between genomes $G_0^2 = +c-d+a+e$ and $G_1^2 = +a-e-d-c$.

between b_1 and b_4. Therefore, gene order evolution studies can present evolutionary evidence, for the establishment of orthologs, that can not be found with the study of local mutations in genes [14].

This example can also be analyzed with the use of matching models and similarity measures. In this case, what is required is to find a maximum matching that minimizes the reversal distance between G_0 and G_1. A matching can be seen as a way to describe an assignment between homologous pairs of genes in two genomes. Maximum matchings assign, for each gene family, as many homologous pairs as possible. There are only two possible maximum matchings between G_0 and G_1: a first one assigning b_1 to b_3 and b_2 to b_4, and a second one assigning b_1 to b_4 and b_2 to b_3 (both assign a_1 to a_2 and c_1 to c_2). By considering matched genes in the two genomes to be equal but distinct from the remaining genes, each matching induces a new order of genes in the two genomes. In this case, the two new orders would be: $G_0^1 = +c-d+a+e$ and $G_1^1 = +a-d-e-c$ for the first matching (b_1 and b_3 are renamed to d, b_2 and b_4 are renamed to e), and $G_0^2 = +c-d+a+e$ and $G_1^2 = +a-e-d-c$ for the second matching (b_1 and b_4 are renamed to d, b_2 and b_3 are renamed to e). Since gene order evolution is being measured in terms of reversals and G_0^1 requires three reversals to be transformed into G_1^1 (the scenario presented in figure 2.5) but G_0^2 requires four reversals to be transformed into G_1^2 (figure 2.6), the optimum matching for this example is the one that assigns b_1 to b_3 and b_2 to b_4. It is said, in this case, that the reversal distance between G_0 and G_1 under a maximum matching model equals three. These results were obtained using the tool GRIMM (`http://grimm.ucsd.edu/GRIMM/`).

The key idea to distinguish orthologous from paralogous genes in two genomes is that orthologous genes should have been less displaced than paralogous genes during the divergence of two genomes [14, 33]. Therefore, finding a matching that optimizes some similarity measure will point out, for each gene family, which pairs of genes have best conserved their relative positioning in the two genomes. The previous example illustrates how the use of matching models and similarity measures can help in a better identification of orthologs [14]. Also, by considering exemplar matchings (i.e., matchings that assign, for each gene family, exactly one homolog pair), this approach can help with the identification of ancestral homologs by finding, for each gene family, which pair of homologous genes have best conserved their positioning in two genomes [33].

3

Genome rearrangements

This chapter presents an overview of the main approaches developed in the context of genome rearrangements. The purpose of these approaches is to compare two genomes, constituted by the same group of distinct genes, in terms of gene order evolution. Each approach models genomes in a different way and considers different types of gene order rearranging events. However, the aim of all the approaches presented in this chapter is to find a minimum number of rearranging operations necessary to transform the order of genes in one genome into the order of genes in the other genome.

The first approach models unichromosomal genomes as (unsigned) permutations and considers reversals as the only possible rearranging event [23]. The second approach mimics the first but considers also the information on gene transcriptional direction [4]. The third and last approach deals with the case of multichromosomal genomes and allows more complex rearranging operations that affect one or more chromosomes [21].

The examples presented for each approach were obtained using the tool GRIMM (available from `http://grimm.ucsd.edu/GRIMM/`).

3.1 Unsigned permutations

The first approach models unichromosomal genomes as (unsigned) permutations of the elements of a finite set of genes. The idea is to use permutations to represent the order by which genes occur in each genome and search for the minimum number of operations needed to transform one permutation into the other. These operations are particular cases of permutations that model the rearrangement event of the reversal of a genomic segment, which is a common evolutionary process in the evolution of unichromosomal genomes (see section 2.3).

If two genomes are constituted by n distinct genes, then, by enumerating each gene from 1 to n, the two genomes can be seen as a different order of the elements of the set $\{1, 2, \ldots, n\}$. Therefore, each genome can be modeled as a permutation of n elements, representing the order by which genes occur in it.

Definition 3.1.1 (permutations). A permutation of the elements of a set X is a bijective function

$f : X \longrightarrow X$. A permutation f of the elements of the set $\{1, 2, \ldots, n\}$ can be represented by a $2 \times n$ matrix:

$$f = \begin{pmatrix} 1 & 2 & \cdots & n \\ f(1) & f(2) & \cdots & f(n) \end{pmatrix}$$

or, in a shorter way, by $f = f(1)f(2)\cdots f(n)$, or simply by $f = f_1 f_2 \cdots f_n$, where f_i denotes $f(i)$.

S_n denotes the set of all permutations of the set $\{1, 2, \ldots, n\}$. The composition of two permutations $\pi, \sigma \in S_n$ is defined as the normal composition of functions and results in a new permutation $\pi \circ \sigma = \pi(\sigma(1)) \cdots \pi(\sigma(n))$ in S_n. S_n forms a group under composition, known as the symmetric group (of $\{1, 2, \ldots, n\}$) with neutral element $I_n = 1 \cdots n$ (the identity permutation). The following properties are either direct or simple consequences of $(S_n, \circ)$ being a group[1]:

- (*neutral element* I_n) for every permutations $\pi \in S_n$:

$$\pi \circ I_n = I_n \circ \pi = \pi \tag{3.1}$$

- (*associativity of* $\circ$) for all permutations $\pi, \sigma, \gamma \in S_n$:

$$(\pi \circ \sigma) \circ \gamma = \pi \circ (\sigma \circ \gamma) \tag{3.2}$$

- (*existence and unicity of inverse*) every permutation $\pi \in S_n$ has a unique inverse in S_n, i.e., for any given permutation $\pi \in S_n$, there exists one and only one permutation in S_n, denoted as π^{-1}, verifying that:

$$\pi^{-1} \circ \pi = \pi \circ \pi^{-1} = I_n \tag{3.3}$$

- for all permutations $\pi, \sigma, \gamma \in S_n$:

$$\gamma \circ \pi = \gamma \circ \sigma \Leftrightarrow \pi = \sigma \Leftrightarrow \pi \circ \gamma = \sigma \circ \gamma \tag{3.4}$$

Definition 3.1.2 (reversal). Consider $i, j, n \in \mathbb{N}$ such that $1 \leq i \leq j \leq n$. A reversal $\rho(i, j) \in S_n$ is a permutation with the following structure:

$$\rho(i,j) \stackrel{def}{=} \begin{pmatrix} 1 & 2 & \cdots & i-1 & i & i+1 & \cdots & j-1 & j & j+1 & \cdots & n \\ 1 & 2 & \cdots & i-1 & j & j-1 & \cdots & i+1 & i & j+1 & \cdots & n \end{pmatrix}$$

A reversal $\rho(i, j) \in S_n$ can also be represented in a shorter way as:

$$\rho(i,j) = 1\ 2 \cdots (i-1)\ j\ (j-1) \cdots (i+1)\ i\ (j+1) \cdots n$$

[1] For a detailed introduction to group theory see [16].

For example, $\rho(1,1) = 1\ 2\ 3\ 4 = I_4$, $\rho(2,3) = 1\ 3\ 2\ 4$, $\rho(2,4) = 1\ 4\ 3\ 2$ and $\rho(1,4) = 4\ 3\ 2\ 1$ are reversals in S_4.

A reversal $\rho(i,j) \in S_n$, when composed on the right with a permutation $\pi \in S_n$, has the effect of reverting the order assigned by π to the elements between i and j (inclusively), i.e.:

$$\begin{aligned} \pi \circ \rho(i,j) = \quad & \pi(\rho(i,j)(1)) \cdots \pi(\rho(i,j)(i-1)) \\ & \pi(\rho(i,j)(i))\pi(\rho(i,j)(i+1)) \cdots \pi(\rho(i,j)(j-1))\pi(\rho(i,j)(j)) \\ & \pi(\rho(i,j)(j+1)) \cdots \pi(\rho(i,j)(n)) \\ = \quad & \pi(1) \cdots \pi(i-1)\pi(j)\pi(j-1) \cdots \pi(i+1)\pi(i)\pi(j+1) \cdots \pi(n) \\ = \quad & \pi_1 \cdots \pi_{i-1}\pi_j\pi_{j-1} \cdots \pi_{i+1}\pi_i\pi_{j+1} \cdots \pi_n \end{aligned} \tag{3.5}$$

For example, for a permutation $\pi = 1\ 4\ 3\ 2\ 5\ 7\ 6$ and reversal $\rho(3,6) = 1\ 2\ 6\ 5\ 4\ 3\ 7$ in S_7:

$$\pi \circ \rho(3,6) = \pi_1\pi_2\pi_6\pi_5\pi_4\pi_3\pi_7 = 1\ 4\ 7\ 5\ 2\ 3\ 6$$

The composition, on the right, of a reversal $\rho(i,j) \in S_n$ with a permutation $g \in S_n$, corresponding to the order of genes in a genome G, models the rearrangement event of the reversal of the genomic segment of G composed by the genes between positions i and j. The approach to the problem of genome rearrangements using permutations is specified in the problem of finding the reversal distance between two permutations.

Definition 3.1.3 (reversal distance). Consider two permutations $\pi, \sigma \in S_n$. The reversal distance $d(\pi, \sigma)$ between π and σ is the minimum number t for which there exists a sequence of reversals $\rho_1, \ldots, \rho_t \in S_n$ such that:

$$\pi \circ \rho_1 \circ \cdots \circ \rho_t = \sigma \tag{3.6}$$

The reversal distance problem consists in finding a sequence of $d(\pi, \sigma)$ reversals $\rho_1, \ldots, \rho_{d(\pi,\sigma)} \in S_n$ that verify equation 3.6 for two given permutations π and σ in S_n.

Let $g = g_1 \cdots g_n$ and $h = h_1 \cdots h_n$ be two permutations representing the order of genes in two genomes G and H. Finding the reversal distance between permutations g and h corresponds in practice to finding the minimum amount of evolution that must have taken place between genomes G and H, assuming that evolution has occurred only in terms of reversals. Solving the reversal distance problem is to provide one of the most parsimonious scenarios of evolution between G and H in terms of reversals.

Example 3.1.1. Consider the permutations $g = 1\ 6\ 5\ 3\ 4\ 2\ 7$ and $h = 7\ 6\ 5\ 1\ 2\ 3\ 4$ in S_7. It requires at least four reversals to transform g into h and this is attained by the sequence of reversals

$\rho(2,3), \rho(4,5), \rho(4,7), \rho(1,4)$:

$$\begin{aligned} g = &\quad 1\ 6\ 5\ 3\ 4\ 2\ 7 \\ g \circ \rho(2,3) = &\quad 1\ 5\ 6\ 3\ 4\ 2\ 7 \\ g \circ \rho(2,3) \circ \rho(4,5) = &\quad 1\ 5\ 6\ 4\ 3\ 2\ 7 \\ g \circ \rho(2,3) \circ \rho(4,5) \circ \rho(4,7) = &\quad 1\ 5\ 6\ 7\ 2\ 3\ 4 \\ g \circ \rho(2,3) \circ \rho(4,5) \circ \rho(4,7) \circ \rho(1,4) = &\quad 7\ 6\ 5\ 1\ 2\ 3\ 4\ = h \end{aligned}$$

Therefore, the sequence of reversals $\rho(2,3), \rho(4,5), \rho(4,7), \rho(1,4)$ solves the reversal distance problem between g and h and $d(g,h) = 4$.

The problem of finding the reversal distance between two permutations is equivalent to the simpler problem of sorting a permutation by reversals. This fact has led researchers to focus on the later problem.

Definition 3.1.4 (sort by reversals). Consider a permutation $\pi \in S_n$. Let $d(\pi)$ denote the minimum number t for which there exist reversals $\rho_1, \ldots, \rho_t \in S_n$ such that:

$$\pi \circ \rho_1 \circ \cdots \circ \rho_t = I_n \tag{3.7}$$

The sort by reversals problem[2] consists in finding a sequence of $d(\pi)$ reversals $\rho_1, \ldots, \rho_{d(\pi)} \in S_n$ that verify equation 3.7, for a given permutation π in S_n [3].

Proposition 3.1.1. *Let $\pi, \sigma \in S_n$. Then $d(\pi, \sigma) = d(\sigma^{-1} \circ \pi)$.*

Proof 3.1.1. Let $\pi, \sigma \in S_n$ and $\rho_1, \ldots, \rho_{d(\pi,\sigma)}$ be a sequence of reversals in S_n that solve the reversal distance problem between π and σ. Let σ^{-1} be the inverse permutation of σ in S_n. Then, equation 3.6 holds and as a consequence of equation 3.4:

$$\pi \circ \rho_1 \circ \cdots \circ \rho_{d(\pi,\sigma)} = \sigma \Leftrightarrow \sigma^{-1} \circ \pi \circ \rho_1 \circ \cdots \circ \rho_{d(\pi,\sigma)} = \sigma^{-1} \circ \sigma = I_n$$

Therefore, the sequence of reversals $\rho_1, \ldots, \rho_{d(\pi,\sigma)}$ solve the sort by reversals problem for the permutation $\sigma^{-1} \circ \pi$ and $d(\pi, \sigma) \geq d(\sigma^{-1} \circ \pi)$. Following a similar reasoning, $d(\sigma^{-1} \circ \pi) \leq d(\pi, \sigma)$ and consequently $d(\pi, \sigma) = d(\sigma^{-1} \circ \pi)$.

□

It is important to note that the sort by reversals problem has a solution for every permutation in S_n. This is a simple consequence of the fact that reversals generate the symmetric group S_n [23]. For every permutation $\pi \in S_n$, there exists a sequence of reversals $\rho_1, \ldots, \rho_t \in S_n$ that generate π (i.e., a

[2] Also known as the problem of sorting a permutation by reversals.

[3] By definition, for $\pi \in S_n$, $d(\pi) = d(\pi, I_n)$. Function d can be therefore called with one or two arguments. The meaning of d should be made clear in the context it is called for.

sequence of reversals verifying that $\rho_1 \circ \cdots \circ \rho_t = \pi$). Therefore, finding a sequence of reversals that solve the sort by reversals problem for a permutation $\pi \in S_n$ is equivalent to finding one of the shortest sequences of reversals that generates π^{-1}. If $\rho_1, \ldots, \rho_t$ is a sequence of reversals in S_n that generates π^{-1}, then $\rho_1 \circ \cdots \circ \rho_t = \pi^{-1}$ and from equation 3.4:

$$\rho_1 \circ \cdots \circ \rho_t = \pi^{-1} \Leftrightarrow \pi \circ \rho_1 \circ \cdots \circ \rho_t = \pi \circ \pi^{-1} = I_n \tag{3.8}$$

Hence, using equation 3.8 and following a similar reasoning of proof 3.1.1, $d(\pi)$ equals the smallest number t for which there exists a sequence of t reversals that generate π^{-1} and the sort by reversals problem for a permutation π always has a solution [23].

Example 3.1.2. Consider once more the two permutations $g = 1\ 6\ 5\ 3\ 4\ 2\ 7$ and $h = 7\ 6\ 5\ 1\ 2\ 3\ 4$. It requires at least four reversals to sort $h^{-1} \circ g$ by reversals, and this is attained by the sequence of reversals presented in example 3.1.1:

$$\begin{aligned}
g &= 1\ 6\ 5\ 3\ 4\ 2\ 7 \\
h^{-1} &= 4\ 5\ 6\ 7\ 3\ 2\ 1 \\
h^{-1} \circ g &= 4\ 2\ 3\ 6\ 7\ 5\ 1 \\
h^{-1} \circ g \circ \rho(2,3) &= 4\ 3\ 2\ 6\ 7\ 5\ 1 \\
h^{-1} \circ g \circ \rho(2,3) \circ \rho(4,5) &= 4\ 3\ 2\ 7\ 6\ 5\ 1 \\
h^{-1} \circ g \circ \rho(2,3) \circ \rho(4,5) \circ \rho(4,7) &= 4\ 3\ 2\ 1\ 5\ 6\ 7 \\
h^{-1} \circ g \circ \rho(2,3) \circ \rho(4,5) \circ \rho(4,7) \circ \rho(1,4) &= 1\ 2\ 3\ 4\ 5\ 6\ 7 = I_7
\end{aligned}$$

From this, $d(h^{-1} \circ g) = 4$ and the sequence of reversals $\rho(4,7), \rho(3,4), \rho(2,6), \rho(1,7)$ solves the sort by reversals problem for $h^{-1} \circ g$.

Despite the efforts put into solving the *sort by reversals problem*, an efficient algorithm for computing $d(\pi)$ remains unknown. However some important work was done in proving $d(\pi, \sigma)$ to be a good indicator of evolutionary relationships between two genomes and in establishing lower bounds for $d(\pi)$ [28, 4].

3.1.1 Reversal diameter and expected reversal distance

Two important concepts in this approach are the ones of the reversal diameter of the symmetric group S_n and of the expected sort by reversal distance of a permutation in S_n. The first concept establishes a limit for $d(\pi)$ for all permutations. The second concept establishes the expected value of $d(\pi)$ for a random permutation.

Definition 3.1.5 (reversal diameter). The reversal diameter of the symmetric group S_n is denoted as $D(n)$ and is defined as the maximum value of $d(\pi)$ over all permutations $\pi \in S_n$:

$$D(n) = \max_{\pi \in S_n} d(\pi)$$

Definition 3.1.6 (expected reversal distance). Let $\pi \in S_n$. The expected reversal distance in S_n is denoted as $E(d)$ and is defined as the average of $d(\pi)$ over all $\pi \in S_n$ [4]:

$$E(d) = \frac{1}{n!} \sum_{\pi \in S_n} d(\pi)$$

Golan conjectured that $D(n) = n - 1$, and that for $n > 2$, γ and its symmetric γ_n^{-1} were the only permutations in S_n reaching the bound of $n - 1$ [4, 31]:

$$\gamma_n = \begin{cases} 3\ 1\ 5\ 2\ 7\ 4\ \cdots\ (n-3)\ (n-5)\ (n-1)\ (n-4)\ n\ (n-2) & \text{, if n is even} \\ 3\ 1\ 5\ 2\ 7\ 4\ \cdots\ (n-6)\ (n-2)\ (n-5)\ n\ (n-3)\ (n-1) & \text{, if n is odd} \end{cases}$$

This conjecture claimed that $n-1$ was the tightest upper bound possible for $d(\pi)$. Bafna and Pevzner [4] proved Golan's conjecture and also presented a lower bound for $E(d)$.

Theorem 3.1.1 (Bafna and Pevzner [4]). $E(d) \geq (1 - \frac{4.5}{\log(n)})n$.

From this bound, one can conclude that, for high values of n, the expected reversal distance comes very close to the reversal diameter $D(n) = n - 1$ of S_n [4]. This result, combined with the fact that $d(\pi, \sigma) = d(\sigma^{-1} \circ \pi)$, means that given two random, non-related permutations, one can expect the reversal distance between them to be close to the maximum possible, this is, to the reversal diameter of the symmetric group [4]. This result shows that one can expect the reversal distance to discern permutations that are related through molecular evolution from simply random permutations.

3.1.2 Breakpoints, breakpoint graph and lower bounds

This section introduces lower bounds for the reversal distance between two permutations. These bounds are attained by presenting lower bounds for $d(\pi)$, based on features that can be derived from π. An important feature is the number of breakpoints in a permutation, a concept first introduced by Watterson [40] and later by Nadeau and Taylor [28].

Definition 3.1.7 (breakpoint). Let $\pi = \pi_1 \cdots \pi_n$ be a permutation. Extend permutation π by adding the elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$. Let $i \sim j$ denote that $|i-j| = 1$ and $i \not\sim j$ denote that $|i-j| \neq 1$. Then, a pair of elements (π_i, π_{i+1}) of π, where $0 \leq i \leq n$, is called:

1. a breakpoint if $\pi_i \not\sim \pi_{i+1}$.
2. an adjacency if $\pi_i \sim \pi_{i+1}$.

The number of breakpoints in π is denoted by $b(\pi)$.

[4] S_n has $n!$ elements.

Example 3.1.3. Consider the permutation $\pi = 2\ 3\ 1\ 4\ 6\ 5\ 7$ in S_7. Extending the permutation π with the elements $\pi_0 = 0$ and $\pi_8 = 8$, we obtain $\pi = \pi_0\pi_1\pi_2\pi_3\pi_4\pi_5\pi_6\pi_7\pi_8 = 0\ 2\ 3\ 1\ 4\ 6\ 5\ 7\ 8$. Then, $b(\pi) = 5$: the pairs of elements (π_0, π_1), (π_2, π_3), (π_3, π_4), (π_4, π_5) and (π_6, π_7) determine breakpoints; the pairs of elements (π_1, π_2), (π_5, π_6) and (π_7, π_8) determine adjacencies.

The number of breakpoints in given a permutation provides some insight on how hard it is to sort the permutation by reversals. It is plain to see that the only permutation with no breakpoints is the identity permutation [31]. When sorting a permutation $\pi \in S_n$ by reversals, the objective is to transform π into I_n, which corresponds to eliminating all breakpoints in π. Since a reversal may eliminate at most 2 breakpoints in π [31], it will take at least $\frac{b(\pi)}{2}$ reversals to eliminate all breakpoints in π. This observation provides a first lower bound for the sort by reversals problem.

Theorem 3.1.2. *Let $\pi \in S_n$, then $d(\pi) \geq \frac{b(\pi)}{2}$*

The estimation based only on the number of breakpoints of π turned out to be very inaccurate [31]. A tighter bound was later introduced by Bafna and Pevzner [4] by taking into consideration another parameter related to breakpoints that can be derived from π: the size of a maximum cycle decomposition of the breakpoint graph.

Definition 3.1.8 (breakpoint graph). The *breakpoint graph* (V, E) of a permutation $\pi = \pi_1 \cdots \pi_n \in S_n$ is an edge-bicolored graph $G(\pi)$ with $n + 2$ vertexes where:

- $V = \{\pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}\} = \{0, 1, \ldots, n, n + 1\}$
- $E = E_b \cup E_g$, where:
  - $E_b = \{(\pi_i, \pi_{i+1}) : 0 \leq i \leq n\}$ (a black line joining all vertexes by the order provided by π)
  - $E_g = \{(i, i + 1) : 0 \leq i \leq n\}$ (a gray line joining vertexes that determine adjacency in π, this is, joining vertexes by the order of the identity permutation)

For example, the breakpoint graph of permutation $\pi = 2\ 3\ 1\ 4\ 6\ 5$ in S_6, taken from [31], is presented in figure 3.1.

A vertex v in an edge-bicolored graph is said to be *balanced* if the number of edges, of one color, being incident on vertex v equals the number of edges, of the other color, incident on v. A graph G is said to be balanced if all of its vertexes are balanced. Thus, by construction, a breakpoint graph of a permutation is always a balanced graph.

A sequence of vertexes $P = x_1 \cdots x_m$ of V is called a *cycle* in a graph $G = (V, E)$ if $(x_i, x_{i+1}) \in E$ for $1 \leq i \leq m - 1$ and $x_1 = x_m$. A cycle in an edge-colored graph G is called *alternating* if the colors of every two consecutive edges (x_i, x_{i+1}) and (x_{i+1}, x_{i+2}) used in the cycle are distinct (including the pair (x_{m-1}, x_m) and (x_1, x_2)). Two cycles are said to be edge-disjoint if they have no edges in common.

If an edge-bicolored graph is balanced then it has a cycle decomposition into edge-disjoint alternating cycles [31], i.e., there exist alternating cycles $P_1 \cdots P_k$, edge-disjoint two by two, that partition the edge

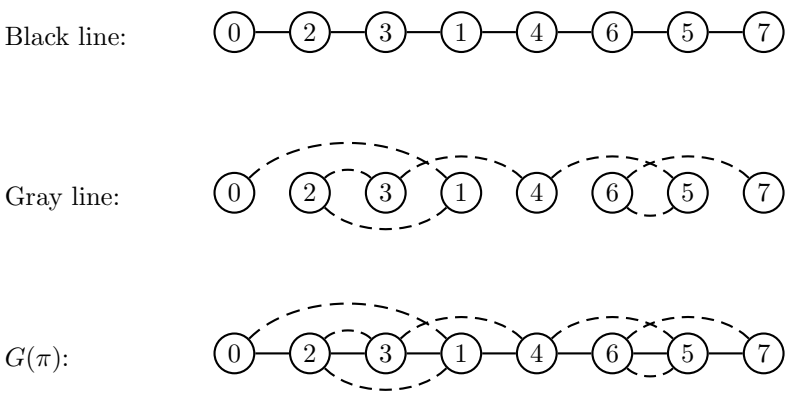


Figure 3.1: The breakpoint graph $G(\pi)$ of the permutation $\pi = 2\ 3\ 1\ 4\ 6\ 5$ in S_6.

set of the graph. The feature of interest here is the maximum number t for which there exists a cycle decomposition of $G(\pi)$ into t edge-disjoint alternating cycles.

Definition 3.1.9 ($c(\pi)$). Let $\pi \in S_n$. $c(\pi)$ is the maximum number t for which there exists a cycle decomposition of $G(\pi)$ into t edge-disjoint alternating cycles.

Example 3.1.4. The breakpoint graph $G(\pi)$ of $\pi = 2\ 3\ 1\ 4\ 6\ 5 \in S_6$ can be decomposed into a maximum number of $c(\pi) = 4$ edge-disjoint alternating cycles. Cycles $P_1 = 2\ 3\ 2$, $P_2 = 6\ 5\ 6$, $P_3 = 4\ 5\ 7\ 6\ 4$ and $P_4 = 0\ 1\ 3\ 4\ 1\ 2\ 0$ form such a decomposition of $G(\pi)$ (see figure 3.2).

Cycle decompositions play an important role in estimating the *sort by reversals distance* of a permutation. Bafna and Pevzner [4] noticed that applying a reversal to a permutation π increases $c(\pi)$ at most by one. This result was proved with the following theorem.

Theorem 3.1.3. *For every permutation π and reversal ρ in S_n, $c(\pi\rho) - c(\pi) \leq 1$.*

It can also be shown that I_n is the only permutation in S_n that reaches the maximum value of $c(I_n) = n + 1$ [4]. Once again, sorting by reversals seeks to transform a permutation $\pi \in S_n$ into I_n by applying successive reversals to π. From this result, and taking into account the previous theorem, it will take at least $c(I_n) - c(\pi) = n + 1 - c(\pi)$ reversals to turn π into I_n. This provides a second lower bound for $d(\pi)$.

Theorem 3.1.4. *Let $\pi \in S_n$, then $d(\pi) \geq n + 1 - c(\pi)$.*

For some biological examples it holds that $d(\pi) = n + 1 - c(\pi)$ [12, 23]. Unfortunately, finding a decomposition of $G(\pi)$ with a maximum number of edge-disjoint alternating cycles is a NP-hard problem [11].

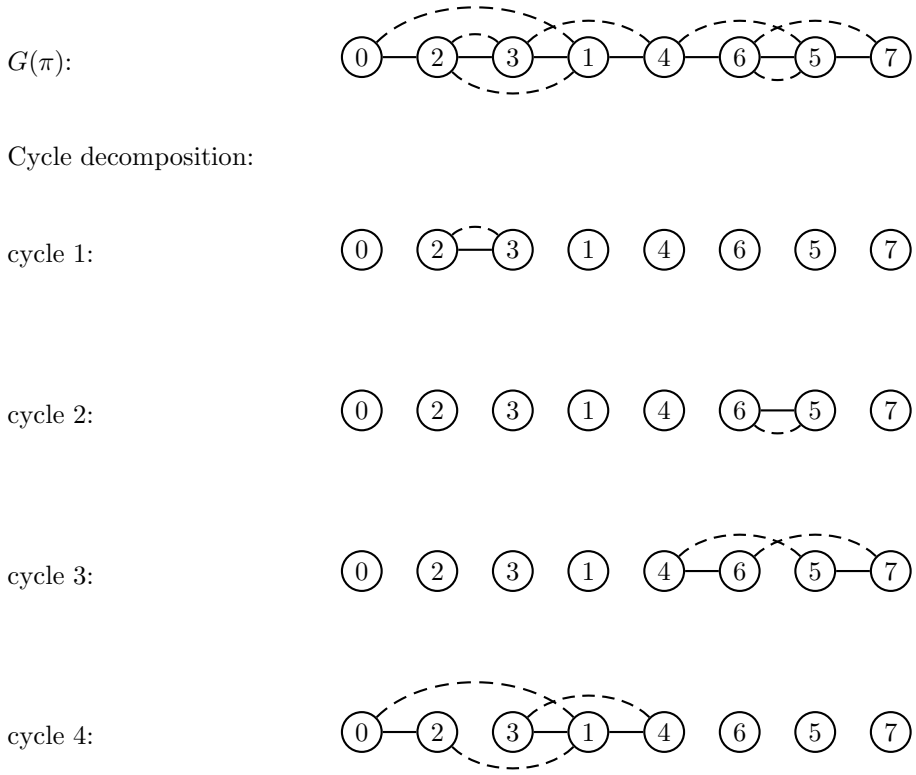


Figure 3.2: A cycle decomposition (taken from [31]) of the breakpoint graph $G(\pi)$ of the permutation $\pi = 2\ 3\ 1\ 4\ 6\ 5$ in S_6, into $c(\pi) = 4$ edge-disjoint cycles.

3.1.3 Algorithms and complexity

Caprara proved the sort by reversals problem to be NP-Hard [11], based on the strong relationship between the sort by reversals problem and the problem of, given a breakpoint graph $G(\pi)$ of some permutation, finding a decomposition of $G(\pi)$ into $c(\pi)$ edge-disjoint alternating cycles. Due to this result, many researchers have tried to devise a practical approximation algorithm for sorting by reversals.

The algorithmic study of the sort by reversals problem started with Watterson *et al.* [40]. The proposed algorithm consisted in successively bringing, by increasing order, elements $1, 2, \ldots, n$ into place. Formally, at step i, perform reversal $\rho(i, \pi^{-1}(i))$ if $\pi(i) \neq i$. This procedure sorts any permutation $\pi \in S_n$ in at most $n - 1$ steps. Waterson's algorithm is worst-case optimal since $d(n) = n - 1$ but it may perform arbitrarily poor. It would take $n - 1$ reversals to sort permutation $\pi = n\ 1\ 2 \ldots\ n - 1$ when optimally it takes only two:

$$n\ 1\ 2 \cdots\ (n-1) \xrightarrow{\rho(2,n)} n\ (n-1)\ (n-2) \cdots\ 1 \xrightarrow{\rho(1,n)} 1\ 2\ 3 \cdots\ n$$

Kececioglu and Sankoff [23] presented a greedy algorithm that sorted a permutation π in at most $b(\pi)$ reversals. Sorting a permutation π by reversals corresponds to eliminating all breakpoints in π. The idea is to chose at each step a reversal that will decrease $b(\pi)$ as much as possible. This algorithm

sorts a permutation $\pi \in S_n$ in $O(n^2)$ time and using no more than $2 \cdot d(\pi)$ reversals[5], meaning also that it is a 2-*approximation algorithm* (because it always returns a solution that is, at most, 2 times larger than the optimal). Later, Bafna and Pevzner [4] came up with a $\frac{7}{4}$-*approximation algorithm*.

Kececioglu and Sankoff [23] also introduced a branch-and-bound exact algorithm that sorts a permutation $\pi \in S_n$ by reversals in $O(mL(n,n))$ time. They define m as the size of the branch-and-bound search tree[6], and $L(n,n)$ as the time needed to solve a linear program of n variables and n constrains[7].

3.2 Signed permutations

A gene is a segment of DNA sequence that encodes a protein and has a reading direction in what concerns the process of transcription (see section 2.2). As a genome evolutionary process, a *reversal* also changes the reading direction of genes in the genome segment it reverses. Unsigned permutations model gene order in a genome. However, when comparing two genomes, information may be available on gene transcriptional reading direction, as well as order. The information on gene transcriptional reading direction can be encoded by associating a sign with each element: "+" for positive transcriptional reading direction, and "−" for negative transcriptional reading direction.

Genomes can be modeled as signed permutations of the set $\{1, \ldots, n\}$. The resulting problem consists in finding the minimum number of (signed) reversals needed to sort a signed permutation in such a way that every element ends with a positive sign. Fortunately, in this more biologically relevant approach, sorting a permutation by reversals becomes a trivial problem.

In order to better illustrate this problem, we present definitions for signed permutations, signed reversals and the signed versions of the reversal distance problem and the sort by reversals problem. Moreover, we define the composition for signed permutations and prove that the set of all signed permutations forms a group under this operation.

In section 3.2.1, a transformation will be defined allowing to see signed permutations as a particular class of (unsigned) permutations. With the use of this transformation, the signed versions of the reversal distance problem and the sort by reversals problem will be reduced to the unsigned versions where only a restricted class of reversals are allowed. We prove this function to be an injective group homomorphism in order to provide a more clear reasoning on the proofs that lead to this result.

Definition 3.2.1 (signed permutation). A signed permutation $\overrightarrow{\pi}$ of the elements of the set $\{1, \ldots, n\}$ is a tuple $\langle \pi, s \rangle$, where π is a permutation in S_n and $s : \{1, \ldots, n\} \longrightarrow \{+, -\}$ is a function that assigns, to the element in position i, a positive or negative sign. A signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ of the elements of the set $\{1, \ldots, n\}$ can be generally represented by the following sequence:

$$\overrightarrow{\pi} = s(1)\pi(1) \cdots s(n)\pi(n)$$

[5] A simple corollary of theorem 3.1.2 [23].
[6] Hence, exponential on the number of variables n.
[7] That can be polynomial on the number of variables n.

For example, the signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ of the elements of the set $\{1, \ldots, 4\}$, where $\pi = 4\ 1\ 3\ 2$ and $s(1) = s(3) = +$ and $s(2) = s(4) = -$, can be represented as $\overrightarrow{\pi} = +4 - 1 + 3 - 2$.

$\overrightarrow{S_n}$ denotes the set of all signed permutations of the set $\{1, \ldots, n\}$. We define the composition of two signed permutations as the binary operation $\odot : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \overrightarrow{S_n}$ that given two signed permutations $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi_2} = \langle \pi_2, s_2 \rangle$ in $\overrightarrow{S_n}$ results in a signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ in $\overrightarrow{S_n}$, denoted as $\overrightarrow{\pi_1} \odot \overrightarrow{\pi_2}$, where $\pi = \pi_1 \circ \pi_2$ is the permutation in S_n resulting of the normal composition π_1 and π_2 and:

$$s(i) = \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

Example 3.2.1. Consider the signed permutations $\overrightarrow{\pi_1} = -1 + 3 - 4 - 2$ and $\overrightarrow{\pi_2} = +4 - 2 - 3 + 1$ in $\overrightarrow{S_4}$. Then, the composition of $\overrightarrow{\pi_1}$ with $\overrightarrow{\pi_2}$ results in the signed permutation $\overrightarrow{\pi_3} = \overrightarrow{\pi_1} \odot \overrightarrow{\pi_2} = -2 - 3 + 4 - 1$.

Proposition 3.2.1. *$(\overrightarrow{S_n}, \odot)$ is a group with neutral element $\overrightarrow{I_n} = +1 \cdots + n$, i.e., it holds that:*

1. *(neutral element $\overrightarrow{I_n}$) for every signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$:*

$$\overrightarrow{\pi} \odot \overrightarrow{I_n} = \overrightarrow{I_n} \odot \overrightarrow{\pi} = \overrightarrow{\pi} \tag{3.9}$$

2. *(associativity of $\odot$) for all signed permutations $\overrightarrow{\pi}, \overrightarrow{\sigma}, \overrightarrow{\gamma} \in \overrightarrow{S_n}$:*

$$(\overrightarrow{\pi} \odot \overrightarrow{\sigma}) \odot \overrightarrow{\gamma} = \overrightarrow{\pi} \odot (\overrightarrow{\sigma} \odot \overrightarrow{\gamma}) \tag{3.10}$$

3. *(existence of inverse) every signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$ has a unique inverse in $\overrightarrow{S_n}$, i.e., for any given signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$, there exists one and only one permutation in $\vec{S_n}$, denoted as $\overrightarrow{\pi}^{-1}$, verifying that:*

$$\overrightarrow{\pi}^{-1} \odot \overrightarrow{\pi} = \overrightarrow{\pi} \odot \overrightarrow{\pi}^{-1} = \overrightarrow{I_n} \tag{3.11}$$

Moreover, for a signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ in $\overrightarrow{S_n}$, $\overrightarrow{\pi}^{-1} = \langle \pi^{-1}, t \rangle$, where, for $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $t(i) = s(\pi^{-1}(i))$.

Proof 3.2.1. In annex (chapter 8). □

As a consequence of $(\overrightarrow{S_n}, \odot)$ being a group it holds that for all signed permutations $\overrightarrow{\pi}, \overrightarrow{\sigma}, \overrightarrow{\gamma} \in \overrightarrow{S_n}$ [8]:

$$\overrightarrow{\gamma} \odot \overrightarrow{\pi} = \overrightarrow{\gamma} \odot \overrightarrow{\sigma} \Leftrightarrow \overrightarrow{\pi} = \overrightarrow{\sigma} \Leftrightarrow \overrightarrow{\pi} \odot \overrightarrow{\gamma} = \overrightarrow{\sigma} \odot \overrightarrow{\gamma} \tag{3.12}$$

Definition 3.2.2 (signed reversal). Consider $i, j, n \in \mathbb{N}$ such that $1 \leq i \leq j \leq n$. A signed reversal

[8] For a detailed introduction to group theory see [16].

$\overrightarrow{\rho(i,j)} \in \overrightarrow{S_n}$ is a signed permutation $\langle \pi, s \rangle$ with the following structure:

$$\pi = \rho(i,j) \qquad s(k) = \begin{cases} + & \text{, if } k < i \text{ or } k > j \\ - & \text{, if } i \leq k \leq j \end{cases}$$

A signed reversal $\overrightarrow{\rho(i,j)} \in \overrightarrow{S_n}$ can be represented as follows:

$$\overrightarrow{\rho(i,j)} = +1 \cdots + (i-1) - j - (j-1) \cdots - (i+1) - i + (j+1) \cdots + n$$

For example, $\overrightarrow{\rho(1,1)} = -1 + 2 + 3 + 4$, $\overrightarrow{\rho(2,3)} = +1 - 3 - 2 + 4$, $\overrightarrow{\rho(2,4)} = +1 - 4 - 3 - 2$ and $\overrightarrow{\rho(1,4)} = -4 - 3 - 2 - 1$ are reversals in $\vec{S_4}$.

It is important to retain from the definition presented here for the composition of two signed permutations that a signed reversal $\overrightarrow{\rho(i,j)} \in \overrightarrow{S_n}$, when composed on the right with a signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ in $\overrightarrow{S_n}$, has the effect of reverting the order assigned by $\overrightarrow{\pi}$ to the elements between i and j, inclusively, as well as the signs of those elements, i.e.:

$$\begin{aligned} \overrightarrow{\pi} \odot \overrightarrow{\rho(i,j)} = \quad & s(1)\pi(1) \cdots s(i-1)\pi(i-1) \\ & \overline{s(j)}\pi(j)\overline{s(j-1)}\pi(j-1) \cdots \overline{s(i+1)}\pi(i+1)\overline{s(i)}\pi(i) \\ & s(j+1)\pi(j+1) \cdots s(n)\pi(n) \end{aligned} \tag{3.13}$$

Where, for $x \in \{+, -\}$:

$$\overline{x} = \begin{cases} + & \text{, if } x = - \\ - & \text{, if } x = + \end{cases}$$

For example, for a signed permutation $\overrightarrow{\pi} = +1 + 4 - 3 + 2 + 5 - 7 - 6$ and reversal $\overrightarrow{\rho(3,6)} = +1 + 2 - 6 - 5 - 4 - 3 + 7$ in $\overrightarrow{S_7}$:

$$\begin{aligned} \overrightarrow{\pi} \odot \overrightarrow{\rho(3,6)} \quad & = s(1)\pi(1)s(2)\pi(2)\overline{s(6)}\pi(6)\overline{s(5)}\pi(5)\overline{s(4)}\pi(4)\overline{s(3)}\pi(3)s(7)\pi(7) = \\ & = +1 + 4 + 7 - 5 - 2 + 3 - 6 \end{aligned}$$

The composition, on the right, of a signed reversal $\overrightarrow{\rho(i,j)} \in \overrightarrow{S_n}$ with a signed permutation $\overrightarrow{g} \in \overrightarrow{S_n}$, representing the order of genes and their transcriptional reading direction in a genome G, models the rearrangement event of the reversal of the genomic segment of G composed by the genes between positions i and j, taking in to account that each of those genes will have their transcriptional reading direction inverted. The approach to the problem of genome rearrangements using signed permutations is specified in the problem of finding the signed reversal distance between two signed permutations.

Definition 3.2.3 (signed reversal distance). Consider two signed permutations $\overrightarrow{\pi}, \overrightarrow{\sigma} \in \overrightarrow{S_n}$. The signed reversal distance $d(\overrightarrow{\pi}, \overrightarrow{\sigma})$ between $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ is the minimum number t for which there exists a

sequence of signed reversals $\overrightarrow{\rho}_1, \cdots, \overrightarrow{\rho}_t \in \overrightarrow{S_n}$ such that:

$$\overrightarrow{\pi} \odot \overrightarrow{\rho}_1 \odot \cdots \odot \overrightarrow{\rho}_t = \overrightarrow{\sigma} \tag{3.14}$$

The signed reversal distance problem consists in finding a sequence of $d(\overrightarrow{\pi}, \overrightarrow{\sigma})$ signed reversals $\overrightarrow{\rho}_1, \cdots, \overrightarrow{\rho}_{d(\overrightarrow{\pi},\overrightarrow{\sigma})}$ that verify equation 3.14 for two given signed permutations $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ in $\overrightarrow{S_n}$.

Example 3.2.2. Consider the signed permutations $\overrightarrow{g} = +1+6-5-3+4-7-2$ and $\overrightarrow{h} = +7-6+5-1-2+3-4$ in $\overrightarrow{S_7}$. It requires at least five signed reversals to transform $\overrightarrow{g}$ into $\overrightarrow{h}$ and this is attained by the sequence of signed reversals $\overrightarrow{\rho(5,7)}$, $\overrightarrow{\rho(3,6)}$, $\overrightarrow{\rho(2,5)}$, $\overrightarrow{\rho(2,6)}$, $\overrightarrow{\rho(1,4)}$:

$$\begin{aligned}
\overrightarrow{g} &= +1+6-5-3+4-7-2 \\
\overrightarrow{g} \odot \overrightarrow{\rho(5,7)} &= +1+6-5-3+2+7-4 \\
\overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} &= +1+6-7-2+3+5-4 \\
\overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} &= +1-3+2+7-6+5-4 \\
\overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} \odot \overrightarrow{\rho(2,6)} &= +1-5+6-7-2+3-4 \\
\overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} \odot \overrightarrow{\rho(2,6)} \odot \overrightarrow{\rho(1,4)} &= +7-6+5-1-2+3-4 = \overrightarrow{h}
\end{aligned}$$

Therefore, the sequence of signed reversals $\overrightarrow{\rho(5,7)}$, $\overrightarrow{\rho(3,6)}$, $\overrightarrow{\rho(2,5)}$, $\overrightarrow{\rho(2,6)}$, $\overrightarrow{\rho(1,4)}$ solves the signed reversal distance problem between $\overrightarrow{g}$ and $\overrightarrow{h}$ and $d(\overrightarrow{g}, \overrightarrow{h}) = 5$.

Likewise the unsigned approach, solving the signed reversal distance problem is equivalent to the simpler problem of sorting a signed permutations by signed reversals.

Definition 3.2.4 (sort by signed reversals). Consider a signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$. Let $d(\overrightarrow{\pi})$ denote the minimum number t for which there exists a sequence of signed reversals $\overrightarrow{\rho}_1, \cdots, \overrightarrow{\rho}_t \in \overrightarrow{S_n}$ such that:

$$\overrightarrow{\pi} \odot \overrightarrow{\rho}_1 \odot \cdots \odot \overrightarrow{\rho}_t = \overrightarrow{I_n} \tag{3.15}$$

The sort by signed reversals problem[9] consists in finding a sequence of $d(\overrightarrow{\pi})$ signed reversals $\overrightarrow{\rho}_1, \cdots, \overrightarrow{\rho}_{d(\overrightarrow{\pi},\overrightarrow{\sigma})} \in \overrightarrow{S_n}$ that verify equation 3.15 for a given signed permutation $\overrightarrow{\pi}$ in $\overrightarrow{S_n}$ [10].

Proposition 3.2.2. *Let $\overrightarrow{\pi}, \overrightarrow{\sigma} \in \overrightarrow{S_n}$. Then $d(\overrightarrow{\pi}, \overrightarrow{\sigma}) = d(\overrightarrow{\sigma}^{-1} \odot \overrightarrow{\pi})$.*

Proof 3.2.2. This result is obtained using the same exact steps presented in proof 3.1.1 but considering signed permutations. □

Example 3.2.3. Consider once more the two signed permutations $\overrightarrow{g} = +1+6-5-3+4-7-2$ and $\overrightarrow{h} = +7-6+5-1-2+3-4$ in $\overrightarrow{S_7}$. It requires at least five signed reversals to sort $\overrightarrow{h}^{-1} \odot \overrightarrow{g}$ by

[9] Also known as the problem of sorting a signed permutation by signed reversals.

[10] By definition, for $\overrightarrow{\pi} \in \overrightarrow{S_n}$, $d(\overrightarrow{\pi}) = d(\overrightarrow{\pi}, \overrightarrow{I_n})$. Again, d can be called with one or two arguments. Moreover, it can be given signed or unsigned permutations as arguments. Nevertheless, the meaning of d should be made clear in the context it is called for.

reversals, and this is attained by the same sequence of signed reversals presented in example 3.2.2:

$$\begin{aligned}
\overrightarrow{g} &= +1+6-5-3+4-7-2 \\
\overrightarrow{h}^{-1} &= -4-5+6-7+3-2+1 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} &= -4-2-3-6-7-1+5 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} \odot \overrightarrow{\rho(5,7)} &= -4-2-3-6-5+1+7 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} &= -4-2-1+5+6+3+7 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} &= -4-6-5+1+2+3+7 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} \odot \overrightarrow{\rho(2,6)} &= -4-3-2-1+5+6+7 \\
\overrightarrow{h}^{-1} \odot \overrightarrow{g} \odot \overrightarrow{\rho(5,7)} \odot \overrightarrow{\rho(3,6)} \odot \overrightarrow{\rho(2,5)} \odot \overrightarrow{\rho(2,6)} \odot \overrightarrow{\rho(1,4)} &= +1+2+3+4+5+6+7 \\
&= \overrightarrow{I_7}
\end{aligned}$$

From this, $d(\overrightarrow{h}^{-1} \odot \overrightarrow{g}) = 5$ and the sequence of signed reversals $\overrightarrow{\rho(5,7)}$, $\overrightarrow{\rho(3,6)}$, $\overrightarrow{\rho(2,5)}$, $\overrightarrow{\rho(2,6)}$, $\overrightarrow{\rho(1,4)}$ solves the sort by signed reversal problem for $\overrightarrow{h}^{-1} \odot \overrightarrow{g}$.

3.2.1 Turning signed permutations into unsigned permutations

Bafna and Pevzner [4] noted that signed permutations in $\overrightarrow{S_n}$ could be modeled by permutations in S_{2n}, according to a transformation $T : \overrightarrow{S_n} \longrightarrow S_{2n}$.

Definition 3.2.5 ($T : \overrightarrow{S_n} \longrightarrow S_{2n}$). The image through T of a signed permutation $\overrightarrow{\pi} = s(1)\pi(1)\cdots s(n)\pi(n)$ is obtained by replacing positive elements $+\pi(i)$ by elements $2\pi(i)-1$, $2\pi(i)$ and negative elements $-\pi(j)$ by elements $2\pi(j)$, $2\pi(j)-1$, i.e., $\pi' = T(\overrightarrow{\pi}) = \pi'(1)\cdots\pi'(2n)$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\pi'(2i-1) = \begin{cases} 2\pi(i)-1 & \text{, if } s(i) = + \\ 2\pi(i) & \text{, if } s(i) = - \end{cases} \quad \wedge \quad \pi'(2i) = \begin{cases} 2\pi(i) & \text{, if } s(i) = + \\ 2\pi(i)-1 & \text{, if } s(i) = - \end{cases}$$

For example, if $\overrightarrow{\pi} = +2-3-1$ (i.e., $\overrightarrow{\pi} = \langle \pi, s \rangle$ where $\pi = 2\ 3\ 1$ and $s(1) = +$ and $s(2) = s(3) = -$), then:

$$\begin{aligned}
T(\overrightarrow{\pi}) &= (2\pi(1)-1)(2\pi(1))(2\pi(2))(2\pi(2)-1)(2\pi(3))(2\pi(2)-1) \\
&= (2\cdot 2-1)(2\cdot 2)(2\cdot 3)(2\cdot 3-1)(2\cdot 1)(2\cdot 1-1) \\
&= 3\ 4\ 6\ 5\ 2\ 1
\end{aligned}$$

Proposition 3.2.3. *$T : \overrightarrow{S_n} \longrightarrow S_{2n}$ is an injective group homomorphism, i.e., it holds that:*

1. *(homomorphism) for any two signed permutations $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ in $\overrightarrow{S_n}$:*

$$T(\overrightarrow{\pi} \odot \overrightarrow{\sigma}) = T(\overrightarrow{\pi}) \circ T(\overrightarrow{\sigma}) \tag{3.16}$$

and it holds that $T(\overrightarrow{I_n}) = I_{2n}$.

2. (injectivity) for any two signed permutations $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ in $\overrightarrow{S_n}$:

$$\overrightarrow{\pi} \neq \overrightarrow{\sigma} \Rightarrow T(\overrightarrow{\pi}) \neq T(\overrightarrow{\sigma}) \tag{3.17}$$

Proof 3.2.3. In annex (chapter 8). □

The following property is a simple consequences of $T : \overrightarrow{S_n} \longrightarrow S_{2n}$ being an injective group homomorphism[11]:

- for any two signed permutations $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ in $\overrightarrow{S_n}$:

$$\overrightarrow{\pi} = \overrightarrow{\sigma} \Leftrightarrow T(\overrightarrow{\pi}) = T(\overrightarrow{\sigma}) \tag{3.18}$$

Proposition 3.2.4. *Let $\overrightarrow{\pi} \in \overrightarrow{S_n}$. Then $d(\overrightarrow{\pi}) \geq d(T(\overrightarrow{\pi}))$.*

Proof 3.2.4. Let $\overrightarrow{\pi} \in \overrightarrow{S_n}$ and $\overrightarrow{\rho_1}, \ldots, \overrightarrow{\rho_{d(\overrightarrow{\pi})}}$ be a sequence of signed reversals in $\overrightarrow{S_n}$ that solve the sort by signed reversals problem for $\overrightarrow{\pi}$. Then, equation 3.15 holds and from equation 3.18:

$$\overrightarrow{\pi} \odot \overrightarrow{\rho}_1 \odot \cdots \odot \overrightarrow{\rho_{d(\overrightarrow{\pi})}} = \overrightarrow{I_n} \Rightarrow T(\overrightarrow{\pi} \odot \overrightarrow{\rho}_1 \odot \cdots \odot \overrightarrow{\rho_{d(\overrightarrow{\pi})}}) = T(\overrightarrow{I_n})$$

Using equation 3.16 and the fact that $T(\overrightarrow{I_n}) = I_{2n}$:

$$T(\overrightarrow{\pi} \odot \overrightarrow{\rho}_1 \odot \cdots \odot \overrightarrow{\rho_{d(\overrightarrow{\pi})}}) = T(\overrightarrow{I_n}) \Rightarrow T(\overrightarrow{\pi}) \circ T(\overrightarrow{\rho_1}) \circ \cdots \circ T(\overrightarrow{\rho_{d(\overrightarrow{\pi})}}) = I_{2n}$$

It is also easy to see that $T(\overrightarrow{\rho(i,j)}) = \rho(2i-1, 2j)$, therefore, $T(\overrightarrow{\rho_1}), \ldots, T(\overrightarrow{\rho_{d(\overrightarrow{\pi})}})$ is a sequence of reversals in S_{2n} that solves the sort by reversals problem for permutation $T(\overrightarrow{\pi})$. Consequently, for every sequence of signed reversals in $\overrightarrow{S_n}$ that solves the sort by signed reversals problem for a signed permutation $\overrightarrow{\pi}$ in $\overrightarrow{S_n}$ there exists a sequence of reversals in S_{2n} that solves the sort by reversals problem for $T(\overrightarrow{\pi})$ and $d(\overrightarrow{\pi}) \geq d(T(\overrightarrow{\pi}))$.

□

Combining proposition 3.2.4 with the lower bound for $d(\pi)$ presented in theorem 3.1.4, one obtains that[12]:

$$d(\overrightarrow{\pi}) \geq d(T(\overrightarrow{\pi})) \geq 2n + 1 - c(T(\overrightarrow{\pi})) \tag{3.19}$$

Notice that, for any signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$ and for any $i \in \mathbb{N}$ such that $1 \leq i \leq n$, vertexes $2i-1$ and $2i$, of the breakpoint graph $G(T(\overrightarrow{\pi}))$ of $T(\overrightarrow{\pi})$, are joined by both gray and black edges (see figure 3.3 for an illustration). Each such pair defines an edge-disjoint alternating cycle of length 2 in $G(T(\overrightarrow{\pi}))$. Clearly, every maximum cardinality decomposition of $G(T(\overrightarrow{\pi}))$ into edge-disjoint alternating cycles will

[11] For a detailed introduction to group theory and group homomorphisms see [16].

[12] This bound has $2n$ instead of n because $\overrightarrow{\pi} \in \overrightarrow{S_n}$ implies $T(\overrightarrow{\pi}) \in S_{2n}$.

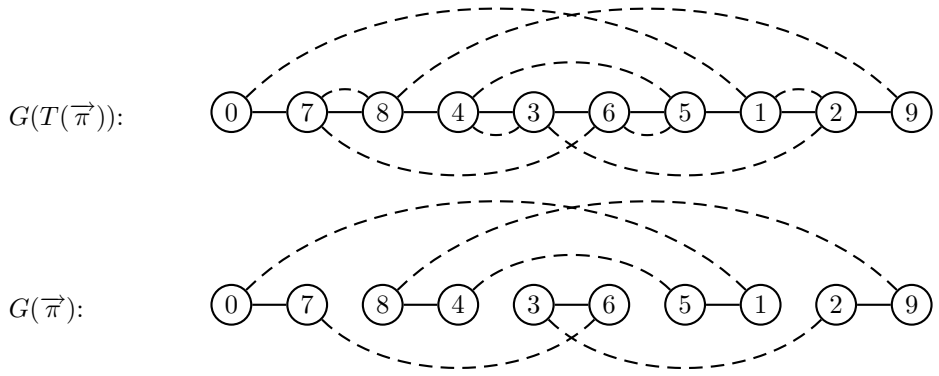


Figure 3.3: The breakpoint graph $G(\overrightarrow{\pi})$ of the signed permutation $\overrightarrow{\pi} = +4-2-3+1$ in $\overrightarrow{S_4}$ compared to the breakpoint graph $G(T(\overrightarrow{\pi}))$ of the permutation $T(\overrightarrow{\pi}) = 7\ 8\ 4\ 3\ 6\ 5\ 1\ 2$ in S_8

include these cycles. The breakpoint graph $G(\overrightarrow{\pi})$ of a signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$ is defined as the breakpoint graph $G(T(\overrightarrow{\pi}))$ of $T(\overrightarrow{\pi})$ with these $2n$ edges removed.

Definition 3.2.6 (breakpoint graph of a signed permutation). Let $\overrightarrow{\pi} \in \overrightarrow{S_n}$, $\pi = T(\overrightarrow{\pi}) \in S_{2n}$ and $G(\pi) = (V, E_b \cup E_g)$. The breakpoint graph $G(\overrightarrow{\pi}) = (V, E'_b \cup E'_g)$ [13] of $\overrightarrow{\pi}$ is a subgraph of $G(\pi)$ having:

- $E'_b = (E_b \setminus \{(2i-1, 2i) : 1 \leq i \leq n\}) \setminus \{(2i, 2i-1) : 1 \leq i \leq n\}$
- $E'_g = E_g \setminus \{(2i-1, 2i) : 1 \leq i \leq n\}$

For example, the breakpoint graph of the signed permutation $\overrightarrow{\pi} = +4-2-3+1$ is presented in figure 3.3.

For any signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$, every vertex of $G(\overrightarrow{\pi})$ has degree 2, and therefore, $G(\overrightarrow{\pi})$ is a collection of edge-disjoint alternating cycles [31]. Let $c(\overrightarrow{\pi})$ denote the number of such cycles in $G(\overrightarrow{\pi})$. Then, the bound in equation 3.19 for $d(\overrightarrow{\pi})$ can be rewritten as follows:

$$d(\overrightarrow{\pi}) \geq 2n + 1 - c(T(\overrightarrow{\pi})) = 2n + 1 - (n + c(\overrightarrow{\pi})) = n + 1 - c(\overrightarrow{\pi}) \tag{3.20}$$

For example, it is easy to verify that the breakpoint graph illustrated in figure 3.3 is already decomposed ($c(\overrightarrow{\pi}) = 1$) and that the bound $n + 1 - c(\overrightarrow{\pi}) = 4$ equals the actual value of $d(\overrightarrow{\pi})$.

As noted in section 3.1.2, the lower bound for $d(\pi)$ presented in theorem 3.1.4 is hard to compute. However, the equivalent bound for signed permutations, presented in equation 3.20, is trivial to compute. Since $G(\overrightarrow{\pi})$ is already decomposed, computing $c(\overrightarrow{\pi})$ is just a matter of counting the number of cycles [31]. Moreover, the bound $n + 1 - c(\overrightarrow{\pi})$ for signed permutations approximates the real value $d(\overrightarrow{\pi})$ extremely well for both simulated and biological data [22, 3,

[13] G can be called for signed or unsigned permutations. However, the meaning of G should be made clear from the argument given.

17]. This lead Hannenhalli and Pevzner [18] to search for another parameter that would close the gap between $d(\overrightarrow{\pi})$ and $n+1-c(\overrightarrow{\pi})$.

3.2.2 Sorting by legal reversals

Hannenhalli and Pevzner [18] found that the sort by signed reversals problem, for a given signed permutation $\overrightarrow{\pi}$, could be reduced to the sort by reversals problem for the permutation $T(\overrightarrow{\pi})$, where only a restricted class or reversals is allowed.

Definition 3.2.7 (legal reversal). Consider $i, j, n \in \mathbb{N}$ such that $1 \leq i \leq j \leq n$. A reversal $\rho(i,j) \in S_n$ is said to be legal reversal i is odd and j is even.

Definition 3.2.8 (sort by legal reversals). Consider a permutation $\pi \in S_{2n}$ that is the image, through T, of some signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$. Let $d_{legal}(\pi)$ denote the minimum number t for which there exists a sequence of legal reversals $\rho_1, \ldots, \rho_t \in S_{2n}$ such that:

$$\pi \circ \rho_1 \circ \cdots \circ \rho_t = I_{2n} \tag{3.21}$$

The sort by legal reversals problem[14] consists in finding a sequence of $d_{legal}(\pi)$ legal reversals $\rho_1, \ldots, \rho_{d_{legal}(\pi)} \in S_{2n}$ that verify equation 3.21 for a given permutation $\pi \in S_n$ that is the image, through T, of some signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$.

Proposition 3.2.5. *Let $\overrightarrow{\pi} \in \overrightarrow{S_n}$. Then $d(\overrightarrow{\pi}) = d_{legal}(T(\overrightarrow{\pi}))$.*

Proof 3.2.5. Let $\overrightarrow{\pi} \in \overrightarrow{S_n}$ and $\overrightarrow{\rho_1}, \ldots, \overrightarrow{\rho_{d(\overrightarrow{\pi})}}$ be a sequence of signed reversals in $\overrightarrow{S_n}$ that solve the sort by signed reversals problem for $\overrightarrow{\pi}$. Notice that since $T(\overrightarrow{\rho(i,j)}) = \rho(2i-1, 2j)$, the image through T of a signed reversals is always a legal reversals and therefore, $T(\overrightarrow{\rho_1}), \ldots, T(\overrightarrow{\rho_t})$ is a sequence of legal reversals in S_{2n}. Following a similar reasoning to the one presented in proof 3.2.4 it holds that $d(\overrightarrow{\pi}) \geq d_{legal}(T(\overrightarrow{\pi}))$.

Let $\pi \in S_{2n}$ such that $\pi = T(\overrightarrow{\pi})$. Let $\rho_1, \ldots, \rho_{d_{legal}(\pi)}$ be legal reversals in S_{2n} that solve the sort by legal reversals for π. Then, equation 3.21 holds. Since T is injective and $T(\overrightarrow{\rho(i,j)}) = \rho(2i-1, 2j)$, then, for every $k \in \mathbb{N}$ such that $1 \leq k \leq d_{legal}(\pi)$, there exists a signed reversals $\overrightarrow{\rho_k} \in \overrightarrow{S_n}$ such that $\rho_k = T(\overrightarrow{\rho_k})$. From this and the fact that $T(\overrightarrow{I_n}) = I_{2n}$ and $\pi = T(\overrightarrow{\pi})$:

$$\pi \circ \rho_1 \circ \cdots \circ \rho_{d_{legal}(\pi)} = I_{2n} \Rightarrow T(\overrightarrow{\pi}) \circ T(\overrightarrow{\rho_1}) \circ \cdots \circ T(\overrightarrow{\rho_{d_{legal}(\pi)}}) = T(\overrightarrow{I_n})$$

Using equation 3.16:

$$T(\overrightarrow{\pi}) \circ T(\overrightarrow{\rho_1}) \circ \cdots \circ T(\overrightarrow{\rho_{d_{legal}(\pi)}}) = T(\overrightarrow{I_n}) \Rightarrow T(\overrightarrow{\pi} \odot \overrightarrow{\rho_1} \odot \cdots \odot \overrightarrow{\rho_{d_{legal}(\pi)}}) = T(\overrightarrow{I_n})$$

[14] Also known as the problem of sorting a permutation by legal reversals.

From equation 3.18:

$$T(\overrightarrow{\pi} \odot \overrightarrow{\rho_1} \odot \cdots \odot \overrightarrow{\rho_{d_{legal}(\pi)}}) = T(\overrightarrow{I_n}) \Rightarrow \overrightarrow{\pi} \odot \overrightarrow{\rho_1} \odot \cdots \odot \overrightarrow{\rho_{d_{legal}(\pi)}} = \overrightarrow{I_n}$$

Therefore, the sequence of signed reversals $\overrightarrow{\rho_1}, \ldots, \overrightarrow{\rho_{d_{legal}(\pi)}}$ in $\overrightarrow{S_n}$ solves the sort by signed reversals problem for $\overrightarrow{\pi}$ and $d_{legal}(T(\overrightarrow{\pi})) \geq d(\overrightarrow{\pi})$. From this, for a given signed permutation $\overrightarrow{\pi} \in \overrightarrow{S_n}$, $d(\overrightarrow{\pi}) = d_{legal}(T(\overrightarrow{\pi}))$.

□

Due to proposition 3.2.5, the bound for $d(\overrightarrow{\pi})$ in equation 3.20 also applies to $d_{legal}(T(\overrightarrow{\pi}))$:

$$d_{legal}(T(\overrightarrow{\pi})) \geq n + 1 - c(\overrightarrow{\pi}) \tag{3.22}$$

In the sort by reversals problem for a permutation $\pi \in S_n$, the objective is to find a sequence of reversals that turns π into I_n. Theorem 3.1.4 reflected the idea that, knowing that a reversal could increase at most $c(\pi)$ by one and that $c(I_n) = n+1$, it would take at least $n+1-c(\pi)$ reversals to sort a permutation $\pi \in S_n$. The lower bound for $d_{legal}(T(\overrightarrow{\pi}))$ states the same idea.

$\overrightarrow{I_n}$ is the only signed permutation in $\overrightarrow{S_n}$ whose breakpoint graph $G(\overrightarrow{\pi})$ has a cycle decomposition into $c(\overrightarrow{I_n}) = n + 1$ edge-disjoint alternating cycles. For all other signed permutations in $\overrightarrow{S_n}$, $c(\overrightarrow{\pi}) < n + 1$. Let $\pi = T(\overrightarrow{\pi})$ for some signed permutation $\overrightarrow{\pi}$. A reversal ρ is called *proper* if it is legal and $c(\overrightarrow{\pi} \odot T^{-1}(\rho)) - c(\overrightarrow{\pi}) = 1$ (where $c(\overrightarrow{\pi} \odot T^{-1}(\rho))$ is the number edge-disjoint alternating cycles in the breakpoint graph of the signed permutation $\overrightarrow{\pi} \odot T^{-1}(\rho)$). Its effect when acting on π is to increase the value of $c(\overrightarrow{\pi})$ by one. If we were able to find a *proper* reversal for every permutation π that is the image of some signed permutation $\overrightarrow{\pi}$, then we would be able to optimally sort any permutation $\overrightarrow{\pi}$ with $n+1-c(\overrightarrow{\pi})$ reversals and the lower bound in equation 3.22 would become an equality. However, there are permutations for which there is no proper reversal [31].

3.2.3 Algorithms and complexity

Hannenhalli and Pevzner [18] found that certain permutations had yet another hidden obstacle to be sorted by legal reversals. The authors presented a result in which $d_{legal}(T(\overrightarrow{\pi}))$, where $\overrightarrow{\pi} \in \overrightarrow{S_n}$, is expressed as a deterministic expression involving polynomial time computable properties of $T(\overrightarrow{\pi})$ and of the breakpoint graph $G(\overrightarrow{\pi})$ of $\overrightarrow{\pi}$ (such as $c(\overrightarrow{\pi})$). From this result, they devised an algorithm for computing $d(\overrightarrow{\pi}) = d_{legal}(\pi)$ for any $\overrightarrow{\pi} \in \overrightarrow{S_n}$ that runs in time $O(n^4)$.

Since sorting a permutation by reversals is NP-Hard [11], but sorting a signed permutation by reversals is a polynomial time solvable problem, one might ask why not using representations of genomes as signed permutations all the time. The problem is that some available data on gene order in genomes, like the ones provided from comparative physical maps (see chapter 2.3), does not have explicit information on the transcriptional reading direction of genes [31]. Sequencing provides both order and direction of genes in a genome and there has been an increase in the number of fully sequenced genomes. However,

sequencing an entire genome is still expensive and unsigned data can still provide valid rearrangement scenarios [31].

3.3 Multichromosomal Genomes

Genomes of simple organisms can be well modeled by signed or unsigned permutations. The reversal distance between two such genomes is also a good indicator of molecular evolution (see section 3.1.1). In genomes of more complex organisms, genes are grouped in conserved blocks called *chromosomes* and evolution takes place by several evolutionary processes besides reversals of segments of chromosomes. For example, the chromosome 7 in the human genome can be viewed as a "shuffling" of different genes from chromosomes 2, 5, 6, 11, and 13 in the mouse genome [31].

This approach for genome rearrangements was first introduced by Kececioglu and Ravi [21] and models genomes as sets of chromosomes. Genes are represented by an integer whose sign (+ or −) represents its transcriptional reading direction. Chromosomes are defined as sequences of genes. Given two genomes Π and Γ, the idea is to find a most parsimonious scenario of evolution of Π into Γ. This corresponds to a shortest sequence of rearrangement events required to transform Π into Γ. It is assumed however, that Π and Γ contain the same set of genes and none of them contains duplicate genes.

Definition 3.3.1 (multichromosomal genome). A chromosome π is an ordered sequence of integers, i.e., $\pi = \pi_1 \cdots \pi_n$ where, for $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $\pi_i \in \mathbb{Z}$. A chromosome with no genes is represented by ϵ. A multichromosomal genome Π is an ordered set $\{\pi^1, \ldots, \pi^N\}$ of chromosomes $\pi^i = \pi^i_1 \cdots \pi^i_{n_i}$, with respective lengths n_i, having no genes in common, i.e.:

$$\forall_{1 \leq i \leq\ N} \quad \forall_{1 \leq j \leq\ N} \quad i \neq j \quad \Rightarrow \quad \Big(\bigcup_{k=1}^{n_i} \{|\pi^i_k|\} \Big) \cap \Big(\bigcup_{k=1}^{n_j} \{|\pi^j_k|\} \Big) = \emptyset \tag{3.23}$$

For example, $\Pi = \{-3\,-2\,-1\,+4\,+5\,+6\,+7\,+11, +9, +10\,+8\}$ is a multichromosomal genome with 3 chromosomes with respective lengths $n_1 = 8$, $n_2 = 1$ and $n_3 = 2$.

Although the reading direction of genes matters, the reading direction of an entire chromosome is irrelevant [19]. Every chromosome π can be viewed from "left to right", as $\pi = \pi_1 \cdots \pi_n$, or from "right to left", as $-\pi = -\pi_n \cdots - \pi_1$, that is, chromosomes π and $-\pi$ are equivalent. Two multichromosomal genomes $\Pi = \{\pi^1, \ldots, \pi^N\}$ and $\Gamma = \{\gamma^1, \ldots, \gamma^N\}$, with a same number of N chromosomes, are said to be equal if they differ only on the reading direction of chromosomes, i.e:

$$\Pi = \Gamma \quad \overset{def}{\Leftrightarrow} \quad \forall_{1 \leq i \leq N} \quad \pi^i = \gamma^i \vee \pi^i = -\gamma^i \tag{3.24}$$

For example, the multichromosomal genomes $\{+3{-}2{+}1, +4{-}7{+}6, -5{+}8\}$, $\{+3{-}2{+}1, -6{+}7{-}4, -5{+}8\}$ and $\{-1+2-3, +4-7+6, -8+5\}$ are equal.

The four most common rearrangement events in multichromosomal genomes are *reversals*, *translo-*

cations, *fusions* and *fissions*.

Definition 3.3.2 (chromossome reversal). Consider a multichromosomal genome $\Pi = \{\pi^1, \ldots, \pi^N\}$ with N chromosomes. Let $k, i, j \in \mathbb{N}$ such that $1 \leq k \leq N$ and $1 \leq i \leq j \leq n_k$. Let $\pi = \pi_1 \cdots \pi_{i-1}\pi_i\pi_{i+1} \cdots \pi_{j-1}\pi_j\pi_{j+1} \cdots \pi_{n_k}$ be the kth chromosome π^k of Π. A chromosome reversal $\rho(k, i, j)$ acts on Π by transforming the kth chromosome of Π into $\pi' = \pi_1 \cdots \pi_{i-1} - \pi_j - \pi_{j-1} \cdots - \pi_{i+1} - \pi_i\pi_{j+1} \cdots \pi_{n_k}$.

Definition 3.3.3 (translocation). Consider a multichromosomal genome $\Pi = \{\pi^1, \ldots, \pi^N\}$ of N chromosomes. Let $k, l, i, j \in \mathbb{N}$ such that $1 \leq k \leq N$, $1 \leq l \leq N$, $k \neq l$, $1 \leq i \leq n_k+1$ and $1 \leq j \leq n_l+1$. Let $\pi = \pi_1 \cdots \pi_{i-1}\pi_i \cdots \pi_{n_k}$ and $\sigma = \sigma_1 \cdots \sigma_{j-1}\sigma_j \cdots \sigma_{n_l}$ be the kth and lth chromosomes π^k and π^l of Π. A translocation $\rho(k, l, i, j)$ acts on Π by transforming the kth and lth chromosome of Π into $\pi' = \pi_1 \cdots \pi_{i-1}\sigma_j \cdots \sigma_{n_l}$ and $\sigma' = \sigma_1 \cdots \sigma_{j-1}\pi_i \cdots \pi_{n_k}$ with $(i-1)+(n_l-j+1)$ and $(j-1)+(n_k-i+1)$ genes respectively.

For a multichromosomal genome Π and a chromosome reversal or translocation ρ concerning chromosomes of Π, $\Pi \circledcirc \rho$ denotes the multichromosomal genome obtained by the application of ρ to the chromosomes of Π that it affects. A *fusion* is a particular case of a translocation $\rho(k, l, n_k + 1, 1)$ (or $\rho(k, l, 1, n_l + 1)$) that concatenates the chromosomes π^k and π^l, resulting in a chromosome $\pi^k_1 \cdots \pi^k_{n_k}\pi^l_1 \cdots \pi^l_{n_l}$ and an empty chromosome ϵ (of length 0). A *fission* is a particular case of a translocation $\rho(k, l, i, 1)$, where $\pi^l = \epsilon$ (or $\rho(k, l, 1, j)$, where $\pi^k = \epsilon$), causing the "breaking" of the kth chromosome π^k into two chromosomes $\pi^k_1 \cdots \pi^k_{i-1}$ and $\pi^k_i \cdots \pi^k_{n_k}$. Fusions and fissions are rather common in mammalian evolution [19]. For example, the major difference in overall genome organization of humans and chimpanzees is the fusion of two chimpanzee chromosomes into one human chromosome [19].

Definition 3.3.4 (genomic distance). Consider two multichromosomal genomes $\Pi = \{\pi^1, \ldots, \pi^M\}$ and $\Gamma = \{\gamma^1, \ldots, \gamma^N\}$, with chromosomes of lengths $n_1, \ldots, n_M$ and $m_1, \ldots, m_N$, containing the same set of genes, i.e., such that:

$$\bigcup_{i=1}^{M}\bigcup_{j=1}^{n_i}\{|\pi^i_j|\} = \bigcup_{i=1}^{N}\bigcup_{j=1}^{m_i}\{|\gamma^i_j|\} \tag{3.25}$$

Without loss of generality assume that $M \leq N$ and add $N-M$ empty chromosomes ϵ to Π, i.e., redefine Π as $\Pi' = \{\pi^1, \ldots, \pi^M, \pi^{M+1}, \ldots, \pi^N\}$, where $\pi^{M+1} = \cdots = \pi^N = \epsilon$ [15]. The genomic distance $d(\Pi, \Gamma)$ between Π and Γ is the minimum number t for which there exists a sequence of chromosome reversals and translocations $\rho_1, \ldots, \rho_t$ such that:

$$((\cdots(\Pi' \circledcirc \rho_1) \circledcirc \cdots) \circledcirc \rho_t) = \Gamma \tag{3.26}$$

The genomic distance problem consists in finding a sequence of $d(\Pi, \Gamma)$ chromosome reversals and

[15] This is a necessary step because no chromosome reversals or translocation can increase or decrease the number of chromosomes in a multichromosomal genome.

translocations $\rho_1, \ldots, \rho_{d(\Pi,\Gamma)}$ that verify equation 3.26 for two given multichromosomal genomes Π and Γ that verify equation 3.25.

Example 3.3.1 (from [31]). Consider the multichromosomal genomes $\Pi = \{\pi^1, \pi^2, \pi^3\}$ and $\Gamma = \{\gamma^1, \gamma^2, \gamma^3\}$, where $\pi^1 = +1+2+3+4$, $\pi^2 = +5+6+7+8$, $\pi^3 = +9+10+11$, $\gamma^1 = -3-2-1+4+5+6+7+11$ and $\gamma^2 = +9$, $\gamma^3 = +10+8$. It requires at least four rearrangement events to transform Π into Γ and this is attained by the sequence of chromosome reversal and translocations $\rho(1,1,3)$, $\rho(1,2,4,3)$, $\rho(1,2,5,1)$, $\rho(3,2,1,1)$:

$$\begin{aligned}
\Pi = & \quad \{+1+2+3+4, +5+6+7+8, +9+10+11\} & \\
\Pi \circledcirc \rho(1,1,3) = & \quad \{-3-2-1+4, +5+6+7+8, +9+10+11\} & = \Pi_1 \\
\Pi_1 \circledcirc \rho(1,2,4,3) = & \quad \{-3-2-1+4, +5+6+7+11, +9+10+8\} & = \Pi_2 \\
\Pi_2 \circledcirc \rho(1,2,5,1) = & \quad \{-3-2-1+4+5+6+7+11, \epsilon, +9+10+8\} & = \Pi_3 \\
\Pi_3 \circledcirc \rho(3,2,1,1) = & \quad \{-3-2-1+4+5+6+7+11, +9, +10+8\} & = \Gamma
\end{aligned}$$

From this, $((((\Pi \circledcirc \rho(1,1,3)) \circledcirc \rho(1,2,4,3)) \circledcirc \rho(1,2,5,1)) \circledcirc \rho(3,2,1,1)) = \Gamma$. Therefore, $d(\Pi,\Gamma) = 4$ and the sequence of chromosome reversals and translocations $\rho(1,1,3)$, $\rho(1,2,4,3)$, $\rho(1,2,5,1)$, $\rho(3,2,1,1)$ solves the genomic distance problem between Π and Γ. Notice that $\rho(1,1,3)$ is a chromosome reversal acting on the first chromosome of Π, $\rho(1,2,4,3)$ is a translocation acting on the first and second chromosomes of Π_1, $\rho(1,2,5,1)$ is a fusion of the first and second chromosomes of Π_2 and $\rho(3,2,1,1)$ is a fission of the third chromosome of Π_3.

3.3.1 Algorithms and complexity

Kececioglu and Ravi [21] devised an algorithm to compute the genomic distance between two genomes with the same number of chromosomes. This limitation was overcome by Hannenhalli and Pevzner [19], who managed to reduce the computation of the genomic distance between two multichromosomal genomes to the computation of the signed reversal distance, where only a restricted class of signed reversals was allowed, between two suitably well defined signed permutations. The same authors also proved that the genomic distance between two multichromosomal genomes could be computed in terms of several polynomial time computable properties of these signed permutations, hence providing a polynomial time algorithm for computing $d(\Pi,\Gamma)$.

4

Matching models and similarity measures

The previously presented approaches for genome rearrangements took for input two different orders of the same set of genes. The idea was to find the minimum number of operations (modeling genome-level evolutionary processes) needed to transform one order of genes into another. While these approaches may be appropriated for some small genomes, they fail to scope to more complex genomes where several copies of the same gene may be scattered across the genome.

This chapter focus on the recent approach for genome rearrangements that consists in finding a matching, of a chosen model, that optimizes a chosen similarity measures. The idea of using matching models and similarity measures for comparing two genomes in terms of gene order evolution was first presented by Sankoff [33] and has been applied in several recent studies [1, 2, 38, 7, 9, 14]. The aim is to overcome the difficulty of having duplicate genes in unichromosomal genomes.

Section 4.1 presents a formalism for this approach (following the one presented in [1, 2]) and defines, from a general perspective, the optimization problems that are the focus of this dissertation. Section 4.2 introduces some of the similarity measures that have been proposed and an extended example is provided in section 4.3. Finally, in section 4.4, the complexity of this method is presented.

4.1 Matching models

In recent studies genomes are modeled as sequences over an alphabet of gene families, where each gene family represents a group of homologous genes. This is due to the increasing number of fully sequenced genomes and the use of sequence alignment tools for deriving gene order in two such genomes (see section 2.3). A genome is said to have *duplicate genes* if it has two genes that belong to the same gene family. It is said to be *duplicate-free* otherwise.

Sankoff stated that when comparing genomes with duplicate genes, it is sometimes reasonable to disregard all genes on the genomes except the *true exemplars* of each gene family [33]. In this way, the reduced genomes would be *duplicate-free* and one could compare them using some previous approach to genome rearrangements (see chapter 3).

Suppose that the most recent common ancestor of two genomes G and H [1] had no duplicate genes. Sankoff's idea of *true exemplars* of gene families is that of the genes, in each genome, that best reflect their positions in that most recent, duplicate-free, common ancestor genome. The idea is that the *true exemplar* genomes (the genomes with all genes erased except the *true exemplars* of each gene family) should be the ones minimizing, for example, the reversal distance between any two *exemplar* genomes (genomes obtained by erasing all but one gene of each gene family). One would then search for the *true exemplar* genomes and take the reversal distance between them as a measure of evolution between G and H.

In a way, the approach of matching models and similarity measures generalizes Sankoff's approach. The idea is to use a matching $\mathcal{M}$ between genes in two genomes G and H (i.e., an assignment between homologous genes in G and H) to induce two new duplicate free genomes over the alphabet of gene families $\{1, \ldots, |\mathcal{M}|\}$ and of lengths $|\mathcal{M}|$ (i.e., signed permutations in $\overrightarrow{S_{|\mathcal{M}|}}$), where $|\mathcal{M}|$ is the size of $\mathcal{M}$. The objective is to find a matching $\mathcal{M}$ between G and H, whose induced duplicate free genomes maximize (or minimize) a chosen function $d_{|\mathcal{M}|} : \overrightarrow{S_{|\mathcal{M}|}} \times \overrightarrow{S_{|\mathcal{M}|}} \longrightarrow \mathbb{N}_0$.

Two matching models have been considered: the exemplar model [33] and the maximum model [13]. In Sankoff's approach, the matching was required to be an *exemplar* matching (i.e., to match exactly one gene, in each genome, for each gene family), while other approaches require the matching to be a *maximum* matching (to match as many genes as possible).

Each choice of matching model and similarity measure results in a different optimization problem. These problems do not measure the minimum amount of rearranging events necessary to transform one genome into another, like the approaches on genome rearrangements do. However, the solution to each problem provides an indication of how closely related are two genomes in terms of gene order evolution and can be used for the construction of phylogenetic trees for sets of genomes. Moreover, finding an optimal exemplar or maximum matching can also provide useful information for the identification of ancestral homologs and true orthologs genes in groups of homologous genes shared by two genomes (see section 2.4).

Definition 4.1.1 (genome). Consider a finite alphabet $\mathcal{A}$ of gene families. A gene is an element of $\mathcal{A}$ having a $+$ or $-$ sign prefixed to it. If g is a gene then:

- $-g$ denotes the gene obtained from g by switching its sign.
- $s(g)$ denotes the sign of g.
- $|g|$ denotes the gene family of g.

A genome G over $\mathcal{A}$ is a sequence $G[1] \cdots G[n_G]$ of genes, where n_G denotes the length of G. For all $a \in \mathcal{A}$ and for all $i, j \in \mathbb{N}$ such that $1 \leq i \leq j \leq n_G$, $occ_G(a, i, j)$ denotes the number of occurrences of genes g in a genome G, between positions i and j (inclusively), such that $|g| = a$. $occ_G(a, 1, n_G)$ is

[1] The genome from which G and H started to diverge.

abbreviated to $occ_G(a)$. A genome G over $\mathcal{A}$ is said to be *duplicate-free* if for all $a \in \mathcal{A}$, $occ_G(a) \leq 1$. G is said to have *duplicate genes* otherwise.

Definition 4.1.2 (matching). A matching $\mathcal{M}$ between two genomes G and H over the same finite alphabet of gene families, is a set $\mathcal{M} \subseteq \{1, \ldots, n_G\} \times \{1, \ldots, n_H\}$ of pairs that are pairwise disjoint and establish a correspondence between genes of the same gene family in both genomes[2], i.e:

1. Every two distinct elements $(i, j), (i', j') \in \mathcal{M}$ verify that $i \neq i'$ and $j \neq j'$.
2. Every element $(i, j) \in \mathcal{M}$ verifies that $|G[i]| = |H[j]|$

The number of elements in a matching $\mathcal{M}$ is denoted as $|\mathcal{M}|$. Genes $G[i]$ and $H[j]$ that are matched by a pair (i, j) in a matching $\mathcal{M}$ are said to be $\mathcal{M}$-saturated.

Definition 4.1.3 (exemplar and maximum matchings). Consider two genomes G and H over the same finite alphabet of gene families $\mathcal{A}$. Let $\mathcal{M}$ be a matching between G and H. Then:

- $\mathcal{M}$ is said to be an *exemplar matching* if, for each gene family $a \in \mathcal{A}$, such that $occ_G(a) \geq 1$ and $occ_H(a) \geq 1$, it saturates exactly one gene, in each genome, that belongs to that family.
- $\mathcal{M}$ is said to be a *maximum matching* if it saturates as many genes of each gene family as possible, i.e., if it is a matching of maximum cardinality.

Proposition 4.1.1. *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and define:*

- $\mathcal{A}' = \{a \in \mathcal{A} : occ_G(a) \geq 1 \wedge occ_H(a) \geq 1\}$.
- $M_a = \max\{occ_G(a), occ_H(a)\}$, *for all* $a \in \mathcal{A}'$.
- $m_a = \min\{occ_G(a), occ_H(a)\}$, *for all* $a \in \mathcal{A}'$.

Then:

1. *there exist* $\prod_{a \in \mathcal{A}'} occ_G(a) \cdot occ_H(a)$ *distinct exemplar matchings between G and H.*
2. *there exist* $\prod_{a \in \mathcal{A}'} A_{m_a}^{M_a}$ *distinct maximum matchings between G and H, where:*

$$A_{m_a}^{M_a} \stackrel{def}{=} M_a(M_a - 1) \cdots (M_a - m_a + 1)$$

3. *if $\mathcal{M}$ is an exemplar matching, then* $|\mathcal{M}| = |\mathcal{A}'|$.
4. *if $\mathcal{M}$ is a maximum matching, then* $|\mathcal{M}| = \sum_{a \in \mathcal{A}'} m_a$.

Proof 4.1.1. In annex (chapter 8). □

[2]Notice that, because of the limits imposed on a pair (i, j) of a matching, a matching between G and H and a matching between H and G are different definitions. However, if $\mathcal{M}$ is a matching between G and H, then $\mathcal{M}' = \{(k, i) : (i, k) \in \mathcal{M}\}$ is a matching between H and G.

A matching establishes a correspondence between homologous genes in two genomes. Given a matching $\mathcal{M}$ between two genomes G and H, the idea is to do the following steps:

1. Discard unmatched genes, maintaining the order of the remaining genes in each genome.
2. Assign a different number, from 1 to $|\mathcal{M}|$, to each element of $(i, j) \in \mathcal{M}$ and rename genes $G[i]$ and $H[j]$ with that number while maintaining their original signs.

The resulting genomes are called $\mathcal{M}$-reduced genomes and are duplicate free genomes over the alphabet of gene families $\{1, \ldots, |\mathcal{M}|\}$ and of lengths $|\mathcal{M}|$ (figure 4.1 provides an illustration of this process).

Definition 4.1.4 ($\mathcal{M}$-reduced genomes). Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$, of lengths n_G and n_H. Consider a matching $\mathcal{M}$ between G and H and a bijective function $f : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$. Define $G_0 = G$, $G_1 = H$, $n_{G_0} = n_G$ and $n_{G_1} = n_H$. Let:

- $(a, b)^{-0} \stackrel{def}{=} (a, b)$ and $(a, b)^{-1} \stackrel{def}{=} (b, a)$, for all $a, b \in \mathbb{N}$.
- $\overline{x} \stackrel{def}{=} 1 - x$, for all $x \in \{0, 1\}$.
- $P^x_{\mathcal{M}}(i) = 1$ if $G_x[i]$ is $\mathcal{M}$-saturated, i.e., if there exists $k \in \mathbb{N}$ such that $1 \leq k \leq n_{\overline{x}}$ and $(i, k)^{-x} \in \mathcal{M}$, for $x \in \{0, 1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$; $P^x_{\mathcal{M}}(i) = 0$ otherwise.
- $k^x_{\mathcal{M}}(i)$ denote the position of the ith $\mathcal{M}$-saturated gene in G_x, i.e., the minimum number t for which $\sum_{j=1}^{t} P^x_{\mathcal{M}}(j) = i$, for $x \in \{0, 1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$.
- $m^x_{\mathcal{M}}(i)$ denote the pair in $\mathcal{M}$ that matches the gene $G_x[i]$, i.e., the pair $(i, k)^{-x} \in \mathcal{M}$ (where $1 \leq k \leq n_{\overline{x}}$), for $x \in \{0, 1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$ and $P^x_{\mathcal{M}}(i) = 1$.

The $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ are defined as follows:

$$\begin{aligned} \mathcal{M}_f(G) &\stackrel{def}{=} \mathcal{M}_f(G_0) \\ \mathcal{M}_f(H) &\stackrel{def}{=} \mathcal{M}_f(G_1) \end{aligned} \tag{4.1}$$

Where for all $x \in \{0, 1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$ [3]:

$$\mathcal{M}_f(G_x)[i] \stackrel{def}{=} s(G_x[k^x_{\mathcal{M}}(i)]) f(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \tag{4.2}$$

It is easy to see from definition that for every finite alphabet of gene families $\mathcal{A}$, every genomes G and H over $\mathcal{A}$, every matching $\mathcal{M}$ between G and H and every bijective function $f : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$, the $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ are duplicate free genomes over the alphabet of gene families $\mathcal{B} = \{1, \ldots, |\mathcal{M}|\}$ and of lengths $|\mathcal{M}|$. In example 4.1.1 we build the $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ for the same genomes G and H, matching $\mathcal{M}$ and bijective function f used in figure 4.1.

[3] $s(G_x[k^x_{\mathcal{M}}(i)])$ is the sign of the gene in position $k^x_{\mathcal{M}}(i)$ of G_x (see definition 4.1.1).

Example 4.1.1. Let $\mathcal{A} = \{a, b, c\}$, $G = +b + a - b - a + c$, $H = -a + b - b + c - a + a$, $\mathcal{M} = \{(2,5), (3,2), (4,1), (5,4)\}$ be a matching between G and H and $f : \mathcal{M} \longrightarrow \{1,2,3,4\}$ be such that $f((2,5)) = 1$, $f((3,2)) = 2$, $f((4,1)) = 3$ and $f((5,4)) = 4$. Define $G_0 = G$, $G_1 = H$, $n_{G_0} = n_G = 5$ and $n_{G_1} = n_H = 6$. Then, $|\mathcal{M}| = 4$, $k^0_{\mathcal{M}}(1) = 2$, $k^0_{\mathcal{M}}(2) = 3$, $k^0_{\mathcal{M}}(3) = 4$, $k^0_{\mathcal{M}}(4) = 5$, $k^1_{\mathcal{M}}(1) = 1$, $k^1_{\mathcal{M}}(2) = 2$, $k^1_{\mathcal{M}}(3) = 4$, $k^1_{\mathcal{M}}(4) = 5$ and :

$$\begin{aligned}
\mathcal{M}_f(G_0) = &\ \mathcal{M}_f(G_0)[1]\mathcal{M}_f(G_0)[2]\mathcal{M}_f(G_0)[3]\mathcal{M}_f(G_0)[4] = \\
= &\ s(G_0[2])f(m^0_{\mathcal{M}}(2))s(G_0[3])f(m^0_{\mathcal{M}}(3))s(G_0[4])f(m^0_{\mathcal{M}}(4))s(G_0[5])f(m^0_{\mathcal{M}}(5)) = \\
= &\ +f((2,5)) - f((3,2)) - f((4,1)) + f((5,4)) = \\
= &\ +1 - 2 - 3 + 4
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_f(G_1) = &\ \mathcal{M}_f(G_1)[1]\mathcal{M}_f(G_1)[2]\mathcal{M}_f(G_1)[3]\mathcal{M}_f(G_1)[4] = \\
= &\ s(G_1[1])f(m^1_{\mathcal{M}}(1))s(G_1[2])f(m^1_{\mathcal{M}}(2))s(G_1[4])f(m^1_{\mathcal{M}}(4))s(G_1[5])f(m^1_{\mathcal{M}}(5)) = \\
= &\ -f((4,1)) + f((3,2)) + f((5,4)) - f((2,5)) \\
= &\ -3 + 2 + 4 - 1
\end{aligned}$$

Therefore, $\mathcal{M}_f(G) \stackrel{def}{=} \mathcal{M}_f(G_0) = +1 - 2 - 3 + 4$ and $\mathcal{M}_f(H) \stackrel{def}{=} \mathcal{M}_f(G_1) = -3 + 2 + 4 - 1$. Figure 4.1 illustrates this construction. Notice that $\mathcal{M}$ is neither an exemplar or maximum matching. $\mathcal{M}$ saturates two genes of the gene family a (therefore $\mathcal{M}$ is not an exemplar matching) but does not saturate the two genes in each genome of the gene family b (therefore $\mathcal{M}$ is not a maximum matching).

There is a natural one to one correspondence between duplicate free genomes over the alphabet of gene families $\{1, \ldots, n\}$ of lengths n and signed permutations in $\overrightarrow{S_n}$. If G is such a genome, then G can be represented as the signed permutation $\langle \pi, s' \rangle \in \overrightarrow{S_n}$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\begin{aligned}
\pi(i) &= G[i] \\
s'(i) &= s(G[i])
\end{aligned} \tag{4.3}$$

where $s(G[i])$ is the sign of the gene $G[i]$. Also, if $\overrightarrow{\pi} \in \overrightarrow{S_n}$, then $\overrightarrow{\pi}$ can be represented as the duplicate free genome G over the alphabet of gene families $\{1, \ldots, n\}$ of length n defined as:

$$G[i] = s(i)\pi(i) \tag{4.4}$$

for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$. Therefore, and henceforth in this dissertation, duplicate free genomes over the alphabet of gene families $\{1, \ldots, n\}$ of lengths n can be viewed as signed permutations in $\overrightarrow{S_n}$ and conversely.

In particular, if G and H are genomes over a finite alphabet of gene families, then, for any matching $\mathcal{M}$ between G and H and for any bijective function $f : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$, the $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ can be viewed as signed permutations in $\overrightarrow{S_{|\mathcal{M}|}}$.

Similarity (or dissimilarity) measures, which will be introduced in section 4.2, are families $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$, of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$, that measure how closely related (respectively how far apart) are two duplicate free genomes, over the alphabet of gene families $\{1,\ldots,n\}$ and of lengths n (i.e., signed permutations in $\overrightarrow{S_n}$), in terms of gene order evolution. The signed reversal distance, for example, is a similarity measure. The idea is to use a similarity measure to evaluate the similarity between genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$. However, for a similarity measure $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ to be adequate for this approach, it is desired that, for any genomes G and H and any matching $\mathcal{M}$ between G and H, the value $d_{|\mathcal{M}|}(\mathcal{M}_f(G), \mathcal{M}_f(H))$ does not depend on the choice of the bijective function $f : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$. We prove that this is the case if and only if each function d_n is invariant on the left for permutations.

Definition 4.1.5 (invariant on the left for permutations). A function $d : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ is said to be invariant on the left for permutations if, for all signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ and all permutations $\pi \in S_n$, it holds that:

$$d(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d(\langle \pi, + \rangle \odot \overrightarrow{\pi_1}, \langle \pi, + \rangle \odot \overrightarrow{\pi_2}) \tag{4.5}$$

where $+ : \{1,\ldots,n\} \longrightarrow \{+,-\}$ is defined as $+(i) = +$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

Proposition 4.1.2. *Let $\mathcal{A}$ be a finite alphabets of gene families, G and H be two genomes over $\mathcal{A}$, $\mathcal{M}$ be a matching between G and H and $f_1, f_2 : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$ be two bijective functions. Then, it holds that:*

$$\begin{aligned} \mathcal{M}_{f_2}(G) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G) \\ \mathcal{M}_{f_2}(H) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H) \end{aligned} \tag{4.6}$$

where $+ : \{1,\ldots,n\} \longrightarrow \{+,-\}$ is defined as $+(i) = +$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

Proof 4.1.2. In annex (chapter 8). □

Proposition 4.1.3. *Consider a family $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$. The two following conditions are equivalent:*

1. *for all $n \in \mathbb{N}$, d_n is invariant on the left for permutations.*
2. *for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$, all matchings $\mathcal{M}$ between G and H and all bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$, it holds that:*

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) \tag{4.7}$$

Proof 4.1.3. In annex (chapter 8). □

Given a matching $\mathcal{M}$ between two genomes G and H, there are $|\mathcal{M}|!$ different bijective functions $f : \mathcal{M} \longrightarrow \{1,\ldots,|\mathcal{M}|\}$. However, we are particularly interested in two: $f^0_{\mathcal{M}}$, that turns G into the increasing sequence $\pm 1 \cdots \pm |\mathcal{M}|$ and $f^1_{\mathcal{M}}$, that turns H into the increasing sequence $\pm 1 \cdots \pm |\mathcal{M}|$.

Definition 4.1.6 ($f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$). Let G and H be two genomes over a finite alphabet of gene families and $\mathcal{M}$ be a matching between G and H. Define $G_0 = G$ and $G_1 = H$. Functions $f^0_{\mathcal{M}}, f^1_{\mathcal{M}} : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ are defined as follows:

$$\begin{aligned} f^0_{\mathcal{M}}((i,k)) &\stackrel{def}{=} \sum_{j=1}^{i} P^0_{\mathcal{M}}(j) \\ f^1_{\mathcal{M}}((i,k)) &\stackrel{def}{=} \sum_{j=1}^{k} P^1_{\mathcal{M}}(j) \end{aligned} \tag{4.8}$$

for all $(i,k) \in \mathcal{M}$. $P^x_{\mathcal{M}}(j)$ are defined as in definition 4.1.4.

Proposition 4.1.4. *Let G and H be two genomes over a finite alphabet of gene families and $\mathcal{M}$ be a matching between G and H. Define $G_0 = G$ and $G_1 = H$. Then:*

1. *For all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:*

$$\begin{aligned} f^0_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) &= i \\ f^1_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))) &= i \end{aligned} \tag{4.9}$$

2. *$f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$ are bijective functions.*

3. *For all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:*

$$\begin{aligned} \mathcal{M}_{f^0_{\mathcal{M}}}(G_0)[i] &= s(G_0[k^0_{\mathcal{M}}(i)])i \\ \mathcal{M}_{f^1_{\mathcal{M}}}(G_1)[i] &= s(G_1[k^1_{\mathcal{M}}(i)])i \end{aligned} \tag{4.10}$$

where $k^x_{\mathcal{M}}(i)$ and $m^x_{\mathcal{M}}(i)$ are defined as in definition 4.1.4.

From proposition 4.1.4, $f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$ are bijective functions for which $\mathcal{M}_{f^0_{\mathcal{M}}}(G) = \pm 1 \cdots \pm |\mathcal{M}|$ and $\mathcal{M}_{f^1_{\mathcal{M}}}(H) = \pm 1 \cdots \pm |\mathcal{M}|$. We are now in conditions of defining the optimization problems that are the focus of this dissertation.

Definition 4.1.7 ($d(G, H, \mathcal{M}, f)$). Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$, $\mathcal{M}$ be a matching between G and H, $f : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ be a bijective function and $d = \bigcup_{n \in \mathbb{N}} \{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$. Then:

$$d(G, H, \mathcal{M}, f) \stackrel{def}{=} d_{|\mathcal{M}|}(\mathcal{M}_f(G), \mathcal{M}_f(H)) \tag{4.11}$$

Definition 4.1.8 ($OPT_e(G, H, d)$ and $OPT_m(G, H, d)$). Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $d = \bigcup_{n \in \mathbb{N}} \{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations. Then:

- $OPT_e(G, H, d)$ is the problem of finding an exemplar matching $\mathcal{M}$, between G and H, such that

$d(G, H, \mathcal{M}, f^0_{\mathcal{M}})$ is optimized (i.e., maximized if d is a similarity measure and minimized if d is a dissimilarity measure).

- $OPT_m(G, H, d)$ is the problem of finding maximum matching $\mathcal{M}$, between G and H, such that $d(G, H, \mathcal{M}, f^0_{\mathcal{M}})$ is optimized.

Each choice of matching model and similarity (or dissimilarity) measure results in a different optimization problem. However, if the chosen similarity (or dissimilarity measure) $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ is such that every function d_n is symmetric, then, solving $OPT_e(G, H, d)$ implies solving $OPT_e(H, G, d)$ and solving $OPT_m(G, H, d)$ implies solving $OPT_m(H, G, d)$. This result is presented in proposition 4.1.7 and is a simple consequence of propositions 4.1.5 and 4.1.6.

Proposition 4.1.5. *Let G and H be two genomes over a finite alphabet of gene families, of lengths n_G and n_H. Let $\mathcal{M} \subseteq \{1, \ldots, n_G\} \times \{1, \ldots, n_H\}$. Let $T(\mathcal{M}) \stackrel{def}{=} \{(k, i) : (i, k) \in \mathcal{M}\}$, then:*

1. *if, $\mathcal{M}$ is a matching between G and H, then, $T(\mathcal{M})$ is a matching between H and G.*
2. *if, $\mathcal{M}$ is an exemplar matching between G and H, then, $T(\mathcal{M})$ is an exemplar matching between H and G.*
3. *if, $\mathcal{M}$ is a maximum matching between G and H, then, $T(\mathcal{M})$ is a maximum matching between H and G.*

Proof 4.1.4. We omit this proof because it an obvious result. □

Proposition 4.1.6. *Consider a family $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations. If, for all $n \in \mathbb{N}$, d_n is symmetric, i.e., for all signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ it holds that:*

$$d_n(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d_n(\overrightarrow{\pi_2}, \overrightarrow{\pi_1}) \tag{4.12}$$

Then, it holds that, for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$ and all matchings $\mathcal{M}$ between G and H:

$$d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) = d(H, G, T(\mathcal{M}), f^0_{T(\mathcal{M})}) \tag{4.13}$$

where $T(\mathcal{M}) \stackrel{def}{=} \{(k, i) : (i, k) \in \mathcal{M}\}$.

Proof 4.1.5. In annex (chapter 8). □

Proposition 4.1.7. *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations and symmetric. Then:*

1. *If $\mathcal{M}$ solves $OPT_e(G, H, d)$, then $T(\mathcal{M})$ solves $OPT_e(H, G, d)$.*

2. If $\mathcal{M}$ solves $OPT_m(G, H, d)$, then $T(\mathcal{M})$ solves $OPT_m(H, G, d)$.

where $T(\mathcal{M}) \stackrel{def}{=} \{(k, i) : (i, k) \in \mathcal{M}\}$.

Proof 4.1.6. In annex (chapter 8). □

4.2 Similarity measures

Several measures have been proposed: *number of common intervals* [39], *Maximum Adjacency Disruption number* [35], *Summed Adjacency Disruption number* [35], *number of breakpoints* [28], *number of conserved intervals* [5] and *signed reversal distance* [40].

These measures were originally defined for either signed, or unsigned, permutations. However, we chose to adapt these definitions to the notation used in this chapter. We will define how to compute each measure for two given duplicate free genomes of lengths n and over an alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$.

In some of the definitions the genomes are said to be unsigned. This is the case when the original measure was defined for (unsigned) permutations. In this case, the construction of the $\mathcal{M}$-reduced genomes is the same but signs are omitted from their representation for computing the measure. This results genomes G that can be viewed as (unsigned) permutations where $|G[i]|$ and $G[i]$ have the same meaning (both can be viewed as natural numbers). Also, if G is such a genome we will refer to G^{-1} as the genome that is the inverse (unsigned) permutation of G.

We omit the definition of the signed reversal distance because it has already been defined in section 3.2 (see Definition 3.2.3) for signed permutations. In proposition 4.2.2, we prove the properties of symmetry and invariance on the left for permutations for the Maximum Adjacency Disruption number, the Summed Adjacency Disruption number, the number of breakpoints and the number of common intervals.

Definition 4.2.1 (common intervals). Let G_0 and G_1 be two duplicate-free genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n. An interval $[k, k + l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n - k$, is said to be a common interval between G_0 and G_1 if the content of the subsequence $G_0[k] \cdots G_0[k + l]$ of G_0 is the same as the content of some subsequence of G_1 (of the same length), i.e., if there exists a number p such that $1 \leq p \leq n - l$ and:

$$\bigcup_{i=k}^{k+l} \{G_0[i]\} = \bigcup_{i=p}^{p+l} \{G_1[i]\}$$

For example, if $G_0 = 1\ 2\ 3\ 4\ 5$ and $G_1 = 1\ 5\ 3\ 4\ 2$, then $[1, 1]$, $[2, 2]$, $[3, 3]$, $[4, 4]$, $[5, 5]$, $[1, 5]$, $[2, 4]$, $[2, 5]$, $[3, 4]$ and $[3, 5]$ are common interval between G_0 and G_1. The number of common intervals between G_0 and G_1 is a symmetric similarity measure.

Definition 4.2.2 (maximum adjacency disruption number). Let G_0 and G_1 be two duplicate-free

unsigned genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n. The maximum adjacency disruption number (MAD) between genomes G_0 and G_1 is defined as the maximum of two values $M_{0,1}$ and $M_{1,0}$, computed as follows:

$$M_{0,1} = \max_{1 \leq i \leq n-1} \left| (G_0^{-1} \circ G_1)[i] - (G_0^{-1} \circ G_1)[i+1] \right| \tag{4.14}$$

$$M_{1,0} = \max_{1 \leq i \leq n-1} \left| (G_1^{-1} \circ G_0)[i] - (G_1^{-1} \circ G_0)[i+1] \right| \tag{4.15}$$

For example if, $G_0 = 1\ 2\ 3\ 4$ and $G_1 = 1\ 3\ 4\ 2$, then $M_{0,1} = 2$ is obtained from $G_0^{-1} \circ G_1 = 1\ 3\ 4\ 2$ and $M_{1,0} = 3$ is obtained from $G_1^{-1} \circ G_0 = 1\ 4\ 2\ 3$, resulting in a maximum adjacency disruption number of 3 between G_0 and G_1. The maximum adjacency disruption number between G_0 and G_1 is a dissimilarity measure.

Definition 4.2.3 (summed adjacency disruption number). Let G_0 and G_1 be two duplicate-free unsigned genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n. The summed adjacency disruption number (SAD) between genomes G_0 and G_1 is defined as the sum of two values $S_{0,1}$ and $S_{1,0}$, computed as follows:

$$S_{0,1} = \sum_{i=1}^{n-1} \left| (G_0^{-1} \circ G_1)[i] - (G_0^{-1} \circ G_1)[i+1] \right| \tag{4.16}$$

$$S_{1,0} = \sum_{i=1}^{n-1} \left| (G_1^{-1} \circ G_0)[i] - (G_1^{-1} \circ G_0)[i+1] \right| \tag{4.17}$$

For example if, $G_0 = 1\ 2\ 3\ 4$ and $G_1 = 1\ 3\ 4\ 2$, then $S_{0,1} = 5$ is obtained from $G_0^{-1} \circ G_1 = 1\ 3\ 4\ 2$ and $S_{1,0} = 6$ is obtained from $G_1^{-1} \circ G_0 = 1\ 4\ 2\ 3$, resulting in a summed adjacency disruption number of 11 between G_0 and G_1. The summed adjacency disruption number between G_0 and G_1 is a dissimilarity measure.

To compute the Maximum Adjacency Disruption number and the Summed Adjacency Disruption number between two genomes G_0 and G_1, one is required to consider the genome $G_0^{-1} \circ G_1$ for computing $M_{0,1}$ and $S_{0,1}$, and the genome $G_1^{-1} \circ G_0$ for computing $M_{1,0}$ and $S_{1,0}$.

Proposition 4.2.1. *Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $\mathcal{M}$ be a matching between G and H. Consider the representation of $\mathcal{M}_{f^0_{\mathcal{M}}}(G)$, $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$ and $\mathcal{M}_{f^1_{\mathcal{M}}}(G)$ as (unsigned) permutations (i.e., excluding signs in their definition). Then:*

$$\left(\mathcal{M}_{f^0_{\mathcal{M}}}(G)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) = \mathcal{M}_{f^0_{\mathcal{M}}}(H) \tag{4.18}$$

$$\left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(G) = \mathcal{M}_{f^1_{\mathcal{M}}}(G) \tag{4.19}$$

Due to proposition 4.2.1, for computing the Maximum Adjacency Disruption number or the Summed

Adjacency Disruption number between $\mathcal{M}_{f^0_{\mathcal{M}}}(G)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$, one can considering the particular $\mathcal{M}$-reduced genome $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$ for computing $M_{0,1}$ and $S_{0,1}$, and the particular $\mathcal{M}$-reduced genome $\mathcal{M}_{f^1_{\mathcal{M}}}(G)$ for computing $M_{1,0}$ and $S_{1,0}$.

Definition 4.2.4 (breakpoints). A gene $G[i]$ is said to precede a gene $G[i+1]$ in a genome G. Let G_0 and G_1 be two duplicate-free genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n. Define the genomes G'_0 and G'_1, of lengths $n+2$, as follows:

- $G'_0[i+1] = s(G_0[i])(|G_0[i]|+1)$ and $G'_1[i+1] = s(G_1[i])(|G_1[i]|+1)$ for $1 \leq i \leq n$.
- $G'_0[1] = G'_1[1] = +1$ and $G'_0[n+2] = G'_1[n+2] = +(n+2)$.

A gene $G'_0[i]$, where $1 \leq i \leq n+1$, is said to determine a breakpoint in G'_0 if neither $G'_0[i]$ precedes $G'_0[i+1]$ in G'_1, nor $-G'_0[i+1]$ precedes $-G'_0[i]$ in G'_1. $G'_0[i]$ is said to determine an adjacency otherwise[4]. For example, if $G_0 = +1+2+3$ and $G_1 = -2-1-3$, then $G'_0 = +1+2+3+4+5$ and $G_1 = +1-3-2-4+5$ and $G'_0[2]$ determines an adjacency and $G'_0[1]$, $G'_0[3]$ and $G'_0[4]$ determine breakpoints. The number of breakpoints between G_0 and G_1 is defined as the number of breakpoints in G'_0. Notice that since $G'_0[i]$ either determines a breakpoint or an adjacency in G'_0, the number of breakpoints in G'_0 plus the number of adjacencies in G'_0 equals $n+1$. Therefore, the number of breakpoints between G_0 and G_1 plus the number of adjacencies between G_0 and G_1 equals $n+1$. The number of breakpoints is a dissimilarity measure.

Definition 4.2.5 (conserved intervals). A gene $G[i]$ is said to precede a gene $G[j]$ in a genome G, if $i < j$. Let G_0 and G_1 be two duplicate-free genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n. An interval $[a, b]$, where a and b are genes from different gene families, is said to be a conserved interval between G_0 and G_1 if:

1. Either a precedes b, or $-b$ precedes $-a$ in G_0.
2. Either a precedes b, or $-b$ precedes $-a$ in G_1.
3. The set of gene families appearing between a and b, regardless of signs, is the same for G_0 and G_1, i.e, if $|G_0[k]| = a$, $|G_0[l]| = b$, $|G_1[k']| = a$ and $|G_1[l']| = b$, then:
   - If $k < l \wedge k' < l'$ then: $\bigcup_{i=k}^{l}\{|G_0[i]|\} = \bigcup_{i=k'}^{l'}\{|G_1[i]|\}$
   - If $k < l \wedge k' > l'$ then: $\bigcup_{i=k}^{l}\{|G_0[i]|\} = \bigcup_{i=l'}^{k'}\{|G_1[i]|\}$
   - If $k > l \wedge k' < l'$ then: $\bigcup_{i=l}^{k}\{|G_0[i]|\} = \bigcup_{i=k'}^{l'}\{|G_1[i]|\}$
   - If $k > l \wedge k' > l'$ then: $\bigcup_{i=l}^{k}\{|G_0[i]|\} = \bigcup_{i=l'}^{k'}\{|G_1[i]|\}$

[4]Notice that $G'_0[1]$ determines a breakpoint in G'_0 if and only if $G_0[1] \neq G_1[1]$ and $G'_0[n+1]$ determines a breakpoint in G'_0 if and only if $G_0[n] \neq G_1[n]$.

Notice that if $[a, b]$ is a conserved interval between G_0 and G_1, then so is $[-b, -a]$. Only one of these should be counted for the number of conserver intervals. For example, if $G_0 = +1+2+3+4+5+6-7+8$ and $G_1 = +1+4-3-2+5-6-7+8$, there are seven conserved intervals between G_0 and G_1: $[1, -7]$, $[1, 5]$, $[1, 8]$, $[2, 3]$, $[5, -7]$, $[5, 8]$ and $[-7, 8]$. The number of conserved intervals between G_0 and G_1 is a similarity measure.

Proposition 4.2.2. *SAD, MAD, the number of breakpoints and the number of common intervals are symmetric and invariant on the left for permutations.*

Proof 4.2.1. In annex (chapter 8). □

4.3 An example

Consider an alphabet of gene families $\mathcal{A} = \{a, b, c\}$ and the the following genomes over $\mathcal{A}$:

$$G_0 = +b + a - b + c \qquad G_1 = -a + b - b - c + a$$

The following table presents the ${}^2A_1 \times {}^2A_2 \times {}^1A_1 = 2 \times 2 \times 1 = 4$ possible maximum matchings between G_0 and G_1, as well as the resulting genomes $\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_0)$, $\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_1)$, $\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_0)$ and $\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_1)$ for each matching $\mathcal{M}^i$:

i	$\mathcal{M}^i$	x	$\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_x)$	$\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_x)$
1	$\{(1,2),(2,1),(3,3),(4,4)\}$	0	$+1+2-3+4$	$+2+1-3+4$
		1	$-2+1-3-4$	$-1+2-3-4$
2	$\{(1,2),(2,5),(3,3),(4,4)\}$	0	$+1+2-3+4$	$+1+4-2+3$
		1	$+1-3-4+2$	$+1-2-3+4$
3	$\{(1,3),(2,1),(3,2),(4,4)\}$	0	$+1+2-3+4$	$+3+1-2+4$
		1	$-2+3-1-4$	$-1+2-3-4$
4	$\{(1,3),(2,5),(3,2),(4,4)\}$	0	$+1+2-3+4$	$+2+4-1+3$
		1	$+3-1-4+2$	$+1-2-3+4$

The values of each proposed measure for each matching $\mathcal{M}^i$ are presented in the next table:

	Breakpoints	Common Intervals	Conserver Intervals	MAD	SAD	Signed reversal distance
$\mathcal{M}^1$	5	**8**	**0**	**2**	**8**	**3**
$\mathcal{M}^2$	**4**	7	**0**	3	11	**3**
$\mathcal{M}^3$	5	7	**0**	3	11	4
$\mathcal{M}^4$	5	5	**0**	3	14	4
Optimal	4	8	0	2	8	3

The following table presents the $occ_{G_0}(a) \times occ_{G_0}(a) \times occ_{G_0}(b) \times occ_{G_0}(b) \times occ_{G_0}(c) \times occ_{G_0}(c) = 1 \times 2 \times 2 \times 2 \times 1 \times 1 = 8$ possible exemplar matchings between G_0 and G_1, as well as the resulting genomes $\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_0)$, $\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_1)$, $\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_0)$ and $\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_1)$ for each matching $\mathcal{M}^i$:

i	$\mathcal{M}^i$	x	$\mathcal{M}^i_{f^0_{\mathcal{M}^i}}(G_x)$	$\mathcal{M}^i_{f^1_{\mathcal{M}^i}}(G_x)$
1	$\{(1,2),(2,1),(4,4)\}$	0	$+1+2+3$	$+2+1+3$
		1	$-2+1-3$	$-1+2-3$
2	$\{(1,2),(2,5),(4,4)\}$	0	$+1+2+3$	$+1+3+2$
		1	$+1-3+2$	$+1-2+3$
3	$\{(1,3),(2,1),(4,4)\}$	0	$+1+2+3$	$+2+1+3$
		1	$-2-1-3$	$-1-2-3$
4	$\{(1,3),(2,5),(4,4)\}$	0	$+1+2+3$	$+1+3+2$
		1	$-1-3+2$	$-1-2+3$
5	$\{(3,2),(2,1),(4,4)\}$	0	$+1-2+3$	$+1-2+3$
		1	$-1+2-3$	$-1+2-3$
6	$\{(3,2),(2,5),(4,4)\}$	0	$+1-2+3$	$+3-1+2$
		1	$+2-3+1$	$+1-2+3$
7	$\{(3,3),(2,1),(4,4)\}$	0	$+1-2+3$	$+1-2+3$
		1	$-1-2-3$	$-1-2-3$
8	$\{(3,3),(2,5),(4,4)\}$	0	$+1-2+3$	$+3-1+2$
		1	$-2-3+1$	$-1-2+3$

The values of each proposed measure for each matching $\mathcal{M}^i$ are presented in the next table:

	Breakpoints	Common Intervals	Conserver Intervals	MAD	SAD	Signed reversal distance
$\mathcal{M}^1$	4	5	0	2	6	3
$\mathcal{M}^2$	**3**	5	0	2	6	**2**
$\mathcal{M}^3$	**3**	5	**1**	2	6	**2**
$\mathcal{M}^4$	4	5	0	2	6	3
$\mathcal{M}^5$	4	**6**	0	**1**	**4**	**2**
$\mathcal{M}^6$	4	5	0	2	6	3
$\mathcal{M}^7$	4	**6**	0	**1**	**4**	3
$\mathcal{M}^8$	4	5	0	2	6	**2**
Optimal	3	6	1	1	4	2

4.4 Complexity

The problems $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$, where d is one of the measures presented in section 4.2, have all been proved to be NP-hard when the considered genomes G and H share genes of a gene family that has duplicates in either one of the genomes, i.e., when there is a gene family $a \in \mathcal{A}$ for which either $occ_G(a) \geq 1$ and $occ_H(a) \geq 2$ or, $occ_H(a) \geq 1$ and $occ_G(a) \geq 2$.

Bryant [10] proved that the problem becomes NP-Hard when the reversal distance or breakpoint distance is considered for exemplar matchings. Blin and Rizzi came to the same conclusion for maximum matchings [6] and also proved the problem to be NP-complete when the conserved interval distance is considered for exemplar or maximum matchings [8]. Chauve et al. [13] came to the same conclusion for the number of common intervals and the MAD and SAD numbers.

For the particular cases where d is SAD or MAD, $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$ have also been proved to be APX-hard problems [13]. The class APX (APX stands for approximable) consists of the problems in NP that allow polynomial-time approximation algorithms with approximation ratio bounded by a constant (i.e., a c-approximation algorithm where c is a constant). The class PTAS (PTAS stands for polynomial-time approximation scheme) consists of the problems A for which, for all constants c, there is polynomial-time c-approximation algorithm for A. PTAS are indeed the most useful sort of approximation algorithms. A problem A is said to be APX-hard if for every problem B in APX there is PTAS reduction of B to A and a consequence of this (and of the assumption that $P \neq NP$) is that no APX-hard problem is in PTAS [30]. Therefore, there is no polynomial-time approximation scheme for the problems $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$, where d is SAD or MAD.

Due to these results, some non-optimal algorithms (i.e., algorithms which are not guaranteed to provide optimal solutions) have been proposed for some of the considered optimization problems [7, 9, 38]. Additionally, others have used Pseudo-Boolean Optimization encodings to compute exact solutions [1, 2].

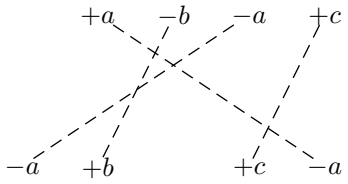
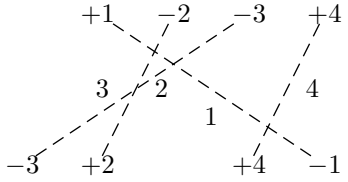
	i	1	2	3	4	5	6
Genomes:	$G =$	$+b$	$+a$	$-b$	$-a$	$+c$	
	$H =$	$-a$	$+b$	$-b$	$+c$	$-a$	$+a$
step 1:							
step 2:							
$\mathcal{M}$-reduced genomes:	$\mathcal{M}_f(G) =$	$+1$	-2	-3	$+4$		
	$\mathcal{M}_f(H) =$	-3	$+2$	$+4$	-1		

Figure 4.1: Building of the $\mathcal{M}$-reduced genomes $\mathcal{M}_f(G)$ and $\mathcal{M}_f(H)$ for the genomes $G = +b+a-b-a+c$ and $H = -a+b-b+c-a+a$, the matching $\mathcal{M} = \{(2,5),(3,2),(4,1),(5,4)\}$ between G and H and the bijective function $f : \mathcal{M} \longrightarrow \{1,2,3,4\}$ defined by $f((2,5)) = 1$, $f((3,2)) = 2$, $f((4,1)) = 3$ and $f((5,4)) = 4$.

5

Pseudo-Boolean Optimization encodings for comparative genomics

This chapter introduces the formalism of Pseudo-Boolean Optimization (PBO) problems and presents the Pseudo-Boolean Optimization encodings for maximizing the number of common intervals [2] and minimizing the number of breakpoints [1] for both exemplar and maximum matchings. Furthermore a new encoding is proposed for minimizing the summed adjacency disruption number for both exemplar and maximum matchings.

Despite being distinct, the six encodings share a general guideline that is described in the beginning of section 5.2. Furthermore, some genome preprocessing steps are applied prior to each encoding. These steps are presented in section 5.2.1.

5.1 Pseudo-Boolean Optimization formalism and complexity

Definition 5.1.1 (linear Pseudo-Boolean constraint). Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables $x_i \in \{0, 1\}$. A literal l_i is either a Boolean variable x_i or its negation $\overline{x_i} \stackrel{def}{=} 1 - x_i$. A linear Pseudo-Boolean constraint over the set of variables X has the following form:

$$\left(\sum_i a_i \cdot l_i \right) \triangleright b$$

where $a_i, b \in \mathbb{Z}$, l_i are literals, $\triangleright$ can be any of the $=, >, \geq, <, \leq, \neq$ operators and $a_i \cdot l_i$ denotes the usual integer multiplication of l_i by a_i.

Definition 5.1.2 (cost function). A cost function over a set $X = \{x_1, \ldots, x_n\}$ of Boolean variables is a linear function of the variables in X, i.e., a function:

$$C(x_1, \ldots, x_n) = \sum_{j=1}^{n} c_j x_j$$

where all c_j are integer numbers.

Definition 5.1.3 (valorations and satisfability). Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A valoration over X is a function $\mathcal{V} : X \to \{0, 1\}$ that assigns to each Boolean variable $x_i \in X$ a "true" (1) or "false" (0) value. For a literal l_i, $\mathcal{V}(l_i)$ denotes $\mathcal{V}(x_i)$ if $l_i = x_i$ and denotes $1 - \mathcal{V}(x_i)$ otherwise. A valoration $\mathcal{V}$ is said to satisfy a linear Pseudo-Boolean constraint $(\sum_i a_i l_i) \triangleright b$, over X, if:

$$\left(\sum_i a_i \cdot \mathcal{V}(l_i) \right) \triangleright b$$

A linear Pseudo-Boolean constraint over X is said to be satisfiable if there exists a valoration over X satisfying it.

Definition 5.1.4 (Pseudo-Boolean Optimization problem). Given a set of Pseudo-Boolean constraints and a cost function C over a set of Boolean variables $X = \{x_1, \ldots, x_n\}$, find, if possible, a valoration $\mathcal{V}$ over X satisfying all the constraints and minimizing the given cost function, i.e., minimizing the value of $C(\mathcal{V}(x_1), \ldots, \mathcal{V}(x_n))$ [1].

Pseudo-Boolean Optimization is a formalism equivalent to that of $0-1$ Integer Linear Optimization ($\{0,1\}-$ILO). A Pseudo-Boolean Optimization problem becomes NP-complete as soon as two or more constraints are considered. Even so, Pseudo-Boolean Optimization solvers are deemed to be among the fastest NP-complete problem specific solvers [15, 27, 37].

5.2 Pseudo-Boolean Optimization encodings

Angibaud et al. presented Pseudo-Boolean Optimization encodings for the problems of finding an exemplar or maximum matching that maximizes the number of common intervals [2] and finding an exemplar or maximum matching that minimizes the number of breakpoints [1]. These are four distinct encodings: one for each choice of matching model and similarity measure.

Let G_0 and G_1 be two genomes and d a similarity (or dissimilarity) measure. The idea of the Pseudo-Boolean Optimization encodings presented in this chapter for the problems $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$, is that of defining:

1. A set of Boolean variables X.
2. A set of linear Pseudo-Boolean constraints over X.
3. A cost function over X.

such that solving the corresponding Pseudo-Boolean optimization problem, i.e., finding a valoration $\mathcal{V}$ over X satisfying all the constraints and minimizing the cost function, induces a valid matching $\mathcal{M}$

[1] If a maximization problem is considered, one needs only multiply the coefficients of the objective function by -1 to obtain an equivalent minimization problem.

(exemplar or maximum) between G_0 and G_1 that optimizes $d(G_0, G_1, \mathcal{M}, f^0_{\mathcal{M}})$ and the value of the cost function for the optimal valoration $\mathcal{V}$ is translated into the value of $d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$.

Since the number of common intervals and the summed adjacency disruption number are defined for (duplicate-free) unsigned genomes, the input genomes G_0 and G_1 will be considered unsigned and therefore, $|G_0[i]|$ will be abbreviated to $G_0[i]$. For the number of breakpoints it will be assumed, without loss of generality, that G_0 and G_1 are genomes over an alphabet of gene families $\mathcal{A} \subset \mathbb{N}$ and signs will not be omitted.

5.2.1 Genome preprocessing

Prior to the definition of each Pseudo-Boolean Optimization encoding, some preprocessing steps are presented for reducing the size of genomes.

The preprocessing of the input genomes G_0 and G_1 will result in two new genomes G'_0 and G'_1, for which the problem of finding an exemplar (or maximum) matching, between G_0 and G_1 that optimizes some measure, is equivalent to finding a matching between G'_0 and G'_1, of the respective model, that optimizes the same measure. Moreover, if G_0 and G_1 are genomes over the alphabet of gene families $\mathcal{A}$, then G'_0 and G'_1 will be genomes over the alphabet of gene families $\mathcal{A}' = \{a \in \mathcal{A} : occ_{G_0}(a) \geq 1 \wedge occ_{G_1}(a) \geq 1\}$.

It is assumed, in the encodings presented in sections 5.2.2, 5.2.3 and 5.2.4, that the input genomes G_0 and G_1 have been preprocessed and therefore verify that every gene that occurs in G_0, also occurs in G_1 (with the same or reverse sign), and conversely.

Step one

A first step is applied for both exemplar and maximum models. This step consists in deleting from the input genomes G_0 and G_1 genes that belong to a gene family with occurrences in only one of the genomes, while maintaining the order of the remaining genes. Clearly, no such gene will be matched in any valid matching between G_0 and G_1, therefore, these genes can simply be deleted from the genomes.

Consider two genomes G_0 and G_1, of lengths n_{G_0} and n_{G_1}, over and alphabet of gene families $\mathcal{A}$. Let:

- $P_x(a)$ denote the minimum number k such that $a < k \leq n_{G_x}$ and for which $G_x[k]$ occurs (with the same, or reverse sign) in both G_0 and G_1, i.e., for $x, a \in \mathbb{N}_0$ such that $0 \leq x \leq 1$ and $0 \leq a < n_{G_x}$:

$$P_x(a) = \min \{k \in \mathbb{N} : k > a \wedge occ_{G_0}(|G_x[k]|) \geq 1 \wedge occ_{G_1}(|G_x[k]|) \geq 1\} \tag{5.1}$$

- $P^i_x(a)$ denote the application of P_x, i times, to a, i.e.:

$$\begin{cases} P^1_x(a) = P_x(a) \\ P^i_x(a) = P_x(P^{i-1}_x(a)) \quad \text{, for } i \in \mathbb{N} \text{ such that } i > 1 \end{cases} \tag{5.2}$$

This step consists in computing the genomes G'_0 and G'_1, over the alphabet of gene families:

$$\mathcal{A}' = \{a \in \mathcal{A} : occ_{G_0}(a) \geq 1 \wedge occ_{G_1}(a) \geq 1\} \tag{5.3}$$

and of lengths $n_{G'_0} = \sum_{a' \in \mathcal{A}'} occ_{G_0}(a')$ and $n_{G'_1} = \sum_{a' \in \mathcal{A}'} occ_{G_1}(a')$. For $x \in \{0, 1\}$ and for $i \in \mathbb{N}$ such that $1 \leq i \leq n_{G'_x}$, $G'_x[i]$ is defined as the i-th gene in G_x that has occurrences in both G_0 and G_1, i.e. [2]:

$$G'_x[i] = G_x[P_x^i(0)] \tag{5.4}$$

Example 5.2.1. Consider genomes $G_0 = +a-c+a+c-b-b+d$ and $G_1 = +e-a-a+a-d+e+e-d-d+b$. Then, $\mathcal{A} = \{a, b, d\}$, $n_{G'_0} = 2 + 2 + 1 = 5$, $n_{G'_1} = 3 + 1 + 3 = 7$, $P_0(0) = 1$, $P_0(1) = 3$, $P_0(3) = 5$, $P_0(5) = 6$, $P_0(6) = 7$, $P_1(0) = 2$, $P_1(2) = 3$, $P_1(3) = 4$, $P_1(4) = 5$, $P_1(5) = 8$, $P_1(8) = 9$, $P_1(9) = 10$ and:

$$\begin{aligned}
G'_0 = &\ G_0[P_0^1(0)]G_0[P_0^2(0)]G_0[P_0^3(0)]G_0[P_0^4(0)]G_0[P_0^5(0)] = \\
= &\ G_0[1]G_0[3]G_0[5]G_0[6]G_0[7] = +a + a - b - b + d \\
G'_1 = &\ G_1[P_1^1(0)]G_1[P_1^2(0)]G_1[P_1^3(0)]G_1[P_1^4(0)]G_1[P_1^5(0)]G_1[P_1^6(0)][P_1^7(0)] = \\
= &\ G_1[2]G_1[3]G_1[4]G_1[5]G_1[8]G_1[9]G_1[10] = -a - a + a - d - d - d + b
\end{aligned}$$

Step two

A second step is applied if the exemplar model is considered. The idea is that in the case of an exemplar matching, consecutive occurrences in a genome of genes of the same gene family can be reduced to just one. An exemplar matching $\mathcal{M}$ between two genomes G_0 and G_1, must match exactly one gene of each gene family in each of the genomes. After the first preprocessing step, this corresponds to choosing, for each gene family, which occurrence of that gene family will be matched in each genome.

If one of the genomes, say G_0, has consecutive occurrences of genes of a gene family $a \in \mathcal{A}$, say in positions $i, i+1, \ldots$ and $i+m$, then, the possible exemplar matchings $\mathcal{M}$ between G_0 and G_1 that chose to match $G_0[i]$, or $G_0[i+1] \ldots$, or $G_0[i+m]$, and differ only on the choice of which of those genes they match, will result in equal $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ genomes (the same goes for $\mathcal{M}_{f^1_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^1_{\mathcal{M}}}(G_1)$)[3]. Therefore, there is no need for considering all these possible solutions and the sequence $G_0[i]G_0[i+1] \cdots G_0[i+m]$ in G_0 is shortened to $G_0[i]$.

However, if signs mater in the computation of the chosen similarity (or dissimilarity) measure (such as in the case of number of breakpoints), only consecutive occurrences of genes of the same gene family, with equal signs can be reduced to one gene.

Consider two genomes G_0 and G_1, of lengths n_{G_0} and n_{G_1}, over and alphabet of gene families $\mathcal{A}$ and such that for all $a \in \mathcal{A}$, $occ_{G_0}(a) \geq 1$ and $occ_{G_1}(a) \geq 1$. Let:

- $E_x(p)$ denote the minimum number k such that $p < k \leq n_{G_x}$ and:

[2] Notice that $P_x(a)$ may not be defined for some values of a. Therefore, $P_x^i(a)$ may also not be defined for some values of a and i. However, this presents no problem in the construction of genomes G'_0 and G'_1.

[3] See definition 4.1.4 and figure 4.1.

- For the number of common intervals and the summed adjacency disruption number: $|G_x[k]| \neq |G_x[p]|$.
- For the number of breakpoints: $G_x[k] \neq G_x[p]$.

for $x \in \{0, 1\}$ and $p \in \mathbb{N}$ such that $1 \leq p < n_{G_x}$.

- $E_x^i(a)$ denote the application of E_x, i times, to a, i.e.:

$$\begin{cases} E_x^1(a) = E_x(a) \\ E_x^i(a) = E_x(E_x^{i-1}(a)) \quad \text{, for } i \in \mathbb{N} \text{ such that } i > 1 \end{cases} \tag{5.5}$$

This step consists in computing the genomes G'_0 and G'_1, over the same alphabet of gene families $\mathcal{A}$, that are defined as follows[4]:

$$\begin{cases} G'_x[1] = G_x[1] \\ G'_x[i+1] = G_x[E_x^i(1)] \quad \text{, for } i \in \mathbb{N} \text{ such that } i > 1 \text{ and } E_x^i(1) \text{ is defined} \end{cases} \tag{5.6}$$

The length $n_{G'_x}$ of G'_x equals the minimum number t for which $E_x^t(1)$ is undefined.

Example 5.2.2. Continuing the previous example, let $G_0 = +a + a - b - b + d$ and $G_1 = -a - a + a - d - d - d + b$. Suppose that the exemplar model is considered for the number of breakpoints. Then, $E_0(1) = 3$, $E_0(3) = 5$, $E_0(5)$ is undefined, $E_1(1) = 3$, $E_1(3) = 4$, $E_1(4) = 7$, $E_1(7)$ is undefined and:

$$G'_0 = G_0[1]G_0[E_0^1(1)]G_0[E_0^2(1)] = G_0[1]G_0[3]G_0[5] = +a - b + d$$
$$G'_1 = G_1[1]G_1[E_1^1(1)]G_1[E_1^2(1)]G_1[E_1^3(1)] = G_1[1]G_1[3]G_1[4]G_1[7] = -a + a - d + b$$

5.2.2 Number of common intervals

Let G_0 and G_1 be two unsigned genomes over an alphabet of gene families $\mathcal{A}$ such that for all $a \in \mathcal{A}$, $occ_{G_0}(a) \geq 1$ and $occ_{G_1}(a) \geq 1$. Let $n_0 = n_{G_0}$ and $n_1 = n_{G_1}$ be the lengths of G_0 and G_1. Consider the following sets of Boolean variables:

$$A = \{a(i,k) : i, k \in \mathbb{N} \wedge 1 \leq i \leq n_0 \wedge 1 \leq k \leq n_1 \wedge G_0[i] = G_1[k]\}$$
$$C = \{c(i,j,k,l) : i, j, k, l \in \mathbb{N} \wedge 1 \leq i \leq j \leq n_0 \wedge 1 \leq k \leq l \leq n_1\}$$

Variables $a(i,k) \in A$ represent pairs of a matching $\mathcal{M}$ between G_0 and G_1, i.e., having a variable $a(i,k) \in A$ assigned to the value "true" represents that $G_0[i]$ is matched to $G_1[k]$. In order to force variables $a(i,k) \in A$ to define a valid matching between G_0 and G_1, we need to guaranty that:

[4]Notice that $E_x(a)$ may not be defined for some values of a. Therefore, $E_x^i(a)$ may also not be defined for some values of a and i. However, this presents no problem in the construction of genomes G'_0 and G'_1.

1. No gene $G_0[i]$ is matched to more than one gene $G_1[k]$, therefore, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n_0$, add the constraint:

$$\sum_{\substack{1 \leq k \leq n_1 \\ G_1[k]=G_0[i]}} a(i,k) \leq 1 \tag{5.7}$$

2. No gene $G_1[k]$ is matched to more than one gene $G_0[i]$, therefore, for all $k \in \mathbb{N}$ such that $1 \leq k \leq n_1$, add the constraint:

$$\sum_{\substack{1 \leq i \leq n_0 \\ G_0[i]=G_1[k]}} a(i,k) \leq 1 \tag{5.8}$$

To force variables $a(i,k) \in A$ to respect the chosen matching model, we need to guaranty that, for all gene families, variables $a(i,k)$ saturate the correct amount of genes of that gene family. Therefore, for all $a \in \mathcal{A}$, add the following constraint if the exemplar model is considered:

$$\sum_{\substack{1 \leq i \leq n_0 \\ G_0[i]=a}} \sum_{\substack{1 \leq k \leq n_1 \\ G_1[k]=a}} a(i,k) = 1 \tag{5.9}$$

or the following constraint if the maximum model is considered:

$$\sum_{\substack{1 \leq i \leq n_0 \\ G_0[i]=a}} \sum_{\substack{1 \leq k \leq n_1 \\ G_1[k]=a}} a(i,k) = \min\{occ_{G_0}(a), occ_{G_1}(a)\} \tag{5.10}$$

It is clear to see that every valoration $\mathcal{V}$ over A, that satisfies the constraints introduced so far, defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ (exemplar or maximum) between G_0 and G_1. Moreover, for every valid matching $\mathcal{M}'$ (exemplar or maximum) between G_0 and G_1, there exists a valoration $\mathcal{V}$ over A that satisfies the constraints introduced so far and such that $\mathcal{M}' = \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$. Therefore, we have a one to one correspondence between valorations $\mathcal{V}$ over A that satisfy the constraints introduced so far and valid matchings (exemplar or maximum) between G_0 and G_1.

Variables $c(i,j,k,l) \in C$ represent common intervals between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, where $\mathcal{M}$ is the matching defined by variables $a(i,k)$. For a variable $c(i,j,k,l)$ to be assigned to the value "true" it will be required that:

1. $G_0[i]$ and $G_0[j]$ are matched to genes in G_1 between positions k and l.

2. $G_1[k]$ and $G_1[l]$ are matched to genes in G_0 between positions i and j.

To ensure this, for all $c(i,j,k,l) \in C$ add the constraint:

$$\begin{aligned} 4 \cdot c(i,j,k,l) - \sum_{\substack{k \leq r \leq l \\ G_0[i]=G_1[r]}} a(i,r) - \sum_{\substack{k \leq s \leq l \\ G_0[j]=G_1[s]}} a(j,s) \\ - \sum_{\substack{i \leq q \leq j \\ G_0[q]=G_1[k]}} a(q,k) - \sum_{\substack{i \leq p \leq j \\ G_0[p]=G_1[l]}} a(p,l) \leq 0 \end{aligned} \tag{5.11}$$

A variable $c(i,j,k,l)$ will be forced to take the value "false" if either:

1. Some gene in G_0, strictly between positions i and j, is matched to some gene in G_1 in a position r smaller than k or greater than l.

2. Some gene in G_1, strictly between positions k and l, is matched to some gene in G_0 in a position p smaller than i or greater than j.

Therefore, for all $c(i,j,k,l) \in C$ and for all $p, r \in \mathbb{N}$ such that either:

1. $i < p < j$, $1 \leq r < k$ and $G_0[p] = G_1[r]$.

2. $i < p < j$, $l < r \leq n_1$ and $G_0[p] = G_1[r]$.

3. $1 \leq p < i$, $k < r < l$ and $G_0[p] = G_1[r]$.

4. $j < p \leq n_0$, $k < r < l$ and $G_0[p] = G_1[r]$.

add the constraint:

$$c(i,j,k,l) + a(p,r) \leq 1 \tag{5.12}$$

Notice that if $\mathcal{V}$ is a valoration over $A \cup C$ that satisfies all these constraints and such that $\mathcal{V}(c(i,j,k,l)) = 1$ for some variable $c(i,j,k,l) \in C$, then, $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ is a valid matching (exemplar or maximum) between G_0 and G_1 and: $G_0[i]$ and $G_0[j]$ are matched to some genes in G_1 between positions k and l; $G_1[k]$ and $G_1[l]$ are matched to some genes in G_0 between positions i and j; if a gene $G_0[p]$, where $i < p < j$, is matched, then it is matched to some gene $G_1[m]$, where $k \leq m \leq l$; if a gene $G_1[m]$, where $k < m < l$, is matched, then it is matched to some gene $G_0[p]$, where $i \leq p \leq j$. Consequently, all $\mathcal{M}$-saturated genes in G_0 between positions i an j are matched to genes in G_1 between positions k and l, and conversely. Therefore, it is easy to see, from construction, that $c(i,j,k,l)$ will correspond to a common interval between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ (see definition 4.1.4). Moreover, it is clear that, due to the constraints introduced for variables in C, if $\mathcal{V}(c(i,j,k,l)) = 1$ for some variable $c(i,j,k,l) \in C$, then, $\mathcal{V}(c(i',j',k,l)) = \mathcal{V}(c(i,j,k',l')) = 0$ for every other variables $c(i,j,k',l'), c(i',j',k,l) \in C$. This means that no common interval between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ is counted more than once.

The objective is to maximize the number of common intervals between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, where $\mathcal{M}$ is the matching defined by variables $a(i,k)$. Therefore, the objective function of this Pseudo-Boolean Optimization problem is the following:

$$\sum_{i=1}^{n_0} \sum_{j=i}^{n_0} \sum_{k=1}^{n_1} \sum_{l=k}^{n_1} c(i,j,k,l) \tag{5.13}$$

Notice that no constraints were defined for forcing a variable $c(i,j,k,l)$ to take the value "true". This is not a problem, because variables $c(i,j,k,l)$ that are not forced to take the value "false" (by some constraint of the type presented in equation 5.12) and are free to take the value "true" (i.e., that

do not contradict some constraint of the type presented in equation 5.11, when assigned to the value "true"), will always be assigned to the value "true" by any valoration that solves this Pseudo-Boolean Optimization problem, because they will increase the value of the objective function[5].

Due to this, if $\mathcal{V}$ is a solution of this Pseudo-Boolean Optimization problem, then for every common interval between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ (where $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$), there exists a variable $c(i,j,k,l) \in C$ for which $\mathcal{V}(c(i,j,k,l)) = 1$. Consequently, if $\mathcal{V}$ is a solution of this Pseudo-Boolean Optimization, then there is a one to one correspondence between variables $c(i,j,k,l) \in C$ for which $\mathcal{V}(c(i,j,k,l)) = 1$ and common intervals between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$.

Therefore finding a solution $\mathcal{V}$ of this Pseudo-Boolean Optimization problem defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ (exemplar or maximum) between G_0 and G_1 for which $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ have a maximum the number of common intervals and this number equals the value of the objective function for $\mathcal{V}$, i.e., $\sum_{i=1}^{n_0}\sum_{j=i}^{n_0}\sum_{k=1}^{n_1}\sum_{l=k}^{n_1}\mathcal{V}(c(i,j,k,l))$.

This Pseudo-Boolean Optimization problem has $O(n_0^2 n_1^2)$ variables and $O(n_0^2 n_1^2)$ constraints. A compact description of this encoding is presented in table 8.1 in the annexes.

Variable reduction rules

We now present some rules for cutting variables $c(i,j,k,l) \in C$, which are to be applied sequentially.

Rule 5.2.1. Delete from C all variables $c(i,i,k,k)$, where $1 \leq i \leq n_0$ and $1 \leq k \leq n_1$.

The idea of rule 5.2.1 is to discard variables in C that represent possible common intervals of size 1. There is a one to one correspondence between common intervals of length 1, between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, and variables $a(i,k) \in A$ assigned to the value "true". Therefore, the number of common intervals of length 1 equals the number of variables $a(i,k) \in A$ assigned to the value "true". These common intervals are represented by variables $c(i,i,k,k)$ in the model and the idea is to discard these variables and add to the objective function the number of variables $a(i,k) \in A$ that will surely be assigned to the value "true". This number equals $|\mathcal{M}| = |\mathcal{A}|$ for the exemplar model and $|\mathcal{M}| = \sum_{a\in\mathcal{A}} \min\{occ_{G_0}(a), occ_{G_1}(a)\}$, for the maximum model (see proposition 4.1.1).

Rule 5.2.2. Delete from C all variables $c(i,j,k,l)$ for which any of the following conditions holds:

1. $occ_{G_1}(G_0[i],k,l) = 0$ or $occ_{G_1}(G_0[j],k,l) = 0$.
2. $occ_{G_0}(G_1[k],i,j) = 0$ or $occ_{G_0}(G_1[l],i,j) = 0$.

or any of the following hold in the case of the exemplar model:

1. $G_0[i] = G_0[j]$.

[5]However, in a suboptimal solution (i.e., a valoration that satisfies all the constraints but does not maximize the objective function), some variables $c(i,j,k,l)$ may be assigned to the value "false" when they actually represent common intervals between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, and should be assigned to the value "true".

2. $G_1[k] = G_1[l]$.

or any of the following hold in the case of the maximum model:

1. $G_0[i] = G_0[j]$ and $occ_{G_1}(G_0[i], k, l) < 2$.

2. $G_1[k] = G_1[l]$ and $occ_{G_0}(G_1[k], i, j) < 2$.

Rule 5.2.2 has the objective of discarding variables $c(i, j, k, l)$ that can never be assigned to the value "true" because of some constraint of the type presented in equation 5.11. If either one of the conditions above conditions is not verified for a variable $c(i, j, k, l)$, then either $G_0[i]$ or $G_0[j]$ cannot be matched to a gene in G_1 between positions k and l, or, either $G_1[k]$ or $G_1[l]$ cannot be matched to a gene in G_0 between positions i and j. Therefore, no such variable can be assigned to the value "true" in this encoding.

Rule 5.2.3. Delete from C all variables $c(i, j, k, l)$ for which there exists $a \in \mathcal{A}$ such that, if the exemplar model is considered, any of the following conditions hold:

1. $occ_{G_0}(a, i, j) = occ_{G_0}(a)$ and $occ_{G_1}(a, k, l) = 0$.

2. $occ_{G_1}(a, k, l) = occ_{G_1}(a)$ and $occ_{G_0}(a, i, j) = 0$.

or, if the maximum model is considered, any of the following conditions hold:

1. $occ_{G_0}(a) < occ_{G_1}(a)$ and $occ_{G_0}(a, i, j) > occ_{G_1}(a, k, l)$.

2. $occ_{G_1}(a) < occ_{G_0}(a)$ and $occ_{G_1}(a, k, l) > occ_{G_0}(a, i, j)$.

3. $|occ_{G_0}(a, i, j) - occ_{G_1}(a, k, l)| > |occ_{G_0}(a) - occ_{G_1}(a)|$

Rule 5.2.3 discards variables $c(i, j, k, l)$ that can never be assigned to the value "true" because of some constraint of the type presented in equation 5.12.

For the exemplar model, condition 1 states that if all the occurrences in G_0, of a gene family $a \in \mathcal{A}'$, take place between positions i an j, and a does not occur in G_1 between positions k an l, then some gene, of the gene family a, in G_0 between positions i an j must be matched to some position r that is smaller than k or greater than l. Therefore, $c(i, j, k, l)$ cannot be assigned to the value "true". Condition 2 states the same idea for gene families whose occurrences in G_1 take place between positions k an l.

For the maximum model, condition 1 states that if the a gene family $a \in \mathcal{A}$ has more occurrences in G_0 than in G_1 (which means that in a maximum matching all the genes of that gene family in G_0 must be matched), then, if the number of occurrences of a in G_0 between positions i and j is greater than the number of occurrences of a in G_1 between positions k and l, then some gene, of the gene family a, in G_0 between positions i an j must be matched to some position r that is smaller than k or greater than l. Therefore, $c(i, j, k, l)$ cannot be assigned to the value "true". Condition 2 states the same idea for gene families that have more occurrences in G_1 than in G_0.

Regarding condition 3 for the maximum model, notice that $|occ_{G_0}(a) - occ_{G_1}(a)|$ equals the number of genes (in G_0 or G_1), of the gene family a, that will remain unmatched in any solution of the problem. $|occ_{G_0}(a,i,j) - occ_{G_1}(a,k,l)|$ corresponds to the number of genes (occurring either in G_0 between positions i and j, or in G_1 between positions k and l), of the gene family a, that must be unmatched in a solution of the problem in order for $c(i,j,k,l)$ not to be forced to the value "false" by some constraint of the the type presented in equation 5.12. Therefore, if $|occ_{G_0}(a,i,j) - occ_{G_1}(a,k,l)| > |occ_{G_0}(a) - occ_{G_1}(a)|$ too many genes, of the gene family a, are required to be unmatched for $c(i,j,k,l)$ to be allowed to take the value "true" and $c(i,j,k,l)$ can be discarded.

Notice that for the maximum model, conditions 1 and 2 can capture variables that condition 3 cannot, and conversely. For example, if $G_0 = a\ c\ c\ b\ b\ b\ c\ b$ and $G_1 = a\ b\ b\ b\ a\ b\ b\ c$, then $c(1,4,1,4)$ is captured by condition 3 and $c(4,6,4,6)$ is captured by condition 1.

Rule 5.2.4. Delete from C all variables $c(i,j,k,l)$ for which the two following conditions hold[6]:

1. For all $p \in \mathbb{N}$, such that $i \leq p \leq j$, it holds that $occ_{G_0}(G_0[p], 1, i-1) + occ_{G_0}(G_0[p], j+1, n_0) + occ_{G_1}(G_0[p], 1, k-1) + occ_{G_1}(G_0[p], l+1, n_1) = 0$

2. For all $r \in \mathbb{N}$, such that $k \leq r \leq l$, it holds that $occ_{G_0}(G_1[r], 1, i-1) + occ_{G_0}(G_1[r], j+1, n_0) + occ_{G_1}(G_1[r], 1, k-1) + occ_{G_1}(G_1[r], l+1, n_1) = 0$

and, if the exemplar model is considered, the following two conditions also hold:

1. $occ_{G_0}(G_0[i]) = occ_{G_0}(G_0[j]) = 1$.

2. $occ_{G_1}(G_1[k]) = occ_{G_1}(G_1[l]) = 1$.

or, if the maximum model is considered, the following two conditions also hold:

1. $occ_{G_0}(G_0[i]) \leq occ_{G_1}(G_0[i])$ and $occ_{G_0}(G_0[j]) \leq occ_{G_1}(G_0[j])$.

2. $occ_{G_1}(G_1[k]) \leq occ_{G_0}(G_1[k])$ and $occ_{G_1}(G_1[l]) \leq occ_{G_0}(G_1[l])$.

Rule 5.2.4 eliminates variables $c(i,j,k,l)$ that will be assigned to the value "true" independently of which variables $a(i,k) \in A$ are assigned to the value "true" in a solution of the PBO problem. It is necessary to guaranty, for both exemplar and maximum models, that no gene occurring in G_0, between positions i and j, can occur in G_0 in a position p, smaller than i or greater than j, or in G_1 in a position p, smaller than k or greater than l (and conversely for genes occurring in G_1, between positions k and l). This implies that if a gene in G_0, between positions i and j, is matched, then it is matched to some gene in G_1 between positions k and l (and conversely for genes occurring in G_1, between positions k and l). Therefore, to assure that $c(i,j,k,l)$ that will be assigned to the value "true", it is only necessary to guaranty that the genes $G_0[i]$, $G_0[j]$, $G_1[k]$ and $G_1[l]$ must be matched (and therefore, matched to

[6] In order to avoid ambiguities, define $occ_{G_x}(a,i,j) = 0$ if $i > j$.

genes inside the interval), independently of which variables $a(i,k) \in A$ are assigned to the value "true". For this, the respective conditions are required for each model.

The number of variables $c(i,j,k,l)$ captured by rule 5.2.4 is added to the objective function.

5.2.3 Number of breakpoints

Let G'_0 and G'_1 be two genomes, of lengths $n_{G'_0}$ and $n_{G'_1}$, over an alphabet of gene families $\mathcal{A}' \subset \mathbb{N}$ such that for all $a' \in \mathcal{A}'$, $occ_{G_0}(a') \geq 1$ and $occ_{G_1}(a') \geq 1$. Define genomes G_0 and G_1 by adding to G'_0 and G'_1 a first gene $+(\min \mathcal{A}' - 1)$ and a last gene $+(\max \mathcal{A}' + 1)$, i.e.:

- $G_0[i+1] = G'_0[i]$ for $1 \leq i \leq n_{G'_0}$.
- $G_1[k+1] = G'_1[k]$ for $1 \leq k \leq n_{G'_1}$.
- $G_0[1] = G_1[1] = +(\min \mathcal{A}' - 1)$ and $G_0[n_{G'_0} + 2] = G_1[n_{G'_1} + 2] = +(\max \mathcal{A}' + 1)$.

G_0 and G_1 are genomes over the alphabet of gene families $\mathcal{A} = \mathcal{A}' \cup \{\min \mathcal{A}' - 1, \max \mathcal{A}' + 1\} \subset \mathbb{N}_0$, of lengths $n_0 = n_{G'_0} + 2$ and $n_1 = n_{G'_1} + 2$.

Consider the following sets of Boolean variables:

$$A = \{a(i,k) : i,k \in \mathbb{N} \wedge 1 \leq i \leq n_0 \wedge 1 \leq k \leq n_1 \wedge |G_0[i]| = |G_1[k]|\}$$
$$B = \{b_x(i) : x \in \{0,1\} \wedge i \in \mathbb{N} \wedge 1 \leq i \leq n_x\}$$
$$C = \{c_x(i,j) : x \in \{0,1\} \wedge i,j \in \mathbb{N} \wedge 1 \leq i < j \leq n_x\}$$
$$D = \{d(i,j,k,l) : i,j,k,l \in \mathbb{N} \wedge 1 \leq i < j \leq n_0 \wedge 1 \leq k < l \leq n_1\}$$

Variables $a(i,k) \in A$ represent pairs of a matching $\mathcal{M}$ between G_0 and G_1, i.e., having a variable $a(i,k) \in A$ assigned to the value "true" represents that $G_0[i]$ is matched to $G_1[k]$. Variables $b_x(i) \in B$ represent $\mathcal{M}$-saturated genes in G_x, according to variables $a(i,k) \in A$.

In order to force variables $a(i,k) \in A$ to define a valid matching between G_0 and G_1 of the chosen matching model, we need to guaranty that:

1. For each gene family, the number of $\mathcal{M}$-saturated genes (where $\mathcal{M}$ is the matching defined by variables $a(i,k) \in A$), of that family, in G_0 and G_1 respects the considered matching model. Therefore, for all $a \in \mathcal{A}$ and for all $x \in \{0,1\}$, add the following constraint if the exemplar model is considered:

$$\sum_{\substack{1 \leq i \leq n_x \\ |G_x[i]| = a}} b_x(i) = 1 \tag{5.14}$$

or, the following constraint if the maximum model is considered:

$$\sum_{\substack{1 \leq i \leq n_x \\ |G_x[i]| = a}} b_x(i) = \min\{occ_{G_0}(a), occ_{G_1}(a)\} \tag{5.15}$$

Notice that a valoration $\mathcal{V}$ over B, that verifies the constraints above, determines which genes in G_0 and G_1 will be $\mathcal{M}$-saturated, where $\mathcal{M}$ is the matching between G_0 and G_1 defined by variables $a(i,k) \in A$.

2. Every gene $G_0[i]$ is matched, at most, to one gene $G_1[k]$ and this is only possible when $b_0(i)$ is assigned to the value "true". Therefore, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n_0$, add the constraint:

$$\sum_{\substack{1 \leq k \leq n_1 \\ |G_1[k]|=|G_0[i]|}} a(i,k) = b_0(i) \tag{5.16}$$

3. Every gene $G_1[k]$ is matched, at most, to one gene $G_0[i]$ and this is only possible when $b_1(k)$ is assigned to the value "true". Therefore, for all $k \in \mathbb{N}$ such that $1 \leq k \leq n_1$, add the constraint:

$$\sum_{\substack{1 \leq i \leq n_0 \\ |G_0[i]|=|G_1[k]|}} a(i,k) = b_1(k) \tag{5.17}$$

It is clear to see that every valoration $\mathcal{V}$ over $A \cup B$, that satisfies the constraints introduced so far, defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ (exemplar or maximum) between G_0 and G_1. Also, there is a one to one correspondence between valorations $\mathcal{V}$ over $A \cup B$ that satisfy the constraints introduced so far and valid matchings (exemplar or maximum) between G_0 and G_1.

Notice that variables $b_0(1)$, $b_1(1)$, $b_0(n_0)$, $b_1(n_1)$, $a(1,1)$ and $a(n_0,n_1)$ will always be assigned to the value "true" by any valoration $\mathcal{V}$ over $A \cup B$ that satisfies the constraints introduced so far. Moreover, if $\mathcal{V}$ is a valoration over $A \cup B$ that satisfies the constraints introduced so far and $\mathcal{M}$ is the matching between G_0 and G_1 defined by the variables $a(i,k) \in A$ (i.e., $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$), then $\mathcal{M}' \stackrel{def}{=} \{(i-1, j-1) : (i,j) \in (\mathcal{M} - \{(1,1),(n_0,n_1)\})\}$ is a valid matching between the original genomes G_0' and G_1'. Furthermore, there is a one to one correspondence between valorations $\mathcal{V}$ over $A \cup B$ that satisfy the constraints introduced so far and valid matchings (exemplar or maximum) between G_0' and G_1'.

Variables $c_x(i,j) \in C$ represent that no gene in G_x, strictly between positions i and j, is $\mathcal{M}$-saturated, where $\mathcal{M}$ is the matching defined by variables $a(i,k)$. If no gene in G_x, strictly between positions i and j, is $\mathcal{M}$-saturated, then $c_x(i,j)$ will be forced to take the value "true". To ensure this, for all $c_x(i,j) \in C$ add the constraint:

$$c_x(i,j) + \sum_{i<p<j} b_x(p) \geq 1 \tag{5.18}$$

A variable $c_x(i,j) \in C$ will be forced to take the value "false" if some gene in G_x, strictly between positions i and j, is $\mathcal{M}$-saturated. Therefore, for all $c_x(i,j) \in C$ and for all $p \in \mathbb{N}$, such that $i < p < j$, add the constraint:

$$c_x(i,j) + b_x(p) \leq 1 \tag{5.19}$$

Notice that a variable $c_x(i,j) \in C$, where $j = i+1$, will always be assigned to the value "true" by the constraint in equation 5.18. Also, in a solution of the problem, if $c_x(i,j)$, $b_x(i)$ and $b_x(j)$ are assigned to the value "true", then $G_x[i]$ and $G_x[j]$ will correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_x)$ and in $\mathcal{M}_{f^1_{\mathcal{M}}}(G_x)$.

The idea of this encoding is to maximize the number of adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$, where $\mathcal{M}' \stackrel{def}{=} \{(i-1,j-1) : (i,j) \in (\mathcal{M} - \{(1,1),(n_0,n_1)\})\}$ and $\mathcal{M}$ is the matching between G_0 and G_1 defined by the variables $a(i,k)$. This corresponds to minimizing the number of breakpoints between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ because the number of breakpoints between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ plus the number of adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ equals $|\mathcal{M}'|+1$ (see definition 4.2.4) and, $|\mathcal{M}'|+1$ is a constant for both the exemplar and maximum model (see proposition 4.1.1).

Variables $d(i,j,k,l) \in D$ represent adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$, where $\mathcal{M}'$ is the matching between G'_0 and G'_1 defined by variables $a(i,k)$. The idea is to establish a one to one correspondence between variables $d(i,j,k,l) \in D$, assigned to the value "true" in a solution of the problem, and adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$. A variable $d(i,j,k,l) \in D$ will represent an adjacency between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ if and only if, the two following conditions hold:

1. $G_0[i]$ and $G_0[j]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$.
2. $G_1[k]$ and $G_1[l]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$.

and either the two following conditions hold:

1. $G_0[i] = G_1[k]$ and $G_0[j] = G_1[l]$.
2. $G_0[i]$ is matched to $G_1[k]$ and $G_0[j]$ is matched to $G_1[l]$.

or the two following conditions hold:

1. $G_0[i] = -G_1[l]$, $G_0[j] = -G_1[k]$.
2. $G_0[i]$ is matched to $-G_1[l]$ and $G_0[j]$ is matched to $-G_1[k]$.

Notice that some variable $d(1,j,1,l) \in D$ and some variable $d(i,n_0,k,n_1) \in D$ will deal with the possible adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ that result from adding, to both $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$, the initial and final genes $+1$ and $+(|\mathcal{M}'|+2)$ (see definition 4.2.4).

A variable $d(i,j,k,l) \in D$, such that $G_0[i] = G_1[k]$ and $G_0[j] = G_1[l]$, will be assigned to the value "true" if and only if variables $a(i,k)$, $a(j,l)$, $c_0(i,j)$ and $c_1(k,l)$ are assigned to the value "true". To ensure this, for all variables $d(i,j,k,l) \in D$, such that $G_0[i] = G_1[k]$ and $G_0[j] = G_1[l]$, add the following

constraints:

$$\begin{gathered} a(i,k) + a(j,l) + c_0(i,j) + c_1(k,l) - d(i,j,k,l) \leq 3 \\ a(i,k) - d(i,j,k,l) \geq 0 \\ a(j,l) - d(i,j,k,l) \geq 0 \\ c_0(i,j) - d(i,j,k,l) \geq 0 \\ c_1(k,l) - d(i,j,k,l) \geq 0 \end{gathered} \tag{5.20}$$

Similarly, for all variables $d(i,j,k,l) \in D$, such that $G_0[i] = -G_1[l]$ and $G_0[j] = -G_1[k]$, add the following constraints:

$$\begin{gathered} a(i,l) + a(j,k) + c_0(i,j) + c_1(k,l) - d(i,j,k,l) \leq 3 \\ a(i,l) - d(i,j,k,l) \geq 0 \\ a(j,k) - d(i,j,k,l) \geq 0 \\ c_0(i,j) - d(i,j,k,l) \geq 0 \\ c_1(k,l) - d(i,j,k,l) \geq 0 \end{gathered} \tag{5.21}$$

A variable $d(i,j,k,l) \in D$ for which neither $G_0[i] = G_1[k]$ and $G_0[j] = G_1[l]$, nor $G_0[i] = -G_1[l]$ and $G_0[j] = -G_1[k]$, cannot represent an adjacency between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ and will be forced to take the value "false". Therefore, for all variables $d(i,j,k,l) \in D$ in this condition, add the constraint:

$$d(i,j,k,l) = 0 \tag{5.22}$$

It is important to note that, due to the constraints presented in equations 5.20 and 5.21 (and also due to the constraints that force variables $a(i,k) \in A$ to define a valid matching between G_0 and G_1), for any valoration $\mathcal{V}$ over $A \cup B \cup C \cup D$ that satisfies all the constraints presented above, and for all $d(i,j,k,l) \in D$, if $\mathcal{V}(d(i,j,k,l)) = 1$, then for every other variables $d(i,j,k',l'), d(i',j',k,l) \in D$ it holds that $\mathcal{V}(d(i,j,k',l')) = \mathcal{V}(d(i',j',k,l)) = 0$. This means that no adjacency between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ is counted more than once. Nevertheless, it is desired to impose this fact with additional (redundant) constraints that will, hopefully, improve the solving time of this Pseudo-Boolean Optimization problem:

- For all $i,j \in \mathbb{N}$ such that $1 \leq i < j \leq n_0$, add the constraint:

$$\sum_{1 \leq k < n_1} \sum_{k < l \leq n_1} d(i,j,k,l) \leq 1 \tag{5.23}$$

- For all $k,l \in \mathbb{N}$ such that $1 \leq k < l \leq n_1$, add the constraint:

$$\sum_{1 \leq i < n_0} \sum_{i < j \leq n_0} d(i,j,k,l) \leq 1 \tag{5.24}$$

From this, it is easy to see that if $\mathcal{V}$ is a valoration over $A \cup B \cup C \cup D$ that verifies all of the presented constraints then, there is a one to one correspondence between variables $d(i,j,k,l)$, such that $\mathcal{V}(d(i,j,k,l)) = 1$, and adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ (where $\mathcal{M}' \stackrel{def}{=} \{(i-1,j-1) : (i,j) \in (\mathcal{M} - \{(1,1),(n_0,n_1)\})\}$ and $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$). Consequently, the number of adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ equals the number of variables $d(i,j,k,l) \in D$ such that $\mathcal{V}(d(i,j,k,l)) = 1$.

The objective is to maximize the number of adjacencies between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$. Therefore, the objective function of this Pseudo-Boolean Optimization problem is the following:

$$\sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} \sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} d(i,j,k,l) \tag{5.25}$$

Finding an optimal solution $\mathcal{V}$ of this Pseudo-Boolean Optimization problem defines a valid matching:

$$\mathcal{M}' = \big\{(i-1,j-1) : (i,j) \in \big(\{(i,k) : \mathcal{V}(a(i,k)) = 1\} - \{(1,1),(n_0,n_1)\}\big)\big\} \tag{5.26}$$

between G'_0 and G'_1 (exemplar or maximum) for which $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ have a maximum number of adjacencies between them and, this number equals the value of the objective function for $\mathcal{V}$, i.e., $\sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} \sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \mathcal{V}(d(i,j,k,l))$. Moreover, the number of breakpoints between $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_0)$ and $\mathcal{M}'_{f^0_{\mathcal{M}'}}(G'_1)$ equals $(|\mathcal{M}'|+1) - \sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} \sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \mathcal{V}(d(i,j,k,l))$.

This Pseudo-Boolean Optimization problem has $O(n_0^2 n_1^2)$ variables and $O(n_0^2 n_1^2)$ constraints. A compact description of this encoding is presented in table 8.2 in the annexes.

Variable reduction rules

We now present some rules for reducing the number of variables $d(i,j,k,l) \in D$. Constraints that concern variables that are captured by these rules will be excluded when building the Pseudo-Boolean Optimization instance. In order to avoid ambiguities in these rules, define $occ_{G_x}(a,i,j) = 0$ if $i > j$.

Exemplar model. If the exemplar model is considered, delete from D all variables $d(i,j,k,l)$ for which any of the following conditions hold:

1. $|G_0[i]| = |G_0[j]|$.
2. $|G_1[k]| = |G_1[l]|$.
3. There exists $p \in \mathbb{N}$ such that $i < p < j$ and $occ_{G_0}(|G_0[p]|, i+1, j-1) = occ_{G_0}(|G_0[p]|)$.
4. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_1}(|G_1[p]|, k+1, l-1) = occ_{G_1}(|G_1[p]|)$.

The idea of this rule is to discard variables $d(i,j,k,l) \in D$ that can never be assigned to the value "true" in a solution of the problem. In the exemplar model, it is required to match one gene for each gene family $a \in \mathcal{A}$. For a variable $d(i,j,k,l)$ to be assigned to the value "true", it is required that genes

$G_0[i]$, $G_0[j]$, $G_1[k]$ and $G_1[l]$ are $\mathcal{M}$-saturated, therefore, if condition 1 or condition 2 holds, either $G_0[i]$ or $G_0[j]$ cannot be $\mathcal{M}$-saturated, or $G_1[k]$ or $G_1[l]$ cannot be $\mathcal{M}$-saturated. Consequently $d(i,j,k,l)$ must take the value "false" and is discarded. Another requirement for a variable $d(i,j,k,l)$ to be assigned to the value "true" is that $G_0[i]$ and $G_0[j]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$. If condition 3 holds, then there exists a gene family whose occurrences in G_0 are contained strictly between positions i and j. Therefore, since one gene of that gene family must be matched, $G_0[i]$ and $G_0[j]$ do not correspond to consecutive genes in $\mathcal{M}_{0,1}(G_0)$ and $d(i,j,k,l)$ must be assigned to the value "false". Condition 4 states the same idea for gene families whole occurrences in G_1 are contained strictly between positions k and l.

Maximum model. If the maximum model is considered, delete from D all variables $d(i,j,k,l)$ for which any of the following conditions hold:

1. There exists $p \in \mathbb{N}$ such that $i < p < j$ and $occ_{G_0}(|G_0[p]|) \leq occ_{G_1}(|G_0[p]|)$.
2. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_1}(|G_1[p]|) \leq occ_{G_0}(|G_1[p]|)$.
3. There exists $p \in \mathbb{N}$ such that $i < p < j$ and $occ_{G_0}(|G_0[p]|, i+1, j-1) > |occ_{G_0}(|G_0[p]|) - occ_{G_1}(|G_0[p]|)|$.
4. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_1}(|G_1[p]|, k+1, l-1) > |occ_{G_0}(|G_1[p]|) - occ_{G_1}(|G_1[p]|)|$.

The rule for the maximum model aims to discard variables $d(i,j,k,l) \in D$ that can never be assigned to the value "true" in a solution of the problem. A maximum matching is required to match as many genes, of each gene family, as possible. If condition 1 holds, then, there is a gene, occurring in G_0 in a position p strictly between positions i and j, that belongs to a gene family that has the same, or more, occurrences in G_1 than in G_0. Therefore, since the maximum model is being considered, all the occurrences of that family in G_0 must be matched, in particular, $G_0[p]$ must be matched. Therefore, $G_0[i]$ and $G_0[j]$ not correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $d(i,j,k,l)$ must be assigned to the value "false". Condition 2 states the same idea for genes that occur in G_1 strictly between positions k and l, that belong to a gene family that has the same, or more, occurrences in G_0 than in G_1. Regarding conditions 3 and 4, notice that $|occ_{G_0}(a) - occ_{G_1}(a)|$ equals the number of genes (in G_0 or G_1), of the gene family a, that will remain unmatched in any solution of the problem. Also, $occ_{G_0}(a, i+1, j-1)$ is the number of genes, of the gene family a, that must be unmatched for $G_0[i]$ and $G_0[j]$ to correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$. The same goes for $occ_{G_1}(a, k+1, l-1)$. Therefore, if condition 3, then too many genes, of some gene family a, are required to be unmatched for $G_0[i]$ and $G_0[j]$ to correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$. Consequently, $G_0[i]$ and $G_0[j]$ do not correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $d(i,j,k,l)$ must be assigned to the value "false". The same reasoning applies to condition 4.

Notice that for the maximum model rule, conditions 1 and 2 can capture variables that conditions 3 and 4 cannot, and conversely. For example, if $G'_0 = +1+3+3+2+2+2+3+2$ and $G'_1 = +1+2+2+2+1+2+2+3$, then $G_0 = +0+1+3+3+2+2+2+3+2+4$ and $G_1 = +0+1+2+2+2+1+2+2+3+4$ and, $d(2,5,2,5)$ is captured by condition 4 and $c(5,7,5,7)$ is captured by condition 1.

Variable value setting constraints

We now present some rules for setting the values of some variables $b_x(i) \in B$, $a(i,k) \in A$ and $d(i,j,k,l) \in D$. These rules are to be applied after the variable reduction rules. Constraints used for defining the values of variables that are captured by these rules will be excluded when building the Pseudo-Boolean Optimization instance.

Some variables $a(i,k) \in A$, $b_x \in B$ and $d(i,j,k,l) \in D$ will always be assigned to the value "true" by any valoration $\mathcal{V}$ that solves the above defined PBO problem (namely, variables $a(i,k) \in A$ and $b_0(i), b_1(k) \in B$ for which genes $G_0[i]$ and $G_1[k]$ belong to a gene family without duplicates in G_0 or G_1). The aim of these rules is to impose this information in the PBO instance, in order to narrow the search for a solution.

Non duplicate gene families. Let $a \in \mathcal{A}$ such that $occ_{G_0}(a) = occ_{G_1}(a) = 1$. Let $i,k \in \mathbb{N}_0$ such that $|G_0[i]| = a$ and $|G_1[k]| = a$. Add the following constraints to the problem:

$$\begin{aligned} b_0(i) &= 1 \\ b_1(k) &= 1 \\ a(i,k) &= 1 \end{aligned} \tag{5.27}$$

Exemplar model. Let $a \in \mathcal{A}$. If the exemplar model is considered and $occ_{G_0}(a) = 1$ and $occ_{G_1}(a) > 1$, then, let $|G_0[i]| = a$ and add the following constraint to the problem:

$$b_0(i) = 1 \tag{5.28}$$

If the exemplar model is considered and $occ_{G_1}(a) = 1$ and $occ_{G_0}(a) > 1$, then, let $|G_1[k]| = a$ and add the following constraint to the problem:

$$b_1(k) = 1 \tag{5.29}$$

Maximum model. Let $a \in \mathcal{A}$. If the maximum model is considered and $occ_{G_0}(a) \leq occ_{G_1}(a)$, then, for all $i \in \mathbb{N}_0$ such that $|G_0[i]| = a$, add the following constraint to the problem:

$$b_0(i) = 1 \tag{5.30}$$

If the maximum model is considered and $occ_{G_1}(a) \leq occ_{G_0}(a)$, then, for all $k \in \mathbb{N}_0$ such that $|G_1[k]| = a$,

add the following constraint to the problem:

$$b_1(k) = 1 \tag{5.31}$$

D variables. Let $d(i, i+1, k, k+1) \in D$ be such that one of the following conditions hold:

1. $G_0[i] = G_1[k]$ and $G_0[i+1] = G_1[k+1]$.
2. $G_0[i] = -G_1[k+1]$ and $G_0[i+1] = -G_1[k]$.

and also, it holds that $occ_{G_0}(|G_0[i]|) = occ_{G_1}(|G_0[i]|) = occ_{G_0}(|G_0[i+1]|) = occ_{G_1}(|G_0[i+1]|) = 1$ (i.e., $G_0[i]$, $G_0[i+1]$, $G_1[k]$ and $G_1[k+1]$ are genes that belong to gene families without duplicates in G_0 and G_1). Then, add the following constraint:

$$d(i, i+1, k, k+1) = 1 \tag{5.32}$$

5.2.4 Summed adjacency disruption number

Let G_0 and G_1 be two unsigned genomes over an alphabet of gene families $\mathcal{A}$ such that for all $a \in \mathcal{A}$, $occ_{G_0}(a) \geq 1$ and $occ_{G_1}(a) \geq 1$. Let $n_0 = n_{G_0}$ and $n_1 = n_{G_1}$ be the lengths of G_0 and G_1 and $\overline{x}$ denote $|x-1|$, for $x \in \{0,1\}$. Consider the following sets of Boolean variables:

$$A = \{a(i,k) : i,k \in \mathbb{N} \wedge 1 \leq i \leq n_0 \wedge 1 \leq k \leq n_1 \wedge G_0[i] = G_1[k]\}$$
$$B = \{b_x(i) : x \in \{0,1\} \wedge i \in \mathbb{N} \wedge 1 \leq i \leq n_x\}$$
$$C = \{c_x(i,j) : x \in \{0,1\} \wedge i,j \in \mathbb{N} \wedge 1 \leq i < j \leq n_x\}$$
$$D = \{d_x(i,j,k,l) : x \in \{0,1\} \wedge i,j,k,l \in \mathbb{N} \wedge 1 \leq i < j \leq n_x \wedge 1 \leq k < l \leq n_{\overline{x}}\}$$
$$E = \{e_x(i,j,p) : x \in \{0,1\} \wedge i,j,p \in \mathbb{N} \wedge 1 \leq i < j \leq n_x \wedge i < p < j\}$$

This encoding uses the variables $a(i,k)$, $b_x(i)$ and $c_x(i,j)$ of the encoding for the number of breakpoints[7]. Variables $a(i,k) \in A$ represent pairs of a matching $\mathcal{M}$ between G_0 and G_1. Variables $b_x(i) \in B$ represent $\mathcal{M}$-saturated genes in G_x. Variables $c_x(i,j) \in C$ represent that no gene in G_x, strictly between positions i and j, is $\mathcal{M}$-saturated. The constraints presented in the encoding for the number of breakpoints for variables $a(i,k)$, $b_x(i)$ and $c_x(i,j)$ are added to this Pseudo-Boolean Optimization problem.

Recall that every valoration $\mathcal{V}$ over $A \cup B$, that satisfies the constraints introduced so far (the ones used in the number of breakpoints for variables $a(i,k)$, $b_x(i)$ and $c_x(i,j)$), defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ (exemplar or maximum) between G_0 and G_1 and, there is a one to one correspondence between valorations $\mathcal{V}$ over $A \cup B$ that satisfy the constraints introduced so far and valid matchings (exemplar or maximum) between G_0 and G_1.

[7]Notice that we have a variable $a(i,k)$ for every position i and k, in G_0 and G_1, such that $G_0[i] = G_1[k]$. For the number of breakpoints it was required that $|G_0[i]| = |G_1[k]|$. However, the summed adjacency disruption number is defined for unsigned genomes, therefore $|G_0[i]|$ and $G_0[i]$ have the same meaning and are abbreviated to $G_0[i]$.

The summed adjacency disruption number is defined as the sum of two numbers $S_{0,1}$ and $S_{1,0}$ (see definition 4.2.3). Due to proposition 4.2.1, for computing the summed adjacency disruption number between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, one can considering the particular $\mathcal{M}$-reduced genome $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ for computing $S_{0,1}$, and the particular $\mathcal{M}$-reduced genome $\mathcal{M}_{f^1_{\mathcal{M}}}(G_0)$ for computing $S_{1,0}$ (where $\mathcal{M}$ is the matching defined by variables $a(i,k)$).

Variables $d_0(i,j,k,l) \in D$ will be used to compute the number $S_{0,1}$ and represent that $G_1[k]$ and $G_1[l]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ that are matched to $G_0[i]$ and $G_0[j]$ (or to $G_0[j]$ and $G_0[i]$), respectively. Variables $d_1(i,j,k,l) \in D$ will be used to compute the number $S_{1,0}$ and represent that $G_0[k]$ and $G_0[l]$ correspond to consecutive genes in $\mathcal{M}_{f^1_{\mathcal{M}}}(G_0)$ that are matched to $G_1[i]$ and $G_1[j]$ (or to $G_1[j]$ and $G_1[i]$), respectively.

Let $a(i,k)^{-0} \stackrel{def}{=} a(i,k)$ and $a(i,k)^{-1} \stackrel{def}{=} a(k,i)$ for all variables $a(i,k) \in A$. A variable $d_x(i,j,k,l) \in D$, such that $G_x[i] = G_{\overline{x}}[k]$ and $G_x[j] = G_{\overline{x}}[l]$, will be assigned to the value "true" if and only if variables $c_{\overline{x}}(k,l)$, $a(i,k)^{-x}$ and $a(j,l)^{-x}$ are assigned to the value "true". To ensure this, for all variables $d_x(i,j,k,l) \in D$, such that $G_x[i] = G_{\overline{x}}[k]$ and $G_x[j] = G_{\overline{x}}[l]$, add the following constraints:

$$\begin{aligned} a(i,k)^{-x} + a(j,l)^{-x} + c_{\overline{x}}(k,l) - d_x(i,j,k,l) &\leq 2 \\ a(i,k)^{-x} - d_x(i,j,k,l) &\geq 0 \\ a(j,l)^{-x} - d_x(i,j,k,l) &\geq 0 \\ c_{\overline{x}}(k,l) - d_x(i,j,k,l) &\geq 0 \end{aligned} \tag{5.33}$$

Similarly, for all variables $d_x(i,j,k,l) \in D$, such that $G_x[i] = G_{\overline{x}}[l]$ and $G_x[j] = G_{\overline{x}}[k]$, add the following constraints:

$$\begin{aligned} a(i,l)^{-x} + a(j,k)^{-x} + c_{\overline{x}}(k,l) - d_x(i,j,k,l) &\leq 2 \\ a(i,l)^{-x} - d_x(i,j,k,l) &\geq 0 \\ a(j,k)^{-x} - d_x(i,j,k,l) &\geq 0 \\ c_{\overline{x}}(k,l) - d_x(i,j,k,l) &\geq 0 \end{aligned} \tag{5.34}$$

A variable $d_x(i,j,k,l) \in D$ for which neither $G_x[i] = G_{\overline{x}}[k]$ and $G_x[j] = G_{\overline{x}}[l]$, nor $G_x[i] = G_{\overline{x}}[l]$ and $G_x[j] = G_{\overline{x}}[k]$, will be forced to take the value "false". Therefore, for all variables $d_x(i,j,k,l) \in D$ in this condition, add the constraint:

$$d_x(i,j,k,l) = 0 \tag{5.35}$$

It is important to note that, due to the constraints presented in equations 5.33, 5.34 and 5.35 (and also due to the constraints that force variables $a(i,k)$ to define a valid matching between G_0 and G_1), for any valoration $\mathcal{V}$ over $A \cup B \cup C \cup D$ that satisfies all the constraints introduced so far, and for all $d_x(i,j,k,l) \in D$, if $\mathcal{V}(d_x(i,j,k,l)) = 1$, then for every other variables $d_x(i,j,k',l'), d_x(i',j',k,l) \in D$ it holds that $\mathcal{V}(d_x(i,j,k',l')) = \mathcal{V}(d_x(i',j',k,l)) = 0$. Nevertheless, it is desired to impose this fact with additional (redundant) constraints that will, hopefully, improve the solving time of this Pseudo-Boolean Optimization problem:

- For all $x \in \{0,1\}$ and for all $i,j \in \mathbb{N}$ such that $1 \leq i < j \leq n_x$, add the following constraint:

$$\sum_{1 \leq k < n_{\overline{x}}} \sum_{k < l \leq n_{\overline{x}}} d_x(i,j,k,l) \leq 1 \tag{5.36}$$

- For all $x \in \{0,1\}$ and for all $k,l \in \mathbb{N}$ such that $1 \leq k < l \leq n_{\overline{x}}$, add the following constraint:

$$\sum_{1 \leq i < n_x} \sum_{i < j \leq n_x} d_x(i,j,k,l) \leq 1 \tag{5.37}$$

Every pair of consecutive genes $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p]$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p+1]$ in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ contributes to $S_{0,1}$ with the value of $|\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p] - \mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p+1]|$ [8]. Also, it is easy to see that for any valoration $\mathcal{V}$ over $A \cup B \cup C \cup D$, that satisfies the constraints introduced so far, $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ is a valid matching (exemplar or maximum) between G_0 and G_1 and there is a one to one correspondence between variables $d_0(i,j,k,l) \in D$ such that $\mathcal{V}(d_0(i,j,k,l)) = 1$, and consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$. Moreover, if $d_0(i,j,k,l)$ is assigned to the value "true", then either $G_0[i]$ is matched to $G_1[k]$ and $G_0[j]$ is matched to $G_1[l]$, or $G_0[i]$ is matched to $G_1[l]$ and $G_0[j]$ is matched to $G_1[k]$. In either case, if a variable $d_0(i,j,k,l)$ is assigned to the value "true" then, the consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ to which $d_0(i,j,k,l)$ corresponds, will contribute to $S_{0,1}$ with an amount equal to the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j, plus one (see figure 5.1). Clearly, this equals $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j.

Therefore, for any valoration $\mathcal{V}$ over $A \cup B \cup C \cup D$ that satisfies the constraints introduced so far, there is a one to one correspondence between variables $d_0(i,j,k,l) \in D$ such that $\mathcal{V}(d_0(i,j,k,l)) = 1$, and consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ (where $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$) and:

- the number $S_{0,1}$ (defined over $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$) equals the sum, over all variable $d_0(i,j,k,l) \in D$ such that $\mathcal{V}(d_0(i,j,k,l)) = 1$, of $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j.

Following a similar reasoning, the number $S_{1,0}$ (defined over $\mathcal{M}_{f^1_{\mathcal{M}}}(G_0)$) equals the sum, over all variable $d_1(i,j,k,l) \in D$ such that $\mathcal{V}(d_1(i,j,k,l)) = 1$, of $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_1, strictly between positions i and j [9].

Variables $e_x(i,j,p) \in E$ are introduced in order to count the number of non-$\mathcal{M}$-saturated genes in G_x, strictly between positions i and j, when it is the case that some variable $d_x(i,j,k,l) \in D$ is assigned to the value "true". A variable $e_x(i,j,p) \in E$ will be assigned to the value "true" if and only if:

1. some variable $d_x(i,j,k,l)$ is assigned to the value "true".

2. $b_x(i)$ and $b_x(j)$ are assigned to the value "true".

[8] Recall that G_0 and G_1 are unsigned genomes, therefore $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ can be viewed as an (unsigned) permutation in $S_{|\mathcal{M}|}$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p]$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)[p+1]$ can be viewed as natural number.

[9] Recall that variables $d_1(i,j,k,l) \in D$ are such that $1 \leq i < j \leq n_1$ and $1 \leq k < l \leq n_0$.

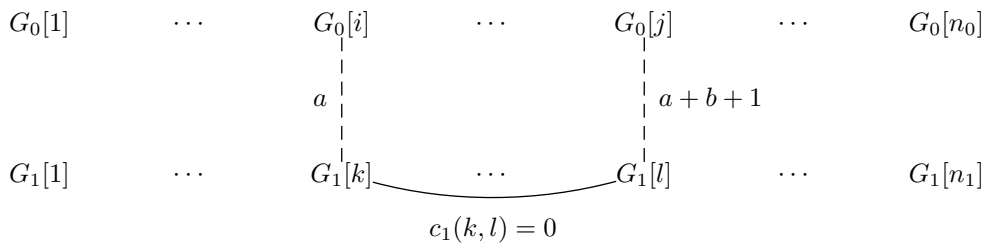


Figure 5.1: Variable $d_0(i,j,k,l)$ is assigned to the value "true" because $c_1(k,l)$, $a(i,k)$ and $a(j,l)$ are assigned to the value "true". a is the number with which $G_0[i]$ and $G_1[k]$ will be renamed in the construction of $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$. It is clear to see that a corresponds to the number genes in G_0, in a position p such that $1 \leq p \leq i$, that are matched by $\mathcal{M}$ (see definitions 4.1.4, 4.1.6 and figure 4.1). From this, it is clear that $a+b+1$ is the number with which $G_0[j]$ and $G_1[l]$ will be renamed in the construction of $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, where b is the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j. Therefore, since $G_1[k]$ and $G_1[l]$ correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, variable $d_0(i,j,k,l)$ will contribute do $S_{0,1}$ with the value $|(a+b+1)-a| = b+1$, i.e., the number of $\mathcal{M}$-saturated genes in G_0, strictly between positions i and j, plus one (where $\mathcal{M}$ is the matching defined by variables $a(i,k)$). If it were the opposite case of having $a(i,l)$ and $a(j,k)$ assigned to the value "true", variable $d_0(i,j,k,l)$ will contribute to $S_{0,1}$ with the same amount, because $|a-(a+b+1)| = |-b-1| = b+1$.

3. $b_x(p)$ is assigned to the value "false".

To ensure this, for all $x \in \{0,1\}$ and for all $i,j,p \in \mathbb{N}$ such that $1 \leq i < p < j \leq n_x$ add the following constraints:

$$\begin{aligned}
\sum_{1\leq k<n_{\overline{x}}} \sum_{k<l\leq n_{\overline{x}}} d_x(i,j,k,l) + b_x(i) + b_x(j) + (1-b_x(p)) - e_x(i,j,p) \leq 3 \\
e_x(i,j,p) - \sum_{1\leq k<n_{\overline{x}}} \sum_{k<l\leq n_{\overline{x}}} d_x(i,j,k,l) \leq 0 \\
e_x(i,j,p) - b_x(i) \leq 0 \\
e_x(i,j,p) - b_x(j) \leq 0 \\
e_x(i,j,p) - (1-b_x(p)) \leq 0
\end{aligned} \tag{5.38}$$

Notice that variables $b_x(i)$ and $b_x(j)$ are required to be assigned to the value "true" for a variable $d_x(i,j,k,l) \in D$ to be assigned to the value "true", therefore, condition 2 is redundant. However, including variables $b_x(i)$ and $b_x(j)$ in these constraints will, hopefully, improve the solving time of this Pseudo-Boolean Optimization problem.

Henceforth, let $\mathcal{V}$ be a valoration over $A \cup B \cup C \cup D \cup E$ that satisfies all the constrains introduced in this model and $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$.

The objective is to minimize the summed adjacency disruption number between genomes $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, which equals the sum of the two number $S_{0,1}$ and $S_{1,0}$. We now explain how the numbers $S_{0,1}$ and $S_{1,0}$ can be expressed as linear functions of the variables $d_x(i,j,k,l) \in D$ and $e_x(i,j,p) \in E$.

For $S_{0,1}$, the idea is to consider all possible pairs of positions i and j, where $1 \leq i < j \leq n_0$, and

determine which verify the following condition:

A) there exists a variable $d_0(i,j,k,l) \in D$ such that $\mathcal{V}(d_0(i,j,k,l)) = 1$.

If a given pair of positions i and j verify condition A), one knows that $G_0(i)$ and $G_0(j)$ are matched to some genes $G_1(k)$ and $G_1(l)$ (where $k < l$ or $l < k$) that correspond to consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$. Every such pair will add $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j, to $S_{0,1}$. However, a pair of positions i and j that does not verify condition A) should simply add 0 to $S_{0,1}$. Both cases are attained by adding to $S_{0,1}$ the following value:

$$\begin{aligned} S_{0,1}(i,j) \stackrel{def}{=} & \quad (j-i) \cdot \left(\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \mathcal{V}(d_0(i,j,k,l)) \right) - \sum_{p=i+1}^{j-1} \mathcal{V}(e_0(i,j,p)) = \\ = & \quad \left(\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} (j-i) \cdot \mathcal{V}(d_0(i,j,k,l)) \right) - \sum_{p=i+1}^{j-1} \mathcal{V}(e_0(i,j,p)) \end{aligned} \tag{5.39}$$

1. If a pair of positions i and j, where $1 \leq i < j \leq n_0$, verify condition A) then, it holds that $\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \mathcal{V}(d_0(i,j,k,l)) = 1$ and consequently, variables $e_0(i,j,p)$, for $i < p < j$, will verify that $\mathcal{V}(e_0(i,j,p)) = 1$ if, and only if, $G_0[p]$ is non-$\mathcal{M}$-saturated. So, when i and j verify condition A), $S_{0,1}(i,j)$ equals $(j-i)$ minus the number of non-$\mathcal{M}$-saturated genes in G_0, strictly between positions i and j.

2. If a pair of positions i and j, where $1 \leq i < j \leq n_0$, does not verify condition A), then, it holds that $\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} \mathcal{V}(d_0(i,j,k,l)) = 0$ and consequently, variables $e_0(i,j,p)$, for $i < p < j$, will verify that $\mathcal{V}(e_0(i,j,p)) = 0$ and so, $S_{0,1}(i,j)$ equals 0.

Therefore, and since there is a one to one correspondence between variables $d_0(i,j,k,l)$ for which $\mathcal{V}(d_0(i,j,k,l)) = 1$ and consecutive genes in $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$, we have that:

$$\begin{aligned} S_{0,1} = & \sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} S_{0,1}(i,j) = \\ = & \sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} \left(\left(\sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} (j-i) \cdot \mathcal{V}(d_0(i,j,k,l)) \right) - \sum_{p=i+1}^{j-1} \mathcal{V}(e_0(i,j,p)) \right) \end{aligned} \tag{5.40}$$

By similar reasoning over all possible pairs of positions i an j, where $1 \leq i < j \leq n_1$, we have that[10]:

$$S_{1,0} = \sum_{i=1}^{n_1-1} \sum_{j=i+1}^{n_1} \left(\left(\sum_{k=1}^{n_0-1} \sum_{l=k+1}^{n_0} (j-i) \cdot \mathcal{V}(d_1(i,j,k,l)) \right) - \sum_{p=i+1}^{j-1} \mathcal{V}(e_1(i,j,p)) \right) \tag{5.41}$$

[10]Recall that variables $d_1(i,j,k,l) \in D$ are such that $1 \leq i < j \leq n_1$ and $1 \leq k < l \leq n_0$, and variables $e_1(i,j,p) \in E$ are such that $1 \leq i < j \leq n_1$ and $i < p < j$.

Therefore, the cost function of this Pseudo-Boolean Optimization problem is the following[11]:

$$\sum_{x=0}^{1}\left(\sum_{i=1}^{n_x-1}\sum_{j=i+1}^{n_x}\left(\left(\sum_{k=1}^{n_{\overline{x}}-1}\sum_{l=k+1}^{n_{\overline{x}}}(j-i)\cdot d_x(i,j,k,l)\right)-\sum_{p=i+1}^{j-1}e_x(i,j,p)\right)\right) \tag{5.42}$$

Finding an optimal solution $\mathcal{V}$ of this Pseudo-Boolean Optimization problem defines a valid matching $\mathcal{M} \stackrel{def}{=} \{(i,j) : \mathcal{V}(a(i,j)) = 1\}$ (exemplar or maximum) between G_0 and G_1 for which the summed adjacency disruption number between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ is minimum. The summed adjacency disruption number between $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ equals the value of the cost function for $\mathcal{V}$, i.e.:

$$\sum_{x=0}^{1}\left(\sum_{i=1}^{n_x-1}\sum_{j=i+1}^{n_x}\left(\left(\sum_{k=1}^{n_{\overline{x}}-1}\sum_{l=k+1}^{n_{\overline{x}}}(j-i)\cdot \mathcal{V}(d_x(i,j,k,l))\right)-\sum_{p=i+1}^{j-1}\mathcal{V}(e_x(i,j,p))\right)\right) \tag{5.43}$$

This Pseudo-Boolean Optimization problem has $O(n_0^2 n_1^2)$ variables and $O(n_0^2 n_1^2)$ constraints. A compact description of this encoding is presented in table 8.3 in the annexes.

Variable reduction rules

Here we introduce variable reduction rules for variables $d_x(i,j,k,l) \in D$ and variables $e_x(i,j,p) \in E$. The rules regarding variables $d_x(i,j,k,l) \in D$ are similar to those used for the number of breakpoints. The only difference between variables $d(i,j,k,l)$ of breakpoints and $d_x(i,j,k,l) \in D$ in the summed adjacency disruption number, is that $c_x(i,j)$ is not required to be assigned to the values "true" for $d_x(i,j,k,l)$ to be assigned to the values "true". Rules were adapted accordingly. It is important to recall, for the understanding of these rules, that variables $d_x(i,j,k,l) \in D$ are such that $1 \leq i < j \leq n_x$ and $1 \leq k < l \leq n_{\overline{x}}$.

The rules for variables $e_x(i,j,p) \in E$ state that if $G_x[p]$ must be matched, then $e_x(i,j,p)$ will be assigned to the value "false" in any solution of the PBO problem. Since variables $e_x(i,j,p) \in E$ contribute negatively to the cost function, these variables are discarded. In order to avoid ambiguities in these rules, define $occ_{G_x}(a,i,j) = 0$ if $i > j$.

Exemplar model. If the exemplar model is considered, delete from D all variables $d_x(i,j,k,l)$ for which any of the following conditions hold:

1. $G_x[i] = G_x[j]$.
2. $G_{\overline{x}}[k] = G_{\overline{x}}[l]$.

or, the following condition holds:

1. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_{\overline{x}}}(G_{\overline{x}}[p], k+1, l-1) = occ_{G_{\overline{x}}}(G_{\overline{x}}[p])$.

Delete from E all variables $e_x(i,j,p)$ for which $occ_{G_x}(G_x[p]) = 1$.

[11] In this encoding, the idea is to minimize a cost function.

Maximum model. If the maximum model is considered, delete from D all variables $d_x(i,j,k,l)$ for which any of the following conditions hold:

1. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_{\overline{x}}}(G_{\overline{x}}[p]) \leq occ_{G_x}(G_{\overline{x}}[p])$.
2. There exists $p \in \mathbb{N}$ such that $k < p < l$ and $occ_{G_{\overline{x}}}(G_{\overline{x}}[p], k{+}1, l{-}1) > |occ_{G_0}(G_{\overline{x}}[p]) - occ_{G_1}(G_{\overline{x}}[p])|$.

Delete from E all variables $e_x(i,j,p)$ for which $occ_{G_x}(G_x[p]) \leq occ_{G_{\overline{x}}}(G_x[p])$.

Variable value setting rules

The variable value setting rules used in of the Pseudo-Boolean Optimization encoding for the number of breakpoints, for variables $a(i,k) \in A$ and $b_x(i) \in B$ are also applied to this encoding. The rule for variables $d(i,j,k,l)$ has been adequately adapted to the rule for variables $d_x(i,j,k,l) \in D$ and new rule is introduced for variables $E_x(i,j,p) \in E$. These rules are to be applied after the variable reduction rules. Constraints used for defining the values of variables that are captured by these rules will be excluded when building the Pseudo-Boolean Optimization instance.

D variables. Let $d_x(i,j,k,k+1) \in D$ be such that one of the following conditions hold[12]:

1. $G_x[i] = G_{\overline{x}}[k]$ and $G_x[j] = G_{\overline{x}}[k+1]$.
2. $G_x[i] = G_{\overline{x}}[k+1]$ and $G_x[j] = G_{\overline{x}}[k]$.

and also, it holds that $occ_{G_0}(G_{\overline{x}}[k]) = occ_{G_1}(G_{\overline{x}}[k]) = occ_{G_0}(G_{\overline{x}}[k+1]) = occ_{G_1}(G_{\overline{x}}[k+1]) = 1$ (i.e., $G_x[i]$, $G_x[j]$, $G_{\overline{x}}[k]$ and $G_{\overline{x}}[k+1]$ are genes that belong to gene families without duplicates in G_0 and G_1). Then, add the following constraint:

$$d_x(i,j,k,k+1) = 1 \tag{5.44}$$

E variables. Let $e_x(i,j,p) \in D$ be such that $occ_{G_0}(G_x[i]) = occ_{G_1}(G_x[i]) = occ_{G_0}(G_x[j]) = occ_{G_1}(G_x[j]) = 1$ (i.e., $G_x[i]$ and $G_x[j]$ are genes that belong to gene families without duplicates in G_0 and G_1). Let k and l be the numbers such that $G_{\overline{x}}[k] = G_x[i]$ and $G_{\overline{x}}[l] = G_x[j]$. Then, if $k = l+1$ or $l = k+1$, add the following constraint:

$$e_x(i,j,p) + b_x(p) = 1 \tag{5.45}$$

Regarding variables $e_x(i,j,p) \in E$, this rule states that if $G_x[i]$ and $G_x[j]$ are non duplicated genes, whose occurrences in $G_{\overline{x}}$ take place in positions k and l that are consecutive (i.e., $|k-l| = 1$), then, if $l = k+1$, variable $d_x(i,j,k,l)$ must be assigned to the value "true", or, if $k = l+1$, variable $d_x(i,j,l,k)$ must be assigned to the value "true". Therefore, $e_x(i,j,p)$ should be assigned to the value "true" if, and only if, $b_x(p)$ is assigned to the value false and the constraint in equation 5.45 is added.

[12]Recall that variables $d_x(i,j,k,l) \in D$ are such that $1 \leq i < j \leq n_x$ and $1 \leq k < l \leq n_{\overline{x}}$.

6

Experimental results

The experimental evaluation of the Pseudo-Boolean Optimization (PBO) encodings presented in section 5.2 was preformed on the same dataset used in [2, 1] consisting in genomes of γ-Protobacteria. In this chapter we present the obtained results and their analysis.

6.1 The data set

The data set of γ-Protobacteria genomes used in [2, 1] was originally studied by Lerat et al. [24] and is composed of thirteen genomes. Only twelve of these genomes were considered, because the thirteenth genome (*Vibrio cholerae*) is separated in two chromosomes and therefore is not in the scope of this study. We list the twelve genomes together with their GenBank accession number[1]:

1. *Buchnera aphidicola APS* (Baphi): GenBank accession number NC_002528.
2. *Escherichia coli K-12* (Ecoli): GenBank accession number NC_000913.
3. *Haemophilus influenzae Rd* (Haein): GenBank accession number NC_000907.
4. *Pseudomonas aeruginosa PAO1* (Paeru): GenBank accession number NC_002516.
5. *Pasteurella multocida Pm70* (Pmult): GenBank accession number NC_002663.
6. *Salmonella typhimurium LT2* (Salty): GenBank accession number NC_003197.
7. *Wigglesworthia glossinidia brevipalpis* (Wglos): GenBank accession number NC_004344.
8. *Xanthomonas axonopodis pv. citri 306* (Xaxon): GenBank accession number NC_003919.
9. *Xanthomonas campestris* (Xcamp): GenBank accession number NC_003902.
10. *Xylella fastidiosa 9a5c* (Xfast): GenBank accession number NC_002488.

[1] GenBank is the United States of America National Institutes of Health (NIH) genetic sequence database. It consists of an annotated collection of all publicly available DNA sequences. These can be accessed at `http://www.ncbi.nlm.nih.gov/Genbank`.

11. *Yersinia pestis CO92* (Ypest-CO92): GenBank accession number NC_003143.

12. *Yersinia pestis KIM* (Ypest-KIM): GenBank accession number NC_004088.

The partition of the whole gene set of these genomes into gene families and the representation of each genome as a signed sequence over the obtained alphabet of gene families (as described in section 2.3) was done by Blin et al. [7]. The dataset was provided by Angibaud et al. [2, 1].

The main properties of each genome in this dataset, namely the number of genes and gene families, as well as the percentage of duplicate genes, is presented in table 6.1. The size of the genomes after the preprocessing steps (described in section 5.2.1) are presented in tables 8.4, 8.5 and 8.6 in the annexes.

Genome	Number of genes	Number of gene families	Percentage of duplicate genes
Baphi	564	549	2.66
Ecoli	4183	3423	18.17
Haein	1709	1531	10.42
Paeru	5540	4500	18.77
Pmult	2015	1811	10.42
Salty	4203	3456	17.77
Wglos	653	627	3.98
Xaxon	4192	3634	13.31
Xcamp	4029	3468	13.92
Xfast	2680	2346	12.46
Ypest-CO92	3599	3021	16.06
Ypest-KIM	3879	3236	16.58

Table 6.1: The number of genes and gene families and the percentage of duplicate genes in each genome of the γ-Protobacteria dataset.

6.2 Computations and validation

Experiments were run on an Intel Xeon 5160 server (3.0GHZ, 1333Mhz, 4GB) running Red Hat Enterprise Linux WS 4. We discarded instances which resulted in files with size greater than 1 Gigabyte and used 10 hours as the cut off value for solving the remaining instances. Results were validated in two ways:

1. By verifying that in all obtained solutions $\mathcal{V}$ (optimal or sub-optimal) all variables were assigned to the correct value, according to what was specified in each encoding. First, we verify that $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ defines a valid matching (of the chosen model) between G_0 and G_1. It is easy to see that the correct value of the remaining variables (of each encoding) is determined by the values assigned to variables $a(i,k)$ and therefore this verification step is preformed[2].

[2]There is an exception for sub-optimal solutions of PBO instances for common intervals, as was mentioned in section 5.2.2.

2. By verifying that in all obtained solutions $\mathcal{V}$ (optimal or sub-optimal) the value of the objective function equaled the value of $d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ (where $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ is the (valid) matching between G_0 and G_1 defined by solution $\mathcal{V}$). This is done by building the $\mathcal{M}$-reduced genomes $\mathcal{M}_{f^0_{\mathcal{M}}}(G_0)$ and $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1)$ and then computing the value of $d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G_0), \mathcal{M}_{f^0_{\mathcal{M}}}(G_1))$ and comparing it to the value of the objective function in the solution $\mathcal{V}$.

On a first evaluation, we used *minisat+*[3] [15], which was able to solve most breakpoints and common intervals instances. Later, we ran *cplex*[4], which is an Integer Linear Optimization (ILO) solver. *Cplex* preformed better than *minisat+* and provided excellent results on our PBO instances for breakpoints and common intervals.

The PBO solver *minisat+* is based on converting PBO instances to a sequence of SAT instances and solving the problem using a SAT solver [15]. However, it turned out that for our PBO instances, the conversion to SAT carried out by *minisat+* resulted in an explosion of clauses in the SAT instances. We find this to be the cause of the inefficiency of *minisat+* for solving our instances.

On the other hand, *cplex* is a branch and bound algorithm for solving ILO problems that uses linear optimization relaxation for obtaining upper (or lower) bounds on optimal solutions in order to narrow the search space. In the case of linear optimization relaxation for $\{0,1\}-$ILO problems, variables are allowed to take any value between 0 and 1. If the goal is to maximize an objective function (or minimize a cost function), the solution of the relaxed linear optimization problem provides an upper bound (respectively, lower bound) for the $\{0,1\}-$ILO problem and sub-optimal solutions provide lower bounds (respectively, upper bounds) for the $\{0,1\}-$ILO problem [27]. We found that the linear optimization relaxation preformed by *cplex* was very effective on solving our instances, begin able to efficiently prune the search tree. We believe this is the reason for the better performance of *cplex* in these instances.

6.3 Results and analysis

Sections 6.3.1, 6.3.2 and 6.3.3 present the results and analysis of the PBO instances for the number of common intervals, the number of breakpoints and the summed adjacency disruption number (SAD), respectively. In each section we present, for each matching model and each pair of genomes G and H of the considered dataset, the optimal (or sub-optimal) value found for the correspondent PBO instance. We do not present the actual (optimal or sub-optimal) solution found for the problems $OPT_e(G,H,d)$ and $OPT_m(G,H,d)$, i.e., the matching $\mathcal{M} \stackrel{def}{=} \{(i,k) : \mathcal{V}(a(i,k)) = 1\}$ between G and H defined by the solution $\mathcal{V}$ of the PBO instance, due to space constraints. Also, since each considered measure is symmetric (see proposition 4.2.2), we fill only half of each table of results. The remaining results are computed by symmetry due to propositions 4.1.6 and 4.1.7.

[3] Available at `http://minisat.se/MiniSat+.html`.
[4] Version 11.2. Available at `http://www.ilog.com/products/cplex`

As in the PBO encoding for breakpoints (see section 5.2.3), border genes were added to each genome prior to the generation of each instance for common intervals and SAD. Initially, this was done because the authors of the encoding for common intervals [2] did so on later studies and it was desirable to have some base of comparison for the results we obtained. We chose to keep border genes for SAD since we already used them for common intervals and breakpoints.

Notice that each of the PBO encodings produces instances with $O(n_0^2 n_1^2)$ variables and $O(n_0^2 n_1^2)$ constraints when given two genomes of lengths n_0 and n_1. However, there is a significant difference in the size and solving times of the instances generated by these encoding for the data set considered. This is due to the variable reduction rules and variable value fixing rules defined for each encoding. These rules aim at simplifying the instances to a feasible point. However, the rules defined for each model are different, some capturing more variables (thus resulting in simpler instances) than others.

Each optimization problem $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$ was proved (individually for each similarity measure d introduced in section 4.2) to be NP-hard as soon as G or H have duplicate genes and, in the case of SAD, also to be APX-hard (see section 4.4). Moreover, some of the variable reduction rules and variable value setting rules for each measure are based on the existence of gene families that occur only once in G and H. Also, it is clear that, if no gene family has duplicates in G or in H, then there exists only one possible solution for the problems $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$. Furthermore, increasing the presence of duplicate genes in G and H, can greatly increase the number of valid matchings between them (see proposition 4.1.1). Due to all these facts, we believe the number of genes, from gene families that occur in both G and H and occur more than once in either G or H, to be an a feature of some interest for determining the difficulty of solving a PBO instance for the problems $OPT_e(G, H, d)$ or $OPT_m(G, H, d)$. These numbers are presented in table 6.2.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Baphi		85	55	93	56	89	30	60	60	49	91	90
Ecoli	295		640	895	678	1135	321	666	655	477	931	951
Haein	121	364		301	300	379	123	229	218	190	322	325
Paeru	352	1092	652		657	1061	389	1025	1034	617	1019	1036
Pmult	138	397	305	342		407	134	250	239	198	340	348
Salty	290	1145	657	897	697		329	684	680	498	957	975
Wglos	38	111	66	111	68	115		83	83	69	113	114
Xaxon	204	600	384	707	379	601	245		872	498	539	545
Xcamp	199	579	354	688	354	580	238	870		494	513	521
Xfast	110	281	201	308	201	293	122	323	314		260	269
Ypest-CO92	301	816	524	782	543	839	326	570	541	432		870
Ypest-KIM	310	888	564	836	594	908	343	611	582	458	921	

Table 6.2: Number of genes from gene families shared by two γ-Protobacteria genomes and with occurrences greater than 1 in one of the genomes. The number assigned to line x and column y is the number of genes in genome x, whose gene families occur both in genomes x and y and having a number of occurrences greater than 1 in either x or y.

From our results, we have concluded that instances for comparing genomes that share a small number

of such genes (like the instances involving the genome Baphi or the genome Wglos) tend to be ones with smaller sizes and faster solving times, in all the of the considered measures. However, we have found that there can be a great difference both in size and solving time of instances that compare pairs of genomes that have a similar (but large) number of shared duplicate genes. This is particularly clear in some of the instances for the number of common intervals and SAD.

For example, considering the number of common intervals under the maximum model, the instance for comparing the genomes Xcamp and Xaxon involves $870 + 872 = 1742$ duplicate genes, while the one for comparing the genomes Salty and Paeru involves $897 + 1061 = 1958$ duplicate genes. *Cplex* runs out of memory when solving the first instance, but solves the second one in less than 18 seconds. Also, the instance for comparing the genomes Ypest-CO92 and Ypest-KIM involves $921 + 870 = 1791$ duplicate genes but exceeds 1 Gigabyte.

From this, we believe that in every considered problem, the structure of each genome must play an important role in determining the size and solving time of instances for comparing genomes that share a large number of duplicate genes.

6.3.1 Number of common intervals

The solutions for the exemplar and maximum model are presented in tables 6.3 and 6.4. The running times of *cplex* for these instances are presented in tables 8.7 and 8.8 in the annexes.

For the number of common intervals, we found that most generated instances had reasonable size, i.e., less than 1 Gigabyte. Indeed, the only instances that exceeded 1 Gigabyte were the ones for comparing the genomes $Ypest - CO92$ and $Ypest - KIM$ under the exemplar model and the maximum model. The size of the remaining instances summed 2.3 Gigabytes (rounded up).

Cplex solved 126 out of 132 instances for common intervals. The longest running time for an instance was of 63.81 seconds for the exemplar model and 387.32 seconds for the maximum model (both refer to the instances generated for comparing the genomes Salty and Ypest-KIM). This is somewhat of a breakthrough on existing publications because we were able to solve every feasible instance with very decent running times, matching the results obtained in [2] for the maximum model and thus providing all but 3 of the missing solutions. Also, we provide 63 out of 66 solutions for the exemplar model which was pointed as the main bottleneck of the approach [2, 1].

6.3.2 Number of breakpoints

The solutions for the number of adjacencies using the exemplar model and the maximum model are presented in tables 6.5 and 6.6. The number of breakpoints using the exemplar model and maximum model are presented in tables 8.9 and 8.10 in the annexes. The running times of *cplex* for these instances are presented in tables 8.11 and 8.12 in the annexes.

The total set of the PBO instances for breakpoints sums 37 Megabytes (rounded up). This is due mainly to the variable reduction rules for variables $d(i, j, k, l) \in D$ being very strict thus preforming a

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	2889										
Haein	1099	2542									
Paeru	1514	3597	1860								
Pmult	1210	3060	3550	2125							
Salty	2856	†	2563	3553	3067						
Wglos	1268	2302	1060	1538	1185	2309					
Xaxon	1171	2443	1345	3495	1509	2453	1208				
Xcamp	1171	2436	1341	3485	1499	2442	1206	†			
Xfast	966	1841	1211	2190	1360	1847	976	6263	6099		
Ypest-CO92	2583	14390	2443	3586	2988	15285	2301	2376	2369	1801	
Ypest-KIM	2145	14818	2246	3447	2778	15821	2078	2338	2327	1749	*

Table 6.3: Results for the number of common intervals using the exemplar model.
(*) Instance exceeded 1 Gigabyte. (†) *Cplex* ran out of memory.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	2882										
Haein	1109	2784									
Paeru	1524	4067	2036								
Pmult	1224	3342	3936	2337							
Salty	2849	†	2820	4070	3376						
Wglos	1275	2328	1085	1558	1214	2335					
Xaxon	1186	2602	1474	3988	1653	2734	1228				
Xcamp	1186	2594	1461	3944	1633	2721	1226	†			
Xfast	979	1877	1295	2377	1466	1981	994	6971	6773		
Ypest-CO92	2585	15193	2694	4106	3298	16323	2318	2611	2585	1949	
Ypest-KIM	2141	15509	2500	3991	3092	16663	2093	2591	2559	1891	*

Table 6.4: Results for the number of common intervals using the exemplar model.
(*) Instance exceeded 1 Gigabyte. (†) *Cplex* ran out of memory.

drastic reduction of the number of variables (and constraints) included in each PBO instance. *Cplex* preformed extremely well on the breakpoints instances, solving all instances within less than 146.49 seconds for the exemplar model and less than one second for the maximum model.

Comparing to previous publications [1], these results are promising. Not only have we solved all instances for both the exemplar and maximum model (and thus completed the work started in [1]), but we have done so with remarkable running times. Indeed the fact that every instance for the maximum model was solved in less than one second indicates that our implementation of this encoding might be able to deal with larger genomes.

6.3.3 Summed adjacency disruption number

The PBO instances for the summed adjacency disruption number have proved to be harder to solve than the PBO instances for common intervals or breakpoints. This comes as no surprise because

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	368										
Haein	157	489									
Paeru	226	570	301								
Pmult	182	593	755	340							
Salty	367	2465	492	559	597						
Wglos	201	371	159	246	188	372					
Xaxon	183	380	217	536	245	376	189				
Xcamp	183	378	216	536	241	375	189	2880			
Xfast	158	301	191	372	213	300	158	980	969		
Ypest-CO92	352	1559	465	571	582	1560	369	372	369	298	
Ypest-KIM	339	1560	460	566	574	1562	356	368	365	292	2815

Table 6.5: Results for the number of adjacencies using the exemplar model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	377										
Haein	161	550									
Paeru	229	651	333								
Pmult	188	662	849	373							
Salty	376	2874	550	644	670						
Wglos	203	380	165	251	197	381					
Xaxon	188	425	241	609	274	434	194				
Xcamp	188	420	238	596	267	427	194	3256			
Xfast	162	324	205	405	234	325	163	1075	1060		
Ypest-CO92	361	1744	522	671	648	1758	377	420	412	323	
Ypest-KIM	348	1747	517	662	639	1761	364	422	412	315	3327

Table 6.6: Results for the number of adjacencies using the maximum model.

the variable reduction rules are much less strict than the ones for breakpoints (see section 5.2.4) and therefore, the encoding for SAD of the problems $OPT_e(G, H, d)$ and $OPT_m(G, H, d)$ results in much larger instances due to a larger number of variables involved. Nonetheless, we have solved some of the SAD instances, namely some that concern the genomes Baphi and Wglos.

The solutions for the exemplar and maximum model for the genomes Baphi and Wglos are presented in tables 6.7 and 6.8. The running times of *cplex* for these instances are presented in tables 8.13 and 8.14 in the annexes.

Let us note that the results presented in this dissertation are the first exact results obtained on these APX-hard problems and can be used to evaluate the accuracy of non-optimal algorithms which are not guaranteed to provide optimal solutions and which may be developed in the future. Moreover, the sub-optimal solutions found are valid and can be used as upper bounds on the optimal solutions. Additionally, *cplex* provides, for every (sub-optimal) solution, a lower bound using linear optimization relaxation (the parameter *gap* establishes how far apart is this bound from the currently found solution).

Therefore, from our sub-optimal solutions, we also provide upper and lower bounds on optimal solutions. The obtained lower bounds and the parameter *gap* from *cplex* are included, for sub-optimal solutions, in tables 6.7 and 6.8.

We believe that better results can be obtained for this model and that the key for reducing the computation time lies in further research on variable reduction rules.

	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	32058	66510	60896‡	64894	31982	37082	55614	53018	56036	42556‡	43718‡
LB			59554							42057	42977
Gap			2.20%							1.17%	1.69%
Wglos	63496‡	72908	75512‡	74676	63126‡		70272	68628	68304	62398‡	59420‡
LB	62431		73167		61827					61353	58694
Gap	1.68%		3.10%		2.06%					1.67%	1.22%

Table 6.7: Results for the summed adjacency disruption number using the exemplar model.
(‡) Sub-optimal solution. (LB) Lower bounds from *cplex*. (Gap) The *gap* value from *cplex* (the greatest gap in sub-optimal solutions was of 3.10%).

	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	34126‡	69462	64925‡	68336	34054‡	37906	58348	55546	59172	45011‡	45783‡
LB	34014		63281		33860					44002	44840
Gap	0.33%		2.53%		0.57%					2.24%	2.06%
Wglos	67793‡	76669	80666‡	78286	66839‡		74012‡	72045	72140	66620‡	63927‡
LB	64823		75383		64612		73604			63557	60807
Gap	4.38%		6.55%		3.33%		0.55%			4.60%	4.88%

Table 6.8: Results for the summed adjacency disruption number using the maximum model.
(‡) Sub-optimal solution. (LB) Lower bounds from *cplex*. (Gap) The *gap* value from *cplex* (the greatest *gap* in sub-optimal solutions was of 6.55%).

7

Conclusion and future work

In this dissertation we began by studying the approaches for genome rearrangements in order to provide a good background for our research in matching models and similarity measures. We found that research on genome rearrangements was in a mature state: NP-completeness has been proved for the reversal distance and polynomial time algorithms provided for both the signed reversal distance and the genomic distance. However, the formalism of the signed reversal distance problem and of the genomic distance was left with some loose ends that we found adequate to complete. A clearer formalism of each approach was presented and led to the inclusion in this dissertation of proofs of some propositions that were taken for granted in most of the bibliography. Namely, we introduced a formalism for signed permutations, defined the composition for these in an adequate way (i.e., to guarantee that signed reversals had the desired effect when composed, on the right, with another signed permutation) and proved this algebraic structure to be a group. Moreover, we proved the transformation T, introduced in section 3.2.1, to be an injective group homomorphism. This led to clear and straightforward proof of the equivalence between sorting a signed permutation $\overrightarrow{\pi}$ by signed reversals and sorting the permutation $T(\overrightarrow{\pi})$ by legal reversals.

Regarding the problem of finding a matching (of some model) that optimizes a chosen measure, much effort was put into finding a clear definition of the problem and this led to the establishment of some interesting results. Also, some effort was put into finding the scope of applications of the approach. We concluded that the approach of matching models and similarity measures can establish gene order evolutionary distances between genomes, but can also help in the identification of orthologous genes and true exemplar genes in two genomes. Additionally, some effort was put into understanding how genomes come to be represented as signed sequences over alphabets of gene families, and what exactly are duplicate genes.

While researching on the Pseudo-Boolean Optimization encodings, we provided some variable reduction rules and constraint value setting rules (in both exemplar and maximum matchings) for the number of breakpoints and common intervals. Experimental results show that *cplex* preforms extremely well on our instances. Moreover, we completed the work started in the number of breakpoints [1] and

were able to obtain nearly all results for the number of common intervals [2] (we missed 6 out of 132). As for the new generic Pseudo-Boolean Optimization encoding of SAD proposed in this dissertation, we found the instances produced to be much harder than those of common intervals or breakpoints, mostly due to the greater amount of variables that are not eliminated by the variable cutting rules. Nevertheless, optimal solutions were obtained for some instances and these can be used to validate the accuracy of approximation algorithms. Moreover, the sub-optimal solutions found using *cplex* are valid, and provide upper bounds (and also lower bounds) on the optimal solutions.

As future work, we suggest the use of Pseudo-Boolean Optimization encodings and algorithms for the remaining measures and find out if whether an optimal solution, on a given model and measure, can impose a bound on the optimal solutions of other measures. Additionally, applying to this model similar heuristics to those presented [2, 1] should reveal interesting results.

Bibliography

[1] S. Angibaud, G. Fertin, I. Rusu, A. Thévenin, and S. Vialette. A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In *5th RECOMB Comparative Genomics Statellite Workshop*, volume 4751 of *Lecture Notes in Computer Science*, pages 16–29, 2007.

[2] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. How pseudo-boolean programming can help genome rearrangement distance computation. In *4th RECOMB Comparative Genomics Statellite Workshop*, volume 4205 of *Lecture Notes in Computer Science*, pages 75–86, 2006.

[3] V. Bafna and P. A. Pevzner. Sorting by reversals: Genome rearrangements in plant organneles and evolutionary history of x chromosome. *Molecular Biology and Evolution*, 12(2):239–246, 1995.

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

[5] A. Bergeron and J. Stoye. On the similarity of sets of permutations and its application to genome comparison. In *9th Annual International Conference on Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 68–79, 2003.

[6] G. Blin, C. Chauve, and G. Fertin. The breakpoint distance for signed sequences. In *1st Algorithms and Computational Methods for Biochemical and Evolutionary Networks*, volume 3 of *Texts in Algorithms*, pages 3–16, 2004.

[7] G. Blin, C. Chauve, and G. Fertin. Genes order and phylogenetic reconstruction: Application to γ-protobacteria. In *3rd RECOMB Comparative Genomics Satellite Workshop*, volume 3678 of *Lecture Notes in Bio-Informatics*, pages 11–20, 2005.

[8] G. Blin and R. Rizzi. Conserved interval distance computation between non-trivial genomes. In *11th Annual International Conference on Computing and Combinatorics*, volume 3595 of *Lecture Notes in Computer Science*, pages 22–31, 2005.

[9] G. Bourque, Y. Yacef, and N. El-Mabrouk. Maximizing synteny blocks to identify ancestral homologs. In *11th Annual International Conference on Computing and Combinatorics (COCOON)*, volume 3678 of *Lecture Notes in Bio-Informatics*, pages 21–35, 2005.

[10] D. Bryant. The complexity of calculating exemplar distances. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Allignment, and the Evolution of Gene Families*, chapter 3, pages 207–212. Kluwer Academic Publishers, 2000.

[11] A. Caprara. Sorting by reversals is difficult. In *1st RECOMB Comparative Genomics*, pages 75–83, 1997.

[12] A. Caprara and S. kiong Ng. A column-generation based branch-and-bound algorithm for sorting by reversals. In *Mathematical Support for Molecular Biology*, volume 47 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 213–226. AMS Press, 1999.

[13] C. Chauve, G. Fertin, R. Rizzi, and S. Vialette. Genomes containing duplicates are hard to compare. In *2nd International Workshop on Bioinformatics Research and Applications*, volume 3992 of *Lecture Notes in Computer Science*, pages 783–790, 2006.

[14] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):302–315, 2005.

[15] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.

[16] R. L. Fernandes and M. Ricou. *Introdução à Álgebra*. The IST Press, Lisboa, Portugal, 2004.

[17] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: A test case. *Genomics*, 30:299–311, 1995.

[18] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed premutations by reversals). In *Journal of the ACM*, pages 178–189, 1995.

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

[20] N. C. Jones and P. A. Pevzner. *An Introduction to Bioinformatics Algorithms*. The MIT Press, Cambrige, Massachusetts. London, England, 2006.

[21] J. Kececioglu and R. Ravi. Of mice and men: Evolutionary distances between genomes under translocations. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, 1995.

[22] J. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In *5th Annual Symposium on Combinatorial Pattern Matching*, pages 307–325, 1994.

[23] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangements. *Algorithmica*, 13:180–210, 1995.

[24] E. Lerat, V. Daubin, and N. A. Moran. From gene trees to organismal phylogeny in prokaryotes: the case of the γ-proteobacteria. *PLoS Biology*, 1(1):101–109, 2003.

[25] E. Lerat, V. Daubin, H. Ochman, and N. A. Moran. Evolutionary origins of genomic repertoires in bacteria. *PLoS Biology*, 3(5):807–814, 2005.

[26] W.-H. Li, Z. Gu, H. Wang, and A. Nekrutenko. Evolutionary analyses of the human genome. *Nature*, 409:847–849, 2001.

[27] V. M. Manquinho and J. Marques-Silva. Effective lower bounding techniques for pseudo-boolean optimization. In *Conference on Design, Automation and Test in Europe*, pages 660–665, 2005.

[28] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences*, 81(3):814–818, 1984.

[29] J. D. Palmer and L. A. Herbon. Plant mitochondrial dna evolved rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28:87–97, 1988.

[30] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[31] P. A. Pevzner. *Computational Molecular Biology, An Algorithmic Approach*. The MIT Press, Cambrige, Massachusetts. London, England, 1995.

[32] D. Sankoff. Analytical approaches to genomic evolution. *Biochimie*, 75(5):409–413, 1993.

[33] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

[34] D. Sankoff. Gene and genome duplication. *Current Opinion in Genetics & Development*, 11(6):681–684, 2001.

[35] D. Sankoff and L. Haque. Power boost for cluster tests. In *3rd RECOMB Comparative Genomics Statellite Workshop*, volume 3678 of *Lecture Notes in Computer Science*, pages 121–130, 2005.

[36] D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B. F. Lang, and R. Cedergren. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences of the United States of America*, 89(14):6575–6579, 1992.

[37] H. M. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-boolean sat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:165–189, 2006.

[38] K. M. Swenson, M. Marron, and J. V. Earnest-deyoung. Approximating the true evolutionary distance between two genomes. In *7th SIAM Workshop on Algorithm Engineering and Experiments*, pages 121–129, 2005.

[39] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

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

8

Annexes

Proof (Proposition 3.2.1). We prove here that $(\overrightarrow{S_n}, \odot)$ is a group with neutral element $\overrightarrow{I_n} = +1 \cdots +n$ and where the inverse element of a signed permutation $\overrightarrow{\pi} = \langle \pi, s \rangle$ in $\overrightarrow{S_n}$ is the signed permutation $\overrightarrow{\pi}^{-1} = \langle \pi^{-1}, t \rangle$, where, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $t(i) = s(\pi^{-1}(i))$.

Remember that for two signed permutations $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi_2} = \langle \pi_1, s_2 \rangle$ in $\overrightarrow{S_n}$, $\overrightarrow{\pi_1} \odot \overrightarrow{\pi_2} = \langle \pi, s \rangle$, where $\pi = \pi_1 \circ \pi_2$ and:

$$s(i) = \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

The proof is separated in three steps:

1. (*neutral element* $\overrightarrow{I_n}$) Let $\overrightarrow{\pi} = \langle \pi, s \rangle$ be an arbitrary signed permutation in $\overrightarrow{S_n}$ and consider the signed permutation $\overrightarrow{I_n} = \langle I_n, u \rangle$, where, I_n is the identity permutation in S_n and for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$, $u(i) = +$. Then:

$$\overrightarrow{I_n} \odot \overrightarrow{\pi} = \langle I_n \circ \pi, t \rangle = \langle \pi, t \rangle$$

where for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$:

$$t(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s(i) = u(\pi(i)) \\ - & \text{, if } s(i) \neq u(\pi(i)) \end{cases} \quad \stackrel{u(\pi(i))=+}{\Rightarrow} t(i) = \begin{cases} + & \text{, if } s(i) = + \\ - & \text{, if } s(i) = - \end{cases} \quad \Rightarrow t = s$$

Consequently $\overrightarrow{I_n} \odot \overrightarrow{\pi} = \overrightarrow{\pi}$. Also:

$$\overrightarrow{\pi} \odot \overrightarrow{I_n} = \langle \pi \circ I_n, t \rangle = \langle \pi, t \rangle$$

where for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$::

$$t(i) \stackrel{def}{=} \begin{cases} + & \text{, if } u(i) = s(I_n(i)) \\ - & \text{, if } u(i) \neq s(I_n(i)) \end{cases} \stackrel{\substack{u(i)=+ \\ I_n(i)=i}}{\Rightarrow} t(i) = \begin{cases} + & \text{, if } + = s(i) \\ - & \text{, if } - = s(i) \end{cases} \Rightarrow t = s$$

Consequently, $\overrightarrow{\pi} \odot \overrightarrow{I_n} = \overrightarrow{\pi} = \overrightarrow{I_n} \odot \overrightarrow{\pi}$ and $\overrightarrow{I_n}$ is the neutral element of $(\overrightarrow{S_n}, \odot)$. ■

2. (*associativity of* $\odot$) Let $\overrightarrow{\pi}_1 = \langle \pi_1, s_1 \rangle$, $\overrightarrow{\pi}_2 = \langle \pi_2, s_2 \rangle$ and $\overrightarrow{\pi}_3 = \langle \pi_3, s_3 \rangle$ be three arbitrary signed permutations in $\overrightarrow{S_n}$. The aim here is to prove that the signed permutations $\overrightarrow{\pi}_1 \odot (\overrightarrow{\pi}_2 \odot \overrightarrow{\pi}_3)$ and $(\overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2) \odot \overrightarrow{\pi}_3$ are equal. Define:

   - $\overrightarrow{\pi}_4 = \overrightarrow{\pi}_2 \odot \overrightarrow{\pi}_3 = \langle \pi_4, s_4 \rangle$, where $\pi_4 = \pi_2 \circ \pi_3$ and:

$$s_4(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s_3(i) = s_2(\pi_3(i)) \\ - & \text{, if } s_3(i) \neq s_2(\pi_3(i)) \end{cases}$$

   - $\overrightarrow{\pi}_5 = \overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_4 = \overrightarrow{\pi}_1 \odot (\overrightarrow{\pi}_2 \odot \overrightarrow{\pi}_3) = \langle \pi_5, s_5 \rangle$, where $\pi_5 = \pi_1 \circ (\pi_2 \circ \pi_3)$ and:

$$s_5(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s_4(i) = s_1(\pi_4(i)) \\ - & \text{, if } s_4(i) \neq s_1(\pi_4(i)) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \quad s_5(i) = \begin{cases} + & \text{, if } \big(s_1(\pi_2(\pi_3(i))) = + \wedge s_3(i) = s_2(\pi_3(i))\big) \vee \\ & \quad \big(s_1(\pi_2(\pi_3(i))) = - \wedge s_3(i) \neq s_2(\pi_3(i))\big) \\ - & \text{, if } \big(s_1(\pi_2(\pi_3(i))) = + \wedge s_3(i) \neq s_2(\pi_3(i))\big) \vee \\ & \quad \big(s_1(\pi_2(\pi_3(i))) = - \wedge s_3(i) = s_2(\pi_3(i))\big) \end{cases}$$

   - $\overrightarrow{\pi}_6 = \overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2 = \langle \pi_6, s_6 \rangle$, where $\pi_6 = \pi_1 \circ \pi_2$ and:

$$s_6(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

- $\overrightarrow{\pi}_7 = \overrightarrow{\pi}_6 \odot \overrightarrow{\pi}_3 = (\overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2) \odot \overrightarrow{\pi}_3 = \langle \pi_7, s_7 \rangle$, where $\pi_7 = (\pi_1 \circ \pi_2) \circ \pi_3) = \pi_5$ and:

$$s_7(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s_3(i) = s_6(\pi_3(i)) \\ - & \text{, if } s_3(i) \neq s_6(\pi_3(i)) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \quad s_7(i) = \begin{cases} + & \text{, if } \big(s_3(i) = + \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))\big) \vee \\ & \quad \big(s_3(i) = - \wedge s_2(\pi_3(i)) \neq s_1(\pi_2(\pi_3(i)))\big) \\ - & \text{, if } \big(s_3(i) = + \wedge s_2(\pi_3(i)) \neq s_1(\pi_2(\pi_3(i)))\big) \vee \\ & \quad \big(s_3(i) = - \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))\big) \end{cases}$$

We now prove that the signed permutations $\overrightarrow{\pi_5} = \langle \pi_5, s_5 \rangle$ and $\overrightarrow{\pi_7} = \langle \pi_7, s_7 \rangle$ are equal, i.e., that $\pi_5 = \pi_7$ and $s_5 = s_7$:

- $(\pi_5 = \pi_7)$ This is a direct consequence of the commutativity of the composition of permutations in S_n:

$$\pi_5 = \pi_1 \circ (\pi_2 \circ \pi_3) = (\pi_1 \circ \pi_2) \circ \pi_3 = \pi_7$$

- $(s_5 = s_7)$ We prove now that for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $s_5(i) = + \Leftrightarrow s_7(i) = +$. Since the sign function s for the composition of two signed permutations is defined as $+$ and $-$ in complementary cases, a trivial corollary of this proof is that for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $s_5(i) = - \Leftrightarrow s_7(i) = -$, and that therefore, $s_5 = s_7$. Let $i \in \mathbb{N}$ such that $1 \leq i \leq n$, then:

  (a) $(s_5(i) = + \Rightarrow s_7(i) = +)$ Suppose that $s_5(i) = +$. Then, either condition i or condition ii holds:

  i. $s_1(\pi_2(\pi_3(i))) = + \wedge s_3(i) = s_2(\pi_3(i))$

  ii. $s_1(\pi_2(\pi_3(i))) = - \wedge s_3(i) \neq s_2(\pi_3(i))$

  Suppose that condition i holds. Then $s_1(\pi_2(\pi_3(i))) = +$ and $s_3(i) = s_2(\pi_3(i))$.

  – Suppose that $s_3(i) = +$. Then $s_2(\pi_3(i)) = s_3(i) = +$ and it holds that:

$$s_3(i) = + \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))$$

  Consequently, $s_5(i) = +$.

  – Suppose that $s_3(i) = -$. Then $s_2(\pi_3(i)) = s_3(i) = -$ and it holds that:

$$s_3(i) = - \wedge s_2(\pi_3(i)) \neq s_1(\pi_2(\pi_3(i)))$$

  Consequently, $s_5(i) = +$.

  Suppose that condition ii holds. Then $s_1(\pi_2(\pi_3(i))) = -$ and $s_3(i) \neq s_2(\pi_3(i))$.

– Suppose that $s_3(i) = +$. Then $s_2(\pi_3(i)) \neq s_3(i) = +$, $s_2(\pi_3(i)) = -$ and it holds that:

$$s_3(i) = + \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))$$

Consequently, $s_5(i) = +$.

– Suppose that $s_3(i) = -$. Then $s_2(\pi_3(i)) \neq s_3(i) = -$, $s_2(\pi_3(i)) = +$ and it holds that:

$$s_3(i) = - \wedge s_2(\pi_3(i)) \neq s_1(\pi_2(\pi_3(i)))$$

Consequently, $s_5(i) = +$.

(b) $(s_7(i) = + \Rightarrow s_5(i) = +)$ Suppose that $s_7(i) = +$. Then, either condition i or condition ii holds:

i. $s_3(i) = + \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))$

ii. $s_3(i) = - \wedge s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))$

Suppose that condition i holds. Then $s_3(i) = +$ and $s_2(\pi_3(i)) = s_1(\pi_2(\pi_3(i)))$.

– Suppose that $s_2(\pi_3(i)) = +$. Then $s_1(\pi_2(\pi_3(i))) = s_2(\pi_3(i)) = +$ and it holds that:

$$s_1(\pi_2(\pi_3(i))) = + \wedge s_3(i) = s_2(\pi_3(i))$$

Consequently, $s_5(i) = +$.

– Suppose now that $s_2(\pi_3(i)) \neq +$. Then $s_2(\pi_3(i)) = -$, $s_1(\pi_2(\pi_3(i))) = s_2(\pi_3(i)) = -$ and it holds that:

$$s_1(\pi_2(\pi_3(i))) = - \wedge s_3(i) \neq s_2(\pi_3(i))$$

Consequently, $s_5(i) = +$.

Suppose that condition ii holds. Then $s_3(i) = -$ and $s_2(\pi_3(i)) \neq s_1(\pi_2(\pi_3(i)))$.

– Suppose that $s_2(\pi_3(i)) = +$. Then $s_1(\pi_2(\pi_3(i))) \neq s_2(\pi_3(i)) = +$, $s_1(\pi_2(\pi_3(i))) = -$ and it holds that:

$$s_1(\pi_2(\pi_3(i))) = - \wedge s_3(i) \neq s_2(\pi_3(i))$$

Consequently, $s_5(i) = +$.

– Suppose now that $s_2(\pi_3(i)) \neq +$. Then $s_2(\pi_3(i)) = -$, $s_1(\pi_2(\pi_3(i))) \neq s_2(\pi_3(i)) = -$, $s_1(\pi_2(\pi_3(i))) = +$ and it holds that:

$$s_1(\pi_2(\pi_3(i))) = + \wedge s_3(i) = s_2(\pi_3(i)$$

Consequently, $s_5(i) = +$.

Consequently, $\overrightarrow{\pi_5} = \overrightarrow{\pi}_1 \odot (\overrightarrow{\pi}_2 \odot \overrightarrow{\pi}_3) = (\overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2) \odot \overrightarrow{\pi}_3 = \overrightarrow{\pi_7}$ ■

3. (*existence of inverse*) Let $\overrightarrow{\pi} = \langle \pi, s \rangle$ be a signed permutation in $\overrightarrow{S_n}$. consider the signed permutations $\overrightarrow{I_n} = \langle I_n, u \rangle$ and $\overrightarrow{\pi}^{-1} = \langle \pi^{-1}, t \rangle$, where, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$, $t(i) = s(\pi^{-1}(i))$ and $u(i) = +$. Then:

$$\overrightarrow{\pi} \odot \overrightarrow{\pi}^{-1} = \langle \pi \circ \pi^{-1}, k \rangle = \langle I_n, k \rangle$$

where for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$:

$$k(i) \stackrel{def}{=} \begin{cases} + & \text{, if } t(i) = s(\pi^{-1}(i)) \\ - & \text{, if } t(i) \neq s(\pi^{-1}(i)) \end{cases}$$

Since by the definition of $\overrightarrow{\pi}^{-1}$, for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$, $t(i) = s(\pi^{-1}(i))$, then, for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$, $k(i) = +$ and $k = u$. Consequently $\overrightarrow{\pi} \odot \overrightarrow{\pi}^{-1} = \overrightarrow{I_n}$. Also:

$$\overrightarrow{\pi}^{-1} \odot \overrightarrow{\pi} = \langle \pi^{-1} \circ \pi, k \rangle = \langle I_n, k \rangle$$

where for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$:

$$k(i) \stackrel{def}{=} \begin{cases} + & \text{, if } s(i) = t(\pi(i)) \\ - & \text{, if } s(i) \neq t(\pi(i)) \end{cases}$$

Since by the definition of $\overrightarrow{\pi}^{-1}$, for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$, $t(i) = s(\pi^{-1}(i))$, then $t(\pi(i)) \stackrel{def}{=} s(\pi^{-1}(\pi(i))) = s(i)$. Therefore, for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$, $k(i) = +$ and $k = u$. Consequently, $\overrightarrow{\pi}^{-1} \odot \overrightarrow{\pi} = \overrightarrow{I_n} = \overrightarrow{\pi} \odot \overrightarrow{\pi}^{-1}$ and $\overrightarrow{\pi}^{-1}$ is the inverse element of $\overrightarrow{\pi}$ in $\overrightarrow{S_n}$. Moreover, it is easy to see that $\overrightarrow{\pi}^{-1}$ is the only permutation in $\overrightarrow{S_n}$ that verifies that $\overrightarrow{\pi}^{-1} \odot \overrightarrow{\pi} = \overrightarrow{I_n} = \overrightarrow{\pi} \odot \overrightarrow{\pi}^{-1}$. ■

□

Proof (Proposition 3.2.3). We prove here that the function $T : \overrightarrow{S_n} \longrightarrow S_{2n}$, defined as $T(\overrightarrow{\pi}) = \pi'(1) \cdots \pi'(2n)$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\pi'(2i-1) = \begin{cases} 2\pi(i) - 1 & \text{, if } s(i) = + \\ 2\pi(i) & \text{, if } s(i) = - \end{cases} \qquad \pi'(2i) = \begin{cases} 2\pi(i) & \text{, if } s(i) = + \\ 2\pi(i) - 1 & \text{, if } s(i) = - \end{cases}$$

is an injective homomorphism between the groups $(\overrightarrow{S_n}, \odot)$ and $(S_n, \circ)$. The proof is separated in two steps:

1. (*homomorphism*) First, notice that $T(\overrightarrow{I_n}) = I_{2n}$ is a trivial consequence of the definition of T.

   Let $\overrightarrow{\pi}$ and $\overrightarrow{\sigma}$ be two arbitrary signed permutations in $\overrightarrow{S_n}$. The objective is to prove that:

$$T(\overrightarrow{\pi} \odot \overrightarrow{\sigma}) = T(\overrightarrow{\pi}) \circ T(\overrightarrow{\sigma})$$

We will construct $T(\overrightarrow{\pi} \odot \overrightarrow{\sigma})$ and $T(\overrightarrow{\pi}) \circ T(\overrightarrow{\sigma})$ from their definitions and prove them to be equal permutations in S_n. Let:

- $\overrightarrow{\pi}_3 = \overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2 = \langle \pi_3, s_3 \rangle$, where $\pi_3 = \pi_1 \circ \pi_2$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$s_3(i) = \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

- $\pi'_1 = T(\overrightarrow{\pi}_1) \in S_{2n}$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\pi'_1(2i-1) = \begin{cases} 2\pi_1(i) - 1 & \text{, if } s_1(i) = + \\ 2\pi_1(i) & \text{, if } s_1(i) = - \end{cases}$$

$$\pi'_1(2i) = \begin{cases} 2\pi_1(i) & \text{, if } s_1(i) = + \\ 2\pi_1(i) - 1 & \text{, if } s_1(i) = - \end{cases}$$

- $\pi'_2 = T(\overrightarrow{\pi}_2) \in S_{2n}$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\pi'_2(2i-1) = \begin{cases} 2\pi_2(i) - 1 & \text{, if } s_2(i) = + \\ 2\pi_2(i) & \text{, if } s_2(i) = - \end{cases}$$

$$\pi'_2(2i) = \begin{cases} 2\pi_2(i) & \text{, if } s_2(i) = + \\ 2\pi_2(i) - 1 & \text{, if } s_2(i) = - \end{cases}$$

Then:

- $\pi'_3 = T(\overrightarrow{\pi}_3) = T(\overrightarrow{\pi}_1 \odot \overrightarrow{\pi}_2) \in S_{2n}$ is such that for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$:

$$\pi'_3(2i-1) \stackrel{def}{=} \begin{cases} 2\pi_1(\pi_2(i)) - 1 & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

$$\pi'_3(2i) \stackrel{def}{=} \begin{cases} 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ 2\pi_1(\pi_2(i)) - 1 & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

- $\pi'_4 = \pi'_1 \circ \pi'_2 = T(\overrightarrow{\pi}_1) \circ T(\overrightarrow{\pi}_2) \in S_{2n}$ is such that for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$:

$$\pi'_4(2i-1) \stackrel{def}{=} \pi'_1(\pi'_2(2i-1))$$

$$\pi'_4(2i-1) \stackrel{def}{=} \pi'_1(\pi'_2(2i))$$

We now prove that for all $i \in \mathbb{N}$, such that $1 \leq i \leq n$:

(a) $\pi'_4(2i-1) = \pi'_3(2i-1)$:

$$\pi'_4(2i-1) \stackrel{def}{=} \pi'_1(\pi'_2(2i-1)) \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} \pi'_1(2\pi_2(i)-1) & \text{, if } s_2(i) = + \\ \pi'_1(2\pi_2(i)) & \text{, if } s_2(i) = - \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = + \wedge s_1(\pi_2(i)) = + \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) = + \wedge s_1(\pi_2(i)) = - \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) = - \wedge s_1(\pi_2(i)) = + \\ 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = - \wedge s_1(\pi_2(i)) = - \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \pi'_3(2i-1)$$

(b) $\pi'_4(2i) = \pi'_3(2i)$:

$$\pi'_4(2i) \stackrel{def}{=} \pi'_1(\pi'_2(2i)) \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} \pi'_1(2\pi_2(i)) & \text{, if } s_2(i) = + \\ \pi'_1(2\pi_2(i)-1) & \text{, if } s_2(i) = - \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) = + \wedge s_1(\pi_2(i)) = + \\ 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = + \wedge s_1(\pi_2(i)) = - \\ 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = - \wedge s_1(\pi_2(i)) = + \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) = - \wedge s_1(\pi_2(i)) = - \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \begin{cases} 2\pi_1(\pi_2(i))-1 & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ 2\pi_1(\pi_2(i)) & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases} \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \pi'_4(2i-1) = \pi'_3(2i-1)$$

Therefore, $T(\overrightarrow{\pi} \odot \overrightarrow{\sigma}) = T(\overrightarrow{\pi}) \circ T(\overrightarrow{\sigma})$ ■

2. (*injectivity*) Let $\overrightarrow{\pi}_1 = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi}_2 = \langle \pi_2, s_2 \rangle$ be two arbitrary signed permutations in $\overrightarrow{S_n}$. The objective is to prove that:

$$\overrightarrow{\pi}_1 \neq \overrightarrow{\pi}_2 \Rightarrow T(\overrightarrow{\pi}_1) \neq T(\overrightarrow{\pi}_2)$$

Let $\pi'_1 = T(\overrightarrow{\pi}_1)$ and $\pi'_2 = T(\overrightarrow{\pi}_2)$. Suppose that $\overrightarrow{\pi}_1 \neq \overrightarrow{\pi}_2$, then there exist $i \in \mathbb{N}$, such that $1 \leq i \leq n$ and either condition (a) or condition (b) (or both) hold:

(a) $\pi_1(i) \neq \pi_2(i)$

(b) $s_1(i) \neq s_2(i)$

- Suppose that condition (a) holds and assume, without loss of generality, that $\pi_1(i) > \pi_2(i)$. Then, the following conditions hold:

  i. $2\pi_1(i) > 2\pi_2(i)$

  ii. $2\pi_1(i) > \pi_2(i) - 1$

  iii. $2\pi_1(i) - 1 > 2\pi_2(i) - 1$

  iv. $2\pi_1(i) - 1 > 2\pi_2(i)$

Conditions i, ii and iii are trivial. Condition iv holds because:

$$\begin{aligned}
& \pi_1(i) > \pi_2(i) && \Leftrightarrow \\
\Leftrightarrow \quad & \pi_1(i) - \pi_2(i) > 0 && \Leftrightarrow \\
\Leftrightarrow \quad & \pi_1(i) - \pi_2(i) \geq 1 && \Leftrightarrow \\
\Leftrightarrow \quad & 2\pi_1(i) - 2\pi_2(i) \geq 2 && \Leftrightarrow \\
\Leftrightarrow \quad & 2\pi_1(i) \geq 2\pi_2(i) + 2 && \Leftrightarrow \\
\Leftrightarrow \quad & 2\pi_1(i) - 1 \geq 2\pi_2(i) + 1 && \Leftrightarrow \\
\Leftrightarrow \quad & 2\pi_1(i) - 1 > 2\pi_2(i)
\end{aligned}$$

Since $\pi'_1(2i-1)$ equals either $2\pi_1(i)$ or $2\pi_1(i)-1$ and $\pi'_2(2i-1)$ equals either $2\pi_2(i)$ or $2\pi_2(i)-1$, then, due to conditions i, ii, iii and iv, $\pi'_1(2i-1) > \pi'_2(2i-1)$, regardless of the values of $s_1(i)$ and $s_2(i)$. Therefore, there exists $j \in \mathbb{N}$ such that $1 \leq j \leq 2n$ and $\pi'_1(j) \neq \pi'_2(j)$ (in this case, $j = 2i-1$). Consequently, $T(\overrightarrow{\pi}_1) \neq T(\overrightarrow{\pi}_2)$.

- Suppose that condition (a) does not hold and that condition (b) holds. Then $\pi_1(i) = \pi_2(i)$ but $s_1(i) \neq s_2(i)$.
  - Suppose that $s_1(i) = +$ and $s_2(i) = -$. Then:

    $$\pi'_1(2i-1) \stackrel{def}{=} 2\pi_1(i) - 1 \neq 2\pi_1(i) \stackrel{def}{=} \pi'_2(2i-1)$$

    Consequently, there exists $j \in \mathbb{N}$ such that $1 \leq j \leq 2n$ and $\pi'_1(j) \neq \pi'_2(j)$. Therefore, $T(\overrightarrow{\pi}_1) \neq T(\overrightarrow{\pi}_2)$.
  - Suppose that $s_1(i) = -$ and $s_2(i) = +$. Then:

    $$\pi'_1(2i-1) \stackrel{def}{=} 2\pi_1(i) \neq 2\pi_1(i) - 1 \stackrel{def}{=} \pi'_2(2i-1)$$

    Consequently, there exists $j \in \mathbb{N}$ such that $1 \leq j \leq 2n$ and $\pi'_1(j) \neq \pi'_2(j)$. Therefore, $T(\overrightarrow{\pi}_1) \neq T(\overrightarrow{\pi}_2)$.

Therefore, $\overrightarrow{\pi}_1 \neq \overrightarrow{\pi}_2 \Rightarrow T(\overrightarrow{\pi}_1) \neq T(\overrightarrow{\pi}_2)$ ■

□

Proof (Proposition 4.1.1). Let G and H be two genomes over the alphabet of gene families $\mathcal{A}$ and define:

- $\mathcal{A}' = \{a \in \mathcal{A} : occ_G(a) \geq 1 \wedge occ_H(a) \geq 1\}$.
- $M_a = \max\{occ_G(a), occ_H(a)\}$, for all $a \in \mathcal{A}'$.
- $m_a = \min\{occ_G(a), occ_H(a)\}$, for all $a \in \mathcal{A}'$.

Statements 3 and 4 of proposition 4.1.1 are trivial consequences of the definitions of exemplar and maximum matchings. Regarding statements 1 and 2, notice that by definition, a pair (i,k) of matching between G and H is required to refer to genes of the same gene family. Therefore, no gene, that belongs to a gene family that occurs in only one the genomes, can be matched. Indeed, only genes that belong to a gene family in $\mathcal{A}'$ are allowed to be matched.

In an exemplar matching, it is required to match exactly one gene in each genome, for each gene family in $\mathcal{A}'$. Therefore, for each gene family $a' \in \mathcal{A}'$, there are $occ_G(a')$ possible positions to chose in G and $occ_H(a')$ possible positions to chose in H. Hence, for each gene family $a' \in \mathcal{A}'$, there are $occ_G(a') \cdot occ_H(a')$ possible choices of pairs (i,k) to match genes of that family. Consequently there exist $\prod_{a \in \mathcal{A}'} occ_G(a) \cdot occ_H(a)$ distinct exemplar matchings between G and H. ■

A maximum matching is required to match as many genes in each genome as possible, for each gene family in $\mathcal{A}'$. Since every two distinct elements (i,k) and (i',k') of a matching between G and H must refer to different genes in G and H, a maximum matching must match, for each gene family $a' \in \mathcal{A}'$, $m_{a'}$ genes, of that family, in both G and H. Let $a' \in \mathcal{A}'$ and assume, without loss of generality, that $M_{a'} = occ_G(a')$ and $m_{a'} = occ_H(a')$. Then, in a maximum matching between G and H, all genes, of the gene family a', in H must be matched and only $m_{a'}$ will be matched in G. Therefore, in a maximum matching between G and H, one has to chose $m_{a'}$ positions out of $M_{a'}$ in G (corresponding to which genes in G, of the gene family a', will be matched) and decide for each of them, to which position in H they will match. Hence, there are $C_{m_{a'}}^{M_{a'}} \cdot m_{a'}! = A_{m_{a'}}^{M_{a'}} = M_a(M_a - 1) \cdots (M_a - m_a + 1)$ different ways to saturate the gene family $a' \in \mathcal{A}'$. Consequently, there exist $\prod_{a \in \mathcal{A}'} A_{m_a}^{M_a}$ distinct maximum matchings between G and H. ■

□

Proof (Proposition 4.1.2). Let $\mathcal{A}$ be a finite alphabets of gene families, G and H be two genomes over $\mathcal{A}$, $\mathcal{M}$ be a matching between G and H and $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ be two bijective functions. Remember that for two signed permutations $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi_2} = \langle \pi_1, s_2 \rangle$ in $\overrightarrow{S_n}$, $\overrightarrow{\pi_1} \odot \overrightarrow{\pi_2} = \langle \pi, s \rangle$, where $\pi = \pi_1 \circ \pi_2$ and:

$$s(i) = \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

We want to prove that:

$$\begin{aligned} \mathcal{M}_{f_2}(G) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G) \\ \mathcal{M}_{f_2}(H) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H) \end{aligned} \tag{8.1}$$

where $+ : \{1, \ldots, n\} \longrightarrow \{+, -\}$ is defined as $+(i) = +$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

Define $G_0 = G$ and $G_1 = H$. From definition, for all $x \in \{0,1\}$, for all $y \in \{1,2\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:

$$\mathcal{M}_{f_y}(G_x)[i] \stackrel{def}{=} s'(G_x[k_{\mathcal{M}}^x(i)]) f_y(m_{\mathcal{M}}^x(k_{\mathcal{M}}^x(i))) \tag{8.2}$$

where $s'(G_x[k_{\mathcal{M}}^x(i)])$ is the sign of gene $G_x[k_{\mathcal{M}}^x(i)]$. Let $\langle \pi_x^y, s_x^y \rangle$ be the representation of the $\mathcal{M}$-reduced

genomes $\mathcal{M}_{f_y}(G_x)$ as signed permutations in $\overrightarrow{S_{|\mathcal{M}|}}$, for all $x \in \{0,1\}$ and all $y \in \{1,2\}$, i.e.:

$$\pi_x^y(i) = f_y(m_{\mathcal{M}}^x(k_{\mathcal{M}}^x(i))) \quad \wedge \quad s_x^y(i) = s'(G_x[k_{\mathcal{M}}^x(i)]) \tag{8.3}$$

for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$.

Notice that, for all $x \in \{0,1\}$:

1. $s_x^1(i) = s'(G_x[k_{\mathcal{M}}^x(i)]) = s_x^2(i)$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$. Therefore, $s_x^1 = s_x^2$.

2. since $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ are bijective functions, then $f_2 \circ f_1^{-1} : \{1, \ldots, |\mathcal{M}|\} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ is a bijective function. Therefore, $f_2 \circ f_1^{-1}$ is a permutation in $S_{|\mathcal{M}|}$.

3. for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:

$$\begin{aligned} ((f_2 \circ f_1^{-1}) \circ \pi_x^1)(i) = \quad & (f_2 \circ (f_1^{-1} \circ \pi_x^1))(i) = f_2(f_1^{-1}(\pi_x^1(i))) = \\ = \quad & f_2(f_1^{-1}(f_1(m_{\mathcal{M}}^x(k_{\mathcal{M}}^x(i))))) = f_2(m_{\mathcal{M}}^x(k_{\mathcal{M}}^x(i))) \\ = \quad & \pi_x^2(i) \end{aligned}$$

   Therefore, $(f_2 \circ f_1^{-1}) \circ \pi_x^1 = \pi_x^2$.

4. $\langle f_2 \circ f_1^{-1}, + \rangle \odot \langle \pi_x^1, s_x^1 \rangle = \langle (f_2 \circ f_1^{-1}) \circ \pi_x^1, s'' \rangle = \langle \pi_x^2, s'' \rangle$, where for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:

$$s''(i) = \begin{cases} + & \text{, if } s_x^1(i) = +(\pi_x^1(i)) = + \\ - & \text{, if } s_x^1(i) \neq +(\pi_x^1(i)) = + \end{cases} \quad \Leftrightarrow \quad s''(i) = \begin{cases} + & \text{, if } s_x^1(i) = + \\ - & \text{, if } s_x^1(i) = - \end{cases}$$

   i.e., $s'' = s_x^1$. Consequently $\langle f_2 \circ f_1^{-1}, + \rangle \odot \langle \pi_x^1, s_x^1 \rangle = \langle \pi_x^2, s_x^1 \rangle = \langle \pi_x^2, s_x^2 \rangle$ (because $s_x^1 = s_x^2$).

Therefore, we have that for all $x \in \{0,1\}$:

$$\mathcal{M}_{f_2}(G_x) = \langle f_2 \circ f_1^{-1}, + \rangle \odot \mathcal{M}_{f_1}(G_x) \tag{8.4}$$

Consequently, we have the desired result:

$$\begin{aligned} \mathcal{M}_{f_2}(G) = \mathcal{M}_{f_2}(G_0) = \langle f_2 \circ f_1^{-1}, + \rangle \odot \mathcal{M}_{f_1}(G_0) = \langle f_2 \circ f_1^{-1}, + \rangle \odot \mathcal{M}_{f_1}(G) \\ \mathcal{M}_{f_2}(H) = \mathcal{M}_{f_2}(G_1) = \langle f_2 \circ f_1^{-1}, + \rangle \odot \mathcal{M}_{f_1}(G_1) = \langle f_2 \circ f_1^{-1}, + \rangle \odot \mathcal{M}_{f_1}(H) \end{aligned} \tag{8.5}$$

□

Proof (Proposition 4.1.3). Let $d = \bigcup_{n \in \mathbb{N}} \{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$. Remember that for two signed permutations $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi_2} = \langle \pi_1, s_2 \rangle$ in $\overrightarrow{S_n}$, $\overrightarrow{\pi_1} \odot \overrightarrow{\pi_2} = \langle \pi, s \rangle$, where $\pi = \pi_1 \circ \pi_2$ and:

$$s(i) = \begin{cases} + & \text{, if } s_2(i) = s_1(\pi_2(i)) \\ - & \text{, if } s_2(i) \neq s_1(\pi_2(i)) \end{cases}$$

1. Suppose that for all $n \in \mathbb{N}$, d_n is invariant on the left for permutations, i.e., that for all signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ and all permutations $\pi \in S_n$, it holds that:

$$d_n(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d_n(\langle \pi, + \rangle \odot \overrightarrow{\pi_1}, \langle \pi, + \rangle \odot \overrightarrow{\pi_2}) \tag{8.6}$$

where $+ : \{1, \ldots, n\} \longrightarrow \{+, -\}$ is defined as $+(i) = +$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

We want to prove that for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$, all matchings $\mathcal{M}$ between G and H and all bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$, it holds that:

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) \tag{8.7}$$

Let $\mathcal{A}$ be a finite alphabets of gene families, G and H be two genomes over $\mathcal{A}$, $\mathcal{M}$ be a matching between G and H and $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ be two bijective functions. Then, proposition 4.1.2 holds, and we have that:

$$\begin{aligned} \mathcal{M}_{f_2}(G) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G) \\ \mathcal{M}_{f_2}(H) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H) \end{aligned} \tag{8.8}$$

From equation 8.8 it follows that:

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) = d_{|\mathcal{M}|}(\left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G), \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H)) \tag{8.9}$$

Notice that $f_2 \circ f_1^{-1} : \{1, \ldots, |\mathcal{M}|\} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ is a bijective function, because both f_1 and f_2 are bijective function. Therefore, $f_2 \circ f_1^{-1}$ is a permutation in $S_{|\mathcal{M}|}$.

Since d_n is invariant on the left for permutations, for all $n \in \mathbb{N}$ (in particular for $n = |\mathcal{M}|$), then, because $\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H) \in \overrightarrow{S_{|\mathcal{M}|}}$ and $f_2 \circ f_1^{-1} \in S_{|\mathcal{M}|}$, we have that:

$$d_{|\mathcal{M}|}(\left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G), \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) \tag{8.10}$$

Combining equations 8.9 and 8.10, we obtain the desired result:

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) \tag{8.11}$$

■

2. Suppose that for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$, all matchings $\mathcal{M}$ between G and H and all bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$, it holds that:

$$d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) \tag{8.12}$$

We want to prove that, for all $n \in \mathbb{N}$, d_n is invariant on the left for permutations, i.e., that for all signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ and all permutations $\pi \in S_n$, it holds that:

$$d_n(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d_n(\langle \pi, + \rangle \odot \overrightarrow{\pi_1}, \langle \pi, + \rangle \odot \overrightarrow{\pi_2}) \tag{8.13}$$

where $+ : \{1, \ldots, n\} \longrightarrow \{+, -\}$ is defined as $+(i) = +$, for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

Let $n \in \mathbb{N}$, $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ and $\overrightarrow{\pi_2} = \langle \pi_2, s_2 \rangle$ be two signed permutations in $\overrightarrow{S_n}$ and $\pi \in S_n$. The proof will take the following steps:

(a) defining a finite alphabet of gene families $\mathcal{A}$, two genomes G and H over $\mathcal{A}$, a matching $\mathcal{M}$ between G and H and two bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$ for which it holds that:

$$|\mathcal{M}| = n \tag{8.14}$$

$$\mathcal{M}_{f_1}(G) = \overrightarrow{\pi_1} \quad \wedge \quad \mathcal{M}_{f_1}(H) = \overrightarrow{\pi_2} \tag{8.15}$$

$$f_2 \circ f_1^{-1} = \pi \tag{8.16}$$

(b) using proposition 4.1.2 to obtain that:

$$\begin{aligned} \mathcal{M}_{f_2}(G) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G) \\ \mathcal{M}_{f_2}(H) &= \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H) \end{aligned} \tag{8.17}$$

(c) using the hypothesis followed by the equalities in equation 8.17 to obtain that:

$$\begin{aligned} d_{|\mathcal{M}|}(\mathcal{M}_{f_1}(G), \mathcal{M}_{f_1}(H)) = &\quad d_{|\mathcal{M}|}(\mathcal{M}_{f_2}(G), \mathcal{M}_{f_2}(H)) = \\ = &\quad d_{|\mathcal{M}|}(\left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(G), \left\langle f_2 \circ f_1^{-1}, + \right\rangle \odot \mathcal{M}_{f_1}(H)) \end{aligned} \tag{8.18}$$

(d) applying to equation 8.18 the equalities in equations 8.14, 8.15 and 8.16 to obtain the desired result:

$$d_n(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d_n(\langle \pi, + \rangle \odot \overrightarrow{\pi_1}, \langle \pi, + \rangle \odot \overrightarrow{\pi_2}) \tag{8.19}$$

We now defining a finite alphabet of gene families $\mathcal{A}$, two genomes G and H over $\mathcal{A}$, a matching $\mathcal{M}$ between G and H and two bijective functions $f_1, f_2 : \mathcal{M} \longrightarrow \{1, \ldots, |\mathcal{M}|\}$:

- Consider the finite alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$.
- Let G and H be the representation of the signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ as duplicate free genomes of lengths n over $\mathcal{A}$, i.e., let:

$$G[i] = s_1(i)\pi_1(i) \quad \wedge \quad H[i] = s_2(i)\pi_2(i) \tag{8.20}$$

for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$.

- Let $\mathcal{M} = \{(\pi_1^{-1}(i), \pi_2^{-1}(i)) : i \in \{1, \ldots, n\}\}$. Notice that $|\mathcal{M}| = n$ and $\mathcal{M}$ is a matching between H and G, because:

  (a) since $n_G = n_H = n$ and $\pi_1^{-1}, \pi_2^{-1} \in S_n$:

  $$\mathcal{M} \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\} = \{1, \ldots, n_G\} \times \{1, \ldots, n_H\} \tag{8.21}$$

  (b) every pair $(\pi_1^{-1}(i), \pi_2^{-1}(i)) \in \mathcal{M}$ is such that:

  $$|G[\pi_1^{-1}(i)]| = \pi_1(\pi_1^{-1}(i)) = i = \pi_2(\pi_2^{-1}(i)) = |H[\pi_2^{-1}(i)]| \tag{8.22}$$

  (c) every two distinct elements $(\pi_1^{-1}(i), \pi_2^{-1}(i)), (\pi_1^{-1}(j), \pi_2^{-1}(j)) \in \mathcal{M}$ (i.e., such that $i \neq j$) verify that:

  $$\pi_1^{-1}(i) \neq \pi_1^{-1}(j) \quad \wedge \quad \pi_2^{-1}(i) \neq \pi_2^{-1}(j) \tag{8.23}$$

  because $\pi_1^{-1}, \pi_2^{-1} \in S_n$ are injective functions.

- Let $f_1 : \mathcal{M} \longrightarrow \{1, \ldots, n\}$ ($|\mathcal{M}| = n$) be defined as:

  $$f_1((i, k)) = \pi_1(i) \tag{8.24}$$

  for all $(i, k) \in \mathcal{M}$. Notice that:

  (a) if (i, k) and (i', k') are distinct pairs in $\mathcal{M}$, then $(i, k) = (\pi_1^{-1}(j), \pi_2^{-1}(j))$ for some $j \in \{1, \ldots, n\}$ and $(i', k') = (\pi_1^{-1}(j'), \pi_2^{-1}(j'))$ for some $j' \in \{1, \ldots, n\}$ such that $j \neq j'$. Moreover, $f_1((\pi_1^{-1}(j), \pi_2^{-1}(j))) = \pi_1(\pi_1^{-1}(j)) = j$ and $f_1((\pi_1^{-1}(j'), \pi_2^{-1}(j'))) = j'$. Therefore, f_1 is injective.

  (b) if $j \in \{1, \ldots, n\}$, then $(\pi_1^{-1}(j), \pi_2^{-1}(j)) \in \mathcal{M}$ and $f_1((\pi_1^{-1}(j), \pi_2^{-1}(j))) = j$. Therefore, f_1 is sobrejective.

  From this, $f_1 : \mathcal{M} \longrightarrow \{1, \ldots, n\}$ ($|\mathcal{M}| = n$) is a bijective function.

- Let $f_2 = \pi \circ f_1 : \mathcal{M} \longrightarrow \{1, \ldots, n\}$ ($|\mathcal{M}| = n$). Since $\pi \in S_n$ and f_1 are bijective functions, then f_2 is also bijective.

To conclude this proof, we show that $\mathcal{A}$, G, H, $\mathcal{M}$, f_1 and f_2, defined above, verify the properties specified in equations 8.14, 8.15 and 8.16:

- Clearly, $|\mathcal{M}| = n$.
- Also, it holds that:

$$f_2 \circ f_1^{-1} = (\pi \circ f_1) \circ f_1^{-1} = \pi \circ (f_1 \circ f_1^{-1}) = \pi \tag{8.25}$$

- Define $G_0 = G$ and $G_1 = H$. Then, from definition

$$\mathcal{M}_{f_1}(G_x)[i] \stackrel{def}{=} s(G_x[k^x_{\mathcal{M}}(i)])f_1(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \tag{8.26}$$

for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$ ($|\mathcal{M}| = n$). Recall that $k^x_{\mathcal{M}}(i)$ denotes the ith $\mathcal{M}$-saturated gene in G_x. Since, all genes in G_0 and G_1 are $\mathcal{M}$-saturated, then, it holds that for all $x \in \{0, 1\}$ and for all $i \in \{1, \ldots, n\}$, $k^x_{\mathcal{M}}(i) = i$. Therefore:

$$\mathcal{M}_{f_1}(G_x)[i] = s(G_x[i])f_1(m^x_{\mathcal{M}}(i)) \tag{8.27}$$

for all $i \in \mathbb{N}$ such that $1 \leq i \leq n$. $m^x_{\mathcal{M}}(i)$ denotes the pair in $\mathcal{M}$ that matches the gene $G_x[i]$. Once more, because all genes in G_0 and G_1 are $\mathcal{M}$-saturated:

- for all $i \in \{1, \ldots, n\}$, $m^0_{\mathcal{M}}(i) = (i, j)$, for some $j \in \{1, \ldots, n\}$.
- for all $i \in \{1, \ldots, n\}$, $m^1_{\mathcal{M}}(i) = (j, i)$, for some $j \in \{1, \ldots, n\}$.

Notice that, by definition, $\pi_1(i) = \pi_2(j)$ for all pairs $(i, j) \in \mathcal{M}$. Consequently it holds that, for all $i \in \{1, \ldots, n\}$:

- $f_1(m^0_{\mathcal{M}}(i)) = \pi_1(i)$.
- $f_1(m^1_{\mathcal{M}}(i)) = \pi_1(j)$ for some $j \in \{1, \ldots, n\}$, where $\pi_1(j) = \pi_2(i)$. Therefore, $f_1(m^1_{\mathcal{M}}(i)) = \pi_2(i)$.

Therefore, for all $i \in \{1, \ldots, n\}$:

$$\mathcal{M}_{f_1}(G)[i] = \mathcal{M}_{f_1}(G_0)[i] = s(G_0[i])f_1(m^0_{\mathcal{M}}(i)) = s(G[i])\pi_1(i) = s_1(i)\pi_1(i) \tag{8.28}$$

$$\mathcal{M}_{f_1}(H)[i] = \mathcal{M}_{f_1}(G_1)[i] = s(G_1[i])f_1(m^1_{\mathcal{M}}(i)) = s(H[i])\pi_2(i) = s_2(i)\pi_2(i) \tag{8.29}$$

i.e., $\mathcal{M}_{f_1}(G)$ and $\mathcal{M}_{f_1}(H)$ are the representations of $\overrightarrow{\pi_1}$ and $\overrightarrow{\pi_2}$ as duplicate free genomes of length n over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$:

$$\mathcal{M}_{f_1}(G) = \overrightarrow{\pi_1} \quad \wedge \quad \mathcal{M}_{f_1}(H) = \overrightarrow{\pi_2} \tag{8.30}$$

■

□

Proof (Proposition 4.1.4). Let G and H be two genomes over a finite alphabet of gene families and $\mathcal{M}$ be a matching between G and H. Define $G_0 = G$, $G_1 = H$, $n_0 = n_G$ and $n_1 = n_H$.

1. Let $i \in \mathbb{N}$ be such that $1 \leq i \leq |\mathcal{M}|$. Then:

- From definition, $k^0_{\mathcal{M}}(i) = t$ where t is the minimum number for which it holds that:

$$\sum_{j=1}^{t} P^0_{\mathcal{M}}(j) = i \tag{8.31}$$

- From definition $m^0_{\mathcal{M}}(t)$ is the pair in $\mathcal{M}$ that matches $G_0[t]$, i.e., $m^0_{\mathcal{M}}(t) = (t, k)$ for the one number $k \in \{1, \ldots, n_1\}$ such that $(t, k) \in \mathcal{M}$.

- Therefore:

$$f^0_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) = f^0_{\mathcal{M}}((t, k)) \overset{def}{=} \sum_{j=1}^{t} P^0_{\mathcal{M}}(j) \overset{8.31}{=} i \tag{8.32}$$

Proving that for all $i \in \mathbb{N}$ be such that $1 \leq i \leq |\mathcal{M}|$, $f^1_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))) = i$ is very similar. ■

2. Since For all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$, $m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))$ and $m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))$ are pairs in $\mathcal{M}$ and:

$$\begin{aligned} f^0_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) &= i \\ f^1_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))) &= i \end{aligned} \tag{8.33}$$

then $f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$ are sobrejective functions. Let (i, k) and (i', k') are two distinct pairs in $\mathcal{M}$ (i.e., such that $i \neq i'$ and $j \neq j'$). We have that $P^0_{\mathcal{M}}(i) = P^0_{\mathcal{M}}(i') = 1$, $m^0_{\mathcal{M}}(i) = (i, k)$ and $m^0_{\mathcal{M}}(i') = (i', k')$. Assume, without loss of generality, that $i < i'$. Then, because $P^0_{\mathcal{M}}(i') = 1$ and for all $j \in \{1, \ldots, n_0\}$ it holds that $P^0_{\mathcal{M}}(j) \in \{0, 1\}$, we have that:

$$f^0_{\mathcal{M}}((i, k)) \overset{def}{=} \sum_{j=1}^{i} P^0_{\mathcal{M}}(j) < \sum_{j=1}^{i'} P^0_{\mathcal{M}}(j) \overset{def}{=} f^0_{\mathcal{M}}((i', k')) \tag{8.34}$$

Therefore, $f^0_{\mathcal{M}}((i, k)) \neq f^0_{\mathcal{M}}((i', k'))$ and $f^0_{\mathcal{M}}$ is injective. Following a similar reasoning we also have that $f^1_{\mathcal{M}}$ is injective. Consequently, $f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$ are bijective functions. ■

3. Since $f^0_{\mathcal{M}}$ and $f^1_{\mathcal{M}}$ are bijective functions, it follows from definition that, for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:

$$\begin{aligned} \mathcal{M}_{f^0_{\mathcal{M}}}(G_0)[i] &\overset{def}{=} s(G_0[k^0_{\mathcal{M}}(i)])f^0_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) \overset{8.33}{=} s(G_0[k^0_{\mathcal{M}}(i)])i \\ \mathcal{M}_{f^1_{\mathcal{M}}}(G_1)[i] &\overset{def}{=} s(G_1[k^1_{\mathcal{M}}(i)])f^1_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))) \overset{8.33}{=} s(G_1[k^1_{\mathcal{M}}(i)])i \end{aligned} \tag{8.35}$$

■

□

Proof (Proposition 4.1.6). Consider a family $d = \bigcup_{n \in \mathbb{N}} \{d_n\}$ of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations. Suppose that, for all $n \in \mathbb{N}$, d_n is symmetric, i.e., for all signed permutations $\overrightarrow{\pi_1}, \overrightarrow{\pi_2} \in \overrightarrow{S_n}$ it holds that:

$$d_n(\overrightarrow{\pi_1}, \overrightarrow{\pi_2}) = d_n(\overrightarrow{\pi_2}, \overrightarrow{\pi_1}) \tag{8.36}$$

We want to prove that for all finite alphabets of gene families $\mathcal{A}$, all genomes G and H over $\mathcal{A}$ and all matchings $\mathcal{M}$ between G and H it holds that:

$$d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) = d(H, G, T(\mathcal{M}), f^0_{T(\mathcal{M})}) \tag{8.37}$$

where $T(\mathcal{M}) \stackrel{def}{=} \{(k, i) : (i, k) \in \mathcal{M}\}$.

Let $\mathcal{A}$ be a finite alphabet of gene families, G and H be genomes over $\mathcal{A}$ and $\mathcal{M}$ be a matching between G and H. Let $T(\mathcal{M}) = \{(k, i) : (i, k) \in \mathcal{M}\}$. From proposition 4.1.5, $T(\mathcal{M})$ is a matching between H and G. The proof will take the following steps:

1. Proving that:
$$\mathcal{M}_{f^0_{\mathcal{M}}}(G) = T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G) \quad \wedge \quad \mathcal{M}_{f^0_{\mathcal{M}}}(H) = T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(H) \tag{8.38}$$

2. Using the hypothesis to get that $d_{|\mathcal{M}|}$ is symmetric and obtain:
$$d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G), \mathcal{M}_{f^0_{\mathcal{M}}}(H)) = d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(H), \mathcal{M}_{f^0_{\mathcal{M}}}(G)) \tag{8.39}$$

3. Use the equalities in equation 8.38 and the fact that $|\mathcal{M}| = |T(\mathcal{M})|$ to get that:
$$d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(H), \mathcal{M}_{f^0_{\mathcal{M}}}(G)) = d_{|T(\mathcal{M})|}(T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(H), T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G)) \tag{8.40}$$

4. Since $d_{|T(\mathcal{M})|}$ is invariant on the left for permutations, $T(\mathcal{M})$ is a matching between H and G and $f^0_{T(\mathcal{M})}, f^1_{T(\mathcal{M})} : T(\mathcal{M}) \longrightarrow \{1, \ldots, |T(\mathcal{M})|\}$ are bijective functions, from proposition 4.1.3 it holds that:
$$d_{|T(\mathcal{M})|}(T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(H), T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G)) = d_{|T(\mathcal{M})|}(T(\mathcal{M})_{f^0_{T(\mathcal{M})}}(H), T(\mathcal{M})_{f^0_{T(\mathcal{M})}}(G)) \tag{8.41}$$

5. Combine equations 8.39, 8.40 and 8.41 to conclude the desired result:
$$\begin{aligned} d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) &\stackrel{def}{=} d_{|\mathcal{M}|}(\mathcal{M}_{f^0_{\mathcal{M}}}(G), \mathcal{M}_{f^0_{\mathcal{M}}}(H)) = \\ &= d_{|T(\mathcal{M})|}(T(\mathcal{M})_{f^0_{T(\mathcal{M})}}(H), T(\mathcal{M})_{f^0_{T(\mathcal{M})}}(G)) \stackrel{def}{=} d(H, G, T(\mathcal{M}), f^0_{T(\mathcal{M})}) \end{aligned} \tag{8.42}$$

To finish this proof, we need only guaranty that equation 8.38 holds.

Define $G_0 = G$, $G_1 = H$, $n_0 = n_G$ and $n_1 = n_H$. From definition, we have that for all $x \in \{0, 1\}$ and for all $i \in \{1, \ldots, |\mathcal{M}|\}$:

$$\mathcal{M}_{f^0_{\mathcal{M}}}(G_x)[i] \stackrel{def}{=} s(G_x[k^x_{\mathcal{M}}(i)]) f^0_{\mathcal{M}}(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \tag{8.43}$$

where:

- $(a,b)^{-0} \stackrel{def}{=} (a,b)$ and $(a,b)^{-1} \stackrel{def}{=} (b,a)$, for all $a,b \in \mathbb{N}$. Notice that:

$$((a,b)^{-1})^{-1} = (b,a)^{-1} = (a,b) \tag{8.44}$$

- $\overline{x} \stackrel{def}{=} 1 - x$, for all $x \in \{0,1\}$.

- $P^x_{\mathcal{M}}(i) = 1$ if $G_x[i]$ is $\mathcal{M}$-saturated, i.e., if there exists $k \in \mathbb{N}$ such that $1 \leq k \leq n_{\overline{x}}$ and $(i,k)^{-x} \in \mathcal{M}$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$; $P^x_{\mathcal{M}}(i) = 0$ otherwise.

- $k^x_{\mathcal{M}}(i)$ denote the position of the ith $\mathcal{M}$-saturated gene in G_x, i.e., the minimum number t for which $\sum_{j=1}^{t} P^x_{\mathcal{M}}(j) = i$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$.

- $m^x_{\mathcal{M}}(i)$ denote the pair in $\mathcal{M}$ that matches the gene $G_x[i]$, i.e., the pair $(i,k)^{-x} \in \mathcal{M}$ (where $1 \leq k \leq n_{\overline{x}}$), for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$ and $P^x_{\mathcal{M}}(i) = 1$.

Define $G'_0 = H$, $G'_1 = G$, $n'_0 = n_H$ and $n'_1 = n_G$. From definition, we have that for all $x \in \{0,1\}$ and for all $i \in \{1,\ldots,|\mathcal{M}|\}$ ($|T(\mathcal{M})| = |\mathcal{M}|$):

$$T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G'_x)[i] \stackrel{def}{=} s(G'_x[k^x_{T(\mathcal{M})}(i)])f^1_{T(\mathcal{M})}(m^x_{T(\mathcal{M})}(k^x_{T(\mathcal{M})}(i))) \tag{8.45}$$

where:

- $P^x_{T(\mathcal{M})}(i) = 1$ if $G'_x[i]$ is $T(\mathcal{M})$-saturated, i.e., if there exists $k \in \mathbb{N}$ such that $1 \leq k \leq n'_{\overline{x}}$ and $(i,k)^{-x} \in T(\mathcal{M})$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n'_x$; $P^x_{T(\mathcal{M})}(i) = 0$ otherwise.

- $k^x_{T(\mathcal{M})}(i)$ denote the position of the ith $T(\mathcal{M})$-saturated gene in G'_x, i.e., the minimum number t for which $\sum_{j=1}^{t} P^x_{T(\mathcal{M})}(j) = i$, for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$ ($|T(\mathcal{M})| = |\mathcal{M}|$).

- $m^x_{T(\mathcal{M})}(i)$ denote the pair in $T(\mathcal{M})$ that matches the gene $G'_x[i]$, i.e., the pair $(i,k)^{-x} \in T(\mathcal{M})$ (where $1 \leq k \leq n_{\overline{x}}$), for $x \in \{0,1\}$ and $i \in \mathbb{N}$ such that $1 \leq i \leq n'_x$ and $P^x_{T(\mathcal{M})}(i) = 1$.

Notice that:

1. Since $G_x = G'_{\overline{x}}$, for all $x \in \{0,1\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$, it holds that:

$$G_x[i] = G'_{\overline{x}}[i] \tag{8.46}$$

2. $G_x[i]$ is $\mathcal{M}$-saturated if and only if $G'_{\overline{x}}[i]$ is $T(\mathcal{M})$-saturated, therefore, for all $x \in \{0,1\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$, it holds that:

$$P^x_{\mathcal{M}}(i) = P^{\overline{x}}_{T(\mathcal{M})}(i) \tag{8.47}$$

3. A pair $(i,k) \in \mathcal{M}$ matches the gene $G_x[i]$ if and only if the pair $(i,k)^{-1} \in T(\mathcal{M})$ matches the gene $G'_{\overline{x}}[i]$, therefore, for all $x \in \{0,1\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq n_x$, it holds that:

$$m^x_{\mathcal{M}}(i) = (m^{\overline{x}}_{T(\mathcal{M})}(i))^{-1} \tag{8.48}$$

4. If t is the minimum number for which $\sum_{j=1}^{t} P^x_{\mathcal{M}}(j) = i$, then, because of equation 8.47, t is also the minimum number for which $\sum_{j=1}^{t} P^{\overline{x}}_{T(\mathcal{M})}(j) = i$. Therefore, for all $x \in \{0,1\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$:

$$k^x_{\mathcal{M}}(i) = k^{\overline{x}}_{T(\mathcal{M})}(i) \tag{8.49}$$

5. For all pairs $(i,k) \in \mathcal{M}$, it holds that $(i,k)^{-1} = (k,i) \in T(\mathcal{M})$ and:

$$f^0_{\mathcal{M}}((i,k)) \overset{def}{=} \sum_{j=1}^{i} P^0_{\mathcal{M}}(j) \overset{8.47}{=} \sum_{j=1}^{i} P^1_{T(\mathcal{M})}(j) \overset{def}{=} f^1_{T(\mathcal{M})}((k,i)) = f^1_{T(\mathcal{M})}((i,k)^{-1}) \tag{8.50}$$

Therefore, for all $x \in \{0,1\}$ and for all $i \in \mathbb{N}$ such that $1 \leq i \leq |\mathcal{M}|$, it holds that:

$$\begin{aligned}
\mathcal{M}_{f^0_{\mathcal{M}}}(G_x)[i] &\overset{def}{=} s(G_x[k^x_{\mathcal{M}}(i)])f^0_{\mathcal{M}}(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) = \\
&\overset{8.49}{=} s(G_x[k^{\overline{x}}_{T(\mathcal{M})}(i)])f^0_{\mathcal{M}}(m^x_{\mathcal{M}}(k^{\overline{x}}_{T(\mathcal{M})}(i))) = \\
&\overset{8.46}{=} s(G'_{\overline{x}}[k^{\overline{x}}_{T(\mathcal{M})}(i)])f^0_{\mathcal{M}}(m^x_{\mathcal{M}}(k^{\overline{x}}_{T(\mathcal{M})}(i))) = \\
&\overset{8.50}{=} s(G'_{\overline{x}}[k^{\overline{x}}_{T(\mathcal{M})}(i)])f^1_{T(\mathcal{M})}((m^x_{\mathcal{M}}(k^{\overline{x}}_{T(\mathcal{M})}(i)))^{-1}) = \\
&\overset{8.48}{=} s(G'_{\overline{x}}[k^{\overline{x}}_{T(\mathcal{M})}(i)])f^1_{T(\mathcal{M})}(((m^{\overline{x}}_{T(\mathcal{M})}(k^{\overline{x}}_{T(\mathcal{M})}(i)))^{-1})^{-1}) = \\
&\overset{8.44}{=} s(G'_{\overline{x}}[k^{\overline{x}}_{T(\mathcal{M})}(i)])f^1_{T(\mathcal{M})}(m^{\overline{x}}_{T(\mathcal{M})}(k^{\overline{x}}_{T(\mathcal{M})}(i))) = \\
&\overset{def}{=} T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G'_{\overline{x}})[i]
\end{aligned} \tag{8.51}$$

From this, for all $x \in \{0,1\}$:

$$\mathcal{M}_{f^0_{\mathcal{M}}}(G_x) = T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G'_{\overline{x}}) \tag{8.52}$$

Since we defined $G_0 = G$, $G_1 = H$, $G'_0 = H$ and $G'_1 = G$, we have the desired result:

$$\mathcal{M}_{f^0_{\mathcal{M}}}(G) = T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(G) \quad \wedge \quad \mathcal{M}_{f^0_{\mathcal{M}}}(H) = T(\mathcal{M})_{f^1_{T(\mathcal{M})}}(H) \tag{8.53}$$

□

Proof (Proposition 4.1.6). Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $d = \bigcup_{n\in\mathbb{N}}\{d_n\}$ be a family of functions $d_n : \overrightarrow{S_n} \times \overrightarrow{S_n} \longrightarrow \mathbb{N}_0$ that are invariant on the left for permutations and symmetric. We want to prove that:

1. If $\mathcal{M}$ solves $OPT_e(G,H,d)$, then $T(\mathcal{M})$ solves $OPT_e(H,G,d)$.
2. If $\mathcal{M}$ solves $OPT_m(G,H,d)$, then $T(\mathcal{M})$ solves $OPT_m(H,G,d)$.

where $T(\mathcal{M}) \overset{def}{=} \{(k,i) : (i,k) \in \mathcal{M}\}$.

1. Suppose that d is a similarity measure:

   (a) Suppose that $\mathcal{M}$ solves $OPT_e(G, H, d)$, i.e., that $\mathcal{M}$ is an exemplar matching between G and H that maximizes $d(G, H, \mathcal{M}, f^0_{\mathcal{M}})$. From propositions 4.1.5 and 4.1.6, we have that $T(\mathcal{M})$ is an exemplar matching between H and G verifying that:

   $$d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) = d(H, G, T(\mathcal{M}), f^0_{T(\mathcal{M})}) \tag{8.54}$$

   Suppose, by absurd, that $T(\mathcal{M})$ does not solve $OPT_e(H, G, d)$, i.e., that there exist an exemplar matching $\mathcal{M}'$ between H and G (other than $T(\mathcal{M})$) for which $d(H, G, \mathcal{M}', f^0_{\mathcal{M}'})$ is maximized and:

   $$d(H, G, \mathcal{M}', f^0_{\mathcal{M}'}) > d(H, G, T(\mathcal{M}), f^0_{T(\mathcal{M})}) \overset{8.54}{=} d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) \tag{8.55}$$

   Then, from propositions 4.1.5 and 4.1.6, $T(\mathcal{M}')$ is an exemplar matching between G and H such that:

   $$d(H, G, \mathcal{M}', f^0_{\mathcal{M}'}) = d(G, H, T(\mathcal{M}'), f^0_{T(\mathcal{M}')}) \tag{8.56}$$

   Therefore, $T(\mathcal{M}')$ is an exemplar matching between G and H verifying that:

   $$d(G, H, T(\mathcal{M}'), f^0_{T(\mathcal{M}')}) \overset{8.56}{=} d(H, G, \mathcal{M}', f^0_{\mathcal{M}'}) \overset{8.55}{>} d(G, H, \mathcal{M}, f^0_{\mathcal{M}}) \tag{8.57}$$

   Consequently, assuming that $T(\mathcal{M})$ does not solve $OPT_e(H, G, d)$ contradicts the assumption that $\mathcal{M}$ solves $OPT_e(G, H, d)$ and we can conclude that $T(\mathcal{M})$ solves $OPT_e(H, G, d)$. ■

   (b) Suppose that d is a dissimilarity measure. Then, following a similar reasoning, $T(\mathcal{M})$ solves $OPT_m(H, G, d)$. ■

2. An identical proof, changing what needs to be changed. ■

□

Proof (Proposition 4.2.1). Let G and H be two genomes over a finite alphabet of gene families $\mathcal{A}$ and $\mathcal{M}$ be a matching between G and H. Consider the representation of $\mathcal{M}_{f^0_{\mathcal{M}}}(G)$, $\mathcal{M}_{f^0_{\mathcal{M}}}(H)$ and $\mathcal{M}_{f^1_{\mathcal{M}}}(G)$ as (unsigned) permutations i.e., define $G_0 = G$, $G_1 = H$, $n_0 = n_G$, $n_1 = n_H$ and:

$$\begin{aligned} \mathcal{M}_{f^0_{\mathcal{M}}}(G_x)[i] &\overset{def}{=} f^0_{\mathcal{M}}(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \\ \mathcal{M}_{f^1_{\mathcal{M}}}(G_x)[i] &\overset{def}{=} f^1_{\mathcal{M}}(m^x_{\mathcal{M}}(k^x_{\mathcal{M}}(i))) \end{aligned} \tag{8.58}$$

for all $x \in \{0, 1\}$ and for all $i \in \{1, \ldots, |\mathcal{M}|\}$. We want to prove that:

$$\left(\mathcal{M}_{f^0_{\mathcal{M}}}(G)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) = \mathcal{M}_{f^0_{\mathcal{M}}}(H) \tag{8.59}$$

$$\left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(G) = \mathcal{M}_{f^1_{\mathcal{M}}}(G) \tag{8.60}$$

From proposition 4.1.4 we have that for all $x \in \{0,1\}$ and for all $i \in \{1,\ldots,|\mathcal{M}|\}$:

$$\begin{aligned} \mathcal{M}_{f^0_{\mathcal{M}}}(G_0)[i] &= f^0_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) = i \\ \mathcal{M}_{f^1_{\mathcal{M}}}(G_1)[i] &= f^1_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(i))) = i \end{aligned} \tag{8.61}$$

Therefore, we have that:

$$\begin{aligned} \mathcal{M}_{f^0_{\mathcal{M}}}(G_0) &= \mathcal{M}_{f^0_{\mathcal{M}}}(G) = I_{|\mathcal{M}|} \\ \mathcal{M}_{f^1_{\mathcal{M}}}(G_1) &= \mathcal{M}_{f^1_{\mathcal{M}}}(H) = I_{|\mathcal{M}|} \end{aligned} \tag{8.62}$$

Since $(I_{|\mathcal{M}|})^{-1} = I_{|\mathcal{M}|}$, equation 8.62 suffices to guaranty that equation 8.59 holds:

$$\begin{aligned} & \mathcal{M}_{f^0_{\mathcal{M}}}(H) = \mathcal{M}_{f^0_{\mathcal{M}}}(H) \Leftrightarrow \\ \Leftrightarrow \quad & I_{|\mathcal{M}|} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) = I_{|\mathcal{M}|} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) \Leftrightarrow \\ \Leftrightarrow \quad & (I_{|\mathcal{M}|})^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) = \mathcal{M}_{f^0_{\mathcal{M}}}(H) \Leftrightarrow \\ \overset{8.62}{\Leftrightarrow} \quad & \left(\mathcal{M}_{f^0_{\mathcal{M}}}(G)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H) = \mathcal{M}_{f^0_{\mathcal{M}}}(H) \end{aligned} \tag{8.63}$$

■

Since $\mathcal{M}_{f^0_{\mathcal{M}}}(G) = I_{|\mathcal{M}|}$, equation 8.60 is equivalent to equation 8.64:

$$\left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} = \mathcal{M}_{f^1_{\mathcal{M}}}(G) \tag{8.64}$$

To prove equation 8.64 we will prove that equation 8.65 holds:

$$\mathcal{M}_{f^0_{\mathcal{M}}}(H) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G) = I_{|\mathcal{M}|} \tag{8.65}$$

Having this result, we can conclude that equation 8.64 (and equation 8.60) holds:

$$\begin{aligned} & \mathcal{M}_{f^0_{\mathcal{M}}}(H) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G) = I_{|\mathcal{M}|} \Leftrightarrow \\ \Leftrightarrow \quad & \left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \circ (\mathcal{M}_{f^0_{\mathcal{M}}}(H) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G)) = \left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \circ I_{|\mathcal{M}|} \Leftrightarrow \\ \Leftrightarrow \quad & (\left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \circ \mathcal{M}_{f^0_{\mathcal{M}}}(H)) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G) = \left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \Leftrightarrow \\ \Leftrightarrow \quad & I_{|\mathcal{M}|} \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G) = \left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \Leftrightarrow \\ \Leftrightarrow \quad & \mathcal{M}_{f^1_{\mathcal{M}}}(G) = \left(\mathcal{M}_{f^0_{\mathcal{M}}}(H)\right)^{-1} \end{aligned} \tag{8.66}$$

To conclude this proof, we show that equation 8.65 holds. Let $i \in \{1,\ldots,|\mathcal{M}|\}$. Then:

- From definition, $k^0_{\mathcal{M}}(i) = t$ where t is the minimum number for which it holds that:

$$\sum_{j=1}^{t} P^0_{\mathcal{M}}(j) = i \tag{8.67}$$

- From definition $m^0_{\mathcal{M}}(t)$ is the pair in $\mathcal{M}$ that matches $G_0[t]$. Since $P^0_{\mathcal{M}}(t) = 1$, let $k \in \{1, \ldots, n_1\}$ be the one number for which $(t, k) \in \mathcal{M}$.

- We have that:

$$f^1_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i))) = f^1_{\mathcal{M}}((t,k)) \stackrel{def}{=} \sum_{j=1}^{k} P^1_{\mathcal{M}}(j) \tag{8.68}$$

- From definition, $k^1_{\mathcal{M}}(\sum_{j=1}^{k} P^1_{\mathcal{M}}(j)) = t'$ where t' is the minimum number for which it holds that:

$$\sum_{j=1}^{t'} P^1_{\mathcal{M}}(j) = \sum_{j=1}^{k} P^1_{\mathcal{M}}(j) \tag{8.69}$$

  Since $P^1_{\mathcal{M}}(k) = 1 = P^0_{\mathcal{M}}(t)$ and, for all $j \in \{1, \ldots, n_1\}$ $P^1_{\mathcal{M}}(j) \in \{0, 1\}$, we have that $t' = k$.

- From definition $m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(\sum_{j=1}^{k} P^1_{\mathcal{M}}(j))) = m^1_{\mathcal{M}}(t') = m^1_{\mathcal{M}}(k)$ is the pair in $\mathcal{M}$ that matches $G_1[k]$. Therefore, $m^1_{\mathcal{M}}(k) = (t, k)$.

- Therefore:

$$f^0_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(\sum_{j=1}^{k} P^1_{\mathcal{M}}(j)))) = f^0_{\mathcal{M}}(m^1_{\mathcal{M}}(k)) = f^0_{\mathcal{M}}((t,k)) \stackrel{def}{=} \sum_{j=1}^{t} P^0_{\mathcal{M}}(j) \stackrel{8.67}{=} i \tag{8.70}$$

From this, we have that for all $i \in \{1, \ldots, |\mathcal{M}|\}$:

$$(\mathcal{M}_{f^0_{\mathcal{M}}}(G_1) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G_0))(i) = f^0_{\mathcal{M}}(m^1_{\mathcal{M}}(k^1_{\mathcal{M}}(f^1_{\mathcal{M}}(m^0_{\mathcal{M}}(k^0_{\mathcal{M}}(i)))))) = i \tag{8.71}$$

i.e., $\mathcal{M}_{f^0_{\mathcal{M}}}(G_1) \circ \mathcal{M}_{f^1_{\mathcal{M}}}(G_0) = I_{|\mathcal{M}|}$ ■

□

Proof (Proposition 4.2.2). We now prove that SAD, MAD, the number of breakpoints and the number of common intervals are symmetric and invariant on the left for permutations. Let G_0 and G_1 be two duplicate-free genomes over the alphabet of gene families $\mathcal{A} = \{1, \ldots, n\}$, of size n.

Let $\overrightarrow{\pi_0} = \langle \pi_0, s_0 \rangle$ and $\overrightarrow{\pi_1} = \langle \pi_1, s_1 \rangle$ be the representation of G_0 and G_1 as signed permutations in $\overrightarrow{S_n}$:

- (SAD): Consider the following permutations:

$$\sigma_{0,1} \stackrel{def}{=} \pi_0^{-1} \circ \pi_1 \qquad \sigma_{1,0} \stackrel{def}{=} \pi_1^{-1} \circ \pi_0 \tag{8.72}$$

  Then, the SAD number between G_0 and G_1 is defined as:

$$\sum_{i=1}^{n-1} |\sigma_{0,1}(i) - \sigma_{0,1}(i+1)| + \sum_{i=1}^{n-1} |\sigma_{1,0}(i) - \sigma_{1,0}(i+1)| \tag{8.73}$$

1. (*invariance on the left for permutations*): Let $\pi \in S_n$. We want to prove that the SAD number between G_0 and G_1 equals the SAD number between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$. Consider the following permutations:

$$\sigma'_{0,1} \stackrel{def}{=} (\pi \circ \pi_0)^{-1} \circ (\pi \circ \pi_1) \qquad \sigma'_{1,0} \stackrel{def}{=} (\pi \circ \pi_1)^{-1} \circ (\pi \circ \pi_0) \tag{8.74}$$

Then, the SAD number between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$ is defined as:

$$\sum_{i=1}^{n-1} \left| \sigma'_{0,1}(i) - \sigma'_{0,1}(i+1) \right| + \sum_{i=1}^{n-1} \left| \sigma'_{1,0}(i) - \sigma'_{1,0}(i+1) \right| \tag{8.75}$$

However, it is easy to see that:

$$\begin{array}{rcl} \sigma'_{0,1} & \stackrel{def}{=} & (\pi \circ \pi_0)^{-1} \circ (\pi \circ \pi_1) = (\pi_0^{-1} \circ \pi^{-1}) \circ (\pi \circ \pi_1) = \\ & = & \pi_0^{-1} \circ (\pi^{-1} \circ \pi) \circ \pi_1 = \pi_0^{-1} \circ \pi_1 \stackrel{def}{=} \sigma_{0,1} \\ \sigma'_{1,0} & \stackrel{def}{=} & (\pi \circ \pi_1)^{-1} \circ (\pi \circ \pi_0) = \pi_1^{-1} \circ \pi_0 \stackrel{def}{=} \sigma_{1,0} \end{array} \tag{8.76}$$

Therefore, the SAD number between G_0 and G_1 equals the SAD number between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$ for an arbitrary permutation $\pi \in S_n$, i.e., SAD is invariant on the left for permutations ■

2. (*symmetry*): We want to prove that the SAD number between G_0 and G_1 equals the SAD number between G_1 and G_0. Consider the following permutations:

$$\sigma'_{0,1} \stackrel{def}{=} \pi_1^{-1} \circ \pi_0 \qquad \sigma'_{1,0} \stackrel{def}{=} \pi_0^{-1} \circ \pi_1 \tag{8.77}$$

Then, the SAD number between G_1 and G_0 is defined as:

$$\sum_{i=1}^{n-1} \left| \sigma'_{0,1}(i) - \sigma'_{0,1}(i+1) \right| + \sum_{i=1}^{n-1} \left| \sigma'_{1,0}(i) - \sigma'_{1,0}(i+1) \right| \tag{8.78}$$

However, it is easy to see that $\sigma'_{0,1} = \sigma_{1,0}$ and $\sigma'_{1,0} = \sigma_{0,1}$. Therefore, the SAD number between G_0 and G_1 equals the SAD number between G_1 and G_0. ■

- (MAD): Identical to SAD. ■

- (Common intervals): An interval $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is said to be a common interval between G_0 and G_1 if there exists a number p such that $1 \leq p \leq n-l$ and:

$$\bigcup_{i=k}^{k+l} \{\pi_0(i)\} = \bigcup_{i=p}^{p+l} \{\pi_1(i)\} \tag{8.79}$$

1. (*invariance on the left for permutations*): Let $\pi \in S_n$. We want to prove that the number of common interval between G_0 and G_1 equals the number of common interval between

$\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$.

An interval $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is said to be a common interval between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$ if there exists a number p such that $1 \leq p \leq n-l$ and:

$$\bigcup_{i=k}^{k+l} \{\pi(\pi_0(i))\} = \bigcup_{i=p}^{p+l} \{\pi(\pi_1(i))\} \tag{8.80}$$

We prove that $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is a common interval between G_0 and G_1, if and only if, $[k, k+l]$ is a common interval between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$:

(a) Suppose that $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is a common interval between G_0 and G_1. Then, there exists a number p such that $1 \leq p \leq n-l$ and:

$$\bigcup_{i=k}^{k+l} \{\pi_0(i)\} = \bigcup_{i=p}^{p+l} \{\pi_1(i)\} \tag{8.81}$$

Then, for all $j \in \{k, \ldots, k+l\}$, there exists $j' \in \{p, \ldots, p+l\}$ such that $\pi_0(j) = \pi_1(j')$. Therefore, for all $j \in \{k, \ldots, k+l\}$, there exists $j' \in \{p, \ldots, p+l\}$ such that $\pi(\pi_0(j)) = \pi(\pi_1(j'))$ and it also holds that:

$$\bigcup_{i=k}^{k+l} \{\pi(\pi_0(i))\} = \bigcup_{i=p}^{p+l} \{\pi(\pi_1(i))\} \tag{8.82}$$

i.e., $[k, k+l]$ is a common interval between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$. ■

(b) Suppose that $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is a common interval between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$. Then, there exists a number p such that $1 \leq p \leq n-l$ and::

$$\bigcup_{i=k}^{k+l} \{\pi(\pi_0(i))\} = \bigcup_{i=p}^{p+l} \{\pi(\pi_1(i))\} \tag{8.83}$$

Then, for all $j \in \{k, \ldots, k+l\}$, there exists $j' \in \{p, \ldots, p+l\}$ such that $\pi(\pi_0(j)) = \pi(\pi_1(j'))$ and, since $\pi \in S_n$ is an injective function, $\pi_0(j) = \pi_1(j')$. Therefore, for all $j \in \{k, \ldots, k+l\}$, there exists $j' \in \{p, \ldots, p+l\}$ such that $\pi_0(j) = \pi_1(j')$ and it holds that:

$$\bigcup_{i=k}^{k+l} \{\pi_0(i)\} = \bigcup_{i=p}^{p+l} \{\pi_1(i)\} \tag{8.84}$$

i.e., $[k, k+l]$ is a common interval between G_0 and G_1. ■

Consequently, the number of common interval between G_0 and G_1 equals the number of common interval between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$. ■

2. (*symmetry*): Suppose that $[k, k+l]$, where $k, l \in \mathbb{N}_0$, $1 \leq k \leq n$ and $0 \leq l \leq n-k$, is a common interval between G_0 and G_1. Then, there exists a number p such that $1 \leq p \leq n-l$ and:

$$\bigcup_{i=k}^{k+l} \{\pi_0(i)\} = \bigcup_{i=p}^{p+l} \{\pi_1(i)\} \tag{8.85}$$

Notice that p is such that $1 \leq p \leq n-l \leq n$, l is such that $0 \leq l \leq n-k$ and k is such that $1 \leq k \leq n-l$ (because $1 \leq k \leq n \Rightarrow 1 \leq k$ and $0 \leq l \leq n-k \Rightarrow k \leq n-l$). Therefore, and due to equation 8.85, $[p, p+l]$ is a common interval between G_1 and G_0.

From this, for every common interval between G_0 and G_1 there exists a common interval between G_1 and G_0. Therefore, the number of common intervals between G_0 and G_1 is smaller or equal to the number of common interval between G_1 and G_0. Following a similar reasoning, it also holds that the number of common intervals between G_1 and G_0 is smaller or equal to the number of common interval between G_0 and G_1. Consequently, the number of common interval between G_0 and G_1 equals the number of common interval between G_1 and G_0. ■

- (Breakpoints): Recall that for $x \in \{+, -\}$:

$$\overline{x} \overset{def}{=} \begin{cases} + & \text{, if } x = - \\ - & \text{, if } x = + \end{cases}$$

1. (*invariance on the left for permutations*): Let $\pi \in S_n$. We want to prove that the number of breakpoints between G_0 and G_1 equals the number of breakpoints between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$. This is equivalent to proving that the number of adjacencies between G_0 and G_1 equals the number of adjacencies between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$ (see definition 4.2.4).

Consider the signed permutations $\overrightarrow{\pi'_0} = \langle \pi'_0, s'_0 \rangle$, $\overrightarrow{\pi'_1} = \langle \pi'_1, s'_1 \rangle$, $\overrightarrow{\pi''_0} = \langle \pi''_0, s''_0 \rangle$ and $\overrightarrow{\pi''_1} = \langle \pi''_1, s''_1 \rangle$ in $\overrightarrow{S_{n+2}}$ defined as follows:

$$\begin{array}{rcrcr} \pi'_x(i+1) = \pi_x(i) + 1 & \wedge & \pi'_x(1) = 1 & \wedge & \pi'_x(n+2) = n+2 \\ s'_x(i+1) = s_x(i) & \wedge & s'_x(1) = + & \wedge & s'_x(n+2) = + \\ \pi''_x(i+1) = (\pi \circ \pi_x)(i) + 1 & \wedge & \pi''_x(1) = 1 & \wedge & \pi''_x(n+2) = n+2 \\ s''_x(i+1) = s_x(i) & \wedge & s''_x(1) = + & \wedge & s''_x(n+2) = + \end{array} \tag{8.86}$$

for all $x \in \{0, 1\}$ and all $i \in \{1, \ldots, n\}$ (notice that for all $x \in \{0, 1\}$, $s''_x = s'_x$). Then, the number of adjacencies between $G_0 = \langle \pi_0, s_0 \rangle$ and $G_1 = \langle \pi_1, s_1 \rangle$ is the number of adjacencies in $\overrightarrow{\pi'_0}$ and the number of adjacencies between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$ is the number of adjacencies in $\overrightarrow{\pi''_0}$.

(a) Let $i \in \{1, \ldots, n+1\}$ and suppose that $s'_0(i)\pi'_0(i)$ determines an adjacency in $\overrightarrow{\pi'_0}$. Then,

there exists $p \in \{1, \ldots, n+1\}$ such that either one of the following conditions holds:

i. $s'_1(p)\pi'_1(p) = s'_0(i)\pi'_0(i)$ and $s'_1(p+1)\pi'_1(p+1) = s'_0(i+1)\pi'_0(i+1)$

ii. $s'_1(p)\pi'_1(p) = \overline{s'_0(i+1)}\pi'_0(i+1)$ and $s'_1(p+1)\pi'_1(p+1) = \overline{s'_0(i)}\pi'_0(i)$

Suppose that condition (i) holds. Then, $\pi'_1(p) = \pi'_0(i) \Rightarrow \pi(\pi'_1(p)) = \pi(\pi'_0(i))$ and $\pi'_1(p+1) = \pi'_0(i+1) \Rightarrow \pi(\pi'_1(p+1)) = \pi(\pi'_0(i+1))$. Therefore, $\pi''_1(p) = \pi''_0(i)$ and $\pi''_1(p+1) = \pi''_0(i+1)$. Also, because $s''_0 = s'_0$, we have that $s'_1(p) = s'_0(i) \Rightarrow s''_1(p) = s''_0(i)$ and $s'_1(p+1) = s'_0(i+1) \Rightarrow s''_1(p+1) = s''_0(i+1)$. From this, it holds that $s''_1(p)\pi''_1(p) = s''_0(i)\pi''_0(i)$ and $s''_1(p+1)\pi''_1(p+1) = s''_0(i+1)\pi''_0(i+1)$, i.e., $s''_0(i)\pi''_0(i)$ determines an adjacency in $\overrightarrow{\pi''_0}$.

Following a similar reasoning, if condition (ii) holds, then $s''_0(i)\pi''_0(i)$ also determines an adjacency in $\overrightarrow{\pi''_0}$. Therefore, the number of adjacencies in $\overrightarrow{\pi'_0}$ is smaller or equal to the number of adjacencies in $\overrightarrow{\pi''_0}$. Consequently, the number of adjacencies between G_0 and G_1 is smaller or equal to the number of adjacencies between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$.

(b) Let $i \in \{1, \ldots, n+1\}$ and suppose that $s''_0(i)\pi''_0(i)$ determines an adjacency in $\overrightarrow{\pi''_0}$. Then, there exists $p \in \{1, \ldots, n+1\}$ such that either one of the following conditions holds:

i. $s''_1(p)\pi''_1(p) = s''_0(i)\pi''_0(i)$ and $s''_1(p+1)\pi''_1(p+1) = s''_0(i+1)\pi''_0(i+1)$

ii. $s''_1(p)\pi''_1(p) = \overline{s''_0(i+1)}\pi''_0(i+1)$ and $s''_1(p+1)\pi''_1(p+1) = \overline{s''_0(i)}\pi''_0(i)$

Suppose that condition (ii) holds. Then, $\pi(\pi'_1(p)) = \pi(\pi'_0(i+1)) \Rightarrow \pi'_1(p) = \pi'_0(i+1)$ and $\pi(\pi'_1(p+1)) = \pi(\pi'_0(i)) \Rightarrow \pi'_1(p+1) = \pi'_0(i)$ because $\pi \in S_n$ is injective. Therefore $\pi'_1(p) = \pi'_0(i+1)$ and $\pi'_1(p+1) = \pi'_0(i)$. Also, because $s''_0 = s'_0$, we have that $s''_1(p) = \overline{s''_0(i+1)} \Rightarrow s'_1(p) = \overline{s'_0(i+1)}$ and $s''_1(p+1) = \overline{s''_0(i)} \Rightarrow s'_1(p+1) = \overline{s'_0(i)}$. From this, it holds that $s'_1(p)\pi'_1(p) = \overline{s'_0(i+1)}\pi'_0(i+1)$ and $s'_1(p+1)\pi'_1(p+1) = \overline{s'_0(i)}\pi'_0(i)$, i.e., $s'_0(i)\pi'_0(i)$ determines an adjacency in $\overrightarrow{\pi'_0}$.

Following a similar reasoning, if condition (i) holds, then $s'_0(i)\pi'_0(i)$ also determines an adjacency in $\overrightarrow{\pi'_0}$. Therefore, the number of adjacencies in $\overrightarrow{\pi''_0}$ is smaller or equal to the number of adjacencies in $\overrightarrow{\pi'_0}$. Consequently, the number of adjacencies between $\langle \pi, + \rangle \odot G_0$ and $\langle \pi, + \rangle \odot G_1$ is smaller or equal to the number of adjacencies between G_0 and G_1.

From (a) and (b) we conclude that the number of adjacencies between $\langle \pi, + \rangle \odot G_0 = \langle \pi \circ \pi_0, s_0 \rangle$ and $\langle \pi, + \rangle \odot G_1 = \langle \pi \circ \pi_1, s_1 \rangle$ is the number of adjacencies in π''_0. ■

2. (*symmetry*): We want to prove that the number of breakpoints between G_0 and G_1 equals the number of breakpoints between G_1 and G_0. This is equivalent to proving that the number of adjacencies between G_0 and G_1 equals the number of adjacencies between G_1 and G_0 (see definition 4.2.4).

Consider the signed permutations $\overrightarrow{\pi'_0} = \langle \pi'_0, s'_0 \rangle$, $\overrightarrow{\pi'_1} = \langle \pi'_1, s'_1 \rangle$ in $\overrightarrow{S_{n+2}}$ defined as follows:

$$\begin{aligned} \pi'_x(i+1) = \pi_x(i) + 1 \quad \wedge \quad \pi'_x(1) = 1 \quad \wedge \quad \pi'_x(n+2) = n+2 \\ s'_x(i+1) = s_x(i) \quad \wedge \quad s'_x(1) = + \quad \wedge \quad s'_x(n+2) = + \end{aligned} \tag{8.87}$$

for all $x \in \{0,1\}$ and all $i \in \{1,\ldots,n\}$. Then, the number of adjacencies between $G_0 = \langle \pi_0, s_0 \rangle$ and $G_1 = \langle \pi_1, s_1 \rangle$ is the number of adjacencies in $\overrightarrow{\pi'_0}$ and the number of adjacencies between $G_1 = \langle \pi_1, s_1 \rangle$ and $G_0 = \langle \pi_0, s_0 \rangle$ is the number of adjacencies in $\overrightarrow{\pi'_1}$.

(a) Let $i \in \{1,\ldots,n+1\}$ and suppose that $s'_0(i)\pi'_0(i)$ determines an adjacency in $\overrightarrow{\pi'_0}$. Then, there exists $p \in \{1,\ldots,n+1\}$ such that either one of the following conditions holds:

i. $s'_1(p)\pi'_1(p) = s'_0(i)\pi'_0(i)$ and $s'_1(p+1)\pi'_1(p+1) = s'_0(i+1)\pi'_0(i+1)$

ii. $s'_1(p)\pi'_1(p) = \overline{s'_0(i+1)}\pi'_0(i+1)$ and $s'_1(p+1)\pi'_1(p+1) = \overline{s'_0(i)}\pi'_0(i)$

Suppose that condition (i) holds. Then $s'_1(p)\pi'_1(p)$ determines an adjacency between $\overrightarrow{\pi'_1}$.
Suppose that condition (ii) holds. Then $s'_1(p)\pi'_1(p) = \overline{s'_0(i+1)}\pi'_0(i+1) \Rightarrow \overline{s'_1(p)}\pi'_1(p) = s'_0(i+1)\pi'_0(i+1)$ and $s'_1(p+1)\pi'_1(p+1) = \overline{s'_0(i)}\pi'_0(i) \Rightarrow \overline{s'_1(p+1)}\pi'_1(p+1) = s'_0(i)\pi'_0(i)$. Consequently, $s'_1(p)\pi'_1(p)$ determines an adjacency between $\overrightarrow{\pi'_1}$.
Therefore, the number of adjacencies in $\overrightarrow{\pi'_0}$ is smaller or equal to the number of adjacencies in $\overrightarrow{\pi'_1}$. Consequently, the number of adjacencies between G_0 and G_1 is smaller or equal to the number of adjacencies between G_1 and G_0.

(b) Following a similar reasoning, we conclude that the number of adjacencies between G_1 and G_0 is smaller or equal to the number of adjacencies between G_0 and G_1.

From (a) and (b) we conclude that the number of adjacencies between G_0 and G_1 equals the number of adjacencies between G_1 and G_0. Therefore, the number of breakpoints between G_0 and G_1 equals the number of breakpoints between G_1 and G_0. ■

□

maximize:	$\sum_{i=1}^{n_0}\sum_{j=i}^{n_0}\sum_{k=1}^{n_1}\sum_{l=k}^{n_1} c(i,j,k,l)$		
subject to:			
Condition	**Equation**	**Constraints**	**Indexes**
	5.7	$\sum_{\substack{1\le k\le n_1 \\ m(i,k)}} a(i,k) \le 1$	$1 \le i \le n_0$
	5.8	$\sum_{\substack{1\le i\le n_0 \\ m(i,k)}} a(i,k) \le 1$	$1 \le k \le n_1$
Exemplar model	5.9	$\sum_{\substack{1\le i\le n_0 \\ m_0(i,a)}} \sum_{\substack{1\le k\le n_1 \\ m_1(k,a)}} a(i,k) = 1$	$a \in \mathcal{A}$
Maximum model	5.10	$\sum_{\substack{1\le i\le n_0 \\ m_0(i,a)}} \sum_{\substack{1\le k\le n_1 \\ m_1(k,a)}} a(i,k) = \min\{occ_{G_0}(a), occ_{G_1}(a)\}$	$a \in \mathcal{A}$
	5.11	$4\cdot c(i,j,k,l) - \sum_{\substack{k\le r\le l \\ m(i,r)}} a(i,r) - \sum_{\substack{k\le s\le l \\ m(j,s)}} a(j,s) - \sum_{\substack{i\le p\le j \\ m(p,k)}} a(p,k) - \sum_{\substack{i\le q\le j \\ m(q,l)}} a(q,l) \le 0$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$
$m(p,r)$	5.12	$c(i,j,k,l) + a(p,r) \le 1$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$ $i < p < j$ $1 \le r < k$
$m(p,r)$	5.12	$c(i,j,k,l) + a(p,r) \le 1$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$ $i < p < j$ $l < r \le n_1$
$m(p,r)$	5.12	$c(i,j,k,l) + a(p,r) \le 1$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$ $k < r < l$ $1 \le p < i$
$m(p,r)$	5.12	$c(i,j,k,l) + a(p,r) \le 1$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$ $k < r < l$ $j < p \le n_1$

Table 8.1: PBO encoding of $OPT_e(G_0, G_1, d)$ and $OPT_m(G_0, G_1, d)$ for the number of common intervals. The input genomes, G_0 and G_1, must first be preprocessed (see section 5.2.1). The predicate $m(i,k)$ is defined true if $G_0[i] = G_1[k]$. The predicate $m_x(i,a)$ is defined true if $G_x[i] = a$.

maximize:	$\sum_{i=1}^{n_0-1} \sum_{j=i+1}^{n_0} \sum_{k=1}^{n_1-1} \sum_{l=k+1}^{n_1} d(i,j,k,l)$		
subject to:			
Condition	**Equation**	**Constraints**	**Indexes**
Exemplar model	5.14	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i) = 1$	$a \in \mathcal{A}$ $0 \le x \le 1$ $1 \le i \le n_x$
Maximum model	5.15	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i) = \min\{occ_{G_0}(a), occ_{G_1}(a)\}$	$a \in \mathcal{A}$ $0 \le x \le 1$ $1 \le i \le n_x$
	5.16	$\sum_{\substack{1\le k\le n_1 \\ m(i,k)}} a(i,k) = b_0(i)$	$1 \le i \le n_0$
	5.17	$\sum_{\substack{1\le i\le n_0 \\ m(i,k)}} a(i,k) = b_1(k)$	$1 \le k \le n_1$
	5.18	$c_x(i,j) + \sum_{i<p<j} b_x(p) \ge 1$	$0 \le x \le 1$ $1 \le i < j \le n_x$
	5.19	$c_x(i,j) + b_x(p) \le 1$	$0 \le x \le 1$ $1 \le i < j \le n_x$ $i < p < j$
$m_+(i,k)$ $\wedge$ $m_+(j,l)$	5.20	$a(i,k) + a(j,l) + c_0(i,j) +$ $+c_1(k,l) - d(i,j,k,l) \le 3$ $a(i,k) - d(i,j,k,l) \ge 0$ $a(j,l) - d(i,j,k,l) \ge 0$ $c_0(i,j) - d(i,j,k,l) \ge 0$ $c_1(k,l) - d(i,j,k,l) \ge 0$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$
$m_-(i,l)$ $\wedge$ $m_-(j,k)$	5.21	$a(i,l) + a(j,k) + c_0(i,j) +$ $+c_1(k,l) - d(i,j,k,l) \le 3$ $a(i,l) - d(i,j,k,l) \ge 0$ $a(j,k) - d(i,j,k,l) \ge 0$ $c_0(i,j) - d(i,j,k,l) \ge 0$ $c_1(k,l) - d(i,j,k,l) \ge 0$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$
$\neg((m_+(i,k)$ $\wedge m_+(j,l))$ $\wedge$ $\neg(m_-(i,l)$ $\wedge m_-(j,k))$	5.22	$d(i,j,k,l) = 0$	$1 \le i < j \le n_0$ $1 \le k < l \le n_1$
	5.23	$\sum_{1\le k<n_1} \sum_{k<l\le n_1} d(i,j,k,l) \le 1$	$1 \le i < j \le n_0$
	5.24	$\sum_{1\le i<n_0} \sum_{i<j\le n_0} d(i,j,k,l) \le 1$	$1 \le k < l \le n_1$

Table 8.2: PBO encoding of $OPT_e(G_0, G_1, d)$ and $OPT_m(G_0, G_1, d)$ for the number of adjacencies. The input genomes, G_0 and G_1, must first be preprocessed (see section 5.2.1) and then extended with border genes (see section 5.2.3). The predicate $m(i,k)$ is defined true if $|G_0[i]| = |G_1[k]|$. The predicate $m_x(i,a)$ is defined true if $|G_x[i]| = a$. The predicate $m_-(i,k)$ is defined true if $G_0[i] = -G_1[k]$. The predicate $m_+(i,k)$ is defined true if $G_0[i] = G_1[k]$.

minimize:	$\sum_{x=0}^{1}\sum_{i=1}^{n_x-1}\sum_{j=i+1}^{n_x}\left(\sum_{k=1}^{n_{\overline{x}}-1}\sum_{l=k+1}^{n_{\overline{x}}}(j-i)\cdot d_x(i,j,k,l)-\sum_{p=i+1}^{j-1}e_x(i,j,p)\right)$		
subject to:			
Condition	**Equation**	**Constraints**	**Indexes**
Exemplar model	5.14	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i)=1$	$a\in\mathcal{A}$ $0\le x\le 1$ $1\le i\le n_x$
Maximum model	5.15	$\sum_{\substack{1\le i\le n_x \\ m_x(i,a)}} b_x(i)=\min\{occ_{G_0}(a),occ_{G_1}(a)\}$	$a\in\mathcal{A}$ $0\le x\le 1$ $1\le i\le n_x$
	5.16	$\sum_{\substack{1\le k\le n_1 \\ m(i,k)}} a(i,k)=b_0(i)$	$1\le i\le n_0$
	5.17	$\sum_{\substack{1\le i\le n_0 \\ m(i,k)}} a(i,k)=b_1(k)$	$1\le k\le n_1$
	5.18	$c_x(i,j)+\sum_{i<p<j} b_x(p)\ge 1$	$0\le x\le 1$ $1\le i<j\le n_x$
	5.19	$c_x(i,j)+b_x(p)\le 1$	$0\le x\le 1$ $1\le i<j\le n_x$ $i<p<j$
$m(i,k,x)$ $\wedge$ $m(j,l,x)$	5.33	$a(i,k)^{-x}+a(j,l)^{-x}+c_{\overline{x}}(k,l)-d_x(i,j,k,l)\le 2$ $a(i,k)^{-x}-d_x(i,j,k,l)\ge 0$ $a(j,l)^{-x}-d_x(i,j,k,l)\ge 0$ $c_{\overline{x}}(k,l)-d_x(i,j,k,l)\ge 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
$m(i,l,x)$ $\wedge$ $m(j,k,x)$	5.34	$a(i,l)^{-x}+a(j,k)^{-x}+c_{\overline{x}}(k,l)-d_x(i,j,k,l)\le 2$ $a(i,l)^{-x}-d_x(i,j,k,l)\ge 0$ $a(j,k)^{-x}-d_x(i,j,k,l)\ge 0$ $c_{\overline{x}}(k,l)-d_x(i,j,k,l)\ge 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
$\neg((m(i,k,x)$ $\wedge m(j,l,x))$ $\wedge$ $\neg(m(i,l,x)$ $\wedge m(j,k,x))$	5.35	$d_x(i,j,k,l)=0$	$0\le x\le 1$ $1\le i<j\le n_x$ $1\le k<l\le n_{\overline{x}}$
	5.36	$\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)\le 1$	$0\le x\le 1$ $1\le i<j\le n_x$
	5.37	$\sum_{1\le i<n_x}\sum_{i<j\le n_x} d_x(i,j,k,l)\le 1$	$0\le x\le 1$ $1\le k<l\le n_{\overline{x}}$
	5.33	$\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)+b_x(i)+b_x(j)+$ $+(1-b_x(p))-e_x(i,j,p)\le 3$ $e_x(i,j,p)-\sum_{1\le k<n_{\overline{x}}}\sum_{k<l\le n_{\overline{x}}} d_x(i,j,k,l)\le 0$ $e_x(i,j,p)-b_x(i)\le 0$ $e_x(i,j,p)-b_x(j)\le 0$ $e_x(i,j,p)-(1-b_x(p))\le 0$	$0\le x\le 1$ $1\le i<j\le n_x$ $i<p<j$

Table 8.3: PBO encoding of $OPT_e(G_0,G_1,d)$ and $OPT_m(G_0,G_1,d)$ for the summed adjacency disruption number. The input genomes, G_0 and G_1, must first be preprocessed (see section 5.2.1). Define $a(i,k)^{-0}=a(i,k)$ and $a(i,k)^{-1}=a(k,i)$ for all variables $a(i,k)$. The predicate $m(i,k,x)$ is defined true if $G_x[i]=G_{\overline{x}}[k]$. The predicate $m_x(i,a)$ is defined true if $G_x[i]=a$.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Baphi		745	503	732	533	738	389	563	558	464	744	747
Ecoli	535		1253	2128	1394	3319	581	1410	1397	926	2469	2524
Haein	437	1529		1337	1430	1540	450	963	938	733	1421	1454
Paeru	473	1931	986		1098	1942	521	1725	1710	1006	1814	1847
Pmult	451	1675	1425	1413		1688	476	1007	984	772	1555	1594
Salty	537	3309	1262	2106	1398		580	1412	1394	930	2489	2546
Wglos	381	791	507	799	542	794		638	630	501	791	798
Xaxon	419	1476	808	2043	878	1495	476		3538	1521	1363	1394
Xcamp	419	1473	802	2056	869	1494	475	3540		1498	1338	1365
Xfast	403	1122	722	1315	769	1135	448	1696	1678		1072	1090
Ypest-CO92	534	2584	1219	2051	1352	2607	578	1332	1310	900		3472
Ypest-KIM	527	2587	1215	2047	1348	2613	569	1328	1304	901	3421	

Table 8.4: Genome size after preprocessing for the maximum model. Every line indicates the number of genes of the genome it represents, when preprocessed against the genomes in each column.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Baphi		728	499	724	528	730	382	553	548	462	729	728
Ecoli	531		1228	2095	1373	3252	570	1372	1362	911	2431	2458
Haein	434	1495		1314	1406	1517	440	938	914	720	1390	1408
Paeru	470	1890	970		1081	1916	510	1683	1670	987	1778	1791
Pmult	448	1640	1391	1389		1663	465	981	958	757	1523	1546
Salty	533	3277	1234	2075	1376		569	1374	1360	915	2450	2480
Wglos	380	773	499	785	535	783		621	614	498	773	772
Xaxon	416	1442	796	2013	864	1471	465		3454	1491	1333	1349
Xcamp	416	1444	793	2028	858	1473	464	3448		1472	1313	1327
Xfast	400	1099	713	1297	757	1120	437	1652	1635		1049	1057
Ypest-CO92	530	2539	1190	2020	1330	2571	567	1298	1275	884		3359
Ypest-KIM	523	2541	1186	2016	1326	2578	558	1294	1269	885	3387	

Table 8.5: Genome size after preprocessing for the number of common intervals and the summed adjacency disruption number under the exemplar model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Y-CO92	Y-KIM
Baphi		730	499	726	528	730	382	555	550	462	730	729
Ecoli	531		1228	2097	1373	3281	570	1374	1364	911	2432	2459
Haein	434	1497		1316	1406	1517	440	939	915	720	1391	1409
Paeru	470	1892	970		1081	1916	510	1685	1672	987	1779	1792
Pmult	448	1642	1391	1391		1663	465	982	959	757	1524	1547
Salty	533	3256	1234	2077	1376		569	1376	1362	915	2451	2481
Wglos	380	775	499	787	535	783		622	615	498	774	773
Xaxon	416	1442	796	2015	864	1472	465		3452	1491	1334	1350
Xcamp	416	1444	793	2030	858	1474	464	3458		1472	1314	1328
Xfast	400	1101	713	1299	757	1120	437	1654	1637		1050	1058
Ypest-CO92	530	2541	1190	2022	1330	2571	567	1300	1277	884		3390
Ypest-KIM	523	2543	1186	2018	1326	2578	558	1296	1271	885	3362	

Table 8.6: Genome size after preprocessing for the number of breakpoints under the exemplar model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	0.04										
Haein	0.01	0.09									
Paeru	0.01	0.46	0.09								
Pmult	0.01	0.09	0.05	0.35							
Salty	0.04	†	0.13	0.65	0.13						
Wglos	0.01	0.24	0.01	0.03	0.01	0.18					
Xaxon	0.01	0.16	0.03	0.43	0.05	2.67	0.01				
Xcamp	0.01	0.21	0.02	0.38	0.03	2.45	0.01	†			
Xfast	0.01	0.03	0.01	0.06	0.02	0.07	0.01	0.51	0.56		
Ypest-CO92	0.03	8.94	0.05	0.83	0.09	9.07	0.18	0.18	0.18	0.02	
Ypest-KIM	0.02	12.58	0.07	0.99	0.10	63.81	0.13	0.23	0.21	0.02	*

Table 8.7: Number of common intervals running times, in seconds, rounded up, for the exemplar model. (†) *Cplex* ran out of memory. (*) Instance exceeded 1 Gigabyte.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	0.05										
Haein	0.01	43.69									
Paeru	0.01	26.51	11.89								
Pmult	0.01	0.74	0.22	17.52							
Salty	0.04	†	0.33	17.59	0.45						
Wglos	0.01	0.25	0.01	0.03	0.02	0.20					
Xaxon	0.01	202.62	0.77	10.94	2.53	15.72	0.01				
Xcamp	0.01	48.28	0.21	1.81	0.23	2.17	0.01	†			
Xfast	0.01	0.11	0.03	0.13	6.66	0.15	0.01	0.58	0.51		
Ypest-CO92	0.03	66.83	0.91	6.10	0.77	225.42	0.13	1.71	0.98	0.14	
Ypest-KIM	0.02	319.80	8.04	34.25	4.49	387.32	0.14	388.32	25.18	0.33	*

Table 8.8: Number of common intervals running times, in seconds, rounded up, for the maximum model. (†) *Cplex* ran out of memory. (*) Instance exceeded 1 Gigabyte.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	154										
Haein	267	612									
Paeru	234	849	552								
Pmult	256	624	499	594							
Salty	156	155	614	864	624						
Wglos	170	185	269	250	264	183					
Xaxon	224	677	475	804	497	686	263				
Xcamp	224	680	475	803	497	686	262	116			
Xfast	233	493	426	501	438	499	266	377	375		
Ypest-CO92	168	428	599	792	599	441	184	626	622	475	
Ypest-KIM	174	424	600	788	603	441	188	626	620	479	34

Table 8.9: Results for the number of breakpoints using the exemplar model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	158										
Haein	272	667									
Paeru	242	1084	617								
Pmult	261	705	527	683							
Salty	160	279	678	1093	706						
Wglos	172	194	279	261	272	192					
Xaxon	228	844	535	1018	559	856	271				
Xcamp	228	847	532	1014	557	856	270	183			
Xfast	238	566	470	574	483	517	274	402	406		
Ypest-CO92	172	598	651	992	673	593	194	762	757	544	
Ypest-KIM	178	609	655	1006	678	608	198	762	751	547	61

Table 8.10: Results for the number of breakpoints using the maximum model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	0.01										
Haein	0.01	0.05									
Paeru	0.01	0.32	0.04								
Pmult	0.01	0.06	0.05	0.07							
Salty	0.01	27.17	0.05	0.24	0.05						
Wglos	0.01	0.02	0.01	0.01	0.01	0.02					
Xaxon	0.01	0.06	0.02	0.15	0.03	1.16	0.01				
Xcamp	0.01	0.05	0.02	0.14	0.03	1.05	0.01	18.82			
Xfast	0.01	0.03	0.01	0.03	0.01	0.01	0.01	0.15	0.08		
Ypest-CO92	0.01	4.82	0.04	0.42	0.06	1.18	0.02	0.07	0.05	0.01	
Ypest-KIM	0.01	6.53	0.06	0.37	0.06	3.41	0.02	0.07	0.06	0.03	146.49

Table 8.11: Number of adjacencies running times, in seconds, rounded up, for exemplar model.

	Baphi	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92
Ecoli	0.01										
Haein	0.01	0.05									
Paeru	0.01	0.26	0.06								
Pmult	0.01	0.07	0.03	0.05							
Salty	0.01	0.18	0.04	0.14	0.05						
Wglos	0.01	0.01	0.01	0.01	0.01	0.01					
Xaxon	0.01	0.09	0.02	0.11	0.02	0.12	0.01				
Xcamp	0.01	0.10	0.02	0.09	0.02	0.06	0.01	0.16			
Xfast	0.01	0.02	0.01	0.03	0.01	0.05	0.01	0.04	0.03		
Ypest-CO92	0.01	0.19	0.05	0.17	0.08	0.12	0.01	0.07	0.09	0.02	
Ypest-KIM	0.01	0.29	0.07	0.41	0.07	0.14	0.01	0.10	0.12	0.03	0.23

Table 8.12: Number of adjacencies running times, in seconds, rounded up, for the maximum model.

	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	13312.76	778.94	§	823.08	23425.57	0.16	2339.82	4743.49	18.52	§	§
Wglos	§	2190.78	§	782.39	§		1177.17	1466.11	96.05	§	§

Table 8.13: Summed adjacency disruption number running times for exemplar model.
(§) *Cplex* reached the time limit of ten hours.

	Ecoli	Haein	Paeru	Pmult	Salty	Wglos	Xaxon	Xcamp	Xfast	Ypest-CO92	Ypest-KIM
Baphi	§	622.25	§	863.30	§	0.43	8887.94	3467.77	9.96	§	§
Wglos	§	1416.38	§	10795.27	§		§	9266.44	1257.11	§	§

Table 8.14: Summed adjacency disruption number running times for the maximum model.
(§) *Cplex* reached the time limit of ten hours.