In silico methods for genome rearrangement analysis: from identification of common markers to ancestral reconstruction.

Géraldine Jean

N° d'ordre : 3704

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Géraldine JEAN**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

In silico methods for genome rearrangement analysis: from identification of common markers to ancestral reconstruction

Soutenue le : 9 Décembre 2008

Après avis des rapporteurs :

Bernard MORET	Professeur
Alain DENISE	Professeur

Devant la commission d'examen composée de :

Michel AIGLE	Professeur	Examinateur
Alain DENISE	Professeur	Rapporteur
Serge DULUCQ	Professeur	Directeur
Guillaume FERTIN	Professeur	Examinateur
Guy MELANCON	Professeur	Examinateur
Macha NIKOLSKI	Chargée de recherche	Co-directrice

– 2008 –

Remerciements

Avant tout, je tiens à remercier très sincèrement Serge Dulucq sans qui je n'aurais jamais osé me lancer dans cette longue aventure. Je vous remercie de m'avoir fait découvrir la bioinformatique, je vous remercie également de m'avoir transmis le virus de la recherche et merci enfin d'avoir accepté de devenir mon directeur de thèse malgré un emploi du temps plus que chargé. Je vous suis plus que reconnaissante pour votre disponibilité et vos précieux conseils.

Je tiens à remercier également très chaleureusement Macha Nikolski qui a accepté de co-diriger ma thèse. Macha, je te remercie d'avoir été présente chaque jour durant ma thèse, d'avoir guidé mes recherches avec bienveillance et de m'avoir toujours encouragée quand je ne trouvais pas forcément de réponses à mes questions.

Mes remerciements vont également aux professeurs Bernard Moret et Alain Denise qui m'ont fait l'honneur de rapporter mon manuscrit de thèse. Je remercie tout autant Michel Aigle, Guy Melancon et Guillaume Fertin pour avoir accepté de lire mon travail et pour avoir été présents le jour J pour faire partie de mon jury de thèse.

Je remercie aussi l'ensemble des collègues de l'équipe MAGNOME dans laquelle j'ai "grandi" durant ces trois années : merci à David Sherman pour m'avoir donné cette opportunité, à Pascal Durrens pour ses explications pédagogiques concernant le monde compliqué de la biologie !

Je souhaite également remercier les biologistes du projet Génolevures notamment Jean-Luc Souciet et Bernard Dujon qui ont pris le temps de m'expliquer leurs problématiques et de s'intéresser à mon travail. Notre collaboration a largement contribué à ma compréhension de la génomique et à faire avancer mes recherches.

Merci également à tous mes collègues et amis du LaBRI qui m'ont toujours soutenue, écoutée et supportée au laboratoire ou en dehors : je pense à Florian Iragne, Cyril Cayla, Emmanuelle Beyne et ses travaux pratiques sur la levure de boulanger certains après-midi, Nicolas Loira avec qui j'ai pu développer mon anglo-hispanique, Hayssam Soueidan et ses petites attentions, Adrien Goëffon et ses précieux conseils, Aïda Ouangraoua pour son amitié, Maxime Amblard et son soutien durant nos longues préparations de TD de graphes, David Renault pour son "grand" côté professionnel et son amour pour le volley-ball, Antoine Rollet pour ses conseils durant mes répétitions de soutenance et surtout pour notre goût commun pour les pâtisseries tunisiennes. Un remerciement tout spécial à Tiphaine Martin qui est devenue au fil du temps une amie et une confidente, qui m'a toujours soutenue, aidée à avancer dans les moments difficiles notamment à la fin de ma thèse et avec qui j'ai pu partager bien plus que des théorèmes et des notions de biologie...

Mes remerciements ne seraient pas complets sans une petite pensée à tous les collègues que j'ai pu côtoyer en tant qu'enseignante. Merci à toute l'équipe pédagogique du département informatique de l'IUT de Bordeaux 1 et plus particulièrement à Michel Billaud, Michel Marcus, Jean-Michel Lépine, Mickaël Montassier, Alexandre Pinlou, Nicholas Journet, Olivier Ly, Isabelle Dutour et Eric Sopena pour leur gentillesse et leur envie contagieuse de partager leur savoir. Enseigner à vos côtés a été un réel plaisir.

Je n'oublie pas non plus le personnel administratif du LaBRI et de l'INRIA, en particulier Philippe Biais, Cathy Roubineau et Marie Sanchez, qui m'ont aidée à de maintes reprises et qui ont toujours eu un mot gentil pour moi.

En dehors du laboratoire, je tiens à remercier vivement les personnes qui m'ont accompagnée tout au long de ma thèse en me permettant de continuer ma passion, la GRS. Je remercie du fond du coeur la GR de Saint Pierre Montlimart et la Laëtitia de Nantes (Evelyne, Aude, Pauline, Pascaline, Hélène, Gaëlle, Elise, Maud, Marion, Bérangère, Nancy, Anne-France) pour les échanges sportifs et les moments passés ensemble malgré la distance. Je remercie également la commission fédérale GRS de la FSCF (Annick, Michel, Cathie, Cécile, Corinne, Céline, Anne-Marie, Nicole, Martine) qui m'a permis de m'investir dans mon sport tout en m'encourageant professionnellement. Enfin, je remercie la section GR de l'Union Saint Bruno (Elodie D., Elodie B., Rémédios, Aurélie, Marion et toutes mes gymnastes) qui m'a très gentiment accueillie lors de mon arrivée à Bordeaux pour me permettre de continuer ma passion. Je remercie toutes ces bordelaises qui sont devenues de vraies amies.

Je remercie bien évidemment mes amis et ma famille. Merci à mes parents de n'avoir jamais cessé de croire en moi et d'avoir montré beaucoup d'intérêt à mes recherches même s'il était parfois difficile de comprendre ce que je faisais. Un immense merci à ma chère soeur, femme de lettres, qui a toujours trouvé les mots justes pour me motiver et me remonter le moral dans les moments difficiles.

Enfin, je souhaite remercier mon homme de l'ombre. Emmanuel, tu as été là chaque jour durant cette longue aventure. Tu as toujours cru en moi, tu y as cru parfois plus que moi. Sans toi, j'aurais sans doute baisser les bras bien des fois. Alors merci.

A mes grands-pères, Raymond et Michel,

Résumé

Méthodes in silico pour l'étude des réarrangements génomiques : de l'identification de marqueurs communs à la reconstruction ancestrale.

L'augmentation du nombre de génomes totalement séquencés rend de plus en plus efficace l'étude des mécanismes évolutifs à partir de la comparaison de génomes contemporains. L'un des principaux problèmes réside dans la reconstruction d'architectures de génomes ancestraux plausibles afin d'apporter des hypothèses à la fois sur l'histoire des génomes existants et sur les mécanismes de leur formation. Toutes les méthodes de reconstruction ancestrale ne convergent pas nécessairement vers les mêmes résultats mais sont toutes basées sur les trois mêmes étapes : l'identification de marqueurs commun dans les génomes contemporains, la construction de cartes comparatives des génomes, et la réconciliation de ces cartes en utilisant le critère de parcimonie maximum.

La quantité importante des données à analyser nécessite l'automatisation des traitements et résoudre ces problèmes représente de formidables challenges computationnels. Affiner les modèles et outils mathématiques existants par l'ajout de contraintes biologiques fortes rend les hypothèses établies biologiquement plus réalistes.

Dans cette thèse, nous proposons une nouvelle méthode permettant d'identifier des marqueurs communs pour des espèces évolutivement distantes. Ensuite, nous appliquons sur les cartes comparatives reconstituées une nouvelle méthode pour la reconstruction d'architectures ancestrales basée sur les adjacences entre les marqueurs calculés et les distances génomiques entre les génomes contemporains. Enfin, après avoir corrigé l'algorithme existant permettant de déterminer une séquence optimale de réarrangements qui se sont produits durant l'évolution des génomes existants depuis leur ancêtre commun, nous proposons un nouvel outil appelé VIRAGE qui permet la visualisation animée des scénarios de réarrangements entre les espèces.

Mots-clés: génome ancestral, génomique comparative, réarrangement, point de cassure, permutation

Abstract

In silico methods for genome rearrangement analysis: from identification of common markers to ancestral reconstruction

The increase in the number of entirely sequenced genomes makes increasingly accurate the study of the mechanisms of evolution through the comparison of contemporary genomes. One of the main problems is to reconstruct plausible ancestral genome architecture, which furnishes hypotheses about both the history of contemporary genomes and the general mechanisms of their formation. While not all methods for the ancestral reconstruction necessarily converge towards the same results, they are all based on the same three steps: identification of common markers in contemporary genomes, construction of comparative maps for these genomes, and reconciliation of these maps under a maximum parsimony criterion.

The quantity of data that must be analyzed requires the automation of processing and meeting these needs induces great computational challenges. Through refinement of computational models and methods, we can obtain more biologically relevant hypotheses by adding biological constraints.

In this thesis, we propose a new method for the identification of common markers to construct comparative maps for evolutionary distant genomes. Next, we apply a new method of ancestral genome reconstruction based on adjacencies of synteny markers and genomic distances between contemporary genomes. Finally, after correcting the existing algorithm for computing an optimal sequence of rearrangements that occured during the evolution of modern genomes from their common ancestor, we propose a new tool called VIRAGE that permits the animated visualization of rearrangement scenarios between species.

Keywords: ancestral genome, comparative genomics, rearrangements, breakpoints, permutation

Contents

List of Figures

List of Tables

Introduction

Genetics is a field of biology that today aims in large part to explain the machinery and functioning of species through the study of their genetic information. Understanding the function and evolutionary processes that act on genomes enables scientists to provide scientific answers and, ultimately, new medical or therapeutic solutions to diseases.

A useful way to understand the structure and evolutionary history of a genome is to compare it to other ones. While *comparative genomics* is still a young field, it is currently undergoing a considerable expansion due notably to the advent of large scale sequencing. The huge amount of data available in sequenced genomes makes computational approaches essential so that the analyzes can be automated and performed on a large scale.

In particular, *in silico* methods are applied to study evolutionary relationships among species. A major problem consists in measuring evolution within a set of species of interest by determining the sequence of evolutionary events that make one genome evolve from another.

Evolutionary events are traditionally characterized by *mutations*. Different levels of mutations can be observed. The most commonly studied are called *punctual mutations* that modify the nucleotidic composition of the genome. Study of this mechanism led to the definition of an *edit distance* for genome sequences [Doo90]. However, considering only gene-level mutations does not provide sufficient clues for inferring evolutionary history between species. In fact, Palmer and Herbon observed in 1988 [PH88] that the major part of genes within *Brassica olearacea* and *Brassica campestri* are identical up to 99% but their genomes differ in their size and gene order.

Large-scale mutations that involve changing the relative order of large segments of DNA, enable whole genome comparison. These global mutations called *genomic rearrangements* constitute another approach to study evolutionary events. This field was pioneered by works of Dobzhansky and Sturtevant [DS38] in 1930's. Since the beginning of the nineties, the interest in the study of genomic rearrangements has increased considerably.

In this thesis, we study evolutionary events through *genomic rearrangements* based on a combinatorial and algorithmic comparison of genomes. Several challenges arise in the study of *genomic rearrangements*. Those addressed in this thesis are presented below.

Rearrangement distances and parsimonious scenarios

While punctual mutations act on a single nucleotide base by insertion, deletion or substitution, genomic rearrangements modify the order of large genome segments by reversals, transpositions and translocations (among others). Understanding evolutionary mechanisms progresses through the reconstruction of the most parsimonious sequences of rearrangements that lead to genome formation: *parsimonious scenarios*.

Computational approaches model genomes by signed permutations where each element represents a block of syntenic genes (i.e. groups of genes whose relative order is conserved between several species). Based on the parsimony criterion, the problem consists in quantifying the minimum number of operations applied to permutations, called *rearrangement distance*, and in determining what these operations are by computing the corresponding scenarios. The *sorting signed permutations by reversals* problem introduced by Sankoff [San92] was widely studied in the literature and led to efficient algorithms for solving this problem in the unichromosomal and multichromosomal cases (Hannenhalli and Pevzner theory [HP95b, HP95a]). However, computational model and associated methods do not totally agree with biological reality. In fact, such a model does not take into the account a certain number of important biological facts, first, by considering only a restrained set of operations and, second, by avoiding some constraints for studied rearrangements like centromere positioning [RAS06]. Moreover, current methods can provide a huge number of different scenarios that correspond to the same rearrangement distance [Sie02]. So, which of these scenarios is the most biologically plausible? Refining existing models by adding new biological constraints and solving these problems efficiently using tractable algorithms is a way to tackle this question. Solving it requires one in turn to address considerable computational challenges.

A related challenge lies in the visualization of plausible results in order to facilitate their interpretation by expert biologists. Indeed, genome modeling in the form of signed permutations makes the analysis and comparison of possible scenarios difficult.

Ancestral genome reconstruction

The central dogma of evolutionary biology postulates that contemporary genomes evolved from a common ancestral genome. However, the large scale study of their evolutionary relationships is frustrated by the unavailability of these ancestral organisms that, indeed, do not exist anymore. Constructing plausible hypotheses about the structural characteristics of these ancestral architectures is a computational task whose results may provide deep insight both into the past histories of particular genomes and the general mechanisms of their formation. This task suffers from the two same important difficulties as that the computation of distance and scenarios: how can we guarantee that the solution is biologically plausible? how can we find these solutions in an efficient manner?

Evolutionary inferences are based on the comparison and reconciliation of rearrangement events within contemporary genomes. Computational reconciliation is most often formulated as the *multiple genome rearrangement problem* [SSK96, HCKP95]: given a set of N contemporary genomes and a distance d, find a tree T with the N genomes as leaf nodes and assign permutations (plausible ancestral architectures) to internal nodes such that $D(T) = \sum_{(\pi,\gamma)\in T} d(\pi, \gamma)$ is minimized. When $N = 3$ this is called the *median genome problem*. Methods were developed according to different distances (breakpoint distance [SB97], reversal distance [Cap99, Cap03], rearrangement distance [BP02]). Although efficient algorithms exist to compute distances, solving the *multiple genome rearrangement problem* was proved to be NP-hard (see [Bry98, PS98] for the breakpoint distance and [Cap99, Cap03] for the reversal distance) and requires heuristics even in the case of 3 genomes.

In addition to the computational intractability of this problem, these *in silico* methods provide one single global solution chosen among a multitude of equivalent ones [Eri07] that, furthermore, do not necessarily correspond with those provided by *in vitro* methods [FCG$^+$06, BTP06]. Knowing that the computed median genome (or the root genome in the rearrangement tree) represents the basic building block for *species tree* reconstruction, this reinforces the claim that more biological knowledge is required in mathematical models [RAS06].

A more realistic approach is to consider what common structural features of ancestral genomes might be found. Partial reconciliation of modern genomes identifies permutations as above but does not necessarily provide a total order between segments. Existing algorithms (see [MZS$^+$06]) for this kind of resolution rely strongly on phylogenetic data. However, nothing suggests that recombinatory evolution coincides with mutational evolution.

Identification of common markers

Mathematical solutions for ancestral genome reconstruction are clearly sensitive to the sample of considered genomes: as the number of fully sequenced genomes increases, sampling becomes larger and ancestral reconstruction more and more accurate. However, another very important step in methods for ancestral reconstruction or distance computation lies in the careful identification of common markers used to define signed permutations. These markers represent regions of the genomes that have not been broken, since *conserved segments* between two (or more) related species indicate chromosomal homology inherited from their common ancestor. Finding conserved segments across species makes it possible to solve a dual problem, that consists in detecting *breakpoints*, which are the points between conserved segments along a genome where rearrangements have occurred.

Several methods have been defined to respond to the need for finding common markers within genomes. Among them only GRIMM-Synteny [PT03a, BPT04, BZB$^+$05] was precisely defined with the goal of rearrangement study. Unfortunately, all reports in the literature of these techniques share a common feature of not systematically providing all the necessary details as for the way that breakpoints are detected, and additionally often depend on several user-specified parameters that affect obtained results. This indicates that breakpoint (or conserved segment) detection is not a trivial problem. However, all existing methods come back to basic computational genomics: the study of punctual mutations by *alignment of genome sequences*, which is made easier by the increase of complete sequencing of genomes.

Alignment algorithms are either global (introduced in [NW70, Sel74]) or local (see Smith and Waterman [SW81]). It has been shown that global alignment of whole genomes is not appropriate for solving breakpoint detection; as an example, for widely studied mammal genomes, comparison of human and mouse led to the observation that less than the half of their genomes can be aligned [WLTB$^+$02].

The insight behind current algorithms relies on the fact that conserved segments can be aligned. This leads to *"seed and extend"* algorithms decomposed into three steps: *anchoring*, *filtering* and *extending*. While the first step is solved similarly by the current methods, the two last ones diverge. Moreover, the latter step is totally ignored in the case of GRIMM-Synteny, since its aim is to study genome rearrangements.

Besides, in this case, conserved markers resulting from this method, called *synteny blocks*, smooth over the noise due to *micro-rearrangements* for inferring possible mechanisms behind rearrangements. Beyond determining which rearrangements took place, synteny blocks (and reciprocally breakpoint detection) enable analysis of regions that were broken by rearrangements. Such analysis can provide clues on the issue of rearrangement *hotspots*. This latter topic has generated a quite lively debate on the differences between *random breakage* and *non-random breakage models* of evolution [KBH$^+$03, PT03a, PT03b, TMS04].

Moreover, all current methods were applied and perform well on the 'low-hanging fruit' of highly similar (e.g. mammalian) genomes, but less well on highly divergent genomes with extensive map reshuffling. Thus, algorithms with the ability to handle species having a large evolutionary span are required.

What this thesis is about

This thesis is divided in four parts.

The first one is dedicated to a large overview of current computational methods for solving *genomic rearrangement* challenges and questions that they raise.

In the first chapter, we introduce the mathematical model for the genome and the mechanisms of evolution. We start by defining the notion of common marker, and more precisely synteny blocks, that represent basic elements in the signed permutation model of genomes. Then, after a brief biological presentation of rearrangements, we sum mathematical operations on permutations that mimic their behaviour. Finally, genomic rearrangement challenges (rearrangement distance, parsimonious scenarios, breakpoint and multiple genome rearrangement problem) are presented according to their corresponding mathematical formulation under the permutation model.

Comparative genomics is a young and dynamic field whose rearrangement challenges are widely documented in the literature. Chapter 2 contains a presentation of main current methods and a discussion of their pertinence for each rearrangement challenge. Here we go quite deeply into the presented techniques, by providing details of algorithms and of certain approaches that are either the subject of our own work, or are of particular relevance for our results. For identification of common markers, we present the GRIMM-Synteny approach [PT03a, BPT04, BZB$^+$05], which is the only one which has been explicitly developed in order to study rearrangement events. We also describe in detail the ADHoRe [VSS$^+$02] method on which we base our work on identification of synteny for distant genomes presented in part II. Next follows a presentation of rearrangement distance and corresponding parsimonious scenarios, focused on the computation of distance based on reversals only and extended to multichromosomal genomes by taking into the account translocations, fusions and fissions as well as reversals. Besides the fact that these rearrangements are considered as the most frequent [BP02], efficient algorithms exist for this set of operations (first suggested by Kececioglu and Sankoff [KS93], then improved by Hannenhalli and Pevzner's theory [HP95a, HP95b] and thus represent adequate bases for solving ancestral reconstruction. For the latter challenge, we present the two main parsimony-based global methods (breakpoint and rearrangement distance), as well as the partial reconstruction approach based on phylogenetic considerations (Ma et al. [MZS$^+$06]).

Parts II, III, and IV are dedicated to our contributions in the domain. The developed approaches were validated on real data from the Génolevures project [DS$^+$04], a large-scale comparative genomics project across the evolutionary range of the *Hemiascomycetous yeast phylum* coordinated by the CNRS and operated by a Consortium of laboratories and research centers affiliated with different institutions. Génolevures provides an ideal application domain, since certain clades of species under study present enough synteny in order to identify common markers and therefore to apply computational methods for ancestral analysis.

We propose in part II an original approach for identifying common markers in evolutionary distant genomes. Chapter 3 presents this algorithm called SyDiG - recovering Synteny in Distant Genomes - based on ADHoRe [VSS$^+$02] results, while chapter 4 proposes a comparison with the

GRIMM-Synteny method and an application to the *Hemiascomycetous* yeasts.

In part III, a new piece-wise method for the reconstruction of ancestral architectures is presented. This method, detailed in chapter 5, is based on the study of both adjacencies between common markers and rearrangement distances between modern genomes. Moreover, it makes it possible to use biological constraints such as centromere position. Without any phylogenetic considerations, this leads to the construction of *super-blocks* that represent common ancestral features. After a comparison with existing global and partial methods of ancestral reconstruction, chapter 6 presents the resulting sets of super-blocks obtained for the signed permutations of *Hemiascomycetous* yeasts computed in their turn by the SyDiG algorithm.

The last part addresses the problems of computing and visualizing optimal rearrangement scenarios between putative reconstructed ancestral genomes and contemporary ones. Chapter 7 proposes a single and coherent classification of the notions involved in existing algorithms for computing a parsimonious scenario between two multichromosomal genomes. This classification makes it possible to pinpoint the fact that current algorithms present errors. In the same chapter, we introduce a correct algorithm with a proof of its correction. Finally, chapter 8 is dedicated to the presentation of a new tool called VIRAGE that we have developed for the interactive visualization of rearrangement mechanisms between genomes and which permits a more comfortable rearrangement analysis by biologists.

Part I

Preliminaries

Chapter 1

Modeling a genome and evolutionary mechanisms

The comparison of genomes is a fundamentally powerful way to understand their structure and evolutionary history. Evolutionary events are traditionally characterized by *mutations*. Two main scales of mutations are observed: punctual mutations and genomic rearrangements.

The comparison of genomes through *punctual mutations* consists in aligning nucleotidic sequences extracted from the entire genome sequences we aim to compare. The study of these gene-level mutations leads to a local sequence-based comparison of species, that does not use all of the available information. A higher level of mutations represents another way to study genomes by comparing them globally. These global mutations called *genomic rearrangements* modify the order and the content in terms of genes within the genomes on which they operate.

The number of entirely sequenced genomes becomes more and more important every year and thus the amount of relevant data becomes so huge that automating processing has become essential. Use of computational methods to study genome rearrangements requires one to define a mathematical model of the genome in order to represent all of the information that it contains relative to the content and the order of its genes. However, the content of genes themselves is not to be modeled since in such a study we are interested in large-scale mutations.

In this chapter, we present the *signed permutation model* commonly used to study genome rearrangements. Permutations are constructed based on common elements between several genomes that are supposed to be inherited from a common ancestor. In section 1.1, we define what are these common markers and more precisely the notion of synteny blocks. In the next section, we present the principal rearrangement operations that are encountered and the corresponding mathematical operations on permutations. Genomic rearrangements are at the heart of several challenges: recovering rearrangements that lead to the formation of a novel species, reconstructing gene architecture of ancestor that have vanished today. The last section of this chapter formulates all of these goals in a mathematical way based on the permutation model.

1.1 Common markers: what is a syntenic block?

The comparison between species and more precisely the study of evolutionary mechanisms of genomes proceeds through the definition of points of comparison, called common markers, situated on the genomic sequences of organisms. This is done by comparing the genomes of the species under study.

1.1.1 Genetic information is contained in the genome

The whole genetic information of a species is encoded in its *DNA (desoxyribonucleic acid)* molecules and constitutes the *genome*.

The DNA has two complementary strands where each strand is composed of sequences of *nucleotides*, or *bases*. The four bases found in DNA are adenine (abbreviated A), cytosine (C), guanine (G) and thymine (T). It is the order, the nature and the number of nucleotides that encode the genetic information. From the sequence of one strand of DNA, it is possible to find its complementary sequence by replacing each base by its complement and reversing the sequence. Adenine and thymine are complementary, as are guanine and cytosine. DNA strands are *oriented* from $5'$ to $3'$ according to links $5'$-$3'$ between desoxyribose rings that join nucleotides. The arrangement of DNA strands is called *antiparallel*: the direction of the nucleotides on one strand is opposite to their direction on the other strand (the $5'$ extremity of one strand gets in contact with the $3'$ extremity of the other strand and vice versa).

The *genome* is divided into one or several *chromosome(s)*, each carrying a set of *genes*. A gene is a region of a chromosome, which contains a coding sequence. The majority of coding sequences are transcribed into mRNAs (messenger RiboNucleic Acids) which, in turn, are translated into proteins. The remaining coding sequences are transcribed into RNAs, which are not translated into proteins. For the sake of simplification, we will refer to protein coding sequences as genes. We can define an orientation for each gene. In fact, a gene is present on the two DNA strands (major and complementary) but the transcription process is performed from only one strand. In the case where a gene is transcribed from the major $5'$-$3'$ strand of the DNA sequence, it is said to be directly oriented. If the transcription process is done from the complementary $3'$-$5'$ strand, the gene has the reverse orientation.

1.1.2 Common markers between species

The genome sequence of an organism is inherited from its parents, and in the context of this work is considered to be the same for all members of the same species. The genome sequence of a species is derived through evolution from the sequence of its ancestor species, and related species will have inherited common sequence features from their last common ancestor. This inheritance is as the core of the study of genome rearrangements.

From genes to synteny blocks

Whole genome sequencing makes possible the comparison of genomes by defining *common markers*. Highly similar DNA sequences are called *homologs*. If sequences correspond to genes, we speak about *homologous genes*, where we distinguish *orthologs*, genes in different species that evolved from a common ancestral gene by speciation, from *paralogs*, genes related by duplication within a genome. These homology points define common markers between the genomes of different species. Common markers can also be defined at a higher level of abstraction. Nadeau and Taylor in [NT84] introduced the notion of *conserved segments* that are segments with preserved gene orders without disruption by rearrangements in different species.

In order to mask multiple microrearrangements in a whole genome comparison, one can use *synteny blocks*, which usually consist of short regions of similarity that may be interrupted by dissimilar regions and gaps (definitions are given in [PT03a]). Intuitively, synteny blocks can be converted into conserved segments by microrearrangements. A detailed discussion of synteny blocks and their construction can be found in chapters 2 and 3.

Sign of a common marker

Once common markers are defined between two or more species, a *sign* can be associated with each of them to indicate relative changes in orientation. Signs of common markers in one genome are determined relative to an arbitrarily chosen *reference genome.*

Let Π and Γ be two genomes, and Π be the reference genome. For a common marker σ of (arbitrarily chosen) sign s in Π, we have:

- if σ is a gene, which has the same orientation in Γ as in Π, then σ has s as sign in Γ. Otherwise, the sign of σ is $-s$ in Γ;
- in the case of a conserved segment (synteny block, respectively), s in Γ depends on the order and the signs of common elements in this segment (synteny block, respectively) compared with those in Π (see section 2.1 for details function to the considered method).

It is sometimes not possible to give a sign to common markers. This can happen for example when gene orientation is unknown or when information about order and orientation is insufficient for making an unambiguous choice.

1.2 Mimicking evolutionary mechanisms by operations on permutations

1.2.1 The genome: a signed or unsigned permutation

A genome is a set of chromosomes while a chromosome is a list of markers. These markers can be genes or syntenic blocks. In this thesis we are not concerned by the comparison of the content of markers, only by the gene order in the genome and on its chromosomes. Thus, in the chosen model, a marker is represented by an identifier, signed or not (see section 1.1) and a chromosome can be seen as a list of signed or unsigned identifiers, that is, a permutation.

Let $\Pi = \{\pi^1, ..., \pi^{N_\Pi}\}$ be a *multichromosomal genome* defined as a set of N_Π *chromosomes.* The i^{th} chromosome $\pi^i = \pi^i_1 ... \pi^i_{n_i}$ is a sequence of n_i markers. The *order* of π^i is n_i. Because of the complementarity of the two DNA strands, any chromosome π can be represented in two distinct ways: “from left to right” (i.e. $\pi = \pi_1\pi_2...\pi_n$) or “from right to left” (i.e. $-\pi = -\pi_n ... - \pi_2 - \pi_1$). These two representations are equivalent. Thus, several equivalent forms are possible for the same genome. For example, the genome $\left\{\pi^1, \pi^2, \pi^3\right\}$ can be written as $\left\{\pi^1, -\pi^2, \pi^3\right\}$ or $\left\{-\pi^1, -\pi^2, -\pi^3\right\}$, etc.

Note 1 *For an unichromosomal genome* $\Pi = \{\pi^1\}$, *the notation* π *represents either the entire genome or the unique chromosome.*

The structure of genomes varies between organisms. Genomes of prokaryotes as well as those of organelles such as mitochondria or chloroplasts are characterized by an unique circular chromosome. For eukaryotes, several linear chromosomes form the genome. Over time, chromosome architecture evolves through rearrangement mechanisms. The different possible rearrangements that can occur in different kinds of genomes are described in section 1.2.2.

1.2.2 Rearrangements: different possible operations

Genomic rearrangements modify the genome content or the gene order. Operations such as duplications, insertions or deletions add or delete DNA fragments in the initial genome without modifying the gene order. Reversals, translocations and transpositions are operations that modify the gene order by moving DNA fragments into a chromosome or from one chromosome to another. Combinations of these operations modify both gene content and gene order.

Presentation of possible operations

Duplication Duplication inserts chromosomal fragments of variable length. In general, the new DNA fragment is inserted besides the repeated one.

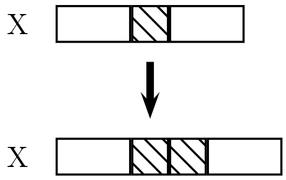


Figure 1.1: Duplication of a gene on the chromosome X.

Insertion and deletion A new DNA fragment can appear on a chromosome during the evolution of a species. This is called gene or segmental *insertion*. The symmetric event of DNA loss is called gene or segmental *deletion*.

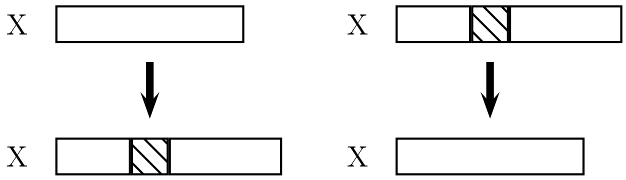


Figure 1.2: Insertion (left) and deletion (right) of a gene on the chromosome X.

Reversal Reversal is a modification of the DNA structure that consists in a 180° rotation of a chromosomal segment most often without loss of genetic material. Thus, a reversal modifies the orientation of involved genes.

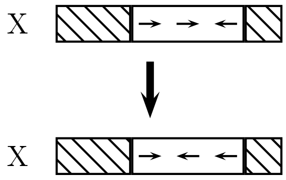


Figure 1.3: Reversal of a gene sequence on the chromosome X. The involved genes belong to the white segment. Small arrows indicate gene orientation.

Translocation Translocation is a mutation that only occurs in multichromosomal genomes, since two chromosomes must be involved. Translocation is a exchange of genetic material between two chromosomes. Figure 1.4 presents a translocation where the sequences at the end of two chromosomes are exchanged.

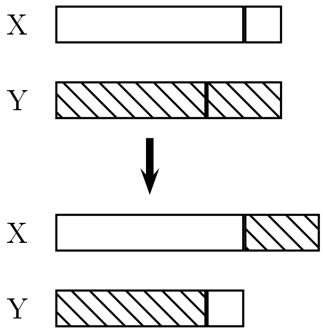


Figure 1.4: Translocation of the chromosomes X and Y.

In the work of certain authors (e.g. Hannenhalli [Han96]), other types of translocations are considered in order to measure evolution between species. In these translocations, other chromosomal segments than suffixes can be combined by a reversal. For example, Hannenhalli in [Han96] presents the prefix-suffix translocation with reversal: the prefix of a chromosome is exchanged with the suffix of an other one and the exchanged sequences are reversed. The figure 1.5 describes this mechanism.

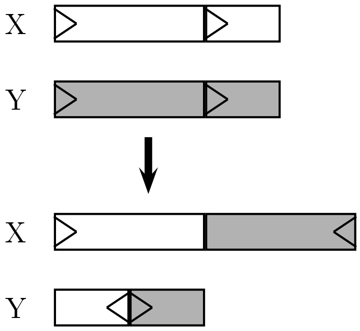


Figure 1.5: Prefix-suffix translocation of the chromosomes X and Y. The prefix of Y and the suffix of X are exchanged: these sequences are reversed during the translocation.

Fission and fusion These rearrangements are particular cases of translocation. Fission is a mechanism that separates a chromosome into two distinct chromosomes (see figure 1.6).

Fusion is the opposite mechanism that joins together two chromosomes into an unique one (see figure 1.7).

Transposition Transposition is a mechanism that consists in moving a DNA sequence along a chromosome. It may or may not involve a reversal as shown on figure 1.8.

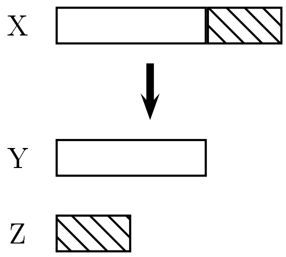


Figure 1.6: Fission of chromosome X into two chromosomes Y and Z.

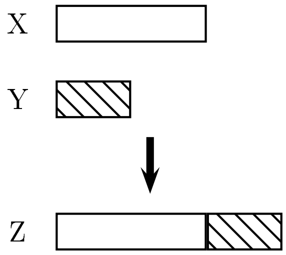


Figure 1.7: Fusion of chromosomes X and Y into the chromosome Z.

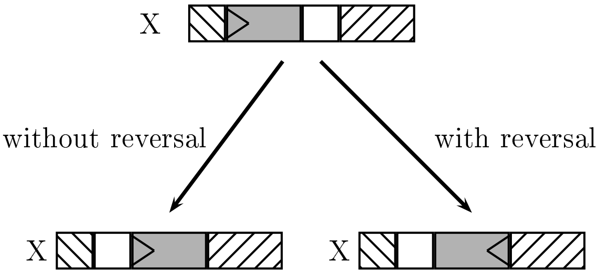


Figure 1.8: Transposition of the genes belonging to the grey segment on the chromosome X with (right) or without (left) reversal.

Mathematical operations for rearrangements

Rearrangements that do not modify gene content (see section 1.2.2) can be modeled by mathematical operations on permutations representing a genome (see section 1.2.1). Table 1.1 shows mathematical operations corresponding to biological rearrangements applied to a multichromosomal and signed genome $\Pi = \{\pi^1, ..., \pi^{N_\Pi}\}$. The application of a rearrangement ϕ to the genome Π results in the genome $\Pi' = \Pi.\phi$.

Rearrangement	**Notation**	**Resulting permutations**
Reversal	$\rho\left(\pi^i, k, l\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{k-1} - \pi^i_l - \pi^i_{l-1} ... - \pi^i_{k+1} - \pi^i_k \pi^i_{l+1} ... \pi^i_n$
Translocation	$t\left(\pi^i, \pi^j, k, l\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{j-1}, \pi^{j\prime}, \pi^{j+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{k-1} \pi^j_l ... \pi^j_{n_j}$ and $\pi^{j\prime} = \pi^j_1 ... \pi^j_{l-1} \pi^i_k ... \pi^i_{n_i}$
Translocation **with reversal**	$t_{rev}\left(\pi^i, \pi^j, k, l\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{j-1}, \pi^{j\prime}, \pi^{j+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{k-1} - \pi^j_{l-1} ... - \pi^j_1$ and $\pi^{j\prime} = -\pi^i_{n_i} ... - \pi^i_k \pi^j_l ... \pi^j_{n_j}$
Fusion	$t\left(\pi^i, \pi^j, n_i + 1, 1\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{j-1}, \pi^{j+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{n_i} \pi^j_1 ... \pi^j_{n_j}$
Fission	$t\left(\pi^i, \emptyset, k, 1\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{N_\Pi+1\prime}, \pi^{i+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{k-1}$ and $\pi^{N_\Pi+1\prime} = \pi^i_k ... \pi^i_n$
Transposition	$r\left(\pi^i, k, l, m\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{N_\Pi}\}$ with $\pi^i\prime = \pi^i_1 ... \pi^i_{k-1} \pi^i_{l+1} ... \pi^i_{m-1} \pi^i_k ... \pi^i_l \pi^i_m ... \pi^i_n$
Transposition **with reversal**	$r_{rev}\left(\pi^i, k, l, m\right)$	$\Pi' = \{\pi^1, ..., \pi^{i-1}, \pi^{i\prime}, \pi^{i+1}, ..., \pi^{N_\Pi}\}$ with $\pi^{i\prime} = \pi^i_1 ... \pi^i_{k-1} \pi^i_{l+1} ... \pi^i_{m-1} - \pi^i_l ... - \pi^i_k \pi^i_m ... \pi^i_n$

Table 1.1: Rearrangements considered as mathematical operations on permutations.

1.3 Mathematical measure of evolution

1.3.1 Rearrangement distance

Measuring the evolutionary distance between two species is one part of comparative genomics analysis. This distance can be formulated in terms of genomic rearrangements. In our case, for a given set of genomic rearrangements, the problem consists in quantifying the minimum number of rearrangements that transform one genome into another. This measure relies on the parsimony principle and defines a distance in the mathematical sense of the word. A distance on a set E is a function $d : E \times E \to \mathbb{R}$ verifying:

(i) $d(x, x) = 0$ for all $x \in E$,

(ii) $d(x, y) > 0$ for all $x, y \in E$ with $x \neq y$,

(iii) $d(x, y) = d(y, x)$ for all $x, y \in E$,

(iv) $d(x, y) \leqslant d(x, z) + d(z, y)$ for all $x, y, z \in E$.

In order to provide results that are more biologically realistic, the operations are sometimes weighted. The weight depends on either the type of the considered operation, or the length of the implied genetic material. The distance (provided that the weight functions are mathematical distances) is then the minimal sum of the costs taken among all the sequences of operations that transform one genome into another.

1.3.2 Parsimonious rearrangement scenario

In the same way, it is important to determine evolutionary scenarios corresponding to a rearrangement distance d, that are called *parsimonious scenarios.* A parsimonious rearrangement scenario between genomes Π and Γ is a sequence of rearrangements $(\phi_1, .., \phi_n)$ that transforms genome Π into Γ such that $d(\Pi, \Gamma) = n$.

Figure 1.9 gives an example of a parsimonious scenario for two unichromosomal genomes π and γ. The number of reversals in the scenario is equal to the reversal distance between them: $d(\pi, \gamma) = 4$.

$$\pi = \quad -9 \quad -8 \quad \underline{+6 \quad +7} \quad +2 \quad +3 \quad +4 \quad +5 \quad +1$$

$$-9 \quad -8 \quad -7 \quad -6 \quad \underline{+2 \quad +3 \quad +4 \quad +5} \quad +1$$

$$-9 \quad -8 \quad -7 \quad -6 \quad -5 \quad -4 \quad -3 \quad -2 \quad \underline{+1}$$

$$\underline{-9 \quad -8 \quad -7 \quad -6 \quad -5 \quad -4 \quad -3 \quad -2 \quad -1}$$

$$\gamma = \quad +1 \quad +2 \quad +3 \quad +4 \quad +5 \quad +6 \quad +7 \quad +8 \quad +9$$

Figure 1.9: One parsimonious scenario between unichromosomal genomes π and γ. The first line represents the genome π, the last, the genome γ and all the lines except for the first are obtained from the previous one by a reversal of the underlined segment.

Note that, although computations of the rearrangement distance and of a parsimonious scenario are two closely related problems, they are often resolved independently in the relevant literature (see section 2.2 of chapter 2).

1.3.3 Breakpoints

Chromosomal segments involved in rearrangements can be identified in permutations by the sets of corresponding consecutive markers. These sets are delineated by *breakpoints.* This notion was introduced by Nadeau and Taylor [NT84] in 1984, and we can distinguish the signed case (see definition 3) from the unsigned one (see definition 2).

Definition 1 *Two consecutive elements π_i and π_{i+1} of a chromosome π are said to be* adjacent *in a genome Π. Denote by $\pi_i.\pi_{i+1}$ an* adjacency *between π_i and π_{i+1}.*

Definition 2 *For two unsigned genomes Π and Γ, if two elements π_i and π_{i+1} are adjacent in Π but neither $\pi_i.\pi_{i+1}$ nor $\pi_{i+1}.\pi_i$ are present in Γ, then the pair $\pi_i.\pi_{i+1}$ forms a* breakpoint *in Π with respect to Γ.*

Definition 3 *For two signed genomes Π and Γ, if two elements π_i and π_{i+1} are adjacent in Π but neither $\pi_i.\pi_{i+1}$ nor $-\pi_{i+1}. - \pi_i$ are present in Γ, then the pair $\pi_i.\pi_{i+1}$ forms a* breakpoint *in Π with respect to Γ.*

When genomes are linear, supplementary adjacencies have to be taken into the account: the ones between the beginning of a chromosome and its first element and the ones between the last element of a chromosome and its end. Figures 1.10, 1.11 and 1.12 present breakpoints in a genome Π according to a genome Γ for different natures of genomes (unichromosomal without information about orientation (i.e. unsigned), multichromosomal, and circular, respectively).

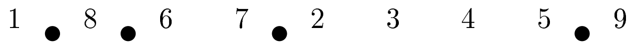


Figure 1.10: Breakpoints in $\Pi = \{1\ 8\ 6\ 7\ 2\ 3\ 4\ 5\ 9\}$ with respect to $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$ where Π and Γ are unichromosomal and linear genomes.

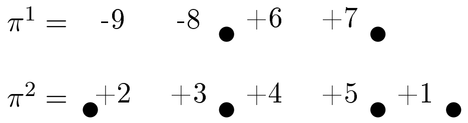


Figure 1.11: Breakpoints in $\Pi = \{-9\ -8\ +6\ +7,\ +2\ +3\ +4\ +5\ +1\}$ with respect to $\Gamma = \{+1\ +2\ +3,\ +4\ +5\ +6\ +7\ +8\ +9\}$ where Π and Γ are multichromosomal and linear genomes.

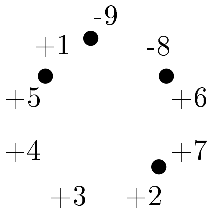


Figure 1.12: Breakpoints in $\Pi = \{-9\ -8\ +6\ +7\ +2\ +3\ +4\ +5\ +1\}$ with respect to $\Gamma = \{+1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9\}$ where Π and Γ are unichromosomal and circular genomes.

The notion of breakpoint leads to a first distance used to measure evolution between species: the *breakpoint distance.* Let Π and Γ be two genomes of respectively N_Π and N_Γ chromosomes. The breakpoint distance $b(\Pi, \Gamma)$ is equal to the number of breakpoints in Π (Γ, respectively) with respect to Γ (Π, respectively). Indeed, the number of breakpoints in Π is equal to the number of those in Γ.

In the case that $N_\Pi < N_\Gamma$, the number of breakpoints is $b(\Pi, \Gamma) = |\{(\pi_i, \pi_{i+1})|\pi_i.\pi_{i+1}$ is a breakpoint in $\Pi\}| + (N_\Gamma - N_\Pi)$ or $b(\Pi, \Gamma) = |\{(\gamma_i, \gamma_{i+1})|\gamma_i.\gamma_{i+1}$ is a breakpoint in $\Gamma\}|$.

The computation of the rearrangement distance and of parsimonious scenarios are closely related to breakpoints, since the transformation of a permutation into another consists in removing breakpoints in order to obtain the target permutation.

1.3.4 Multiple genome rearrangement problem

Measuring the evolutionary distance between contemporary species goes through the implicit reconstruction of ancestral genomes: since the ancestral species no longer exist, we do not know their true genomes. Evolutionary relationships between species, extinct or contemporary, are expressed through a *species tree*.

Computational inference of species trees is most often formulated as the *multiple genome rearrangement problem* [SSK96, HCKP95]: given a set of N contemporary genomes and a distance d, find a tree T with the N genomes as leaf nodes and assign permutations (plausible ancestral architectures) to internal nodes such that

$$D(T) = \textstyle\sum_{\{\pi,\gamma\}\in T} d(\pi,\gamma) \text{ is minimized.}$$

When $N = 3$ this is called the *median genome problem*. In the general case, this formulation corresponds to the well-known *Steiner tree* mathematical problem [HRW92], which was shown to be NP-complete (see [Bry98, PS98] for the breakpoint distance and [Cap99, Cap03] for the reversal distance).

Note that species trees reconstructed from this definition do not necessarily coincide with *phylogenetic trees*, which are trees in which each node with descendants represents the most recent common ancestor of the descendants, and the edge lengths correspond to time estimates.

Chapter 2

From common markers to evolution scenarios

In chapter 1, we presented the mathematical model of signed permutations commonly used to study genome rearrangements in computational approaches. In this thesis, we are interested in three main rearrangement challenges that are strongly related and for which the mathematical formulation is given in chapter 1.

- *Identifying common markers between genomes*: common markers are at the origin of the construction of the permutation encoding a genome. Their identification requires careful attention, since all of the rearrangement studies and hence all of the inferred biological hypothesis are based on the obtained permutations.
- *Computing evolutionary distances and parsimonious scenarios between two genomes*: measuring evolution between species implies the reconstruction of the sequence of rearrangements that separates one genome from another. Finding the minimal number of rearrangements leads to the computation of the rearrangement distance between two genomes.
- *Recovering ancestral architectures*: modern genomes evolved from a common ancestral genome that no longer exists. Finding common structural features of ancestral genomes makes possible understanding the past history and evolutionary mechanisms that lead to contemporary genomes.

All of these rearrangement challenges represent computational tasks that are widely documented in literature. Since the beginning of this research field pioneered by Dobzhansky and Sturtevant [DS38], two main problems are addressed: how can we find these solutions in an efficient manner? how can we guarantee that the uncovered solution is biologically plausible?

Chapter 2 proposes the current state of existing computational methods for all of these challenges. We give a fully detailed presentation of certain approaches on which our work was more precisely focused. We also discuss the pertinence of their solutions and provide a brief presentation of the debates that they have sparked.

2.1 Identification of genome synteny

Studying evolution mechanisms of genomes through the analysis of signed permutations and their transformations makes sense only if these permutations faithfully describe biological information contained in the genomes. Elements of these permutations represent common markers between species that have to be carefully defined. These markers, in turn, represent *conserved segments* that have not been broken, since between two (or more) related species, they indicate chromosomal homology inherited from their common ancestor.

Many methods developed for this purpose are *"seed and extend"* algorithms decomposed in three main steps. First, genome sequences are *anchored* by detecting strongly conserved regions through local alignments. The two last steps are different according to the considered methods. Second step consists in filtering the *anchors*: removing anchors obtained by chance and choosing an exemplar of duplicated regions is done by clustering or chaining anchors. Finally, the obtained conserved segments are aligned.

In what follows, we present in a more detailed way the GRIMM-Synteny method [PT03a, BPT04, BZB$^+$05] that is the only one explicitly defined in the aim of rearrangement study. We also detail i-ADHoRe [VSS$^+$02, SVSP04, SJSV08] method on which we base our work on identification of synteny for distant genomes presented in part II.

2.1.1 Grimm-Synteny

GRIMM-Synteny method [PT03a, BPT04, BZB$^+$05] was developed in the aim of rearrangement study. That is why, the latter step of traditional "seed and extend" algorithm is ignored. Moreover, the GRIMM-Synteny method does not compute conserved segments but *syntenic blocks*. These blocks correspond to conserved segments up to microrearrangements.

Careful readers well remark that a very similar method to GRIMM-Synteny is evoked in [ST05] and explained under the name ST-synteny in [PPT06]. This method need not be considered. In fact, Sankoff in [San06] explains that it is not "an alternative way of constructing syntenic blocks; the so-called ST-synteny was only a (bungled) attempt to mimic Pevzner and Tesler's method, based on our reading or misreading of their paper [PT03a]".

Anchoring

The first step of the method detailed in [BPT04] consists in finding potential regions of homology, as the *anchors*, that represent the starting point of synteny blocks. It proceeds in two successive processing steps: a filtering step of anchors called GRIMM-Anchor is applied after their computation by local alignments.

Local alignments Anchors are found by preprocessing alignments. Initially, GRIMM-Synteny uses gapped alignments given by PatternHunter [MTL02]. A more recent version of GRIMM-Synteny [BZB$^+$05] identifies anchors based on BLASTZ algorithm [SKS$^+$04], which provides the best results on non-coding regions. They also evoke in [BZB$^+$05] a large-scale detection based on genes for treating more distant genomes. Next, anchors are restrained to a set of non-overlapping and unique ones by applying GRIMM-Anchor.

GRIMM-Anchor This preprocessing is used to separate unique hits from repeats. The level of an anchor indicates the number of genomic intervals it concerns, one per involved genome.

For two genomes, the method consists in building a graph where each vertex corresponds to a maximally contiguous region of genomic intervals, called *superinterval*, and where an edge between two superintervals is added if at least two regions of them share an alignment. Such alignments are called *supporting alignments*. Only alignments of unique regions corresponding to connected components consisting of only one edge are retained. They are transformed into two-way anchors only if all corresponding supporting alignments have the same sign, otherwise the connected component is also discarded. In this case, the coordinates of constructed two-way anchors whose sign is the one of their supporting alignments, are defined by those of corresponding superintervals.

The search for N-way anchors ($N > 2$) consists in keeping only the intersecting genomic interval from two-way anchors of the considered genomes. For example, to provide three-way anchors for genomes G_1, G_2 and G_3, all triples of two-way anchors (G_1^1, G_2^1, σ^1), (G_1^2, G_3^2, σ^2) and (G_2^3, G_3^3, σ^3), where G_j^i represent coordinate superintervals and $\sigma^i = +1$ or -1, are identified. A three-way anchor is defined if signs are consistent ($\sigma^1\sigma^2\sigma^3 = 1$) and intervals G_1^1 and G_1^2 (G_2^1 and G_2^3, G_3^2 and G_3^3 respectively) of genome G_1 (G_2 and G_3 respectively) overlap. This three-way anchor is represented by the interval $G_1^1 \cap G_1^2$ with sign 1 in G_1, $G_2^1 \cap G_2^3$ with sign σ^1 in G_2 and $G_3^2 \cap G_3^3$ with sign σ^2 in G_3.

Clustering

Then, the computation of syntenic blocks consists in combining close anchors together without consideration of order and orientation. This clustering step is based on the anchors whose level is equal to the number of species under study.

The proximity between anchors is based on the Manhattan distance. Let $\{G_i\}$ be a set of N genomes and g_i^j be a coordinate within G_i. The Manhattan distance between two points $(g_1^1, g_2^1, .., g_N^1)$ and $(g_1^2, g_2^2, .., g_N^2)$ in the same chromosome tuple is $\sum_{i=1}^{N} |g_i^2 - g_i^1|$. If two points are not defined on the same chromosome tuple, their Manhattan distance is defined as infinite. Hence, the Manhattan distance between two N-way anchors on the same chromosome tuple is the Manhattan distance between their nearest endpoints (There are two terminals for each anchor determining by the signs of the alignments).

In [PT03a], two anchors are joined together if their Manhattan distance is inferior to a user-specified threshold. In [BZB$^+$05], this clustering step is done in a slightly different way. First, the nearest endpoints of the two anchors are determined thanks to the Manhattan distance. Then, GRIMM-Synteny combines or not these two anchors according to per species distances: if, in all species, the distance $|g_i^2 - g_i^1|$ is less than the per-species threshold for G_i, then the anchors are joined together.

Finally, within the obtained set of anchor clusters, those considered as too small are discarded following the hypothesis that short blocks may be caused by chance. In the original version of GRIMM-Synteny [PT03a], a user-specified parameter allows one to keep only clusters whose span is at least a minimum size in the reference species (i.e human). In [BZB$^+$05], authors propose to fix a minimum size per species.

Ordering and signing

Ordering and signing anchor clusters are two important steps that require careful attention due to consequences that involve during rearrangement analysis. However, details about them are quite nebulous in the literature about GRIMM-Synteny.

Clusters are not supposed to overlap, but their span intervals may overlap within one of the considered species. That is why, the authors in [PT03a] compute the *center of mass* of all anchors forming a cluster and order clusters according to the coordinates of their centers of masses. This leads to the numbering of clusters according to their order in a reference species. However, the notion of center of mass is not clearly defined.

Concerning the assignation of cluster orientation, the method is detailed in [PT03c] and is based on the notion of separable permutations. Let the permutation $\pi = (1, .., m)$ be a cluster of anchors in the reference species G_1 and $\gamma = (\gamma_1, .., \gamma_m)$ be the signed permutation corresponding to the same cluster in another species G_2. Permutation γ is *separable* if $(\gamma_1, .., \gamma_r)$ is a signed permutation of $(1, ..r)$ for some $r = 1, .., m - 1$. Sign of π being 1, the sign of γ denoted by σ is defined as follows:

- if $m = 1$, $\sigma = \delta$ such that $\gamma = (\delta)$,
- for $m > 1$, if γ is separable, then $\sigma = 1$,
- for $m > 1$, if $-\gamma = (-\gamma_m, .., -\gamma_1)$ is separable, then $\sigma = -1$,
- otherwise, it is not possible to define clearly the sign of γ. Authors in [PT03c] choose $\sigma = 1$ by default or discard this cluster.

In the case of more than 2 genomes, the signs of a cluster are all determined relative to the one in reference species.

Strips of clusters

The last step defines synteny blocks by combining clusters into *strips*. A strip is a sequence of consecutive signed clusters $\pi_1, .., \pi_n$ in the reference species that either appear consecutively in the same way or in the inverse $-\pi_n, .., -\pi_1$ in another genome. Strips are formed without any consideration of distance between clusters.

2.1.2 I-AdHoRe

Generally, existing methods detect similar sequences either based on nucleotide comparison, or on the gene level. In the latter case, the study of genes enables the detection of homology between chromosomal regions that are highly divergent. I-AdHoRe (iterative Automatic Detection of Homologous Regions) method [VSS+02, SVSP04, SJSV08] is based on this approach: the method consists in identifying chromosomal regions showing a conservation of gene order and content. Obtained results are called *multiplicons*, where the level indicates the number of homologous segments it contains. I-AdHoRe first detects multiplicons of level two by AdHoRe (Automatic Detection of Homologous Regions) routine. Next, by iterating the process, new genomic segments are added to existing multiplicons in order to increase their level.

Input data

AdHoRe and i-AdHoRe methods require the data set of genes with their absolute or relative position on a genomic sequence and their orientation. Homologous genes are determined using BLASTP [AGM+90], which compares amino acid sequences instead of traditional nucleotidic ones.

Detection of multiplicons of level two

Gene Homology Matrix The AdHoRe method [VSS+02] tries to determine chromosomal regions said to be *collinear*, that is, regions sharing a significative conservation of gene order and content. The AdHoRe algorithm first constructs a *Gene Homology Matrix* (GHMs) for each pair of chromosomes. Within this matrix, lines and columns correspond to positions of genes in chromosomes. A non-zero value is assigned to cells whose the line and the column form a pair of homologous genes. A positive or negative sign is attributed to this kind of cells, whether homologous genes have the same orientation or not. Non-zero cells represent the anchors, on which is based the detection of collinear regions.

Anchor Clustering Collinear regions primarily correspond to a set of anchors that have the same sign and that present a proximity within the matrix. This proximity is measured by a special “distance function”, which gives priority to anchors close in the diagonal rather than in the vertical or horizontal axes. This measure, called DPD (Diagonal Pseudo Distance), is not a distance in the mathematical sense of the term, since the triangle inequality is not verified. For two points (x_1, y_1) and (x_2, y_2) in the matrix, the DPD is:

$$d = 2\max(|x_2 - x_1|, |y_2 - y_1|) - \min(|x_2 - x_1|, |y_2 - y_1|).$$

A user-specified parameter fixes the maximal pseudo-distance DPD between two anchors in the same collinear regions and the determination of such regions is realized by successive iterations of anchor clustering by gradually increasing values of DPD until a fixed threshold. Moreover, before each iteration, a quality filter conserves only the most significative clusters in terms of the number of anchors, of the quality of the diagonal and so on. Finally, certain clusters called *base clusters* are merged into a larger one if their DPD is lower than the threshold. Final clusters are called *metaclusters*, which are formed of one or several base clusters.

The clustering process is first distinctly realized on the set of positive anchors and on the set of negative ones. A post-processing consists in combining both orientation classes by clustering clusters from different orientation sets if possible. However, in this case, it is not clearly stated in [VSS+02] how orientation is chosen for the resulting multiplicons.

Detection of higher-level multiplicons

In order to detect multiplicons of higher level, i-AdHoRe algorithm is based on multiplicons of level two for which it tries to add, in an iterative way, one or several genomic segment(s).

Segments that constitute existing multiplicons (of level two initially) are used to create *profiles*. A profile is a multiplicon whose segments are aligned in a such way that homologous genes are located at the same position. Then, these profiles are compared to gene lists (i.e chromosomes) from input data in a way analogous to the AdHoRe algorithm [VSS+02]: GHMs are constructed where lines correspond to positions of genes in a chromosome while columns represent positions of genes in a profile. If an additional segment is detected in the matrix, it is added to the existing multiplicon and the corresponding profile is updated. The whole process is repeated in order to find potential multiplicons of superior levels.

Note that, whatever its level, a multiplicon corresponds to a metacluster and hence is formed of one or several base cluster(s). Moreover, extremities of genomic segments that define a multiplicon are determined by the leftmost and rightmost coordinates of its anchors in the metacluster.

2.1.3 Other methods

Several methods have been defined to respond to the need for finding common markers within genomes. Recently, in [LS08], Claire Lemaitre and Marie-France Sagot propose a survey on the methods for detection of conserved segments. They focus their work on GRIMM-Synteny, which was already presented in section 2.1.1, CHAINNET [KBH+03], MAUVE [DMBP04] and an algorithm provided by Couronne and Patcher [CPB+03] (denoted by CP). They claim that these four methods are representative of the numerous methods that exist in the domain.

While GRIMM-Synteny was developed in order to study rearrangements, the others were computed for other goals as alignments of conserved regions. Because alignments of whole genome sequences are not appropriate for this purpose (see [WLTB+02]), all are defined as "seed and extend" algorithms decomposed in three steps: (1) anchoring, (2) filtering and (3) aligning.

The first step requires local alignments of genome sequences: anchors are defined from ungapped (CHAINNET) or gapped (GRIMM-Synteny and CP) local alignments using tools like BLASTZ [SKS+04] or PatternHunter [MTL02], or exact matches (MAUVE). MAUVE can be more stringent than other methods, since it was developed for bacterial organisms, that share a much higher proportion of coding regions than mammals studied by the other approaches.

The second step is required to remove anchors obtained by chance and choose an exemplar of duplicated regions. This is done by clustering close anchors by computing a distance between them (GRIMM-Synteny, CP) or by chaining anchors according to anchor order and orientation as well as their distance.

Except GRIMM-Synteny, which computes synteny blocks, all of the other methods proceed in a third step in order to provide final alignments of the genomic sequences.

To conclude, all methods have in common that their description in the literature does not always provide all the details concerning the way of detecting breakpoints and to often depend on user-specified parameters that affect obtained results.

2.1.4 Fragile breakpoint model versus random breakpoint model

Finding conserved segments across species enables one to solve a dual problem, that consists in detecting *breakpoints*, which are the regions between conserved segments along a genome where rearrangements have occurred. Breakpoints are less conserved regions that were broken by rearrangements and their analysis can give clues on the issue of *hotspots* of rearrangements. A quite lively debate between *random breakage* and *non-random breakage models* of evolution divides authors in two groups.

The proponents of the non-random distribution of breakpoints along a genome build their theory on two main observations. The analysis of breakpoint sequences shows that they are highly shuffled due to numerous micro-rearrangements [KBH+03]. The concentration of micro-rearrangements within these regions tends to say that they are more prone to rearrangements. A higher level analysis proposed by Pevzner et al. [PT03a, PT03b] consists in studying genomic rearrangements on signed permutations obtained from synteny blocks. They observed that some regions between two markers are re-used suggesting that these regions correspond to hotspots. The "re-use" issue is also a widely debated topic about hotspots [PPT06, San06, ST05].

Trinh et al. [TMS04] defend the thesis of the random model by analyzing in details the small segments within breakpoints: they claim that the loss of similarity between conserved blocks are due to alignment errors or artifacts.

2.2 Evolutionary distances between two genomes

Once common markers are defined, signed permutations can be constructed, and from this model, we can provide a measure of the evolution between two species. In fact, permutations lead to the computation of a mathematical distance that correspond to the minimal number of rearrangements that transform one genome into another. The distance computation is based on a set of rearrangements. In the relevant literature, the considered rearrangements are not always the same. In this section, we focus on the method based on reversals only and its extension to the multichromosomal case by the addition of translocations, fusions and fissions.

2.2.1 The reversal distance for unichromosomal genomes

In 1995 Hannenhalli and Pevzner [HP95a] defined the exact reversal distance between two signed permutations and provide the first polynomial-time algorithm to parsimoniously transform a signed permutation into another using reversals. Their results presented below have been reformulated by Setubal and Meidanis in [SM97]. The studied genomes are represented by one signed permutation according to the previously described formalism. The rearrangement operations considered are restricted to reversals only. Moreover, genomes are defined on the same set of markers without duplications, insertions and deletions.

Figure 1.9 presents a parsimonious scenario transforming the permutation π into the permutation γ. How can one be sure that the obtained scenario is in fact a parsimonious one? In section 1.3.3, the notion of breakpoint was introduced. Computing a parsimonious scenario and thus the rearrangement distance consists in finding the minimum number of rearrangements which remove all of the breakpoints. Thus, the study of breakpoints provides a lower bound for the reversal distance (see lemma 1). In fact, a reversal ρ can remove at most two breakpoints: $b(\pi,\gamma) - b(\pi.\rho,\gamma) \leq 2$.

Lemma 1 *Let π and γ be two permutations and $b(\pi,\gamma)$ be the breakpoint distance between these two permutations. Then, the reversal distance $d(\pi,\gamma)$ verifies: $\frac{b(\pi,\gamma)}{2} \leq d(\pi,\gamma)$.*

The approximation given by the lemma 1 is not very precise. The aim of many works has been to refine this bound. Hannenhalli and Pevzner [HP95a] propose a theory based on a graph introduced by Bafna and Pevzner in [BP93] which leads to an exact formula for the computation of the reversal distance between two signed and unichromosomal genomes.

Breakpoint graph

To transform a signed permutation π into a signed permutation γ, both defined on the same set of n elements, the *breakpoint graph* $G(\pi,\gamma)$ is built. $G(\pi,\gamma)$ is an edge-colored graph built from unsigned representations of two signed permutations. A signed permutation $\pi = \pi_1 .. \pi_n$ over n elements is transformed into an unsigned representation $u(\pi)$ in the following way. Each positive element $+x$ from π is replaced by two vertices labeled $2x-1$ and $2x$ while each negative element $-x$ is replaced by two vertices labeled $2x$ and $2x-1$ (see figure 2.1). If permutations represent linear genomes, vertices $\pi_0 = 0$ and $\pi_{2n+1} = 2n+1$ are added to take into account adjacencies with the first and the last elements. Thus, the graph has $2n+2$ vertices. Note that if genomes are circular, unsigned permutations are defined over $2n$ elements. Edges of G represent adjacencies either in π (edges $\{\pi_{2i}, \pi_{2i+1}\}$, drawn with solid lines), or in γ (edges $\{\gamma_{2i}, \gamma_{2i+1}\}$, drawn with dashed lines) for $i = 0,..,n$ (see figure 2.2 for an example).

$2\pi_i - 1$
$2\pi_i$
$+\pi_i$
$2\pi_i$
$2\pi_i - 1$
$-\pi_i$

(a) π_i is positive (b) π_i is negative

Figure 2.1: Vertices of G obtained from an element of the permutation π.

A reversal applied to the permutation π can also be applied to the breakpoint graph. The particularity of the breakpoint graph defined from two identical permutations is to have solid

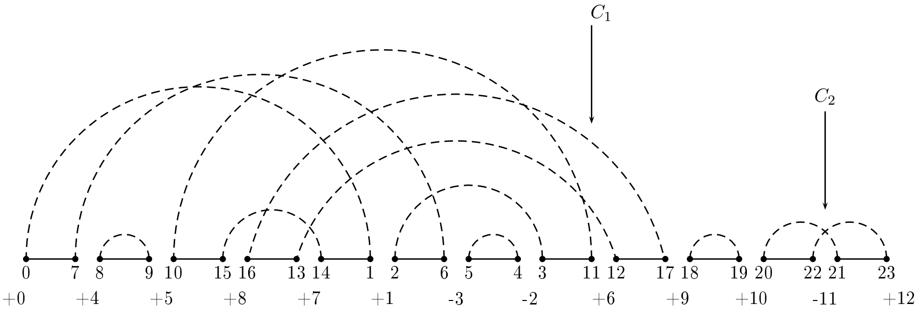


Figure 2.2: Breakpoint graph for the linear permutations $\pi = +4\ +5\ +8\ +7\ +1\ -3\ -2\ +6\ +9\ +10\ -11$ and $\gamma = +1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9\ +10\ +11$.

and dashed edges that link the same vertices. Thus, transforming the permutation π into the permutation γ consists in making the solid and dashed edges coincide (for example this is the case for vertices 5 and 4 in the breakpoint graph of the figure 2.2). The number of cycles in the breakpoint graph defined from two identical permutations with n signed elements is maximal: this number is equal to $n + 1$. Hence, the transformation of π into γ consists in increasing the number of cycles in order to obtain the permutation γ. The number of cycles in the graph $G(\pi, \gamma)$ is denoted by $c(\pi, \gamma)$.

A reversal on the breakpoint graph is defined by two solid edges u and v: elements between u and v are reversed. Only some reversals increase the number of cycles, depending on the considered edges. A traversal (in arbitrary direction) of a cycle provides an orientation for the solid edges. Based on the relative orientation of solid edges, we can define an *orientation* for all pairs of solid edges in a cycle.

Definition 4 *If two solid edges u and v belong to the same cycle of a breakpoint graph and have the same orientation, they are said to be* unoriented. *Otherwise, they are* oriented.

Based on definition 4, we distinguish two kinds of cycles according to the edge orientation: *oriented* and *unoriented* cycles.

Definition 5 *A cycle of a breakpoint graph is* unoriented *if all of its solid edges are pairwise unoriented. Otherwise, the cycle is called to be* oriented.

It is also possible to define an orientation for a dashed edge according to the positions of its incident vertices.

Definition 6 *A dashed edge $\{\pi_i, \pi_j\}$ in $G(\pi, \gamma)$ is* oriented *if $|j - i|$ is even, otherwise it is* unoriented.

The orientation of a cycle can then be redefined based on definition 6.

Definition 7 *A cycle of a breakpoint graph is* unoriented *if all of its dashed edges are unoriented. Otherwise, the cycle is said to be* oriented.

For example in the breakpoint graph of figure 2.2, edges $u = \{20, 22\}$ and $v = \{21, 23\}$ are oriented in the cycle C_2. Thus, the cycle C_2 is also oriented. However, the cycle C_1 is unoriented because it has only two solid edges that are both unoriented ($u = \{16, 13\}$ and $v = \{12, 17\}$).

Theorem 1 (Setubal and Meidanis [SM97]) *Let ρ be a reversal defined on two solid edges u and v of $G(\pi, \gamma)$ with π and γ two signed permutations. Then:*

(i) if u and v belong to two different cycles, $c(\pi.\rho, \gamma) = c(\pi, \gamma) - 1$,

(ii) if u and v belong to the same cycle and are unoriented, $c(\pi.\rho, \gamma) = c(\pi, \gamma)$,

(iii) if u and v belong to the same cycle and are oriented, $c(\pi.\rho, \gamma) = c(\pi, \gamma) + 1$.

The bound provided by breakpoints is refined thanks to theorem 1. In fact, for a given parsimonious scenario $\rho_1\rho_2..\rho_k$ that transforms a signed permutation π of order n into a signed permutation γ, we have:

$$c(\pi.\rho_1.\rho_2...\rho_k, \gamma) = c(\gamma, \gamma) = n + 1$$

According to the theorem, we have:

$$\begin{gathered} c(\pi.\rho_1, \gamma) - c(\pi, \gamma) \leq 1 \\ c(\pi.\rho_1.\rho_2, \gamma) - c(\pi.\rho_1, \gamma) \leq 1 \\ ... \\ c(\pi.\rho_1.\rho_2...\rho_k, \gamma) - c(\pi.\rho_1.\rho_2...\rho_{k-1}, \gamma) \leq 1 \end{gathered}$$

By adding all the terms, we obtain

$$d(\pi, \gamma) \geq c(\pi.\rho_1.\rho_2...\rho_k, \gamma) - c(\pi, \gamma) \text{ and so } d(\pi, \gamma) \geq n + 1 - c(\pi, \gamma).$$

For many permutations, this approximation is very close to the parsimonious distance. Nevertheless, for some cases, this approximation is not exact. If the breakpoint graph of π and γ has only oriented cycles, there exists a scenario such that the number of cycles increases at each step (see theorem 1, item (i)). Thus, the estimate $n + 1 - c(\pi, \gamma)$ is an exact formula in this case. It becomes false when there are one or several unoriented cycle(s), since reversals on this type of cycle do not modify the number of cycles (see theorem 1, item (ii)). Actually, there is a configuration of the breakpoint graph with unoriented cycles for which the formula is correct.

Definition 8 *Two dashed edges $\{\pi_i, \pi_j\}$ and $\{\pi_k, \pi_l\}$ in $G(\pi, \gamma)$* interleave *when $[i, j]$ and $[k, l]$ overlap, but no one of their intervals contains the other.*

Definition 9 *Two cycles C_1 and C_2 in $G(\pi, \gamma)$* interleave *when they have interleaving dashed edges $g_1 \in C_1$ and $g_2 \in C_2$.*

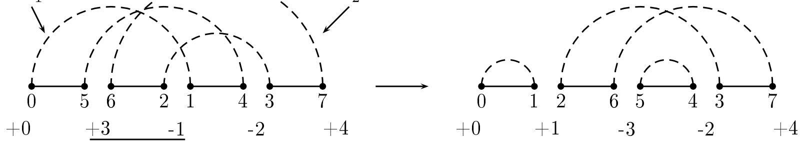


Figure 2.3: Example of a breakpoint graph where the oriented cycle C_1 and the unoriented one C_2 interleave. Applying the reversal defined by $\{0, 5\}$ and $\{1, 4\}$ solid edges within C_1 orients cycle C_2.

If an unoriented cycle interleaves with an oriented one, then applying a reversal to two edges from the oriented cycle increases the number of cycles but orients the unoriented cycle (for an example, see figure 2.3). Thus, the estimation $n + 1 - c(\pi, \gamma)$ for the reversal distance is still exact for this configuration.

Interleaving graph

Unoriented cycles that do not interleave with oriented ones cannot be oriented by the resolution of neighbour cycles. To solve the problem of this kind of unoriented cycles, Hannenhalli and Pevzner introduced the *interleaving graph.*

Definition 10 *An* interleaving graph $I(G)$ *is a graph where each vertex represents a non-trivial cycle (with more than 2 edges) of the breakpoint graph* $G = G(\pi, \gamma)$*. Two vertices are linked by an edge if they are interleaving.*

This graph can be decomposed into connected components.

Definition 11 *The* span *of a connected component* K *of* $I(G)$ *is* $[i, j]$ *where* π_i *and* π_j *are the leftmost and rightmost vertices of any cycle of* K *in* G*.*

Components are classified according to their *orientation.* For example, the breakpoint graph in figure 2.4 has six non-trivial cycles. Cycles C_3 et C_6 are oriented while all the others are unoriented. Figure 2.5 represents the interleaving graph obtained from the breakpoint graph of that in figure 2.4. Three components belong to this graph: two oriented ones and one unoriented formed by the two cycles C_1 et C_5.

Definition 12 *A connected component* K *of the interleaving graph is* oriented *if at least one of its vertices corresponds to an oriented cycle in the breakpoint graph. Otherwise,* K *is* unoriented.

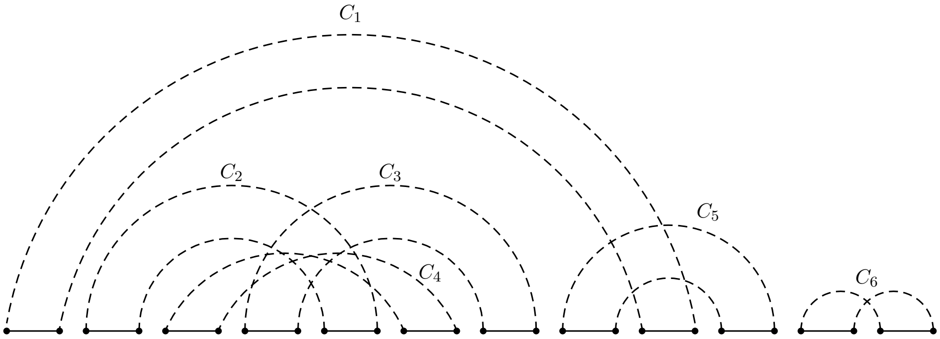


Figure 2.4: Example of a breakpoint graph with six non-trivial cycles C_1 through C_6.

Oriented components are resolved by applying reversals to two oriented edges that increase the number of cycles. Sorting unoriented components is more complex. We have seen that a reversal applied to two solid edges belonging to an unoriented cycle can make it oriented without modifying the number of cycles in the breakpoint graph (theorem 1, item (ii)). In this case, an unoriented component to which the unoriented cycle belongs becomes oriented. Thus, the approximation of the reversal distance can be refined by taking into the account the number of unoriented components. However, not all of the unoriented components require a reversal in order to become oriented.

Hannenhalli and Pevzner [HP95a] give a classification for unoriented components based on the notion of *component separation* defined below.

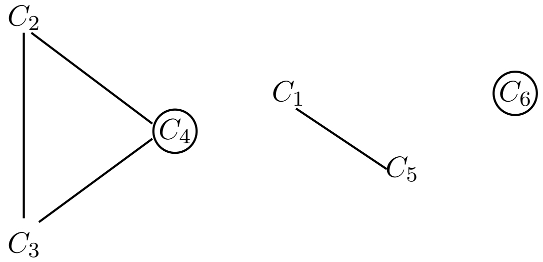


Figure 2.5: Interleaving graph $I(G)$ of the breakpoint graph from figure 2.4. Oriented cycles are encircled. $I(G)$ has 2 oriented components $K_1 = \{C_2, C_3, C_4\}$, $K_2 = \{C_6\}$ and one unoriented $K_3 = \{C_1, C_5\}$.

Definition 13 *Let K_1, K_2 and K_3 be 3 connected components of $I(G)$ and let S_{K_2} and S_{K_3} be the spans of K_2 and K_3.* K_1 separates K_2 *from* K_3 *if there exists a dashed edge $\{\pi_i, \pi_j\}$ in K_1 such that $S_{K_2} \subset [i,j]$ and $S_{K_3} \not\subset [i,j]$.*

Based on this definition, unoriented components are classified into *non hurdles* and *hurdles.* We distinguish *minimal hurdles* from *the greatest hurdle.* In figure 2.6, components K_1 and K_3 are two minimal hurdles separated by the non hurdle K_2.

Definition 14 *A* hurdle *is an unoriented component which does not separate two other unoriented components. Otherwise, it is a* non hurdle.

Definition 15 *A hurdle is* minimal *if its span does not contain the span of any other hurdle.* The greatest hurdle *is a hurdle whose the span contains the spans of all other hurdles.*

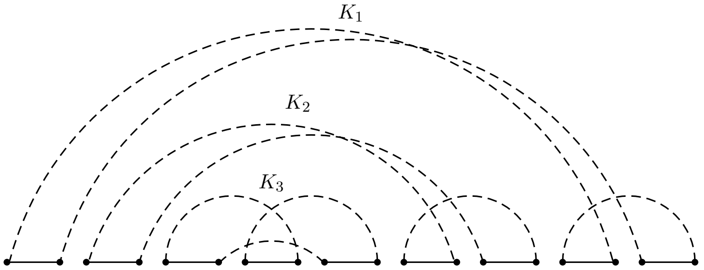


Figure 2.6: Breakpoint graph composed of 3 components K_1, K_2 and K_3. All of them are unoriented and are formed by only one unoriented cycle.

A reversal applied to solid edges belonging to two different cycles decreases the number of cycles (see theorem 1, item (iii)), but if the implied cycles are unoriented they are transformed into an oriented cycle as well as are all unoriented components that separate them. Thus, non hurdles can become oriented by applying reversals to hurdles which they separate. Let $h(\pi, \gamma)$ be the overall number of hurdles in the breakpoint graph of π and γ. The new approximation of reversal distance is then given by the formula:

$$d(\pi, \gamma) \geq n + 1 - c(\pi, \gamma) + h(\pi, \gamma)$$

Nevertheless, “hard-to-sort” permutations exist where the resolution of all the hurdles cannot remove all of the non hurdles. In this case, a supplementary reversal is needed. The configuration of this kind of permutation is called a *fortress* and is based on the notion of *protection*.

Definition 16 *A hurdle K_1* protects *a non hurdle K_2 if removing K_1 transforms K_2 into a hurdle. A* super hurdle *is a hurdle that protects a non hurdle. Otherwise, it is a* simple hurdle*.*

Components K_1 and K_3 are super hurdles belonging to the breakpoint graph of the figure 2.6. If the number of super hurdles is odd and all of them are super hurdles, then it is not possible to remove all of the non hurdles. A supplementary reversal is needed.

Definition 17 *We call a* fortress *a breakpoint graph that has an odd number of hurdles that are all super.*

Let $f(\pi, \gamma)$ be the function that returns 1 if the breakpoint graph is a fortress, and 0 otherwise. Then, the reversal distance is given by theorem 2.

Theorem 2 (Hannenhalli and Pevzner [HP95a]) *For two unichromosomal genomes π and γ, $d(\pi, \gamma) = n + 1 - c(\pi, \gamma) + h(\pi, \gamma) + f(\pi, \gamma)$.*

In [HP95a], Hannenhalli and Pevzner present a construction of the breakpoint graph and the other structures for computing the reversal distance in $\mathcal{O}(n^2)$ for permutations π and γ of order n. Thus, the reversal distance $d(\pi)$ is also computed in $\mathcal{O}(n^2)$. Later, Berman and Hannenhalli in [BH96] improved the algorithm for computing connected components of the interleaving graph and proposed to solve the reversal distance in $\mathcal{O}(n\alpha(n))$, where α is the inverse Ackerman function. In [BMY01], Bader et al. again improved the connected component computation and gave a linear-time algorithm for reversal distance.

2.2.2 Extension to multichromosomal genomes

Hannenhalli and Pevzner [HP95b] extended their theory for reversal distance computation to the multichromosomal case. They propose a polynomial algorithm that computes the minimum number of rearrangements for transforming one multichromosomal genome into another, all of them defined on the same set of markers without repetition. Rearrangements specific to multichromosomal genomes are taken into the account as well as reversals: translocations, fusions and fissions. However, both the formula for rearrangement distance and the algorithm for computing a parsimonious scenario present errors. These were partially corrected by Tesler in [Tes02a]. Ozery-Flato and Shamir in their turn redefine some notions and suggest further corrections for these problems [OFS03]. In what follows, we present using our notations the last results for the rearrangement distance computation based on Hannenhalli and Pevzner's theory and obtained after Tesler, and Ozery-Flato and Shamir's corrections.

Unichromosomal vision for a multichromosomal genome

Hannenhalli and Pevzner propose mimicking the behaviour of a multichromosomal genome through the unichromosomal model. Two steps are needed to transform a multichromosomal genome into an unichromosomal genome: *capping* and *concatenate*. Let Π and Γ be two multichromosomal genomes defined over the same set of N_g gene markers.

A *capping* of Π and Γ consists in adding two ordinals called *caps* to the extremities of each chromosome. Let $C = \{c_0, c_1, .., c_n\}$ with $n = 2\max(N_\Pi, N_\Gamma) - 1$ be the set of distinct caps different from the N_g gene markers in Π and Γ. We denote by $\hat{\Pi} = \{\hat{\pi}^1, ..., \hat{\pi}^{\max(N_\Pi, N_\Gamma)}\}$ a capping of Π where the i^{th} chromosome is $\hat{\pi}^i = c_{2(i-1)}\ \pi^i_1 ... \pi^i_{n_i}\ c_{2(i-1)+1}$. If $N_\Gamma > N_\Pi$, the $N_\Gamma - N_\Pi$ last chromosomes of $\hat{\Pi}$ are empty chromosomes composed of 2 successive caps. From C, we similarly define $\hat{\Gamma}$ with $N_\Pi - N_\Gamma$ empty chromosomes if $N_\Pi > N_\Gamma$. A *concatenate* $\hat{\pi}$ of $\hat{\Pi}$ is a signed permutation $\hat{\pi}$ obtained by concatenating chromosomes after choosing an orientation and an order for each of them. At the end of these two steps, we obtain an unique permutation in which each reversal can be read as a multichromosomal rearrangement. See for illustration example 2.7.

Genomes:	$\Pi = \{1\ 2, 3\ 4, 5\ 8\ 7\ 6\}$	$\Gamma = \{1\ 2\ 3\ 4, 5\ 6\ 7\ 8\}$
Cappings:	$\hat{\Pi} = \{\mathbf{9}\ 1\ 2\ \mathbf{10}, \mathbf{11}\ 3\ 4\ \mathbf{12}, \mathbf{13}\ 5\ 8\ 7\ 6\ \mathbf{14}\}$	$\hat{\Gamma} = \{\mathbf{9}\ 1\ 2\ 3\ 4\ \mathbf{10}, \mathbf{11}\ 5\ 6\ 7\ 8\ \mathbf{12}, \mathbf{13}\ \mathbf{14}\}$
Concatenates:	$\hat{\pi} = \mathbf{9}\ 1\ 2\ \mathbf{10}\ \mathbf{11}\ 3\ 4\ \mathbf{12}\ \mathbf{13}\ 5\ 8\ 7\ 6\ \mathbf{14}$	$\hat{\gamma} = \mathbf{9}\ 1\ 2\ 3\ 4\ \mathbf{10}\ \mathbf{11}\ 5\ 6\ 7\ 8\ \mathbf{12}\ \mathbf{13}\ \mathbf{14}$

Figure 2.7: Example from [Tes02a] of a capping and a concatenate for two genomes Π and Γ. Caps are indicated by bold characters.

Breakpoint graph

The breakpoint graph for multichromosomal genomes is built from permutations $\hat{\pi}$ and $\hat{\gamma}$. The distance value computed on $G(\hat{\pi}, \hat{\gamma})$ depends on the chosen capping and concatenate. Let $G(\Pi, \Gamma)$ be the graph obtained by removing all edges that involve concatenate and capping from $G(\hat{\pi}, \hat{\gamma})$, that is, all dashed edges incident to cap vertices and all solid edges between two cap vertices or between a cap vertex and the first or the last element. Then we can distinguish three types of vertices: (1) isolated vertices called *tails*, (2) cap vertices of degree 1 called Π-*caps*, and (3) other vertices of degree 1 called Γ-*tails*. Figure 2.8 shows the transformation of a graph $G(\hat{\pi}, \hat{\gamma})$ into the graph $G(\Pi, \Gamma)$.

Cycles and paths

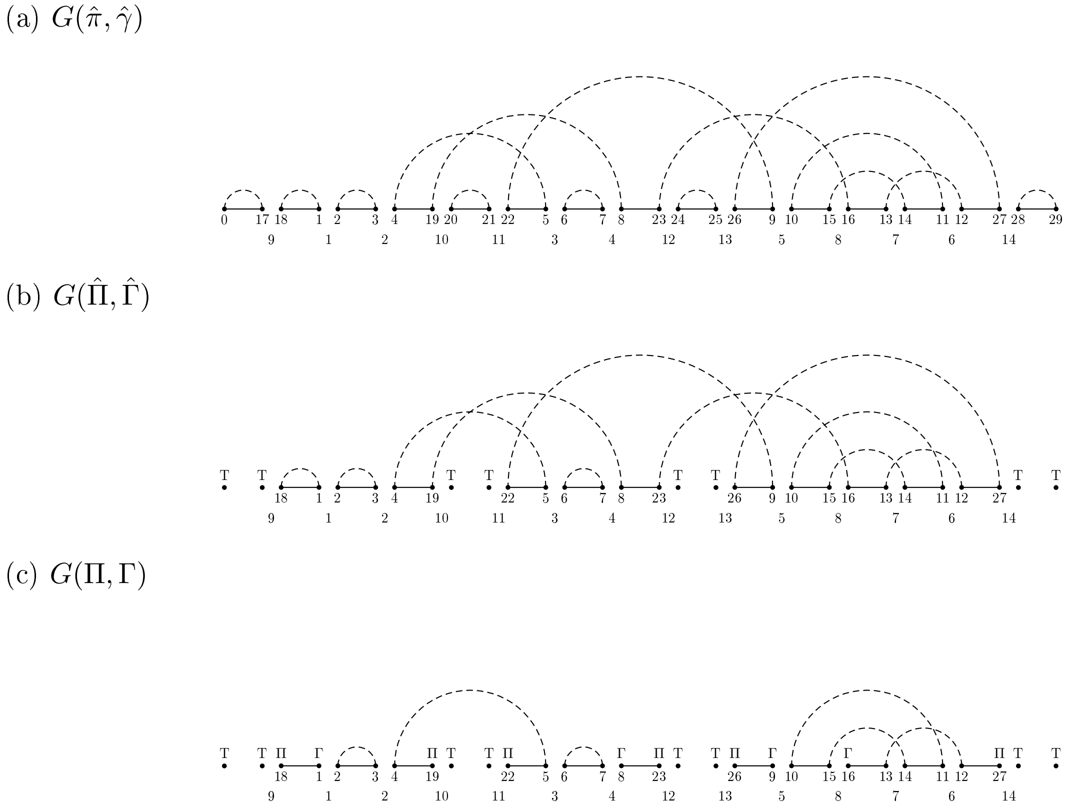
As in the unichromosomal case, the graph $G(\Pi, \Gamma)$ can be decomposed into cycles but also into paths. If a path starts and ends with Π-caps (two Γ-tails, or one Π-cap and one Γ-tail, respectively) then it is a $\Pi\Pi$-*path* ($\Gamma\Gamma$-*path* or $\Pi\Gamma$-*path*, respectively). Orientation for cycles and paths in the multichromosomal case is defined in a way analogous to cycle orientation for the unichromosomal case.

Definition 18 *A cycle or a path of a breakpoint graph is* unoriented *if all its dashed edges are unoriented. Otherwise, the cycle is said to be* oriented.

New notions specific to multichromosomal genomes are also defined for edges and for cycles and paths of breakpoint graph: *interchromosomality* and *intrachromosomality.*

Definition 19 *A dashed edge of a breakpoint graph is* intrachromosomal *if its vertices belong to the same chromosome. It is said* interchromosomal *otherwise.*

Definition 20 *A cycle or path of a breakpoint graph is* interchromosomal *if one of its dashed edges is interchromosomal. Otherwise, it is* intrachromosomal.


(a) $G(\hat{\pi}, \hat{\gamma})$

(b) $G(\hat{\Pi}, \hat{\Gamma})$

T T 18 1 2 3 4 19 T T 22 5 6 7 8 23 T T 26 9 10 15 16 13 14 11 12 27 T T
9 1 2 10 11 3 4 12 13 5 8 7 6 14

(c) $G(\Pi, \Gamma)$

Figure 2.8: Example from [Tes02a] of the transformation of $G(\hat{\pi}, \hat{\gamma})$ into $G(\Pi, \Gamma)$ by removing edges representing the chosen concatenate (from (a) to (b)) and capping (from (b) to (c)). Genomes $\hat{\pi}$ and $\hat{\gamma}$ are the same as those in figure 2.7. Specific vertices are denoted by T (Tails), Π (Π-cap) and Γ (Γ-tails).

Interleaving graph

An *Edge interleaving* defined for unichromosomal genomes (definition 8) is applied to dashed edges of a breakpoint graph representing multichromosomal genomes and extended to cycles and paths.

Definition 21 *Two cycles or paths C_1 and C_2 in $G(\Pi, \Gamma)$* interleave *when they have interleaving edges $g_1 \in C_1$ and $g_2 \in C_2$.*

Then, for multichromosomal genomes, the *interleaving graph* $I(G)$ is a graph where each vertex represents a non-trivial path or cycle of the breakpoint graph $G = G(\Pi, \Gamma)$. Two vertices are linked by an edge if they are interleaving.

In the same way as in definition 12, we define *orientation* for each component of $I(G)$ according to the orientation of its vertices and we distinguish oriented components from unoriented ones.

Moreover, in the same way as for cycles and paths, a component K of $I(G)$ is *interchromosomal* if one of its vertices is interchromosomal, it is *intrachromosomal* otherwise. Let $\mathcal{U}(G)$ be the set of unoriented components of $I(G)$, $\mathcal{IU}(G)$ the set of unoriented and intrachromosomal ones. Within unoriented and intrachromosomal components, we distinguish *real* components from *unreal* components. Denote by $\mathcal{RU}(G)$ the set of real components.

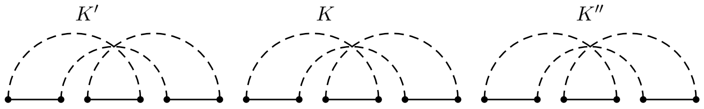


Figure 2.9: Counterexample of the separation notion given by the definition 23. Any element $k \in K$ is such that $K'_{\max} < k < K''_{\min}$. However, K does not separate K' from K'' as it should according to [HP95a].

Definition 22 *A connected component K of $I(G)$ is* real *if K is intrachromosomal, unoriented, and it has no Π-cap or Γ-tail in its span.*

The notion of *component separation* (see definition 13) is defined in the same way as in the unichromosomal case partitions of $\mathcal{U}(G)$, $\mathcal{IU}(G)$ and $\mathcal{RU}(G)$: *hurdles* and *non hurdles* for the first, *knots* and *non knots* for the second, and *real knots* and *non-real knots* for the third.

Note that the definition that Hannenhalli and Pevzner give for the notion of separation in their paper on the multichromosomal case [HP95b] (see definition 23) is different from the definition 13 previously given by the same authors [HP95a] and is incorrect (see counterexample 2.9). A connected component K corresponds to the set of integers $\bar{K} = \{i : i \in C \in K\}$ representing the set of positions of the permutation belonging to cycles or paths of K. For a set of integers K define $K_{\min} = \min_{k \in K} k$ and $K_{\max} = \max_{k \in K} k$.

Definition 23 (Hannenhalli et Pevzner [HP95b]) *A component K* separates *K' from K'' if there exists $k \in K$ such that $K'_{\max} < k < K''_{\min}$.*

A hurdle is *super* if it *protects* (see definition 16) a non hurdle, otherwise it is *simple.* A hurdle can be the *greatest* one if its span contains all the spans of the others hurdles, otherwise it is a *minimal hurdle.* These notions are defined similarly for knots and real knots. The graph G is a *fortress* (*fortress of knots*, or *fortress of real knots*, respectively) if it contains an odd number of hurdles (knots, or real knots, respectively) that are all super.

Within the set of unreal components, Ozery-Flato and Shamir [OFS03] distinguish those called *semi-real knots*, which are characterized by their potential of becoming real knots.

Definition 24 *A* semi-real knot *is a component in $\mathcal{IU}(G) \backslash \mathcal{RU}(G)$ that does not contain a* $\Gamma\Gamma$-*path in its span and that becomes a minimal real knot or the greatest simple real knot after closing its* $\Pi\Gamma$-*paths.*

The *greatest semi-real knot* is a semi-real knot that becomes the greatest simple real knot after closing its $\Pi\Gamma$-paths. A semi-real knot is called a *minimal semi-real knot* if closing its $\Pi\Gamma$-paths makes it a minimal real knot. From the semi-real knot, Ozery-Flato and Shamir [OFS03] define the notions of *simple component* and *weak fortress of real knots.*

Definition 25 *A* simple component *is a component of $I(G)$ with at least one* $\Pi\Gamma$-*path that is not a semi-real knot.*

Definition 26 *A graph G is a* weak fortress of real knots *if (a) G has an odd number of real knots, (b) there exists the greatest real knot in G, (c) all real knots are super except the greatest one and (d) the number of semi-real knots in G is not zero.*

Note that a weak fortress of real knots becomes a fortress of real knots by closing the $\Pi\Gamma$-paths in a semi-real knot. Example 1 gives the details of the components for the breakpoint graph of figure 2.10.

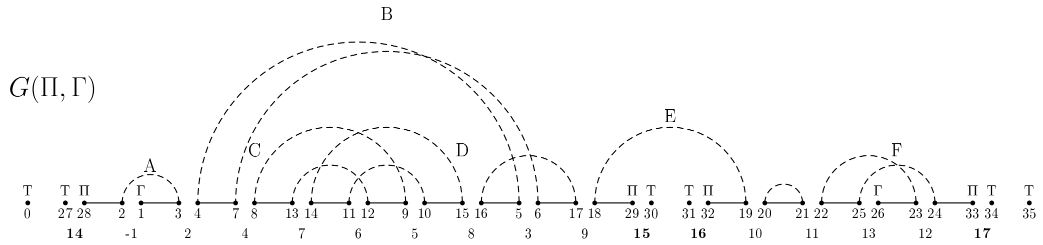


Figure 2.10: Breakpoint graph $G(\Pi, \Gamma)$ for $\Pi = \{-1\ 2\ 4\ 7\ 6\ 5\ 8\ 3\ 9, 10\ 11\ 13\ 12\}$ and $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\}$. Tails vertices are marked by T, Π-caps by Π and Γ-tails by Γ. Non-trivial cycles and paths are denoted by letters from A to F. The interleaving graph $I(G)$ corresponding to $G(\Pi, \Gamma)$ is composed of 5 connected components: $K_1 = \{A\}$, $K_2 = \{B\}$, $K_3 = \{C, D\}$, $K_4 = \{E\}$ and $K_5 = \{F\}$.

Example 1 *Figure 2.10 presents a breakpoint graph $G(\Pi, \Gamma)$. The component K_1 of $I(G)$ is intrachromosomal oriented, $\mathcal{U} = \{K_2, K_3, K_4,\ K_5\}$, $\mathcal{IU} = \{K_2, K_3, K_5\}$ and $\mathcal{RU} = \{K_2, K_3\}$. K_3 is a super hurdle while K_4 and K_5 are simple hurdles, and K_3 and K_5 are super knots. However, K_2 and K_3 are real knots (K_2 is the greatest one), while K_5 is a minimal semi-real knot and K_1 is a simple component.*

Rearrangement distance

Ozery-Flato and Shamir [OFS03] give an exact formula for distance between two multichromosomal genomes Π and Γ as shown in theorem 3. Denote by $\bar{G}(\Pi, \Gamma)$ the graph obtained by closing all the $\Pi\Gamma$-paths in simple components of $G(\Pi, \Gamma)$.

Theorem 3 (Ozery-Flato [OFS03])
$d(\Pi, \Gamma) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + p_{\Gamma\Gamma}(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s'(\Pi,\Gamma) - gr'(\Pi,\Gamma) + fr'(\Pi,\Gamma)}{2} \rceil$.

The parameters of the formula are the following:

- $b(\Pi, \Gamma)$ is the number of solid edges in $G(\Pi, \Gamma)$ ($b = N_g + \max(N_\Pi, N_\Gamma)$),
- $c(\Pi, \Gamma)$ is the number of cycles and paths,
- $p_{\Gamma\Gamma}(\Pi, \Gamma)$ is the number of $\Gamma\Gamma$-paths,
- $r(\Pi, \Gamma)$ is the number of real knots,
- $s'(\Pi, \Gamma)$ is the number of semi-real knots in $G(\Pi, \Gamma)$,
- $gr'(\Pi, \Gamma)$ is equal to 1 if $\bar{G}$ has the greatest real knot and $s' > 0$, and is 0 otherwise,
- $fr'(\Pi, \Gamma)$ is equal to 1 if either (i) $\bar{G}$ is a fortress of real knots and the greatest semi-real knot does not exist in $\bar{G}$, or (ii) $\bar{G}$ is a weak fortress of real knots.

By adapting the linear-time algorithm of Bader et al. for unichromosomal genomes [BMY01], Tesler in [Tes02a] computes the rearrangement distance in linear time.

2.2.3 Other distances

The distance computation methods previously presented rely on reversals and translocations including fissions and fusions, which are specific cases. Although these rearrangements are considered as the most frequent operations during species evolution, different sets of rearrangements and the corresponding genomic distance and scenarios are also investigated in the literature.

Certain studies looked into translocations only. Kececioglu and Ravi [KR95] were the first ones to propose a 2-approximation for computing distance by translocations. In 1996, Hannenhalli [Han96] presents the first polynomial-time algorithm for the signed translocation distance, subsequently corrected by Ozery-Flato and Shamir in [OFS06]. Recently, Li et al. [LQWZ04] proposed a linear implementation for distance computing and Wang a quadratic algorithm to find an optimal sequence of translocations.

Transforming a permutation by transpositions into another (see section 1.2.2) has also been widely studied. However, the complexity of this problem is still open. Bafna and Pevzner [BP98] gave a 1.5-approximation algorithm to find the minimum number of transpositions to transform one genome into another. Hartman et Shamir [HS03] proposed a simpler 1.5-approximation algorithm for the same time complexity. Walter et al. [WSO$^+$05] improved the time complexity of the initial algorithm by giving a $\mathcal{O}(n^3)$ implementation. To date, the best known algorithm is a 1.375-approximation provided by Elias and Hartman in [EH05].

The complexity of the genomic distance problem is still unknown for certain sets of considered rearrangements. In fact, there are efficient algorithms when only one rearrangement is taken into the account, but combinatory problems become more difficult by the addition of new rearrangement types. However, the theory of Hannenhalli and Pevzner [HP95b] presented in this section leads to a linear algorithm [Tes02a] for computing distance in terms of reversals, translocations, fusions and fissions.

2.3 Parsimonious scenarios

The rearrangement distance estimates the minimum number of rearrangements that separate two genomes, while parsimonious scenarios consist in clearly defining which rearrangements occurred during their evolution. These two problems are strongly related but they are often solved independently. This section proposes an overview of the method based on the Hannenhalli and Pevzner's theory [HP95b] for recovering one rearrangement scenario.

2.3.1 Computing a parsimonious scenario for unichromosomal genomes

There are several algorithms for computing a parsimonious scenario between two unichromosomal and signed genomes by reversals. Many of them are based on the Hannenhalli and Pevzner model of the breakpoint graph (see section 2.2.1). From their theory, Hannenhalli and Pevzner developed the first polynomial algorithm for this problem and proposed an $\mathcal{O}(n^4)$ implementation where n is the permutation order. Other more efficient algorithms were developed thereafter: Berman and Hannenhalli [BH96], Kaplan et al. [KST97] and Bader et al. [BMY01] algorithms require $\mathcal{O}(n^2)$, while the one proposed by Bergeron in [Ber01] and [BS01] requires $\mathcal{O}(n^3)$. More recently, Tannier and Sagot in [TS04] solve this problem with a $\mathcal{O}(n\sqrt{n}\log n)$-time algorithm.

All of the quoted algorithms except the last one are based on *safe reversals*. A reversal is *safe* if it decreases the reversal distance by one. There are two types of safe reversals: *proper* safe reversals and *hurdle-cutting* safe reversals. The latter consist in solving the problem of unoriented components and this is done in the same way by all the algorithms. Algorithms differ in the way proper safe reversals in oriented components are found: although the methods are all based on the interleaving graph or the overlap graph (easily obtained from the interleaving graph), the notion of safe reversal is defined differently.

2.3.2 Computation of an optimal scenario for multichromosomal genomes

In order to make the problem easier, finding a parsimonious scenario between two multichromosomal genomes in terms of reversals and translocations is reduced to the unichromosomal case in a way analogous to the distance problem. For two multichromosomal genomes Π and Γ, computing *optimal cappings* Π^* and Γ^* and then *optimal concatenates* π^* and γ^* are needed to obtain unichromosomal permutations to which existing algorithms for the unichromosomal case can be applied from the breakpoint graph $G(\pi^*, \gamma^*)$. Each reversal in the obtained scenario is interpreted as a rearrangement, either a translocation or a reversal.

As was the case for distance resolution, the initial theory of Hannenhalli and Pevzner for this problem [HP95b] was corrected first by Tesler [Tes02a], and then in turn Ozery-Flato and Shamir [OFS03]. In what follows, we present in detail the last results [HP95b, Tes02a, OFS03] for the two main steps that lead to the construction of $G(\pi^*, \gamma^*)$: optimal cappings and optimal concatenates.

Optimal cappings

Optimal cappings Π^* and Γ^* formalize the problem of finding positions and signs for caps in the genome Γ such that $d(\Pi^*, \Gamma^*) = d(\Pi, \Gamma)$ (see lemma 4). This is done for any arbitrary capping in Π. In the breakpoint graph, it consists in adding $2N_\Gamma$ edges linking a Π-cap to a Γ-tail and $N_\Pi - N_\Gamma$ edges between two Π-caps if $N_\Pi > N_\Gamma$. Hannenhalli and Pevzner prove in [HP95b] a set of technical lemmas required to build optimal cappings.

Lemma 2 ([HP95b]) *For every* $\Pi\Pi$*-path and* $\Gamma\Gamma$*-path in* $G(\Pi, \Gamma)$*, there exists either an interchromosomal or an oriented dashed edge which joins these paths into a* $\Pi\Gamma$*-path.*

Lemma 3 ([HP95b]) *For every two unoriented* $\Pi\Gamma$*-paths, there exists either an interchromosomal or an oriented dashed edge which joins these paths into a* $\Pi\Gamma$*-path.*

Let Γ' be the set of the $2\max(N_\Pi, N_\Gamma)!$ possible cappings for Γ.

Lemma 4 ([HP95b]) $d(\Pi, \Gamma) = \min_{\hat{\Gamma} \in \Gamma'} b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + h(\hat{\Pi}, \hat{\Gamma}) + f(\hat{\Pi}, \hat{\Gamma})$.

Optimal cappings Π^* and Γ^* verify: $d(\Pi, \Gamma) = b(\Pi^*, \Gamma^*) - c(\Pi^*, \Gamma^*) + h(\Pi^*, \Gamma^*) + f(\Pi^*, \Gamma^*)$. Ozery-Flato and Shamir give in [OFS03] an algorithm for construction of the sequence of dashed edges leading to optimal capping Γ^* (see algorithm 1).

Despite corrections for optimal capping problem brought by Ozery-Flato and Shamir in [OFS03], the algorithm they propose remains incorrect. In chapter 7, we show a counterexample for Ozery-Flato and Shamir's algorithm and we introduce a correct algorithm for optimal capping as well as the proof of its correction.

Optimal concatenates

Hannenhalli and Pevzner in [HP95b] indicate that it is sometimes necessary to flip (i.e. reverse) some chromosomes in order to obtain optimal final permutations. Tesler in [Tes02a] specifies that at most one reversal of one or several entire chromosome(s) is required during the computation

Algorithm 1 Optimal_Capping

```
Construct the graph $G = G(\Pi, \Gamma)$
while there is a $\Pi\Pi$-path in $G$ do
    Find an interchromosomal or an oriented edge joining this $\Pi\Pi$-path with a $\Gamma\Gamma$-path (lemma 2) and add it to $G$
end while
while $G$ has more than two semi real-knots do
    Find an interchromosomal or an oriented edge joining $\Pi\Gamma$-paths in any two semi real-knots (lemma 3) and add it to $G$
end while
Close all $\Pi\Gamma$-paths in simple components in $G$
if $G$ has two semi real-knots but it is not a fortress of real-knots then
    Find an interchromosomal or an oriented edge joining $\Pi\Gamma$-paths in these semi real-knots (lemma 3) and add it to $G$
end if
Close any remaining $\Pi\Gamma$-paths in $G$
Find a capping $\hat{\Gamma}$ defined by the graph $G(\hat{\Pi}, \hat{\Gamma})$
```

of an optimal scenario based on optimal permutations. However, Tesler shows that reversing some chromosomes is not always sufficient to obtain optimal permutations. Some chromosomes need to be reordered as well to avoid non-biological operations which just exchange two caps. Then, optimal permutations verify the following theorem:

Theorem 4 (Tesler [Tes02a]) *Let $d(\Pi, \Gamma)$ denote the distance between two multichromosomal genomes, Π and Γ. There is a constructive polynomial-time algorithm to produce two permutations π^* and γ^* whose reversal distance is $d_{rev}(\pi^*, \gamma^*) = d$ or $d+1$ such that optimal reversal scenarios between these permutations directly mimic optimal rearrangement scenarios between genomes Π and Γ. When $d_{rev} = d+1$, one reversal step mimics flipping a block of consecutive whole chromosomes, which does not count as an operation in a multichromosomal rearrangement scenario; there are examples when such a step is required.*

Tesler determines optimal concatenates π^* and γ^* based on two steps: *proper flipping* and *proper bonding* of chromosomes [Tes02a].

Proper flipping Chromosome orientation can modify the nature of the interchromosomal components of the corresponding breakpoint graph. An optimal orientation induces a breakpoint graph without unoriented interchromosomal components: in this case, the breakpoint graph is said to be *properly flipped*. For that, each chromosome has to be properly flipped as well.

Definition 27 ([HP95b]) *A chromosome π^i of a genome Π is* properly flipped *in $G = G(\hat{\pi}, \hat{\gamma})$ if every interchromosomal edge originating from it belongs to an oriented component of G.*

Definition 28 ([HP95b]) *The graph $G(\hat{\pi}, \hat{\gamma})$ is* properly flipped *if all chromosomes are properly flipped.*

Definitions 27 and 28 applied to graphs $G = G(\hat{\pi}, \hat{\gamma})$ are extended to graphs $G(\hat{\Pi}, \hat{\Gamma})$ by Tesler in [Tes02a] despite the absence of edges incident to tail vertices.

Tesler also extends lemma 5 to graphs $G(\hat{\Pi}, \hat{\Gamma})$ and presents algorithm *Proper_Flip_Left* (algorithm 2) which leads to a properly flipped graph. Example 2.11 presents an application of the algorithm 2.

Lemma 5 ([HP95b]) *If a chromosome π^i is not properly flipped in $G = G(\hat{\pi}, \hat{\gamma})$, then it is properly flipped in the graph G' obtained by flipping that chromosome. Moreover, every properly flipped chromosome in G remains properly flipped in G'.*

Algorithm 2 Proper_Flip_Left(G)

```
1: Determine components of $G$
2: Classify components of $G$
3: Determine all distinct chromosomes $i_1, i_2, .., i_k$ that contain the leftmost vertex of one or
   more interchromosomal unoriented components
4: Flip chromosomes $i_1, i_2, .., i_k$
```

Proper bonding *Proper bonding* consists in reordering chromosomes in $\hat{\Pi}$ and $\hat{\Gamma}$ in order that the pairs of caps that separate two chromosomes are the same within both genomes, which consequently avoids non-biological operations that simply exchange two caps during the construction of a parsimonious scenario.

Definition 29 ([Tes02a]) *A* bond *is a couple of caps (c_1, c_2) such that c_1 is the right signed cap of the chromosome i and c_2 is the left signed cap of the chromosome $i + 1$.*

The set of the bonds of a concatenate $\hat{\pi}$ is then the following

$$\{(0, \pi_0^1), (\pi_{n_1+1}^1, \pi_0^2), .., (\pi_{n_{N_C-1}+1}^{N_C-1}, \pi_0^{N_C}), (\pi_{n_{N_C}+1}^{N_C}, n+1)\}.$$

Bonds $(0, \pi_0^1)$ and $(\pi_{n_{N_C}+1}^{N_C}, n+1)$ are called *external bonds* while the others are called *internal bonds.*

Definition 30 ([Tes02a]) *A bond (a, b) of the permutation $\hat{\gamma}$ is a* proper bond *when either (a, b) or $(-b, -a)$ is a bond in $\hat{\pi}$.*

As it is shown by Tesler in [Tes02a], optimal concatenates π^* et γ^* can be obtained from optimal cappings so that following conditions are verified:

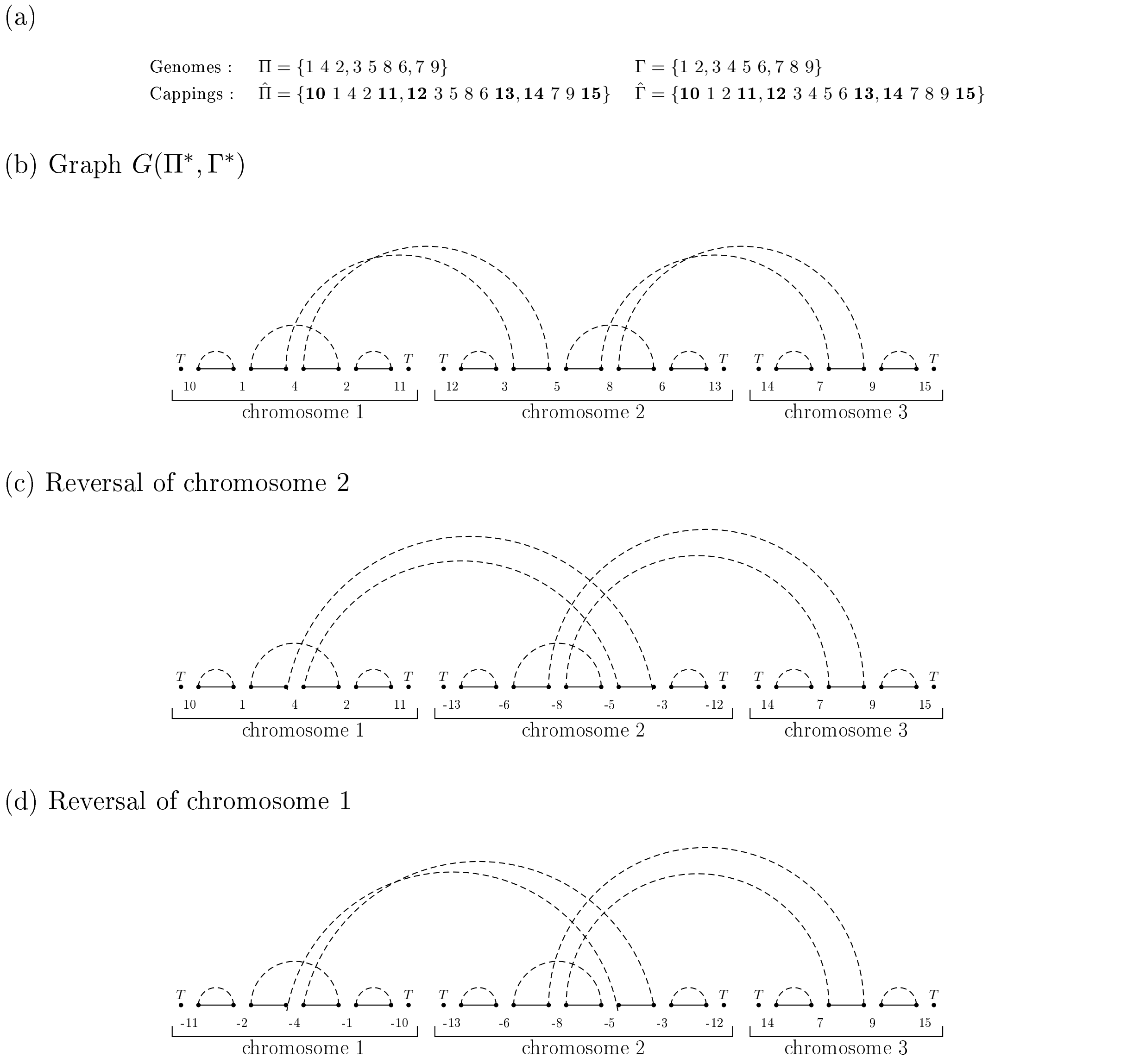
1) $G(\pi^*, \gamma^*)$ is properly flipped, and

2) Either

   (i) all internal and external bonds in γ^* are proper relative to π^*; or

   (ii) there is one improper internal bond and one improper external bond.

Methods developed by Tesler in [Tes02a] for building optimal concatenates π^* and γ^* from optimal cappings Π^* and Γ^* consist in concatenating at each step two chromosomes A and B of Π^* to create a novel bond between these two chromosomes. The concatenate $A + B$ is thus obtained by creating a bond (a, b) with a the right cap of A and b the left cap of B. We look



(a)

Genomes : $\Pi = \{1\ 4\ 2, 3\ 5\ 8\ 6, 7\ 9\}$ $\Gamma = \{1\ 2, 3\ 4\ 5\ 6, 7\ 8\ 9\}$

Cappings : $\hat{\Pi} = \{\mathbf{10}\ 1\ 4\ 2\ \mathbf{11}, \mathbf{12}\ 3\ 5\ 8\ 6\ \mathbf{13}, \mathbf{14}\ 7\ 9\ \mathbf{15}\}$ $\hat{\Gamma} = \{\mathbf{10}\ 1\ 2\ \mathbf{11}, \mathbf{12}\ 3\ 4\ 5\ 6\ \mathbf{13}, \mathbf{14}\ 7\ 8\ 9\ \mathbf{15}\}$

(b) Graph $G(\Pi^*, \Gamma^*)$

(c) Reversal of chromosome 2

(d) Reversal of chromosome 1

Figure 2.11: Application of the algorithm Proper_Flip_Left to genomes Π and Γ. (a) Entry data. (b) Graph $G(\Pi^*, \Gamma^*)$ obtained from optimal cappings $\Pi^* = \hat{\Pi}$ and $\Gamma^* = \hat{\Gamma}$. There are two interchromosomal and unoriented components: chromosomes 1 and 2 are those to flip. (c) Proper flipping of chromosome 2. (d) Proper flipping of chromosome 1. Obtained graph is properly flipped.

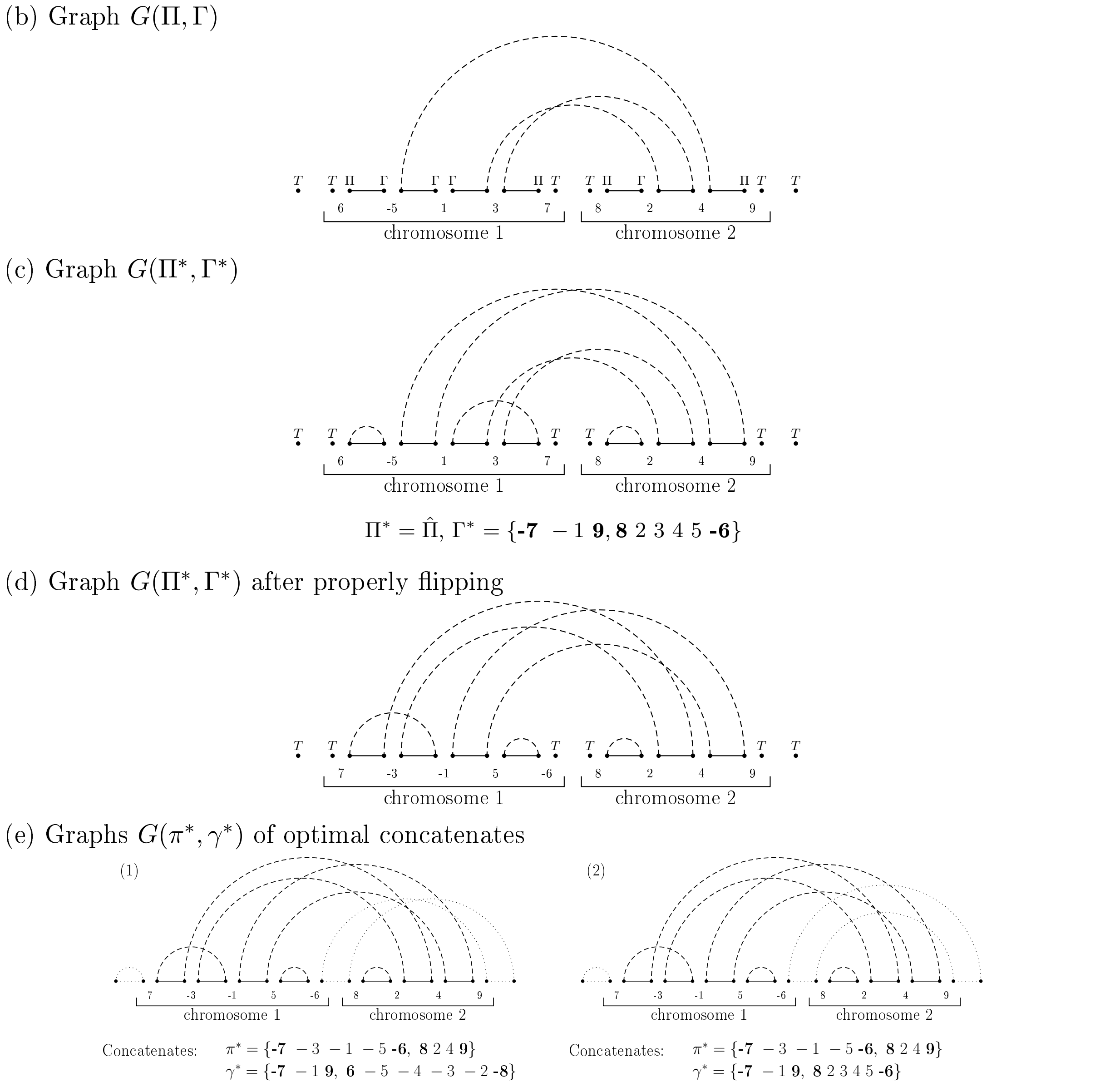
for the same bond in Γ^* with $A' + B'$ obtained by concatenating two of these chromosomes A' and B'. If a and b are located on two different chromosomes in Γ^*, Π^* and Γ^* can have the same bond: the concatenate $A + B$ is said to be *legal* in this case. On the contrary, if a and b are on the same chromosome of Γ^*, creating the bond (a, b) in Γ^* is impossible: $A + B$ is said to be *illegal*. Of course, flipping chromosomes is allowed for creating proper bonds as long as chromosomes are properly flipped (condition (1) of optimal concatenates).

Tesler proposes the algorithm *form_optimal_concatenate* (algorithm 3) that builds optimal concatenates. Steps (1), (2) and (17)-(21) are computed in $\mathcal{O}(n)$. In the worst case, steps (5)-(12) have to be done $(N_C - 1)$ times, which induces a complexity in $\mathcal{O}((N_C - 1)n)$. However, at

(a)

Genomes: $\Pi = \{-5\ 1\ 3, 2\ 4\}$ $\Gamma = \{1, 2\ 3\ 4\ 5\}$

Cappings: $\hat{\Pi} = \{\mathbf{6}\ -5\ 1\ 3\ \mathbf{7}, \mathbf{8}\ 2\ 4\ \mathbf{9}\}$ $\hat{\Gamma} = \{\mathbf{6}\ 1\ \mathbf{7}, \mathbf{8}\ 2\ 3\ 4\ 5\ \mathbf{9}\}$


(b) Graph $G(\Pi, \Gamma)$

(c) Graph $G(\Pi^*, \Gamma^*)$

$\Pi^* = \hat{\Pi}$, $\Gamma^* = \{\textbf{-7}\ -1\ \mathbf{9}, \mathbf{8}\ 2\ 3\ 4\ 5\ \textbf{-6}\}$

(d) Graph $G(\Pi^*, \Gamma^*)$ after properly flipping

(e) Graphs $G(\pi^*, \gamma^*)$ of optimal concatenates

Concatenates: $\pi^* = \{\textbf{-7}\ -3\ -1\ -5\ \textbf{-6}, \mathbf{8}\ 2\ 4\ \mathbf{9}\}$
$\gamma^* = \{\textbf{-7}\ -1\ \mathbf{9}, \mathbf{6}\ -5\ -4\ -3\ -2\ \textbf{-8}\}$

Concatenates: $\pi^* = \{\textbf{-7}\ -3\ -1\ -5\ \textbf{-6}, \mathbf{8}\ 2\ 4\ \mathbf{9}\}$
$\gamma^* = \{\textbf{-7}\ -1\ \mathbf{9}, \mathbf{8}\ 2\ 3\ 4\ 5\ \textbf{-6}\}$

Figure 2.12: Example from [Tes02a] of the construction of optimal concatenates. (a) Entry data. (b) Graph $G(\Pi, \Gamma)$ on which rearrangement distance is computed: $d = 7-4+0+0+\lceil\frac{0+0+0}{2}\rceil = 3$. (c) Graph $G(\Pi^*, \Gamma^*)$ of optimal cappings. (d) Properly flipping of the graph $G(\Pi^*, \Gamma^*)$ by reversing chromosome 1. (e) Graphs $G(\pi^*, \gamma^*)$ of optimal concatenates. The bond $(-6, 8)$ is illegal and reversing chromosome 1 is not possible. Optimal concatenate γ^* is building from two chromosomes of Γ^*. Two concatenates for γ^* are possible: (1) There exists an oriented cycle (dotted lines) between 4 Tail vertices. (2) There exists an unoriented cycle (dotted lines) between 4 Tail vertices but which overlap an oriented component.

each iteration, only one cap among the $2(i-1)$ caps of $\hat{\pi}^1$,..,$\hat{\pi}^{i-1}$ can form an illegal bond with the cap of $\hat{\pi}^i$. So, the probability of doing steps (6) to (11) is $\frac{1}{2(i-1)}$. And hence the average complexity is $\mathcal{O}((\frac{1}{2}+\frac{1}{4}+..+\frac{1}{2(N_C-1)})n) = \mathcal{O}(n\ln(N_C))$.

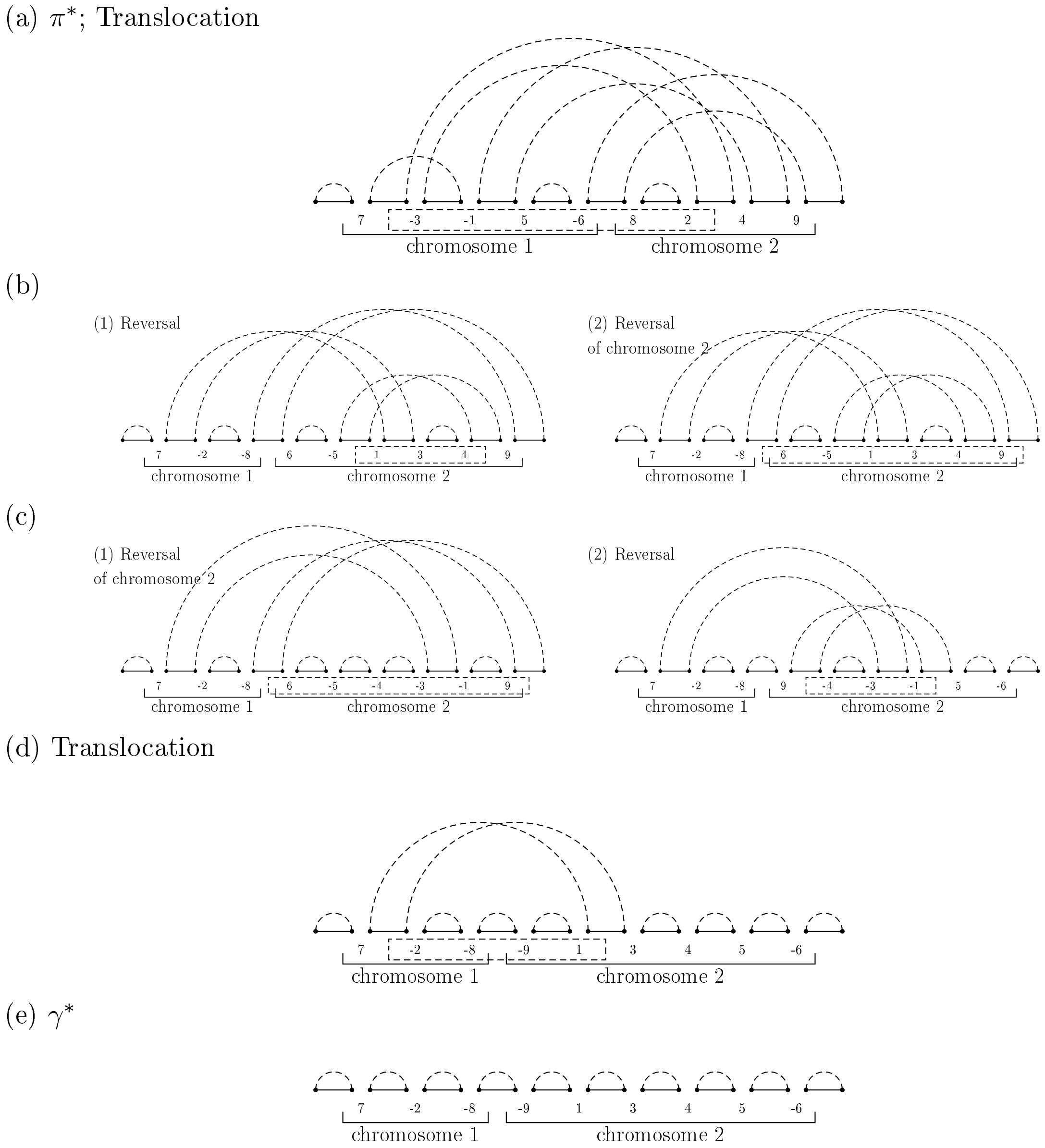
Algorithm 3 form_optimal_concatenate(G, $\hat{\pi}$, $\hat{\gamma}$)

```
 1: Initialize the list of pairs of caps on the chromosomes of $\Gamma$
 2: $G = proper\_flip\_left(G)$
 3: $i = N_C$
 4: while $i \geq 2$ do
 5:    if the bond from $\hat{\pi}^{i-1}$ to $\hat{\pi}^i + .. + \hat{\pi}^{N_C}$ is illegal then
 6:       if $i > 2$ then
 7:          $\hat{\pi}^{i-2}, \hat{\pi}^{i-1} = -\hat{\pi}^{i-1}, -\hat{\pi}^{i-2}$
 8:       else
 9:          $\hat{\pi}^{i-1} = -\hat{\pi}^{i-1}$
10:       end if
11:       $G = Proper\_Flip\_Left(G)$
12:    end if
13:    Form the bond $\hat{\pi}^{i-1} + (\hat{\pi}^i + .. + \hat{\pi}^{N_C})$.
14:    Update the list of bonds and block caps of $\Gamma^*$ (if step 9 occurred this iteration, and this is
       not possible, skip it).
15:    $i = i - 1$
16: end while
17: $\hat{\pi} = \hat{\pi}^1 + .. + \hat{\pi}^{N_C}$
18: if There are no improper bonds then
19:    Form the concatenate $\gamma^*$ starting with the same cap as $\pi^*$ and with the same internal
       bonds.
20: else
21:    Concatenate the two blocks of $\Gamma^*$ together so that $\gamma^*$ and $\pi^*$ start with the same cap.
22: end if
```

Optimal scenario

Unichromosomal methods for building parsimonious scenarios are easily adapted to the multichromosomal case by using optimal concatenates π^* and γ^* as permutations. For methods that need the breakpoint graph, the graph $G(\pi^*, \gamma^*)$ obtained after optimal concatenates can be directly used. In this case, each reversal is interpreted as a multichromosomal rearrangement (reversal, translocation, fusion or fission). However, reversals delimited by caps are strongly constrained. In fact, only reversals starting at a left cap and ending at a right cap are allowed because they correspond to a reversal of a whole chromosome. All of the algorithms previously presented in section 2.3.1 respect this constraint because the reversals to apply are determined by dashed edges and their orientation in the breakpoint graph. Yet, during optimal capping and optimal concatenate constructions, cycles including caps are either trivial (and do not require a reversal) or interchromosomal and oriented. In the latter case, the edges chosen for reversal connect two caps or two non-cap elements. For an example of a multichromosomal scenario, see figure 2.13.

As in the case of the rearrangement distance, optimal cappings can be found by a linear-time algorithm that relies on identification of connected components. Tesler [Tes02a] provides a quadratic-time algorithm to compute optimal concatenates. Then, the time to compute a rearrangement scenario is $\mathcal{O}(n^2)$ using the Bader et al. quadratic-time algorithm [BMY01] for parsimonious scenario by reversals.


(a) π^*; Translocation

(b)

(c)

(d) Translocation

(e) γ^*

Figure 2.13: Two parsimonious scenarios from optimal concatenates obtained in the figure 2.12 (second solution). Each rearrangement is delineated by a rectangle on the permutation. (a) Only one edge is oriented: it determines the translocation to apply. (b) and (c) The reversal of elements 1, 3 and 4 and the reversal of the chromosome 2 are independent: the application order is arbitrary. (d) and (e) A last translocation leads to γ^*.

2.3.3 Why is giving only one optimal scenario misleading?

Sections 2.2 and 2.3 introduced the two linked problems of finding a genomic distance between two genomes and a sequence of rearrangements that realizes this distance. If the first task is considered as a good approximation for the real evolutionary distance, the second one may provide clues about evolutionary mechanisms that occurred during history of the two species.

In the case of the reversal distance (translocation and reversal distance by extension), algorithms previously presented in 2.3.1 compute one parsimonious scenario. Nevertheless, a study led by Siepel in [Sie02] - where he proposes an algorithm to find all safe reversals - shows that there exists a huge number of parsimonious scenarios. For example, for two permutations of order $n = 100$ and reversal distance $d = 0.5n$, hundreds of safe reversals are possible. Bergeron et al. in [BCHSO02] proposed the following theorem to evaluate the number of parsimonious scenarios

Theorem 5 (Bergeron et al. [BCHSO02]) *If π is a random permutation on n elements, and if ρ a random oriented reversal of π, then the probability that ρ is unsafe is $\mathcal{O}(\frac{1}{n^2})$.*

The parsimony principle is thus not enough to provide a sequence of rearrangements that make possible an evolutionary study that is also biologically realistic. In order to reduce the number of parsimonious scenarios in a useful way, one should take into consideration additional biological constraints. Several approaches have been developed to constrain the sorting of permutations. One of these approaches consists in taking into the account the length of reversed segments: Lefebvre et Al. [LEMTS03] proceed according to the principle that small reversals prevail, as a large number of those can be observed in comparing genomes of related species [CNN$^+$00]. Other publications determine parsimonious scenarios that conserve common structures between the two studied genomes all along the sequence (see [Fig04] and [BBCP07]).

2.4 Global methods for ancestral reconstruction

The large scale study of molecular evolution through the comparison of contemporary genomes is frustrated by the impossibility of knowing with certainty the architecture of the common ancestral genomes. Constructing plausible hypothesis about the structural characteristics of these ancestral architectures is a computational task whose results may provide deep insight both into the past histories of particular genomes and the general mechanisms of their formation. This task has two important difficulties: how can we guarantee that the solution is biologically plausible? how can we find these solutions in an efficient manner?

Ancestral reconstruction methods require three basic steps: identification of common markers in the contemporary genomes (see section 2.1), construction of comparative maps of the genomes (using the permutation model, see section 1.2), and reconciliation of these maps using a criterion of maximum parsimony to reconstruct ancestral maps. Computational reconciliation is most often formulated as the *multiple genome rearrangement problem* [SSK96, HCKP95]: given a set of N contemporary genomes and a distance d, find a tree T with the N genomes as leaf nodes and assign permutations (plausible ancestral architectures) to internal nodes such that $D(T) = \sum_{(\pi,\gamma)\in T} d(\pi,\gamma)$ is minimized. When $N = 3$ this is called the *median genome problem.* Sankoff and Blanchette [SB97] developed a method based on the breakpoint distance for unichromosomal genomes, while Caprara used the reversal distance [Cap99, Cap03] to find an ancestral genome for 3 permutations. As for Bourque and Pevzner, they provide algorithms to recover ancestral multichromosomal genomes based on rearrangement distance [BP02]. In both cases the median genome problem was proved to be NP-hard (see [Bry98, PS98] for the breakpoint distance and [Cap99, Cap03] for the reversal distance).

All of these methods provide a global solution to the median genome problem, which is the basic problem in the reconstruction of evolutionary trees. In what follows, we will present the breakpoint-based and rearrangement-based methods respectively proposed by Sankoff and Blanchette [SB97, SB98], and Bourque and Pevzner [BP02]. Finally, we will show that for whichever distance on which the resolution of the median genome is based, the lack of biological constraints in *in silico* methods leads to non representative medians and thus to problematic reconstructed trees.

2.4.1 Breakpoint-based method

Sankoff and Blanchette [SB97] propose to resolve the genome median problem based on breakpoint analysis by reducing it to the *Travelling Salesman Problem* (TSP) (introduced in [BLW76]). They give an algorithm for three unsigned unichromosomal genomes which is easily extensible to the ancestral reconstruction for signed genomes and for more than three genomes. Finally, based on the resolution of the genome median problem, several strategies are considered to reconstruct the phylogenetic tree [BBS97, SB98]. Algorithms presented below are integrated in the software BpAnalysis and reimplemented in GRAPPA [BMW$^+$, MWB$^+$01] which propose faster running times [MTWW02].

Median genome problem

In what follows, we present the initial algorithm given in [SB97] for the median problem in the case of unichromosomal and unsigned genomes defined on the same set of markers $\mathcal{G}$, then we present its extension to unichromosomal and signed genomes.

Reduction to TSP for unsigned genomes To reduce the median genome problem to TSP, genomes and their adjacencies are interpreted in terms of the graph theory. Genomes are represented by a complete weighted graph G. Vertices of G are elements of $\mathcal{G}$. An edge $\{g, h\}$ linking two vertices g and h represents the adjacency between the elements of $\mathcal{G}$ corresponding to g and h. Let $u(gh)$ be the frequency of this adjacency in the 3 genomes, that is, the number of genomes in which it appears (from 0 to 3). TSP consists in determining an Hamiltonian path of minimal cost, the weight of an edge $\{g, h\}$ being defined by $w(gh) = 3 - u(gh)$. Thus, applying TSP to (G, w) leads to an optimal genome A that minimizes the breakpoint number between A and the considered genomes. Sankoff and Blanchette use a branch-and-bound algorithm for which they define a lower bound.

Denote by $P \subseteq E(G)$ the set of available edges. This set is disjoint from the *fragment* $F \subseteq E(G)$, that corresponds to the selected edges at a given instant in the construction of A. Let $\mathit{score} = \sum_{\{g,h\} \in F} w(gh)$. Clearly, it is not necessary to go through branches of the search tree that have a possible minimum score greater than the best score that has already been computed.

Definition 31 *The* availability *of a vertex $g \in V(G)$, denoted by $a(g)$, is equal to 2, 1 or 0 depending on whether g is incident to 0, 1 or more than one edge in F, respectively.*

Let $\mu(g)$ be the sum of the $a(g)$ smallest weight(s) of edges in P incident to g. A path A of weight W_A providing a solution to TSP, is constructed from the set of edges in F with some edges from P. Let $\nu(g)$ be the sum of weights of the $a(g)$ edges from A in P incident to g. Clearly, $\mu(g) \leq \nu(g)$. Then,

$$W_A = \mathit{score} + \sum_{\{g,h\} \in E(A) \cap P} w(gh),$$

$$W_A = score + \frac{1}{2} \sum_{g|\{g,h\} \in E(A) \cap P} w(gh).$$

The weight of an edge in $E(A) \cap P$ is doubly counted:

$$W_A = score + \frac{1}{2} \sum_{g|\{g,h\} \in E(A) \cap P} \nu(g).$$

Since $\mu(g) \leq \nu(g)$, the lower bound is defined by:

$$L(P) = \frac{1}{2} \sum_{g|\{g,h\} \in E(A) \cap P} \mu(g).$$

$L(P)$ is used as a lower bound in the branch-and-bound algorithm *BBF* (algorithm 4) used by algorithm 5 to compute a median genome. The search is recursive. The algorithm is greedy until it finds the first solution whose the score represents an upper bound for the rest. If its cost $U = L(E(G))$, then this solution is optimal. Other bounds exist but Sankoff and Blanchette chose this one because it is easily adaptable to ancestral search for more than 3 genomes.

Algorithm 4 BBF$(P, F, A, score, best)$

```
if $|F| = |V(G)|$ and $score < best$ then
  Conserve $A = F$ as best current solution
  $best \leftarrow score$
end if
if $|F| < |V(G)|$ then
  if $L(P) + score < best$ then
    choose $\{g, h\} \in P$ to add to $F$
    where $a(g) > 0$, $a(h) > 0$ and $w(gh)$ as small as possible,
    and $F \cup \{\{g, h\}\}$ is not a cycle of less than $|V(G)|$ vertices.
    BBF($P - \{\{g, h\}\}$, $F \cup \{\{g, h\}\}$, $A$, $score + w(gh)$, $best$)
    BBF($P - \{\{g, h\}\}$, $F$, $A$, $score$, $best$)
  end if
end if
```

Algorithm 5 genome median computation

```
Require: A completed and weighted graph $(G, w)$
Ensure: A solution $A$ to TSP for $(G, w)$
  $V(A) \leftarrow V(G)$
  $F \leftarrow \emptyset$
  $P \leftarrow E(G)$
  $score \leftarrow 0$
  $best \leftarrow \infty$
  BBF($P$,$F$,$A$,$score$,$best$)
```

Adaptation to the signed case When marker signs are known, they participate in the determination of breakpoints (see section 1.3.3): for an adjacency $g.h$ between two elements g

and h in a signed genome, there is no breakpoint if either $g.h$ or $-h.-g$ appears in the other genome. In addition to the determination of the order of elements, reduction to TSP has also to find the sign of each element. To do so, the graph model of genomes has to be slightly modified.

Two vertices of G are associated with each element g: g and $-g$. Thus, the set of vertices of G is $V = \{g_1, g_2, .., g_n, -g_1, -g_2, .., -g_n\}$ for a set of $\mathcal{G} = \{g_1, g_2, .., g_n\}$ markers. The signed element g is then represented by the edge $\{g, -g\}$. Consequently, for each edge $\{g, h\}$ in $E(G)$, denote by $u(gh)$ the number of genomes where $-g$ and h are adjacent. Weights of edges are then computed in the following way: $w(gh) = 3 - u(gh)$ if $g \neq -h$; if $g = -h$, this edge is used to link two vertices representing an unique element and has to be cover by the solution path. A value $-M$ has to be attributed to $w(gh)$ such that M is sufficiently high in order to force the presence of this edge in the obtained path.

Proposition 1 (Sankoff and Blanchette [SB97]) *If $s = s_1, -s_1, s_2, -s_2, .., s_n, -s_n$ is a solution to TSP on the graph (G, w) then the genome median is given by $S = s_1 s_2 ... s_n$.*

In the same way, it is possible to compute a lower bound $L(G)$ such that $\mu(g) = -M + m$ with m the smallest weight of edges incident to g.

Generalization to more than 3 genomes The median problem can be applied for $N > 3$ genomes. In this case, it corresponds to a completely unresolved tree where there are $N + 1$ vertices with N leaves (contemporary genomes) and one vertex of degree N that is the median genome. Based on the procedure *BBF* given before (algorithm 4), this is done by modifying $w(gh)$ which becomes $N - u(gh)$.

Phylogenetic tree reconstruction

To solve the multiple rearrangement problem, Blanchette et al. [BBS97] and Sankoff and Blanchette [SB98] give a heuristic analogous to the iterative improvement method of Sankoff et al. [SCL76] adapted for the genomics context in [SSK96, FNS96].

The latter is based on a fixed phylogenetic topology seen as an unrooted binary tree T. The N leaves of T correspond to considered genomes and the ancestral genomes that are sought are represented by its $N - 2$ internal nodes. This is a phylogenetic version of the Steiner problem that consists in iteratively improving ancestral genomes by solving the median genome problem for the 3-stars defined by an intermediate vertex and its immediate neighbours.

This strategy requires one to initialize internal permutations. In fact, the global optimality of the obtained tree depends on this initialization step. That is why Sankoff and Blanchette [SB98] (see also [BBS97]) propose several initialization strategies. Assigning values to internal nodes can be done arbitrarily by assigning random permutations. A more reasonable solution assigns permutations by consensus from the three closest genomes in extremities. However, simulations realized by the authors to compare initialization strategies show that more complex methods prove to be more efficient. These methods are based on the resolution of an initial TSP where edge-weights are either the average of the corresponding edge-weights at the three immediately neighbours, or computed by dynamic programming minimizing adjacency disruptions and creations.

2.4.2 Rearrangement-based method

Section 2.4.1 presents Sankoff and Blanchette's work on the median problem based on breakpoint study. The breakpoint number between two genomes leads to a lower bound for the rearrangement distance between the two same genomes.

Although these two distance measures are closely related, it turns out that the study of rearrangements for reconstruction of phylogenetic trees is more representative from the biological point of view than the one of breakpoints [SM01, MSTL02]. Bourque and Pevzner were interested in this problem in the unichromosomal case as well as the multichromosomal one and implemented a program for tree reconstruction called MGR [BP02] that relies on another tool for distance computation, namely GRIMM [Tes02b]. To present Bourque and Pevzner's method, we first apply it to $N = 3$ genomes (the median genome problem) and then give extensions for $N > 3$ genomes (the multiple genome rearrangement problem).

Median genome problem

Unichromosomal genome method Let G_1, G_2, G_3 be three unichromosomal and signed genomes defined over the same set of gene markers $\mathcal{G}$. For this kind of genome, only one type of rearrangement is taken into the account: reversals. Bourque and Pevzner's method [BP02] consists in applying successive reversals to G_1, G_2 or G_3. From the parsimony principle, reversals to apply, called *good reversals*, are intuitively those which make contemporary genomes closer to the searched ancestor. But which are these reversals since the median genome is unknown? Bourque and Pevzner indicate and confirm by simulation that a reversal applied to a genome which movers this genome closer to the other two can reasonably be considered as a good reversal.

Thus, the proposed algorithm applies good reversals to G_1, G_2 or G_3 in order to make them converge towards an unique permutation: the ancestor.

Definition 32 *Let G_1, G_2, G_3 be the considered genomes for the median problem. A* good reversal *ρ applied to G_1 is a reversal such that: $d(G_1.\rho, G_2) < d(G_1, G_2)$ and $d(G_1.\rho, G_3) < d(G_1, G_3)$. Defined similarly for G_2 and G_3.*

Denote by $\Delta(\rho)$ the global reduction of reversal distances $\Delta(\rho) = d(G_1, G_2) + d(G_1, G_3) - (d(G_1.\rho, G_2) + d(G_1.\rho, G_3))$. A reversal decreases the distance between two genomes by at most 1, then a good reversal ρ verifies $\Delta(\rho) = 2$. It is possible to enumerate all the possible good reversals applicable to G_1, G_2 or G_3. However, there are two problems: if several good reversals exist, which should one apply? If there is no good reversal, which reversal should be applied then?

It is important to note that there are interactions between reversals. If two reversals have disjoint spans then applying one has no consequence on the other. Nevertheless, if their spans overlap, applying one reversal can modify the "quality" of another. Thus, the number of good reversals in resulting permutations can vary as a function of the good reversal applied. Bourque and Pevzner base their method on the hypothesis that good reversals applied in the correct order affect the less likely good reversals that are available, and so they define the notion of *best reversal.*

Definition 33 *Let n_ρ the number of good reversals after applying ρ. A* best reversal *ρ among good reversals is such that n_ρ is maximal.*

When the number of good reversals is sufficient to converge towards an unique permutation, the three genomes form a *perfect triangle* (see figure 2.14 for an example). In the contrary case, if all of the good reversals are used up, a best reversal ρ with $\Delta(\rho) < 2$ has to be found. Bourque and Pevzner propose a search of depth k in the tree of possible reversals which minimizes the global sum of reversal distances for each pair of genomes. Let $\rho_1, \rho_2, .., \rho_k$ be a sequence of k reversals applied to G_1, then they define $\Delta(\rho_1, \rho_2, .., \rho_k) = d(G_1, G_2) + d(G_1, G_3) - (d(G_1.\rho_1..\rho_k, G_2) + d(G_1.\rho_1..\rho_k, G_3))$ as the global reduction of reversal distances for this sequence of reversals.

Definition 34 *Let $\rho_1, \rho_2, .., \rho_k$ be the sequence of reversals applied to G_1 such that $\Delta(\rho_1, \rho_2, .., \rho_k)$ is maximal. If there is no good reversal, the* best reversal *in G_1 is the first reversal ρ_1 of the sequence such that Δ is maximal. Defined similarly for G_2 and G_3.*

G_1:	$1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$
G_2:	$1\ 2\ -3\ 4\ -6\ -5\ 7\ 9\ -8$
G_3:	$-1\ 2\ -3\ -4\ 5\ 6\ 7\ 9\ -8$
A:	$1\ 2\ -3\ 4\ 5\ 6\ 7\ 9\ -8$

Genomes	G_1	G_2	G_3
G_1	0	4	5
G_2	4	0	3
G_3	5	3	0

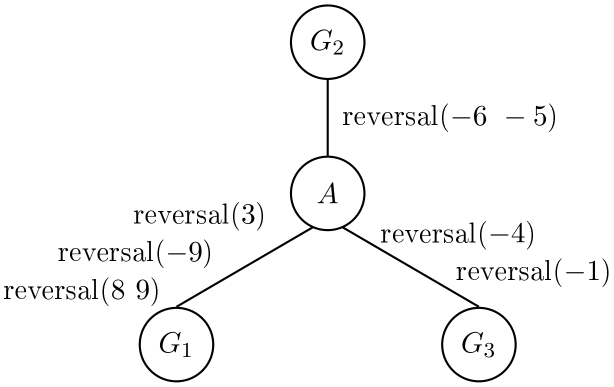


Figure 2.14: Perfect triangle formed by genomes G_1, G_2 et G_3 from [BP02]. MGR gives an optimal ancestor A for these genomes as well as optimal scenarios. The table indicates distances for each couple of genomes: they are equal to those found in the constructed genomic tree going through A.

The algorithm consists in applying a succession of best reversals first taken among good reversals.

Adaptation to the multichromosomal case In the case of multichromosomal genomes, the number of considered operations is higher: translocations, fusions and fissions added to reversals are the most frequent rearrangements in multichromosomal genomes.

Bourque and Pevzner generalize the algorithm given for unichromosomal genomes using the rearrangement distance rather than the reversal distance. Notions of *global reduction* $\Delta(\rho)$ for a reversal ρ, *good*, and *best reversals* are extended to multichromosomal case as global reduction $\Delta(\rho)$ for a rearrangement ρ, good, and best rearrangements according to the rearrangement distance.

However, the choice of the rearrangement to apply is more constrained in the multichromosomal case. In fact, there exists a situation specific to multichromosomal genomes: for 3 multichromosomal genomes, all possible couples of genomes can have a rearrangement distance equal to 1 (see example 2.15). Thus, reconstructed ancestor can be equally G_1, G_2 or G_3. In order to resolve this ambiguity, Bourque and Pevzner give priority to reversals and translocations against fissions and fusions in the choice of good and best rearrangements, starting from the observation that the two first types of operations are the most frequent in studied species (i.e. mammalian genomes).

$$\begin{aligned} G_1 = & \quad \{1\ 2\ 3\ 4\ 5\} \\ G_2 = & \quad \{1\ 2\ -5\ -4\ -3\} \\ G_3 = & \quad \{1\ 2,\ 3\ 4\ 5\} \end{aligned}$$

Figure 2.15: Example from [BP02] of three multichromosomal genomes, G_1, G_2 and G_3, all at distance 1 from each other. A reversal separates G_1 from G_2, a fission separates G_1 from G_3 and G_2 from G_3.

Another biological constraint is presented in [BP02]. It is based on the following hypothesis: a good rearrangement is a rearrangement that does not break a *conserved adjacency.*

Definition 35 *A pair of elements $g.h$ is a* conserved adjacency *if $g.h$ or its opposite, $-h.-g$, is present in all genomes as consecutive elements.*

In fact, according to the parsimony principle, it is less likely that nature breaks an adjacency to form it again later. However, the hypothesis such as it is formulated by Bourque and Pevzner, does not seem to bring a new constraint in ancestral reconstruction. By construction, a conserved adjacency between two genomes cannot be broken during the computation of a parsimonious scenario being reconstructed later. This runs counter to the parsimony criterion. Thus, rearrangements that break an adjacency conserved in N genomes cannot exist in a parsimonious scenario.

Multiple genome rearrangement problem

Resolving the multiple genome rearrangement problem is based on the same principle as for three genomes. However, the notion of good rearrangement has to be redefined with respect to N genomes. This is done by redefining the global reduction $\Delta(\rho)$ of rearrangement distances for the rearrangement ρ applied to the genome G_i:

$$\Delta(\rho) = \sum_{j \neq i} d(G_i, G_j) - \sum_{j \neq i} d(G_i.\rho, G_j)$$

Definition 36 *Let N be the number of considered genomes. A* good rearrangement *ρ applied to G_i is a rearrangement that decreases the rearrangement distance between G_i and all the $N-1$ other genomes by $\Delta(\rho) = N-1$.*

Contrary to the median genome problem, we must determine the starting point for the tree reconstruction. Two strategies are considered: the first considers all N genomes and progresses bit-by-bit towards a common ancestor; the second starts from the median problem (for 3 genomes) and, by successive additions of one genome, determines a phylogenetic tree.

The first method described is without constraint: good rearrangements are applied until 2 genomes converge towards an unique permutation. The operation is done again for $N-2$ genomes and the reconstructed intermediate ancestor. This process is reiterated until the complete resolution of the median problem for the three last genomes. This method is hardly applicable when

N is high and good rearrangements are quickly used up. That is why Bourque and Pevzner propose the second method.

The second technique is constrained by rearrangement distances. In fact, the starting point consists in solving the median problem with the 3 closest genomes in terms of rearrangements. Then, supplementary genomes are successively added to the partially constructed tree T. Let $G_1, G_2, .., G_l$ be the genomes already placed into the tree T. In order to place the genome G_{l+1} into the tree, one has first to determine which edge of the tree has to be divided to insert G_{l+1}, and second to minimize rearrangement distances between leaves. The placement heuristic chosen by Bourque and Pevzner to locate G_{l+1} is still based on rearrangement distances: the edge to divide is the one for which its two extremities and the genome G_{l+1} form a perfect triangle or at least come to it as close as possible. Thus, for each edge $\{u, v\}$ of T, the median genome A of u, v and G_{l+1} is computed. Bourque and Pevzner define then the *addition cost of a genome to an edge.*

Definition 37 *The* addition cost of a genome G_{l+1} to an edge $\{u, v\}$ *is:* $C(u, v) = d(u, A) + d(v, A) + d(G_{l+1}, A) - d(u, v)$ *where* A *is the median genome of* u, v *and* G_{l+1}.

The edge $\{u, v\}$ to divide for inserting G_{l+1} is the one for which $C(u, v)$ is minimal. By construction, the inferred ancestor converges towards species that are close to each other.

2.4.3 Other works based on parsimony

The multiple genome rearrangement problem is widely treated in the literature. We have already mentioned the method based on the reversal distance proposed by Caprara [Cap03] based on the breakpoint graph model of Hannenhalli and Pevzner [HP95a]. Another approach was proposed by Siepel and Moret [SM01] that permits the extension of GRAPPA software [BMW$^+$] by replacing the breakpoint median routine by a reversal one.

Other repertoires of operations were considered to solve the multiple rearrangement problem. For example, Adam and Sankoff [AS08] developed an approach similar to that of Bourque and Pevzner [BP02], but taking into account transpositions and block-interchanges which can be seen as a generalization of transpositions (exchanged segments in block-interchange can not be contiguous) as well as reversals and translocations. This set of operations is grouped in the DCJ (Double-Cut-and-Join) model introduced by Yancopoulos et al. [YAF05].

All of these studies implicitly start from genomes with the same marker content where each marker is present in exactly one copy. It is not rare that studied genomes have several copies for a marker (e.g. marker families). Starting from a contemporary genome where each marker appears twice, El-Mabrouk and Sankoff [EMS03] propose to recover the ancestral duplicated genome under the whole-genome duplication hypothesis by minimizing the number of reversals and/or translocations based on Hannenhalli and Pevzner's theory [HP95a, HP95b]. Zheng et al. [ZZAS08] adapted this method to the *genome halving problem* by guiding the reconstruction with one or several outgroup genome(s) that diverged before the genome duplication event.

As well as whole genome duplication event, duplications at a segmental level exist. The latter case was studied by El-Mabrouk [EM02] who proposed an algorithm that computes an ancestral genome without duplication from a genome having marker families of any size by minimizing reversals and duplication transpositions. In the same paper [EM02], this method is used in order to extend the multiple genome rearrangement algorithm based on breakpoint analysis [SB97, BBS97, SB98] by taking into account duplication events.

2.4.4 Lack of biological constraints

Medians are not unique

A considerable drawback to formulating the problem as the search for a single complete assembly that minimizes the sum of genome distances, is that the set of mathematically equivalent solutions is quite large and widespread. For example, in [BZB$^+$05] more than 3000 solutions are found for the human-murid ancestor, and indeed a statistical study of the variance between minimal solutions by [Eri07] suggests that reporting an unique median architecture is misleading, particularly when medians are the basis of phylogenetic tree reconstruction. A more realistic approach is to consider what common structural features of ancestral genomes might be found. Partial reconciliation of comparative maps identifies permutations of markers as above but does not necessarily provide a total order between segments (see section 2.5).

In silico versus cytogenetic methods

A wider debate exists between the proponents of the *in silico* approach through rearrangement-based methods and the proponents of the cytogenetic approach. Exemplified by Froenicke *et al.* [FCG$^+$06], the latter group argues essentially that under-sampling in the *in silico* approach combined with the tendency of closely related genomes to attract the median, leads to non-unique results that diverge from those found using cytogenetic methods. Bourque *et al.* in their response [BTP06] argue that under-sampling will disappear with time and that the distinction between strong and weak adjacencies (present or not in all explored reconstructions) identified in the *in silico* method permits reliable comparison between the different approaches. Moreover, *in silico* method overcome certain problems of cytogenetic reconstruction: small segments (< 1 Kb), interchromosomal and intrachromosomal rearrangements as well as marker orientation can be studied.

Rocchi *et al.* in their perspective [RAS06] suggest that a combination of the two approaches should lead to more realistic ancestral architectures, but furthermore that it is necessary to better model biological considerations, especially centromere repositioning and segmental duplication.

2.5 Piece-wise reconstruction

In the previous section, we have seen that reporting an unique global median architecture is misleading. A more realistic approach is to consider what common structural features of ancestral genomes might be found. Partial reconciliation of comparative maps identifies permutations of markers as above but does not necessarily provide a total order between segments.

In what follows, we present the method of Ma et al. [MZS$^+$06] for finding *contiguous ancestral regions (CARs)* by assigning to each node of a given phylogenetic tree a set of adjacencies that represent a consensus between those found in contemporary genomes, computed using a method analogous to Fitch's parsimony method [Fit71] and relying on knowledge of the phylogenetic tree. However, we will show that consideration of phylogeny for the reconstruction of ancestral architecture is not completely justified since no proof has been provided that recombinatory evolution coincides with mutational evolution.

2.5.1 Method from phylogenetic data

Ma et al. [MZS$^+$06] propose a computational method to predict the order and orientation of conserved segments in the ancestor through the detection of *CARs* (Contiguous Ancestral

Regions), that represent consistent parts in the ancestor. Their method is based on adjacencies in contemporary genomes, requires a phylogenetic tree and is quite similar to Fitch's parsimony method [Fit71], nucleotides being replaced by adjacencies as elements of phylogeny.

Predecessor and successor graphs

Let T_p be the considered phylogenetic tree where leaf nodes are contemporary genomes. A modern genome is represented by permutations as it is described in section 1.2.1. Duplication events are not take into the account. However, it is not explicitly specified whether contemporary genomes share exactly the same set of markers.

Inferring CARs consists in finding an unique predecessor and successor for each element in the ancestral genome. First, Ma et al. independently solve predecessor and successor searches by a two-step method. In what follows, we present the predecessor search.
The first stage computes a set $P_u(i)$ of possible predecessors for an element i in the node u in a bottom-up fashion. In the case where u is a leaf node, $P_u(i)$ is a singleton representing the unique predecessor of i in u. Otherwise, u has two child nodes, v and w, and $P_u(i) = P_v(i) \cup P_w(i)$ or $P_u(i) = P_v(i) \cap P_w(i)$ depending on whether sets $P_v(i)$ and $P_w(i)$ are disjoint or not. This is done for all nodes of T_p including outgroups until the common ancestor R of all species is reached.

The information on predecessors can be summed into a graph called *Predecessor graph* for each node u. The predecessor graph for a node u of T_p is a directed graph where each marker is represented by two vertices (positive and negative versions). Two special vertices (symbol 0 for both) are added to represent the beginning and the end of a chromosome. An edge (a, b) of a predecessor graph means that the element a belongs to the set $P_u(b)$.

The second step consists in refining, for ancestral nodes, predecessor graphs built during the first stage by propagating $P_R(i)$ down the tree. During the descent in the tree, designate by A and D ancestor and its descendant along a branch. For each i of D, $P_D(i)$ is refined in the following way: $P_D(i) = P_D(i) \cap P_A(i)$ if $P_D(i) \cap P_A(i) \neq \emptyset$; otherwise, $P_D(i)$ remains unchanged.
Similarly, sets of successors for each element i of a node u of T_p, $S_u(i)$ are inferred and lead to *successor graph* construction. In a successor graph of a node u, an edge (a, b) means that the element b belongs to the set $S_u(a)$.

Graph reconciliation into CARs

Clearly, predecessor and successor graphs of a leaf node are identical while those for ancestral nodes generally differ. However, they are not totally different and common parts can be extracted from a new graph G obtained by the intersection of the predecessor and successor ones.

Ambiguities for some elements may still remain: an element i can (a) have several possible predecessors in G, or (b) have several possible successors, or (c) participate in a cycle of G. In order to choose an unique predecessor and successor for an element in G, G is transformed into a weighted graph according to phylogenetic information. The weight $w_A(i, j)$ of an edge (i, j) of the graph G of a ancestral node A is 1 if neither i nor j are in ambiguous case (a) or (b); otherwise,

$$w_A(i, j) = \frac{\mathcal{L}(A,L).w_R(i,j) + \mathcal{L}(A,R).w_L(i,j)}{\mathcal{L}(A,L) + \mathcal{L}(A,R)}$$

where $\mathcal{L}(A, R)$ ($\mathcal{L}(A, L)$, respectively) is the length of the branch linking an ancestral node A to its right (left, respectively) child. Note that if L (R, respectively) is a leaf node, then $w_L(i, j) = 1$ if edge (i, j) belongs to its predecessor graph, and $w_L(i, j) = 0$ otherwise.

Based on this weighted graph for an ancestral node A, Ma et al. propose a greedy heuristic approach to compute a set of paths that cover all the nodes in G, trying to maximize the total edge weights in all of them. This is done by a constructive algorithm that tries to add edges to paths representing CARs starting from the edges of greatest weight. An edge is retained in resulting paths if its addition does not cause an ambiguous case (a) or (b). This process is repeated until no more edges can be added. To solve the ambiguous case (c) in the resulting graph, Ma et al. claim that if such a case appears then the weight of each edge in the formed cycle is 1. Consequently, discarding any edge to break the cycle is sufficient. See example 2 for a complete illustrated case.

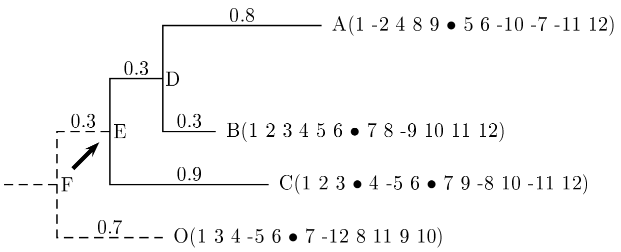

0.8
A(1 -2 4 8 9 • 5 6 -10 -7 -11 12)
0.3
D
0.3
E
0.3
B(1 2 3 4 5 6 • 7 8 -9 10 11 12)
F
0.9
C(1 2 3 • 4 -5 6 • 7 9 -8 10 -11 12)
0.7
O(1 3 4 -5 6 • 7 -12 8 11 9 10)

Figure 2.16: The phylogeny of genomes A,B, C [MZS$^+$06]. The target ancestor is E, and O is the outgroup. The bullet symbol separates chromosomes. Branch lengths are above each branch.

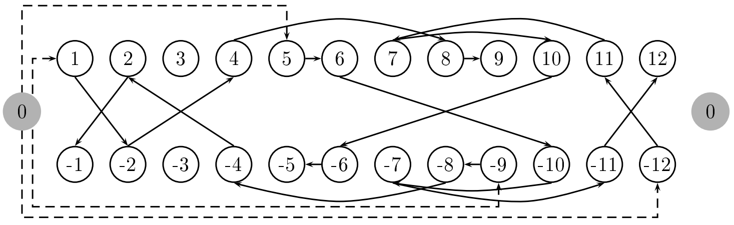


Figure 2.17: Predecessor graph of A from [MZS$^+$06].

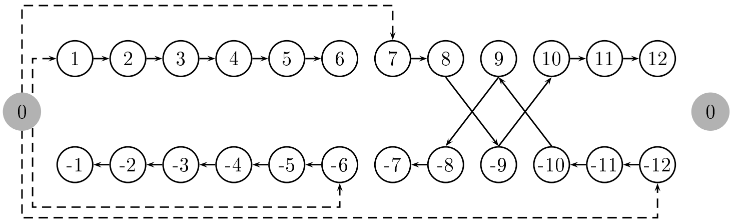


Figure 2.18: Predecessor graph of B from [MZS$^+$06].

Example 2 *Figures 2.17 to 2.26 are those of the practical example given by Ma et al. in [MZS$^+$06]. Given a phylogeny between genomes A, B and C (see figure 2.16), predecessor graphs of A, B and C are directly constructed from the leaf genomes (see figures 2.17, 2.18 and 2.19). Figures 2.20, 2.21 and 2.22 represent the predecessor graphs of internal nodes D, E and F obtained after the bottom-up step. Predecessor graph of E (see 2.23) is adjusted by propagating the predecessor graph of F. In the same way, the final successor graph of E is obtained (see 2.24).*

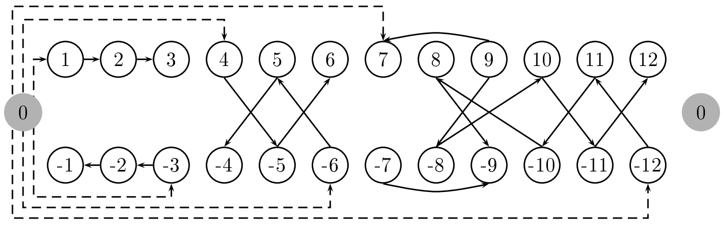


Figure 2.19: Predecessor graph of C from [MZS+06].

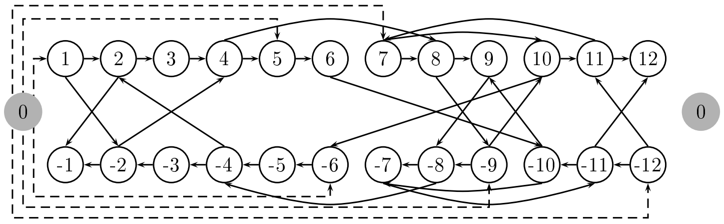


Figure 2.20: Predecessor graph of D from [MZS+06].

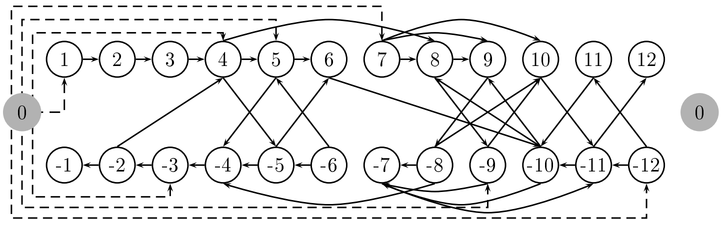


Figure 2.21: Predecessor graph of E from [MZS+06].

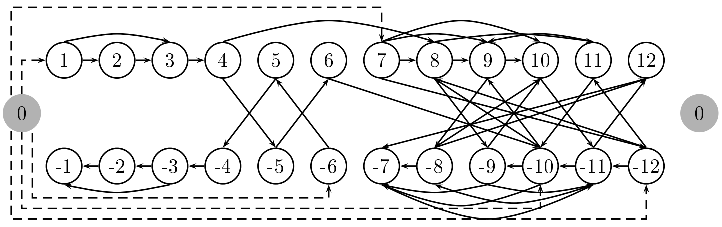


Figure 2.22: Predecessor graph of F from [MZS+06].

Resulting CARs (see figure 2.26) are determined from intersection of predecessor and successor graphs of E (figure 2.25) where ambiguities are solved based on phylogeny information.

CARs with duplications

The initial method of Ma et al. for inferring CARs does not incorporate duplication events. Recently, in [MRR+08], Ma et al. propose a heuristic algorithm called DUPCAR that is an extension of CARs method by including duplications based on a set of gene trees in addition to a phylogenetic tree and a set of contemporary genomes.

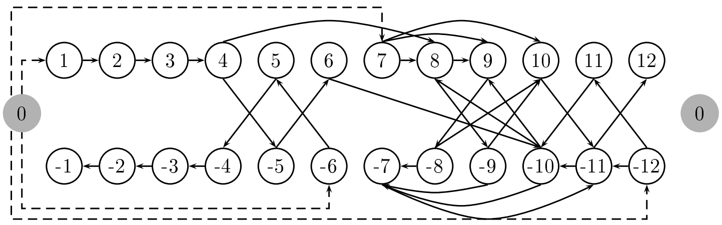


Figure 2.23: Predecessor graph of E after being adjusted by F from [MZS+06].

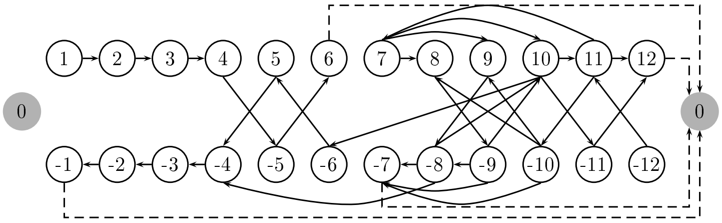


Figure 2.24: Successor graph of E from [MZS+06].

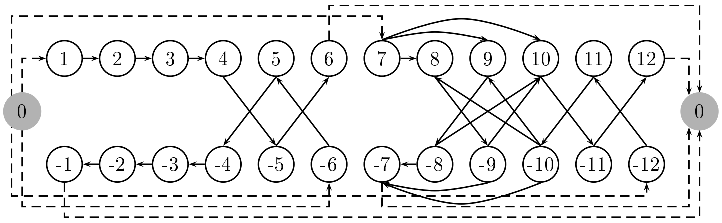


Figure 2.25: Intersection of the predecessor and successor graphs of E from [MZS+06].

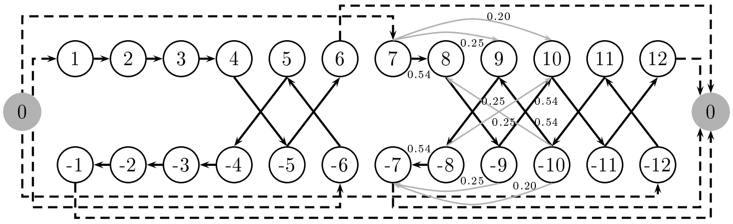


Figure 2.26: The resulting CARs from [MZS+06].

2.5.2 Phylogeny vs evolution mechanisms

The method proposed by Ma et al. does not try to solve the *multiple genome rearrangement problem* and clearly leans on phylogenetic data to predict an ancestral genome. Phylogeny relationships between species are inferred according to the rate of mutations in genomic sequences. Another evolutionary measure between species consists in computing rearrangement or breakpoint distances based on a mathematical model for genomes. While the former implies a temporal notion, the latter does not provide information on the time-scales of the rearrangement events. Although these two measures may converge towards similar results, it is not systematic. Indeed, some authors propose to study the relationship between the phylogenetic distribution of species and the disruption of syntenic blocks via chromosomal inversion events (see [BSR$^+$08] for application to *Drosophila* genomes).

However, in the case of the *multiple genome rearrangement problem* for which we present two methods (see section 2.4.2 for rearrangement-based method and section 2.4.1 for breakpoint-based one), the authors speak in terms of phylogenetic tree reconstruction. According to the paragraph above, the use of this term is somewhat misleading since rearrangements represent a different measure of evolution: thus we will prefer using the notion of *rearrangement tree*, that can be by definition different from the phylogenetic tree for the same set of species.

Part II

SyDiG: uncovering Synteny in Distant Genomes

Chapter 3

SyDiG algorithm

Comparative analysis of complete genomes has over the past ten years provided increased understanding of the processes and mechanisms of evolution, development, and gene regulation. One area where significant insight has been obtained is genome rearrangements, where the mechanisms of chromosomal dynamics have been explored through comparison of chromosomal maps within and between species. A key prerequisite for such studies is the accurate identification of *genome synteny*, since conserved gene order between two (or more) related species indicates chromosomal homology inherited from their common ancestor.

In section 2.1, we presented several computational methods for the identification of genome synteny. In particular, we focused our attention on GRIMM-Synteny [PT03a, BPT04, BZB^{+}05], which determines synteny blocks with the explicit aim of studying rearrangements. However, all of these methods perform well on the 'low-hanging fruit' of highly similar (e.g. mammalian) genomes, but less well on highly divergent genomes with extensive map reshuffling.

In this chapter we present a new algorithm, called SyDiG (Synteny in Distant Genomes) that processes complete genome sequences in order to infer cross-species synteny, and algorithms with the ability to handle species having a large evolutionary span. Our method computes synteny blocks for $N \geq 2$ genomes. It is a three-step process. First, we perform a pre-processing step that consists in determining homologous genes and, from those, in computing multiplicons of level two using AdHoRe routine [VSS^{+}02]. Multiplicons constitute the starting point of our study and all of the homology information contained in them is described in terms of graph theory through the *synteny graph*. Second, based on this graph, we try to extend certain homologies by transitivity. Finally, initial homology information and supplementary homologous elements are used to reconstruct synteny blocks.

3.1 Pre-processing

The starting point for synteny identification is the definition of pairwise homology relationships between genomes. We use the consensus clustering algorithm [NS07], although raw clustering methods can be used such as [EDO02].

Sequence similarity is generally detected either at the DNA level or by relying on genomic maps. In the latter case, the study of gene order makes it possible to detect homology even for highly divergent chromosomic regions. This is exactly the role of i-ADHoRe [SVSP04, SJSV08], a method, explained in section 2.1.2, for identifying segments of chromosomal homology (multiplicons) through the identification of gene order and content conservation.

Recall that a multiplicon is formed by one or several homologous genomic segments and its level indicates the number of segments it contains. In this study i-ADHoRe is solely used to compute level two multiplicons that will be simply called multiplicons in the rest of the chapter. Notice that i-ADHoRe determines the multiplicons based on gene order. Hence, the coordinate system used is at gene level: each element of a genomic segment is mapped to a gene and each chromosomic segment is delimited by two genes, one on each side.

Multiplicons obtained by i-ADHoRe correspond to homologies between two genomic segments (belonging to the same genome or not). The goal now is to refine these homologies into synteny blocks for the set of considered genomes $\{G_1, .., G_N\}$. We do this by analyzing the composition of each multiplicon and computing the synteny blocks using transitivity relations.

3.2 Synteny graph

The first step is to assemble all the information contained in the multiplicons into a graph. This graph has to represent two types of information: first, homology between genomic segments; second, possible overlaps between multiple segments of the same chromosome. Let $\{G_1, .., G_N\}$ be the set of genomes for which we want to compute synteny blocks and $\mathcal{M}$ be the set of multiplicons obtained by AdHoRe for these genomes.

For the needs of the method, we propose a more formal definition of the notion of multiplicon. Let $M = \langle I_1, I_2, A\rangle$ be a (level two) multiplicon where I_1 and I_2 denote the genomic segments that it contains, and A is the set of anchors within it. We note a genomic segment I_i as a sequence of genes $I_i = (g^i_b, .., g^i_e)$ such that g^i_b and g^i_e represent the gene boundaries of this segment. A gene g^i_j of a genomic segment I_i is a pair $\langle p_j, c_j\rangle$ such that p_j is its relative position on the chromosome c_j. If two genes $g^1_i \in I_1$ and $g^2_j \in I_2$ form an anchor in M, then $\langle g^1_i, g^2_j\rangle \in A$.

Figure 3.1 shows an example of multiplicons for $N = 5$ genomes. This same example will be followed through the chapter.

The *synteny graph* G is defined from the set $\mathcal{M}$ of multiplicons for the N genomes under study.

Definition 38 *A* synteny graph $G = (V, E)$ *is a non-oriented edge-colored graph such that*

- $V = \{g^i_j \mid g^i_j \in I_i \in M \in \mathcal{M}\}$ *is the set of all genes participating in a multiplicon,*
- E *is the edge set such that* $\forall e = \{g^i_n, g^j_m\} \in E$ *either* g^i_n *and* g^j_m *form an anchor in a multiplicon of* $\mathcal{M}$ *(dashed edge), or* g^i_n *and* g^j_m *are consecutive on the same chromosome (black edge).*

In this graph, we can distinguish three types of vertices:

(1) *boundary vertices* correspond to gene boundaries of genomic segments participating in a multiplicon,

(2) *anchor vertices* correspond to genes that form an anchor with some other gene and are not boundaries of any genomic segments,

(3) *interleaving vertices* are the other vertices that are neither a boundary nor an element of an anchor.

Note that boundary vertices always form an anchor, since AdHoRe computes multiplicons in such a way that extremities of genomic segments that define them are determined by the leftmost and rightmost coordinates of their anchors. Thus, a gene can be both a boundary of one or several

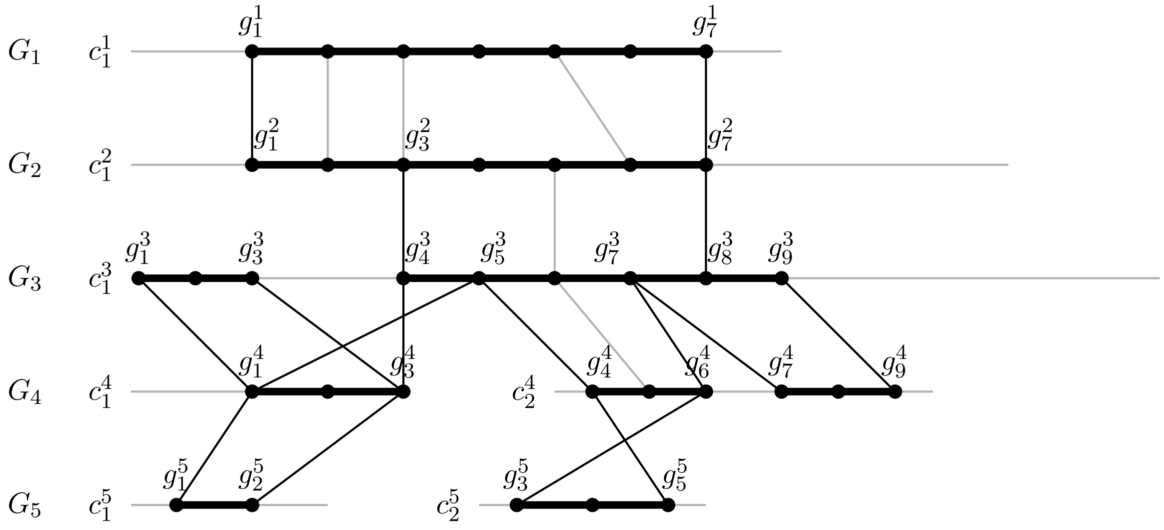


Figure 3.1: Level 2 multiplicons for genomes $\{G_1, .., G_5\}$. Each genome G_i is shown on a separate line with chromosomes denoted by c_1^i, c_2^i,.., c_k^i (k the total number of chromosomes for G_i). A genomic segment $(g_j^i, g_{j+1}^i, ..., g_{k-1}^i, g_k^i)$ on G_i is represented by a bold line on the chromosome and is explicitly delimited by its boundaries g_j^i and g_k^i. Dots along chromosomes represent gene locations. A grey (dark, respectively) line materializes an anchor formed by genes (gene boundaries, respectively) at its extremities.

genomic segments taking part in multiplicons and a simple anchors in other segments. An anchor vertex is a gene that forms an anchor strictly inside one or several multiplicons.

Figure 3.2 shows the synteny graph obtained for the 5 genomes and their multiplicons of figure 3.1.

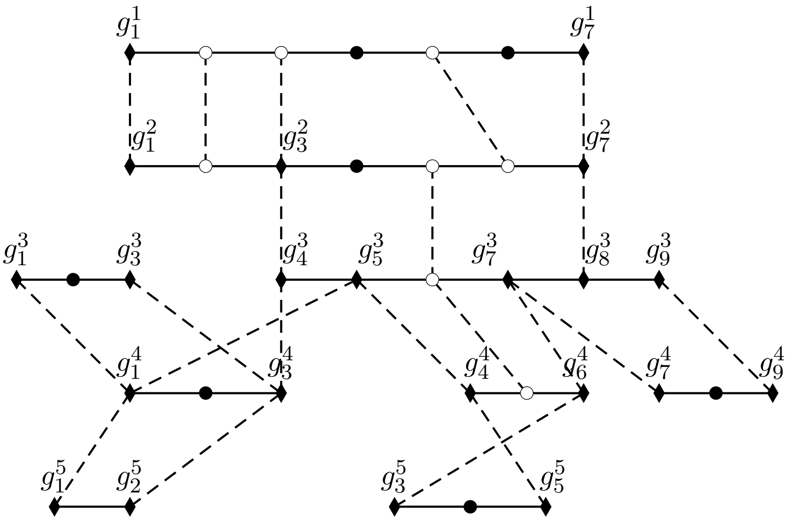


Figure 3.2: Synteny graph obtained for the data shown on figure 3.1. Dashed edges represent homologies while black ones represent gene adjacency. Boundary vertices (anchor vertices, and gene vertices, respectively) are represented by diamonds (white circles and full circles, respectively).

3.3 Extension of homologous boundaries

The synteny graph represents gene relationships within and between genomes: physical relationships are modeled by black edges, which represent gene adjacencies, while dashed edges model homology information contained in multiplicons. From synteny graph, we define two kinds of dependency between elements.

3.3.1 Extended segments

Genomic segments taking part in multiplicons can be physically dependent, since some of them overlap. It is from this kind of dependency that we can infer new homology relationships by combining the information contained in multiplicons related by overlapping genomic segments. Thus, we isolate the set of genes that are dependent only due to chromosomic overlaps. All of these genes will belong to the same *extended segment.*

Definition 39 *An* extended segment *for a given genome is a maximal genomic segment* $I_{max} = (g_b, .., g_e)$ *defined from the set of genomic segments* $\{I_1, .., I_k\}$ *belonging to the same chromosome* c *such that*

(i) $g_b = \langle p_{b_{min}}, c\rangle \in I_i$ *with* $1 \leq i \leq k$ *such that* $p_{b_{min}} = \min(\{p_i \mid g_i = \langle p_i, c\rangle \in I_i, 1 \leq i \leq k\})$,

(ii) $g_e = \langle p_{e_{max}}, c\rangle \in I_i$ *with* $1 \leq i \leq k$ *such that* $p_{e_{max}} = \max(\{p_i \mid g_i = \langle p_i, c\rangle \in I_i, 1 \leq i \leq k\})$,

(iii) 2 consecutive genes on the extended segment belong to the same genomic segment: $\forall g_i, g_{i+1} \in I_{max}$, $\exists I \in \{I_1, .., I_k\}$ *such that* $g_i \in I$ *and* $g_{i+1} \in I$,

(iv) the extended segment satisfies the criterion of maximality: $\forall I \notin \{I_1, .., I_k\}$ *et* $\forall j \in [1, k]$, I *and* I_j *do not overlap.*

The set of extended segments obtained for a given synteny graph G is computed from the connected components of the subgraph of G induced by the black edges of G. In fact, the resulting subgraph can be decomposed into chains that correspond to extended segments.

The synteny graph of figure 3.2 contains 9 extended segments, namely:

- $S^1_1 = (g^1_1, .., g^1_7)$ belonging to G_1,
- $S^2_1 = (g^2_1, .., g^2_7)$ belonging to G_2,
- $S^3_1 = (g^3_1, .., g^3_3)$ and $S^3_2 = (g^3_4, .., g^3_9)$ belonging to G_3,
- $S^4_1 = (g^4_1, .., g^4_3)$, $S^4_2 = (g^4_4, .., g^4_6)$ and $S^4_3 = (g^4_7, .., g^4_9)$ belonging to G_4,
- $S^5_1 = (g^5_1, .., g^5_2)$ and $S^5_2 = (g^5_3, .., g^5_5)$ belonging to G_5.

3.3.2 Groups of homologous genes and boundaries

The goal of our algorithm is to determine synteny blocks for N genomes under study. We use transitivity of the relation defined by the multiplicons in order to solve the missing homologies between genomic segments. Thus, if I_1 is homologous to I_2 that is itself homologous to I_3, we consider that I_1 and I_3 are also homologous. Not all homologies are that simple to solve. For example, in figure 3.1, the genomic segment $I^2_2 = (g^2_3, .., g^2_7)$ of genome G_2 does not have a homolog (direct or by transitivity) with any genomic segment of genome G_1. However, I^2_2 is included in $I^2_1 = (g^2_1, .., g^2_7)$ that itself is homologous with $I^1_1 = (g^1_1, .., g^1_7)$ of G_1. This homology makes it possible to deduce a new boundary in G_1 that "cuts" I^1_1 into two distinct intervals such that one of them is homologous with I^2_2.

Recovering homology relationships between genomic segments can then be reduced to looking for specific genes that are boundaries, and reconstructing the corresponding genomic segments. In order to do that we first partition genes forming at least an anchor in *groups of homologous genes* and in parallel, by considering only the set of gene boundaries, *groups of homologous boundaries.*

Definition 40 Groups of homologous genes *are a partition of genes forming at least one anchor such that a part of this partition is a set of genes that are either directly homologous, or that share a gene with which they form an anchor.*

Definition 41 Groups of homologous boundaries *are a partition of gene boundaries such that a part of this partition is a set of boundaries that are either directly homologous, or that share a gene with which they form an anchor.*

Groups of homologous genes (boundaries, respectively) obtained for a given synteny graph G are computed from the connected components of the subgraph of G induced by the anchor and

Groups of homologous boundaries
g_1^1, g_1^2
g_7^1, g_7^2, g_8^3
$g_1^3, g_1^4, g_1^5, g_5^3, g_4^4, g_5^5$
$g_3^3, g_3^4, g_2^5, g_4^3, g_3^2$
$g_3^5, g_6^4, g_7^3, g_7^4$
g_9^3, g_9^4

Table 3.1: Groups of homologous boundaries obtained from the synteny graph of figure 3.2

Groups of homologous genes
g_1^1, g_1^2
g_2^1, g_2^2
g_5^1, g_6^2
g_7^1, g_7^2, g_8^3
g_5^2, g_6^3, g_5^4
$g_1^3, g_1^4, g_1^5, g_5^3, g_4^4, g_5^5$
$g_3^3, g_3^4, g_2^5, g_4^3, g_3^2, g_3^1$
$g_3^5, g_6^4, g_7^3, g_7^4$
g_9^3, g_9^4

Table 3.2: Groups of homologous genes obtained from the synteny graph of figure 3.2

boundary vertices, and the dashed edges of G. The group of homologous boundaries is obtained for a given synteny graph G, in an analogous way, from the the subgraph of G induced by the boundary vertices, and the dashed edges of G.

Starting from the synteny graph from figure 3.2, we obtain groups of homologous genes and groups of homologous boundaries shown respectively in tables 3.2 and 3.1.

3.3.3 Adding and positioning of new boundaries

The next step is to check each boundary to see whether it creates new boundaries in other genomes. Each extended segment is a genomic segment defined by a maximal set of overlapping genomic segments. Hence, in each extended segment there exist boundaries of genomic segments that are included into other ones. For example, boundary g_3^2 of I_2^2 is included in the interval I_1^2 of the extended segment S_1^2. However, genomic segment I_1^1 homologous to I_1^2 does not contain any boundary homologous to g_2^2. This is precisely the situation where the need for adding new boundaries arises. In order to do this we search in the groups of homologous genes for a boundary homologous to g_3^2 in I_1^1 (see table 3.2). In this case, we find the gene g_3^1.

The algorithm *add_boundaries* implements this operation. Function *extended_segment* returns the extended segment to which a given genomic segment belongs. In the case of the addition of a supplementary boundary, if the current boundary has no homologous gene in the target genomic segment, then it is necessary to pick a gene in this segment as the homologous one. This is done by the routine *locate*: the homologous gene is the one that is proportionately located in the target segment at the same place than the current boundary in its genomic segment.

Algorithm 6 add_boundaries($\mathcal{S}$)

```
Require: Set of extended segments $\mathcal{S}$
Ensure: Set of extended segments $\mathcal{S}$ with new boundaries
Let $\mathcal{B}$ be the set of boundaries for $\mathcal{S}$
while $\mathcal{B} \neq \emptyset$ do
    $b = shift(\mathcal{B})$
    Let $\mathcal{I}$ be the set of genomic segments in which $b$ is included
    for all $I \in \mathcal{I}$ do
        for all $I'$ such that $\exists M = \langle I, I', A \rangle \in \mathcal{M}$ do
            if $\nexists$ $b_h \in I'$ such that $b_h$ and $b$ are two homologous boundaries then
                if $\exists$ $b_a \in I'$ such that $b_a$ and $b$ are two homologous genes then
                    $S' = extended\_segment(I', \mathcal{S})$
                    Mark $b_a$ as boundary in $S'$
                    Add $b_a$ in $\mathcal{B}$
                    Add $b_a$ in the group of boundaries homologous to $b$
                else
                    $S' = extended\_segment(I', \mathcal{S})$
                    $locate(b_{new}, S')$
                    Mark $b_{new}$ as boundary in $S'$
                    Add $b_{new}$ in $\mathcal{B}$
                    Add $b_{new}$ in the group of gene homologous to $b$
                    Add $b_{new}$ in the group of boundaries homologous to $b$
                end if
            end if
        end for
    end for
end while
return $\mathcal{S}$
```

For the example of figure 3.1, six new boundaries are added. All the new boundaries are shown in figure 3.3. The resulting groups of homologous boundaries are shown in table 3.3.

3.4 Reconstructing synteny blocks

Once boundary homology is completely solved, we define the genomic segments and their homology relations. In an extended segment, two boundaries form a genomic segment that is necessarily homologous with at least one other genomic segment. In order to obtain genomic segments that are disjoint for a given chromosome, it is sufficient to go through each extended segment in order,

Final groups of homologous boundaries
g_1^1, g_1^2
$g_7^1, g_7^2, g_8^3, b_6 = g_8^4$
$g_1^3, g_1^4, g_1^5, g_5^3, g_4^4, g_5^5, , b_2 = g_4^1, b_4 = g_4^2$
$g_3^3, g_3^4, g_2^5, g_4^3, g_3^2, b_1 = g_3^1$
$g_3^5, g_6^4, g_7^3, g_7^4, b_3 = g_5^1, b_5 = g_6^2$
g_9^3, g_9^4

Table 3.3: Groups of homologous boundaries obtained from groups of table 3.1 after *add_ boundaries* routine.

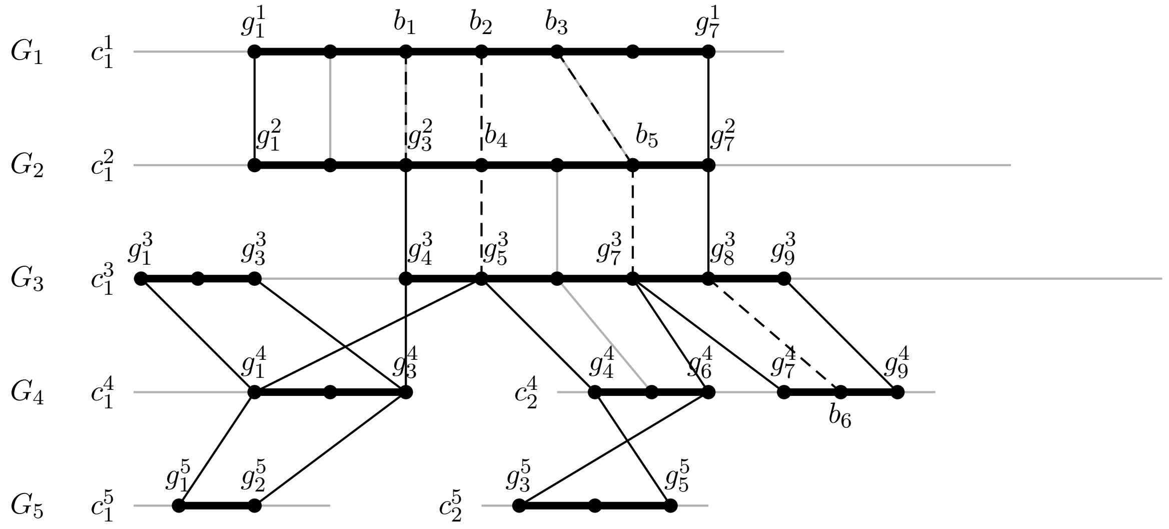


Figure 3.3: New gene boundaries $\{b_1, .., b_6\}$ added for the example from figure 3.1. Boundaries connected by edges represent homologous boundaries. The dashed edges show the homology between the new boundaries and those originally present.

where two successive boundaries delimit a genomic segment. Then, from boundary homology, we deduce homologies between segments delimited by these boundaries. This implies finding the two corresponding boundaries in another genome. If the boundaries are ordered in the same way for the two segments, then the mutual interval orientation is positive; if not, then it is negative. The result is the set of groups of homologous genomic segments.

Finally, in order to obtain *synteny blocks* for the N genomes under study, these groups are filtered in order to keep only those that contain at least one segment per genome. Final synteny blocks for the example of figure 3.1 are shown in figure 3.4.

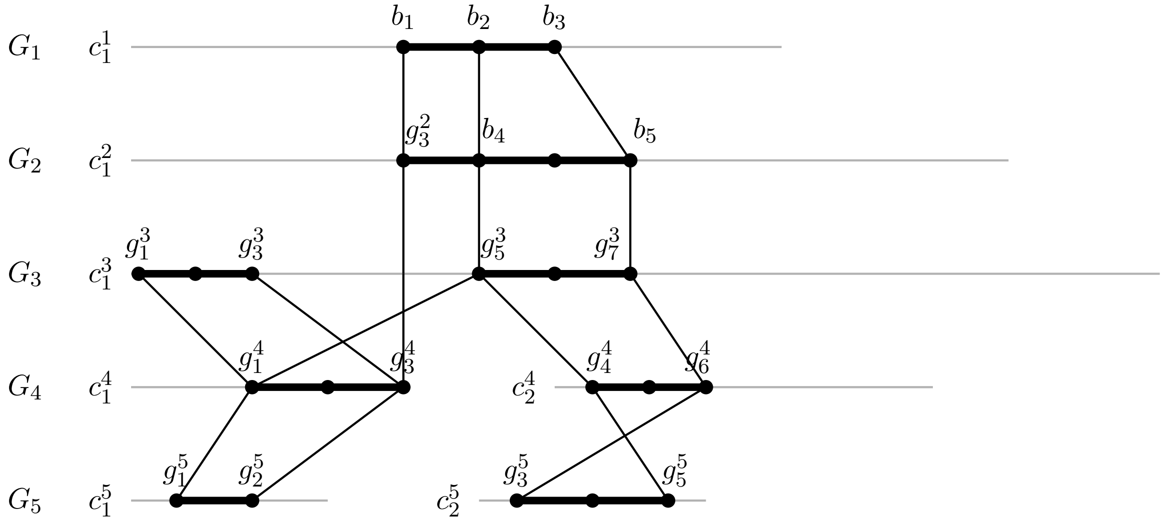


Figure 3.4: Final syntenic blocks for the example from figure 3.1. Genomic segment (g_3^3, g_4^3) is excluded in favour of (g_1^3, g_2^3) because the latter is larger. Genomic segments $(g_1^1, .., g_3^1)$, $(g_1^2, .., g_3^2)$, $(g_5^1, .., g_7^1)$, (g_6^2, g_7^2), $(g_7^3, .., g_9^3)$, $(g_7^4, .., g_9^4)$ are also excluded, since they do not participate in synteny blocks for all of the considered genomes (i.e there are no segments homologous to them in certain genome(s)).

Moreover, additional filters make it possible to adapt obtained synteny blocks as common markers used in the elaboration of signed permutations in order to study rearrangement events.

3.4.1 Duplications

Generally, the permutation model does not allow duplication events, so the SyDiG algorithm proposes to keep only the longest segment in a synteny block where more than one segment belongs to one genome. The intuition behind this filter parameter is that the longer the segment, the smaller the probability that synteny was computed by chance. Nevertheless, other parameters to choose between duplicate segments should be considered such as for example synteny block neighbouring. This is the subject of future work.

3.4.2 Concatenation

In the same permutation model, identifiers represent synteny blocks. Under the parsimony criterion, two identifiers that are adjacent in all the considered genomes cannot be separated to be joined again later. That is why, two modes are implemented in SyDiG algorithm. The first one provides all the synteny blocks and permits one to study their respective genomic segments. The second mode consists in concatenating synteny blocks that appear consecutively in all the considered genomes. This leads to the construction of signed permutations with fewer identifiers, but encoding exactly the same information as far as a study of rearrangements is concerned.

3.5 Complexity

The SyDiG algorithm determines synteny blocks by constructing synteny graph and performing operations on this graph. Synteny graph construction is linear in the number of genes involved in genomic segments participating in multiplicons. Moreover, computing extended segments in the particular subgraph of the synteny graph induced by black edges can be computed in linear time in terms of the number of vertices. Finding groups of homologous genes and groups of homologous boundaries can be computed in both cases in linear time in terms of the number of vertices and dashed edges. Boundaries addition is computed in $O(n^3)$ in the worst case, where n denotes the number of vertices in the synteny graph. Finally, the reconstruction of synteny blocks is realized by scanning the extended segments and the groups of homologous boundaries: the complexity in time is thus $O(n^2)$. Thus, SyDiG algorithm can be computed in a simple way in $O(n^3)$ where n denotes the number of genes involved in genomic segments.

Chapter 4

Applications

Nadeau and Taylor [NT84] were the first to define *conserved segments* as segments having a preserved gene order with no rearrangements between them. *Synteny blocks* are built of these conserved segments, smoothing over the noise due to microrearrangements. These blocks constitute gene markers that are the starting points for further analysis.

Synteny information has various applications for comparative genomics, such as computing rearrangement distances [HP95a] or scenarios [Tes02b], inferring the least common ancestor and rearrangement trees [BP02]. The implications of the analysis of genomic synteny can reach even further, providing insights into the manner by which evolution proceeds. This latter topic has generated a quite lively debate on the differences between random breakage and non-random breakage models of evolution [PT03a, PT03b, TMS04].

Several methods have been defined to respond to the need for finding common markers within genomes in order to study rearrangements. The main methods presented in section 2.1, GRIMM-Synteny ([PT03a], [BPT04], [BZB$^+$05]) and CHAINNET developed by Kent [KBH$^+$04] and used by Ma [MZS$^+$06], are applied to mammal data (human, mouse, rat and chicken for GRIMM-Synteny and human, mouse, rat and dog for the other) and rely on nucleotide-level alignments as obtained with tools such as BLASTZ (for example [SKS$^+$04]).

Comparative genomics analyses obviously rely on the quality of the primary computation of genomic synteny. In this chapter, we revisit the most commonly used algorithm for synteny computation, namely GRIMM-synteny [PT03a, BPT04, BZB$^+$05] (see 2.1.1, page 20 for details of the method). We argue that this algorithm, which works well for the mammalian genomes for which it was developed, produces results whose quality dramatically decreases with the increase of the evolutionary distance. We further identify the issue as a need for more careful homology identification as a preliminary step.

In the first section, we compare Grimm-Synteny and SyDiG on mammal and yeast genome sets. Then, we present a practical application to Hemiascomycetous yeasts that leads to rearrangement analysis presented in chapter 6.

4.1 GRIMM-Synteny versus SyDiG algorithm

In order to realize this comparison, we re-implemented GRIMM-Synteny as the software is not publicly available. Our reimplementation was validated using back-to-back comparison with results available on the author's webpage [Tes04].

The first challenge for comparing the behavior of these algorithms is the judicious choice of data processing. Indeed, GRIMM-Synteny and SyDiG rely on different data. The former proceeds by direct sequence alignment at DNA level (cleaned up by RepeatMasker [SHG04]). The latter relies on the existence of pre-computed protein families. While DNA alignments such as BLASTZ are reasonable for closely related genomes such as mammals, only alignments at protein level can recover distant similarities for species such as yeasts [Duj06]. The data presented below was retrieved from public databases on the 17th of June 2008.

- Mammal genomes: we have considered human, mouse and rat genomes. For these genomes, two sets of data have been retrieved from Ensembl (release 49) and Uniprot (UniRef50, release 13.5, the 10th of june 2008) data.

- Yeast genomes (*Ashbya gossypii* (Ergo), *Kluyveromyces lactis* (Klla), *Kluyveromyces thermotolerans* (Klth), *Zygosaccharomyces rouxii* (Zyro), and *Saccharomyces kluyveri* (Sakl)): data were provided by Génolevures and are available as of the 3rd of September 2008.

4.1.1 Yeast results

In order to apply Grimm-synteny to yeast data, we have computed 3 data sets from TBLASTX alignments:

- unrefined alignments (206191 alignments),

- the longest alignments when several ones overlap (51085 alignments),

- the shortest alignments when several ones overlap (59028 alignments).

Anchors were computed by GRIMM-Anchor for the levels from 2 to 5. Results for levels 2 to 4 are shown for each data set in tables 4.1, 4.2 and 4.3. No 5-level anchors are found for unrefined and longest sets and only one 5-way anchor is found for the shortest set. Based on these results, we do not use GRIMM-Synteny routine to find synteny blocks, since the number of anchors is too small.

SyDiG was used to compute synteny blocks for the same species. Numbers of synteny blocks for respectively two, three and four organisms are shown in tables 4.4, 4.5 and 4.6. A total of 640 synteny blocks are defined for the set of the 5 genomes (without concatenation).

genomes	unrefined	longest	shortest
Ergo-Klth	3659	3887	4629
Ergo-Sakl	3383	3615	4288
Ergo-Zyro	3578	3792	4353
Klla-Ergo	3159	3333	3856
Klla-Klth	3221	3407	3974
Klla-Sakl	3028	3202	3716
Klla-Zyro	2926	3107	3537
Sakl-Klth	3152	3376	3961
Zyro-Klth	3313	3567	4116
Zyro-Sakl	3249	3508	4044

Table 4.1: 2-level anchors on Hemiascomycete yeasts obtained by GRIMM-Synteny

genomes	unrefined	longest	shortest
Ergo-Sakl-Klth	174	184	440
Ergo-Zyro-Klth	214	202	441
Ergo-Zyro-Sakl	104	103	262
Klla-Ergo-Klth	320	181	353
Klla-Ergo-Sakl	348	211	387
Klla-Ergo-Zyro	314	162	345
Klla-Sakl-Klth	47	82	156
Klla-Zyro-Klth	112	87	170
Klla-Zyro-Sakl	98	124	296
Zyro-Sakl-Klth	89	187	247

Table 4.2: 3-level anchors on Hemiascomycete yeasts obtained by GRIMM-Synteny

genomes	unrefined	longest	shortest
Ergo-Zyro-Sakl-Klth	0	0	14
Klla-Ergo-Sakl-Klth	4	4	20
Klla-Ergo-Zyro-Klth	17	3	21
Klla-Ergo-Zyro-Sakl	11	1	24
Klla-Zyro-Sakl-Klth	1	3	12

Table 4.3: 4-level anchors on Hemiascomycete yeasts obtained by GRIMM-Synteny

genomes	number
Ergo-Klth	278
Ergo-Sakl	248
Ergo-Zyro	338
Klla-Ergo	384
Klla-Klth	328
Klla-Sakl	303
Klla-Zyro	381
Sakl-Klth	93
Zyro-Klth	247
Zyro-Sakl	199

Table 4.4: 2-level synteny blocks on Hemiascomycete yeasts obtained by SyDiG

4.1.2 Mammal results

Results for GRIMM-Synteny are available on the webpage "Human-mouse-rat alignments" (by Glenn Tesler, the 16th of March 2004) [Tes04]. In order to run SyDiG on the mammalian genome data, an approximation of protein families is required. We have considered two different sets:

- the Ensembl mcl clustering results (pairwise homology relationships and gene ordered lists) [HAB+07],

- the UniRef50 clusters (pairwise homology relationships and gene ordered lists) [Con08].

genomes	number
Ergo-Sakl-Klth	324
Ergo-Zyro-Klth	439
Ergo-Zyro-Sakl	405
Klla-Ergo-Klth	490
Klla-Ergo-Sakl	484
Klla-Ergo-Zyro	554
Klla-Sakl-Klth	386
Klla-Zyro-Klth	480
Klla-Zyro-Sakl	472
Zyro-Sakl-Klth	284

Table 4.5: 3-level synteny blocks on Hemiascomycete yeasts obtained by SyDiG

genomes	number
Ergo-Zyro-Sakl-Klth	465
Klla-Ergo-Sakl-Klth	542
Klla-Ergo-Zyro-Klth	619
Klla-Ergo-Zyro-Sakl	604
Klla-Zyro-Sakl-Klth	526

Table 4.6: 4-level synteny blocks on Hemiascomycete yeasts obtained by SyDiG

To run the AdHoRe [VSS$^+$02] routine on our data, we explored different sets of i-AdHoRe parameters. To be concordant with results obtained in [BP02], we have chosen a gap size of 15, a cluster gap size of 20 and 9 as minimum number of anchor points. The number of synteny blocks obtained by the SyDiG algorithm is shown in table 4.7 for each set of data.

genomes	Ensembl	UniRef50
Human-Mouse	144	380
Human-Rat	137	215
Mouse-Rat	147	244
Human-Mouse-Rat	230	465

Table 4.7: Synteny blocks on mammals obtained by SyDiG algorithm

4.1.3 Discussion

The number of synteny blocks obtained by the two studied methods concerning mammalian genomes is quite similar. However, the number of anchors for yeast genomes obtained by GRIMM-synteny is low comparing to the number of alignments and moreover the signal within genomes is lost bit-by-bit when the number of considered genomes increases (no anchor for the 5 species for example).

The main issue comes down to the observation that homologous genes correspond neither to DNA alignments, nor to anchors of level 2. Indeed, two anchors of level 2 cannot consist of the same nucleotide sequences from the same genome. Quite to the contrary, one gene from one

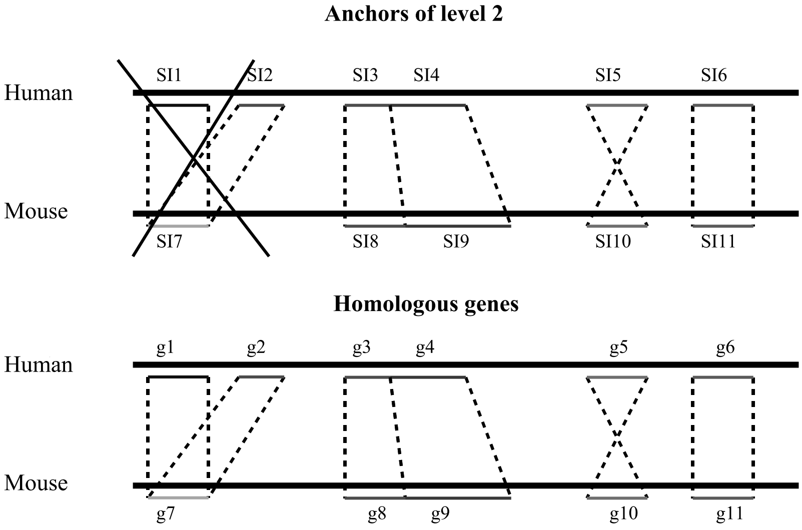


Figure 4.1: Difference between anchors and homologous genes. We have gene homologies between $(g_1, g_7), (g_2, g_7), (g_3, g_8), (g_4, g_9), (g_5, g_{10}), (g_6, g_{11})$. However, (SI_1, SI_7) and (SI_2, SI_7) can not correspond to any anchor since SI_7 is common to two couples. Other couples $(SI_3, SI_8), (SI_4, SI_9), (SI_5, SI_{10}), (SI_6, SI_{11})$ represent anchors.

genome G_i can be homologous to 2 (or indeed many more) genes in another genome G_j (see figure 4.1).

Analysis of these results shows that for mammalian genomes SyDiG performs as well as Grimm-Synteny. While two data sets (UniRef50 and Ensembl) generate slightly different results, they are both comparable (for appropriately-chosen i-ADHoRe parameters) with the results published in [BPT04].

On the other hand, when dealing with distant species such as yeasts, GRIMM-Synteny performs quite poorly. The only way to coax out a signal was to perform quite strong alignment pre-filtering of the TBLASTX results.

A particularly acute problem is that the GRIMM-Synteny procedure discards n-ary homologies. Not only do these paralogous families contain biologically pertinent information, they are often the best candidates for conserved markers between genomes: in the yeasts, for example, half of the genes conserved between species are members of paralogous families of up to 30 members, and discarding these homologies can lead to drastic under-identification of chromosomal homology.

4.2 Application to yeast genomes

We have applied the SyDiG algorithm in the context of the Génolevures project [DS$^+$04] for the case of non-WGD Hemiascomycetous yeasts. The data consists in 5 completely sequenced yeasts from the *Saccharomycetacae* clades: *Kluyveromyces lactis* (Klla), *Saccharomyces kluyveri* (Sakl), *Zygosaccharomyces rouxii* (Zyro), *Ashbya (Eremothecium) gossypii* (Ergo) and *Kluyveromyces thermotolerans* (Klth). These genomes have little genome redundancy and a relatively high (for yeasts) conservation of synteny.

From orthology and synteny relations identified using Génolevures protein families [NS07], the SyDiG algorithm obtains 487 synteny blocks for these genomes (mean size 51 genes). These syntenic blocks contain 8–200 genes (mean size 14 genes) and cover roughly 60% of each genome. Basing these permutations only on protein-coding genes is sufficient, since yeast genomes are highly compact (protein-coding genes cover approximately 80% of the genome), and gene relics are quite rare (approximately 4%) [Duj06]. By combining pairwise syntenies, each genome was factored into a sequence of ordered syntenic blocks, from which a set of distinct blocks common to all genomes was determined. An arbitrary reference genome was chosen, and all the blocks forming this genome were numbered by unique sequential identifiers from 1 to n. By keeping the longest blocks, permutations of 120 identifiers are constructed, that are representative of the pairwise evolutionary distances for these genomes. We are able to place active and inactive centromeres in each genome permutation by locating the flanking genes. Each of 9 centromeres is encoded by two identifiers, resulting in 15 additional blocks. Thus, each genome is represented as a signed permutation of 135 elements, in which chromosomal rearrangements (fusion, fission, translocation, inversion) can be studied (see chapter 6 for an application).

Comparative genome maps are painted (see figure 4.2) with *K. thermotolerans* as reference. Active centromeres are represented by red ovals, telomeres are represented by triangles. The assigned letter indicates the agreement of this centromere across the five species. Markers are well distributed on the chromosomes, so the choice of these markers is representative of the architecture of the contemporary genomes. A high degree of synteny, and a limited number of large-scale rearrangements, is observed between *K. thermotolerans* and *S. Kluyveri*; they share many common adjacencies and their rearrangement distance is half of that seen between other pairs of genomes. Note that *K. lactis* presents two syntenic breaks in centromere areas: the centromere of *Klla0F* is located between the flanking genes of centromeres h and b, and the centromere of *Klla0A* is located between the flanking genes of centromeres h and e. Moreover, *S. kluyveri* has an active centromere (the centromere i), that was disabled in all the other studied genomes.

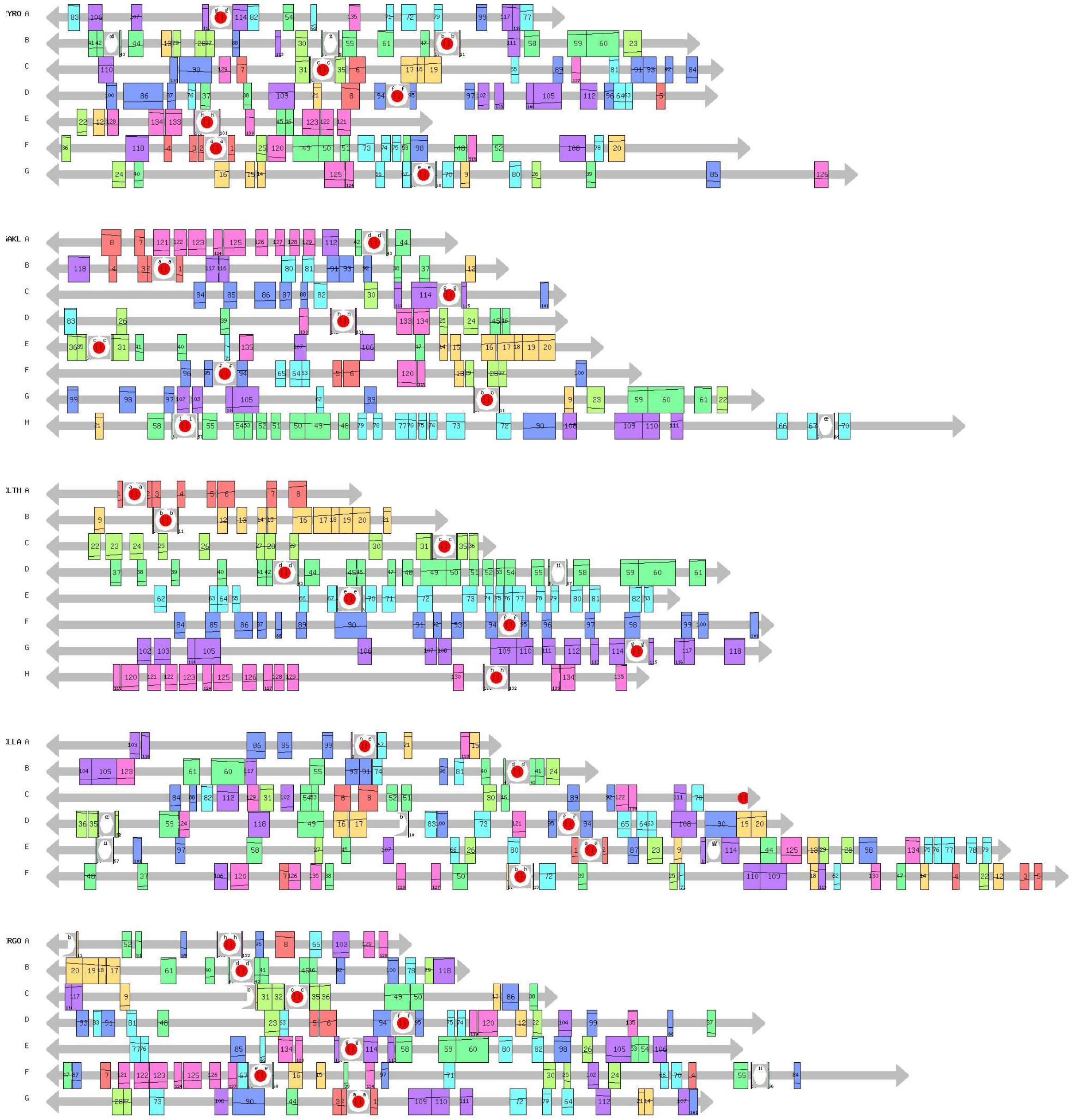


Figure 4.2: Distribution of the 120 longest common synteny blocks representing major conserved segments within *Hemiascomycete yeasts*. Each unique numbered synteny block is given a color indicating its chromosome in the reference genome (*Klth*), and a diagonal bar indicating its relative position on the chromosome. Other genomes are signed permutations of these colored blocks; a change of slope in the diagonal bar indicates an inversion. Block widths are to scale and the size of interleaving non-syntenic regions is shown by large grey lines. Red circles: centromeres; gray triangles: telomeres.

Part III

From super-blocks to constrained median assemblies

Chapter 5

Super-block construction

The study of evolutionary mechanisms is made more and more accurate by the increase in the number of fully sequenced genomes. One of the main problems is to reconstruct plausible ancestral genome architectures based on the comparison of contemporary genomes.

In chapter 2, we presented current methods that have largely focused on finding complete architectures for ancestral genomes, and, due to the computational difficulty of the problem, stop after a small number of equivalent minimal solutions have been found. Recent results suggest, however, that the set of minimum complete architectures is very large and heterogeneous [Eri07]. In fact these solutions are collections of conserved blocks, freely rearranged.

In this chapter, we propose an approach for identifying common ancestral features for the general, N-genome instance, that builds a bridge between breakpoint and rearrangement methods and additionally permits the use of biological constraints. The main contribution is the computation of *super-blocks*, sequences of markers chosen in function of the frequency of the corresponding adjacencies without any use of phylogeny. Here we follow the hypothesis that adjacencies having support in two or more contemporary genomes constitute the semantic basis of an ancestral architecture [SB97]. Super-blocks can of course be joined to produce final assemblies; algorithmically, it is an optimization problem in terms of rearrangement distance of the sequence of fusions of super-blocks. The solution space of genome medians is thus reduced, and only architectures respecting the adjacency semantics are returned. Although the mathematical model does not allow the consideration of segmental duplication, centromere positions are introduced and constrain the final assemblies by allowing only one active centromere in each chromosome of the ancestral architecture.

We show that in theory our method allows for solutions that are either minimal or reasonably close to the minimal in the mathematical model. Although the addition of biological constraints can in principle lead to non-optimal mathematical solutions, practically this does not occur and the key advantage of our method is that it decreases the number of mathematically equivalent solutions by using biological constraints as a filter on the solution space.

This chapter is organized as follows. Section 5.1 gives the necessary preliminaries. In section 5.2, we introduce the notion of dependency for the adjacencies and show the relationship between adjacencies and distances. Section 5.3 provides the methodology for reconstruction of super-blocks from adjacencies, and the strategy for building final assemblies by an optimal sequence of fusions. All this work is under revision in [JSN].

5.1 Preliminaries

Let $\Pi = \{\pi^1, .., \pi^{N_\Pi}\}$ and $\Gamma = \{\gamma^1, ..., \gamma^{N_\Gamma}\}$ be two multichromosomal genomes defined according to the mathematical model presented in section 1.2. As a reminder, the number of breakpoints b between two genomes is a distance such that for 2 multichromosomal genomes Π and Γ with $N_\Pi < N_\Gamma$, the number of breakpoints is $b = |\{(\pi_i, \pi_{i+1})|\pi_i.\pi_{i+1}$ is a breakpoint in $\Pi\}| + (N_\Gamma - N_\Pi)$ or $b = |\{(\gamma_i, \gamma_{i+1})|\gamma_i.\gamma_{i+1}$ is a breakpoint in $\Gamma\}|$.

Let $G_1, ..., G_N$ be N multichromosomal genomes defined over the same set of distinct gene markers $\mathcal{G}$. We denote by $u(g.h)$ the frequency of the adjacency $g.h$ in the N genomes, that is, the number of genomes in which it appears. We denote by $\mathcal{A}$ the set of all adjacencies in $G_1, .., G_N$.

Following Hannenhalli and Pevzner [HP95a], we will use the unsigned representation of a signed genome in terms of *breakpoint graph* (see section 2.2.2 page 31 for more details). The notions of adjacencies and breakpoints are transferred to the breakpoint graph quite naturally. As the choice of added vertices at the extremities of each chromosome is arbitrary, we denote by 0 any telomere without taking into the account its chromosome. Hence, for a chromosome $\pi = \pi_1 ... \pi_n$ we introduce two supplementary adjacencies denoted by $0.\pi_1$ and $\pi_n.0$. In what follows, we will systematically use greek letters to denote elements of a signed permutation and latin letters to denote elements of a non-signed permutation: we will note by $(g_i\, h_i).(g_j\, h_j)$ the adjacency corresponding to $\pi_i.\pi_j$ except for adjacencies with telomeres that will be noted $(0).(g_1\, h_1)$ and $(g_n\, h_n).(0)$. For any adjacency $a = \pi_i.\pi_j = (g_i\, h_i).(g_j\, h_j)$, its reversal $-a$ is defined by $-\pi_j. - \pi_i$ in the signed permutation, and by $(h_j\, g_j).(h_i\, g_i)$ in the non-signed permutation.

Example 3 *Let us consider four genomes* $G_1 = \{1\ 2\ 3\ 4,\ 5\ 6\}$, $G_2 = \{1\ 2\ 34, -5, -6\}$, $G_3 = \{2\ 1\ 3\ 4, -6\ 5\}$ *and* $G_4 = \{3\ 1\ 4\ 2\ -5,\ 6\}$. *Their adjacencies can then be partitioned according to frequency of occurrence in* G_i *as shown in table 5.1.*

frequency	adjacency
4	6.0
3	3.4, 0.5, 4.0
2	0.1, 1.2, 0.6, 5.0,2.3
1	5.6, 0.2, 2.1, 1.3, 4.2, 3.1, 1.4, 0.3, −5.6, 2. − 5

Table 5.1: Adjacencies for genomes G_1, G_2, G_3 and G_4 sorted by frequency.

5.2 Dependent adjacencies

The construction of *super-blocks* is based on the study of adjacencies. This study consists in defining the frequency of adjacencies in the genomes and the adjacency relationships themselves.

The intuition behind our approach is that an adjacency of higher frequency should be preferentially present in a median genome. Mathematically, we are looking for an ancestral architecture that represents a compromise between the rearrangement distance and the number of breakpoints under the parsimony criterion.

In what follows, the considered rearrangement distance is expressed in terms of reversals, fusions, fissions and translocations and is computed according to Hannenhalli and Pevzner's theory [HP95a] (see section 2.2.2).

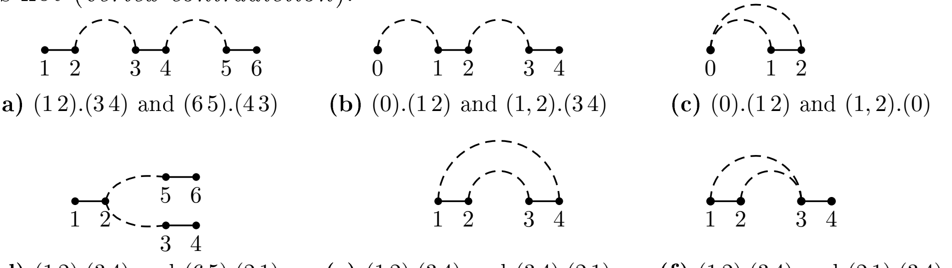
5.2.1 Pairwise adjacency relationships

Let A be a subset of the set of all adjacencies $\mathcal{A}$ for genomes $G_1, ..., G_N$. We build the *adjacency graph* $G = (V, E)$ for A in the following way. For any adjacency $(g_i\, h_i).(g_j\, h_j)$, we create four vertices (g_i, h_i, g_j and h_j) and three edges. Two of the edges represent elements of the original permutation: $e_1 = (g_i, h_i)$ and $e_2 = (g_j, h_j)$. One of the edges represents the adjacency itself: $e_3 = (h_i, g_j)$.

Two adjacencies are *dependent* if their elements are related, either by completing or by contradicting each other. Let a and b be two adjacencies $a = (g_1^a\, h_1^a).(g_2^a\, h_2^a)$ and $b = (g_1^b\, h_1^b).(g_2^b\, h_2^b)$, and $G = (V, E)$ the adjacency graph for $\{a, b\}$.

Definition 42 *We say that a and b* complement each other *if either (i) $\exists\, v_1, v_2 \in V$ such that $d(v_1) = d(v_2) = 1$ and $\forall v \neq v_i, i \in [1, 2]$ we have $v \neq 0$ and $d(v) = 2$, or (ii) $\exists v \in V$ such that $v = 0$ and $\forall v \in V$ we have $d(v) = 2$. We say that a and b* contradict each other *if either (i) $\exists\, v \in V$ such that $d(v) > 2$, or (ii) $\forall v \in V$ we have $v \neq 0$ and $d(v) = 2$.*

For example, adjacencies $(1\,2).(3\,4)$ and $(6\,5).(4\,3)$ complement each other. Indeed, we can form the sequence 1 2 3 4 5 6. On the contrary, $(1\,2).(3\,4)$ and $(6\,5).(2\,1)$ are in contradiction, as are $(1\,2).(3\,4)$ and $(2\,1).(3\,4)$. As can be seen on figure 5.1, the two contradictions are slightly different. Indeed, the latter involves the presence of a cycle (*cycle contradiction*), while the former does not (*vertex contradiction*).



(a) $(1\,2).(3\,4)$ and $(6\,5).(4\,3)$ **(b)** $(0).(1\,2)$ and $(1,2).(3\,4)$ **(c)** $(0).(1\,2)$ and $(1,2).(0)$

(d) $(1\,2).(3\,4)$ and $(6\,5).(2\,1)$ **(e)** $(1\,2).(3\,4)$ and $(3\,4).(2\,1)$ **(f)** $(1\,2).(3\,4)$ and $(2\,1).(3\,4)$

Figure 5.1: Adjacency graphs showing (a), (b) and (c) two adjacencies that complement each other, (d), (e) and (f) two adjacencies that contradict each other. Element edges are represented by solid lines; adjacency edges are represented by dashed lines.

When adjacencies complement each other there is no problem to put them together in order to form a coherent chromosome. However, when two adjacencies a and b are in contradiction, we need to choose one or the other. The intuition given in the beginning of this section is to prefer adjacencies with higher frequencies. However, it is possible to have a median genome in terms of rearrangement distances with an adjacency of lower frequency that is in contradiction with an adjacency of higher frequency as illustrated in the example 4. Notice that the adjacency 3.2 that is present in M_1 has frequency 2, while the adjacency 2.3 present in M_2 is of frequency 1. Because of a better global number of common adjacencies (11 breakpoints against 12 for M_2), M_1 appears as the best median genome in terms of rearrangement distances and breakpoint number but M_2 is also a good candidate for ancestral gene order in terms of rearrangement distances.

Example 4 *Consider three genomes $G_1 = \{1\,2\,3\,4\,5\,6\,7\}$, $G_2 = \{1\,3\,2\,4\,5,\ 6\,7\}$ and $G_3 = \{1\,4\,3\,2\,5\,6,\ 7\}$. Their pairwise rearrangement distances are: $d(G_1, G_2) = 3, d(G_1, G_3) = 5$ and $d(G_2, G_3) = 5$. Two optimal (median) solutions M_1 and M_2 are possible for these genomes: $M_1 = \{1\ \boldsymbol{-2}\ \boldsymbol{-3}\,4\,5,\ 6\,7\}$*

and $M_2 = \{1\,\boldsymbol{\mathit{-3}}\,\boldsymbol{\mathit{-2}}\,4\,5,\ 6\,7\}$*. The rearrangement distances from* M_1 *and* M_2 *to* G_1, G_2 *and* G_3 *are shown below.*

	G_1	G_2	G_3
M_1	2	1	4
M_2	1	2	4

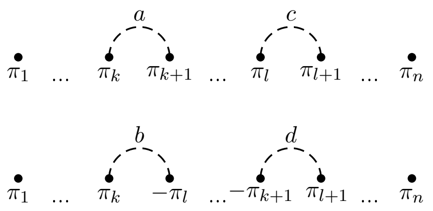
Notice that the adjacency 3.2 *that is present in* M_1 *has frequency* 2*, while the adjacency* 2.3 *present in* M_2 *is of frequency* 1.

5.2.2 Adjacencies and distances

Example 4 is in apparent contradiction with the intuition that the adjacencies of higher frequencies should be preferred. In this section, we analyze in more detail in which cases it is appropriate to follow this intuition.

Bounds for rearrangement distances

If two genomes Π and Γ are not equal, then $d(\Pi, \Gamma)$ is at least 1. If $d(\Pi, \Gamma) = 1$, then there are exactly two breakpoints in Π (say a and c), and two in Γ (say b and d). See figure 5.2 for illustration. We say then that Π and Γ are *identical up to* a, c (symmetrically b, d).



a
c
π_1 ... π_k π_{k+1} ... π_l π_{l+1} ... π_n
b
d
π_1 ... π_k $-\pi_l$ ... $-\pi_{k+1}$ π_{l+1} ... π_n

(a) breakpoints in Π

(b) breakpoints in Γ

Figure 5.2: Π and Γ are identical up to a, c (or b, d). This implies (a) the existence of 2 breakpoints $a = \pi_k.\pi_{k+1}$ and $c = \pi_l.\pi_{l+1}$ in Π, and (b) of 2 breakpoints $b = \pi_k. - \pi_l$ and $d = -\pi_{k+1}.\pi_{l+1}$ in Γ.

Lemma 6 *Let* Π_a, Π_b *and* Γ_a *be three genomes such that an adjacency* a *is present in genomes* Π_a *and* Γ_a*, but not present in* Π_b*. Furthermore, let* Π_a *and* Π_b *be identical up to 2 adjacencies, one of these adjacencies in* Π_a *being* a*, and one in* Π_b *being* b*. Then,* $|d(\Pi_a, \Gamma_a) - d(\Pi_b, \Gamma_a)| \leq 1$.

Proof: Let us denote the respective distances $d(\Pi_a, \Gamma_a) = d_a$ and $d(\Pi_b, \Gamma_a) = d_b$. We know that Π_a and Π_b are identical up to 2 adjacencies, hence $d(\Pi_a, \Pi_b) = 1$.

Rearrangement scenarios between genomes in Π_a, Π_b and Γ_a are represented on the sketch here below. Arrows represent scenarios and the value on them is the corresponding rearrangement distance.

$$\begin{tikzcd}
\Pi_a \arrow[dr, bend left, "d_a"] \arrow[dd, shift right, "1"'] & \\
& \Gamma_a \\
\Pi_b \arrow[ur, bend right, "d_b"'] \arrow[uu, shift right, "1"'] &
\end{tikzcd}$$

There exists a scenario between Π_a and Γ_a *via* Π_b (see the above sketch). Thus $d(\Pi_a, \Gamma_a) \leq d(\Pi_b, \Gamma_a) + 1$. Similarly, there exists a scenario between Π_b and Γ_a *via* Π_a. Thus $d(\Pi_b, \Gamma_a) \leq d(\Pi_a, \Gamma_a) + 1$. So $|d(\Pi_a, \Gamma_a) - d(\Pi_b, \Gamma_a)| \leq 1$. □

This lemma can be generalized to N genomes G_i, each having either the adjacency a, or b, or none. In theorem 6 we consider two genomes M_a and M_b identical up to two adjacencies, and we bound the difference of the sum of rearrangement distances between G_i and these two genomes.

Let $\mathcal{A}$ be the adjacency set of genomes $G_1, ..., G_N$, and let C be the set of all pairs of contradictory adjacencies from $\mathcal{A}$.

Theorem 6 *For any pair of adjacencies $\{a, b\} \in C$ and two genomes M_a and M_b identical up to 2 adjacencies with $a \in M_a$ and $b \in M_b$, it holds that*

$$\left| \sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \right| \leq N.$$

Proof: Let G_i^a (G_i^b and G_i^o, respectively) be the genomes having the adjacency a (b and none, respectively) with $|G_i^a| = N_a$ ($|G_i^b| = N_b$ and $|G_i^o| = N_o$, respectively). Since $N = N_a + N_b + N_o$, we have:

$$\begin{aligned}
\sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \quad = \quad & \sum_i^{N_a} d(M_a, G_i^a) - \sum_i^{N_a} d(M_b, G_i^a) + \\
& \sum_i^{N_b} d(M_a, G_i^b) - \sum_i^{N_b} d(M_b, G_i^b) + \\
& \sum_i^{N_o} d(M_a, G_i^o) - \sum_i^{N_o} d(M_b, G_i^o).
\end{aligned}$$

According to lemma 6 we have:

$$\begin{aligned}
\left| \sum_i^{N_a} d(M_a, G_i^a) - \sum_i^{N_a} d(M_b, G_i^a) \right| \quad & \leq \quad N_a, \text{ and} \\
\left| \sum_i^{N_b} d(M_b, G_i^b) - \sum_i^{N_b} d(M_a, G_i^b) \right| \quad & \leq \quad N_b.
\end{aligned}$$

In the case of genomes G_i^o, there exists a scenario between G_i^o and M_a *via* M_b, as shown on the sketch here below.

$$\begin{tikzcd}
M_a \arrow[dd, bend right, "1"'] & \\
& G_i^o \arrow[ul, bend right, "d(G_i^o, M_a)"'] \arrow[dl, bend left, "d(G_i^o, M_b)"] \\
M_b \arrow[uu, bend right, "1"'] &
\end{tikzcd}$$

So, $d(G_i^o, M_a) \leq d(G_i^o, M_b) + 1$. Similarly, there exists a scenario between G_i^o and M_b *via* M_a, and so $d(G_i^o, M_b) \leq d(G_i^o, M_a) + 1$.

As the distances are symmetric, we can apply the inequality:

$$|d(M_a, G_i^o) - d(M_b, G_i^o)| \leq 1 \; \forall i \in [1..N_o]$$

and thus

$$\left| \sum_i^N d(M_a, G_i^o) - \sum_i^N d(M_b, G_i^o) \right| \leq N_o$$

We conclude that

$$\left| \sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \right| \leq N$$

□

Types of rearrangements

Theorem 6 provides a general theoretical bound. Let us consider that $u(a) > u(b)$. The worst case difference $\sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \approx N$ is in fact rarely met. Lemma 7 below and its corollary analyze the problematic cases in terms of distances and breakpoints.

Sankoff and Trinh in [ST05] show that the rearrangement distance can be decomposed into different types of rearrangements according to the number of deleted breakpoints. The rearrangement distance d between two genomes can be written as $d = d_2 + d_1 + d_0$, where d_2, d_1 and d_0 are the numbers of rearrangements that delete two, one and no breakpoints, respectively. Moreover, this decomposition is unique. If b is the number of breakpoints between the two same genomes, then $b = 2d_2 + d_1$. Let b_a and b_b (d_a and d_b) be the number of breakpoints (rearrangement distances) between Π_a and Γ_a and between Π_b and Γ_a, respectively. We can decompose distances and breakpoints for Π_a and Π_b with respect to Γ_a:

$$\begin{aligned} b_a &= 2d_a^2 + d_a^1 \quad \text{and} \quad d_a = d_a^2 + d_a^1 + d_a^0, \\ b_b &= 2d_b^2 + d_b^1 \quad \text{and} \quad d_b = d_b^2 + d_b^1 + d_b^0. \end{aligned}$$

From these decompositions introduced by Sankoff and Trinh, we propose a more detailed analysis of no-breakpoint rearrangements. In the rest of this section we show that the number of no-breakpoint rearrangements d_b^0 can be bounded in terms of d_a^0.

Lemma 7 *Let Π_a, Π_b and Γ_a be three genomes as in lemma 6. Then, $d_b = d_a^2 + d_a^1 + d_b^0 + 1$ and $d_a^0 - 2 \leq d_b^0 \leq d_a^0$.*

Proof: Let a and c be the two breakpoints in Π_a with respect to Π_b. Two cases are possible:

1. Γ_a has the adjacency a but not c, then $b_b = b_a + 1$,

2. Γ_a has the adjacencies a and c, then $b_b = b_a + 2$.

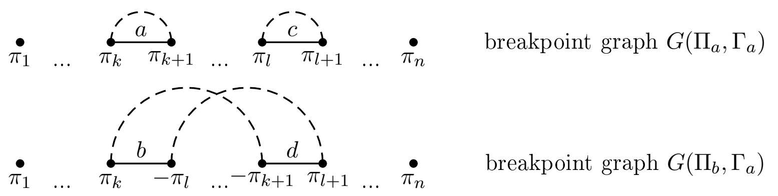
In the first case, we have $b_b = 2d_a^2 + d_a^1 + 1$ and $0 \leq |d_a - d_b| \leq 1$ (lemma 6). The distance equality $d_b = d_a^2 + (d_a^1 + 1) + d_b^0$ follows. And so, $d_a^0 - 2 \leq d_b^0 \leq d_a^0$.

In the second case, two sub-cases have to be considered:

(a) $b_b = 2(d_a^2 + 1) + d_a^1$, or

(b) $b_b = 2d_a^2 + d_a^1 + 2$.

Parts of the breakpoint graphs $G(\Pi_a, \Gamma_a)$ and $G(\Pi_b, \Gamma_a)$ are shown below for the case where Γ_a has adjacencies a and c. In $G(\Pi_b, \Gamma_a)$, the rearrangement defined by the edges $\{\pi_k, -\pi_l\}$ and $\{-\pi_{k+1}, \pi_{l+1}\}$ deletes two breakpoints.


a
c
π1 ... πk πk+1 ... πl πl+1 ... πn
breakpoint graph G(Πa, Γa)
b
d
π1 ... πk −πl ...−πk+1 πl+1 ... πn
breakpoint graph G(Πb, Γa)

We can easily see that only sub-case (a) is possible. So, we have $d_b = (d_a^2 + 1) + d_a^1 + d_b^0$, and so $d_a^0 - 2 \leq d_b^0 \leq d_a^0$. □

Corollary 1 *Let Π_a, Π_b and Γ_a be three genomes as in lemma 6. Then:*

1. *if $d_b = d_a + 1$, then $d_b^0 = d_a^0$,*
2. *if $d_b = d_a$, then $d_b^0 = d_a^0 - 1$,*
3. *if $d_b = d_a - 1$, then $d_b^0 = d_a^0 - 2$.*

Corollary 1 provides a range of possible situations under the hypothesis that $u(a) > u(b)$. Indeed, in the first case, the intuition of preferring a over b is valid. In the second case it still remains valid since for the same distance, we delete breakpoints. But in the third case, this intuition no longer holds.

In practice the third case is very infrequent, and Sankoff and Trinh even disregard it completely in [ST05]. Indeed, unoriented components needing no-breakpoint rearrangements are uncommon in breakpoint graphs. This situation for a pair of genomes implies that in order to obtain $\sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \approx N$, the difference $\sum_i^{N_a} d(M_a, G_i^a) - \sum_i^{N_a} d(M_b, G_i^a)$ has to be close to N_a, where N_a is the number of genomes having the adjacency a. According to corollary 1, $d(M_a, G_i^a) = d(M_b, G_i^a) + 1$ implies that $d_0(M_a, G_i^a) = d_0(M_b, G_i^a) + 2$. These rearrangements being infrequent, it is unlikely to have $d(M_a, G_i^a) = d(M_b, G_i^a) + 1$ and hence $\sum_i^N d(M_a, G_i) - \sum_i^N d(M_b, G_i) \approx N$.

Bounds for breakpoints

The result of theorem 6 can be transposed to breakpoints as shown in theorem 7. Let us denote N_a the number of genomes with adjacency a, N_b the number of genomes with adjacency b and N_o the number of genomes with neither adjacencies a nor b.

Theorem 7 *For any pair of adjacencies $\{a, b\} \in C$ such that $u(a) > u(b)$ and two genomes M_a and M_b identical up to 2 adjacencies with $a \in M_a$ and $b \in M_b$, it holds that*

$$N_a - 2N_b - N_o \leq \sum_i^N b(M_b, G_i) - \sum_i^N b(M_a, G_i) \leq 2N_a - N_b + N_o.$$

Proof: Let G_i^a (G_i^b and G_i^o, respectively) be the genomes having adjacency a (b and none, respectively) with $N = N_a + N_b + N_o$. We have:

$$\begin{aligned}\sum_i^N b(M_b, G_i) - \sum_i^N b(M_a, G_i) &= \sum_i^{N_a} b(M_b, G_i^a) - \sum_i^{N_a} b(M_a, G_i^a) + \\ &\quad \sum_i^{N_b} b(M_b, G_i^b) - \sum_i^{N_b} b(M_a, G_i^b) + \\ &\quad \sum_i^{N_o} b(M_b, G_i^o) - \sum_i^{N_o} b(M_a, G_i^o)\end{aligned}$$

We already know (see lemma 7) that

$$\begin{aligned}1 \leq b(M_b, G_i^a) - b(M_a, G_i^a) &\leq 2 \text{ for all } i \in [1..N_a], \text{ and} \\ -2 \leq b(M_b, G_i^b) - b(M_a, G_i^b) &\leq -1 \text{ for all } i \in [1..N_b].\end{aligned}$$

We know that since M_a and M_b are identical up to a (or b), there are two more adjacencies c and d that differ between these genomes (see figure 5.2). Consequently, any genome G_i^o can have either c, or d, or neither. Hence,

$$-1 \leq b(M_b, G_i^o) - b(M_a, G_i^o) \leq 1 \text{ for all } i \in [1..N_o].$$

And so we have:

$$N_a - 2N_b - N_o \leq \sum_i^N b(M_b, G_i) - \sum_i^N b(M_a, G_i) \leq 2N_a - N_b + N_o \quad \square$$

Theorem 7 provides a theoretical bound for the number of breakpoints. Let us consider again $u(a) > u(b) \geq 1$. In practice, the worst difference $\sum_i^N b(M_b, G_i) - \sum_i^N b(M_a, G_i) \approx N_a - 2N_b - N_o$ is not as bad as it seems.

For example, for 5 genomes the worst case $N_a - 2N_b - N_o$ is superior or equal to 0 for 3 of all possible 5 cases. If an adjacency a is present in all 5 genomes, then clearly $N_a - 2N_b - N_o = 5$. Table 5.2 shows possible values for N_a, N_b and N_o in the 4 remaining cases.

Nevertheless, for two cases the value $N_a - 2N_b - N_o$ is negative (but very close to 0). Fortunately, the worst case (case 4, table 5.2) never occurs. Indeed, recall that we are considering genomes M_a and M_b, identical up to a, c (symmetrically b, d). In this 4th case, the genomes G_i^a have adjacency a but not c, the genomes G_i^b have the adjacencies b and d while the N_o genomes which have neither a nor b, possess d. Hence, the frequency of adjacency d is 3, those of a and c are equal to 2, while b has frequency 1. Thus, we will have resolved the conflict between d and a or c which is an already studied case (case 2, table 5.2).

case #	N_a	N_b	N_o	$N_a - 2N_b - N_o$
1	4	1	0	2
2	3	2	0	-1
3	3	1	1	0
4	2	1	2	-2

Table 5.2: Given 5 genomes and contradictory adjacencies a and b s.t. $u(a) > u(b) \geq 1$, 4 possible cases of presence of a or b in current genomes arise. Given are the possible values for N_a,N_b and N_o, and the worst case difference of the total number of breakpoints.

5.3 From adjacencies to final assemblies

Section 5.2 implies that we can choose adjacencies with higher frequencies because they lead to a reasonable compromise between the breakpoint and the rearrangement distance approaches. Based on the adjacencies we propose to build *super-block assemblies* of median genomes.

The construction of super-blocks is done in two steps. First, we build a partition P of adjacencies where each part is composed of inter-dependent adjacencies. P is partially ordered by adjacency frequency of the parts' elements. Second, P is inspected in decreasing order of its parts, and the super-block sets are constructed by favoring adjacencies with higher frequency.

Finally, to find adjacencies not yet resolved, the last part of our method looks for a sequence of fusions of super-blocks that minimizes the rearrangement distances.

5.3.1 Groups of dependent adjacencies

We have seen previously that there exist different relationships between adjacencies. They can complement each other and, in this case, we can assemble them together in order to form a coherent chain of elements. When two adjacencies are in contradiction, then either there are different possibilities to complement the same element (*vertex contradiction*), or these two adjacencies have the same elements up to their order or to their orientation (*cycle contradiction*).

It is reasonable (see section 5.2) to prefer adjacencies with higher frequencies when there is a conflict. That is why, if a and b are two adjacencies in vertex contradiction, then we will have a preference for a if $u(a) > u(b)$, and need to consider both possibilities only if $u(a) = u(b)$. However, in the case of cycle contradiction, a and b are very similar because of the presence of the two same elements and example 4 shows that a median genome can have either one or the other. Hence, for a cycle contradiction, we relax the constraint and consider both adding a or adding b even for different frequencies.

Let $\mathcal{P}(A)$ be a partition of $\mathcal{A}$. We define $\mathcal{P}_0(\mathcal{A})$ by the membership in the same elementary cycle without 0 (that is a cycle containing 2 adjacencies). Parts of $\mathcal{P}_0(\mathcal{A})$ are either singletons or sets of adjacencies where every pair is in cycle contradiction. For a given set of adjacencies A, the highest frequency of its elements is denoted $u(A) = \max_{a \in A} u(a)$ and is called *set frequency.* We denote by G the adjacency graph containing all the adjacencies of $\mathcal{A}$.

We define the merging of parts $\sqcup : \mathcal{P}(\mathcal{A}) \rightarrow \mathcal{P}(\mathcal{A})$ as follows.

Definition 43 *$\sqcup(\mathcal{P}(\mathcal{A}))$ is a partition of $\mathcal{A}$ such that for any $p \in \sqcup(\mathcal{P}(\mathcal{A}))$*

- *$\exists\ p_1 \in \mathcal{P}(A)$ s.t. $p = p_1$ or*
- *$\exists\ p_1, p_2 \in \mathcal{P}(\mathcal{A})$ s.t. $p = p_1 \cup p_2$ and moreover $\exists\ a \in p_1$ and $\exists\ b \in p_2$ s.t. $u(a) = u(b) =$*

$u(p_1) = u(p_2)$ and either a and b are dependent or a and b participate in a cycle $c \in G$ without vertex $v = 0$ s.t. $\forall v \in c$ we have $u(v) \geq u(a)$.

Starting from $\mathcal{P}_0(\mathcal{A})$, the merging of parts $\sqcup$ defines a sequence of partitions $\mathcal{P}_i(\mathcal{A})$ where $\forall i > 0, \mathcal{P}_i(\mathcal{A}) = \sqcup(\mathcal{P}_{i-1}(\mathcal{A}))$. Obviously, there exists an n for which $\sqcup$ reaches its fixed point denoted by $\sqcup^n(\mathcal{P}(A))$, that is $\mathcal{P}_n(\mathcal{A}) = \sqcup(\mathcal{P}_n(\mathcal{A}))$.

Definition 44 *A* group *g is a part of $\sqcup^n(\mathcal{P}(A))$.*

Example 5 *The adjacencies of the example 3 are partitioned into groups as shown in table 5.3.*

grp. freq.	adjacencies
4	6.0(4)
3	3.4(3), 4.0(3)
3	0.5(3)
2	0.1(2), 1.2(2), 2.1(1), 2.3(2)
2	0.6(2)
2	5.0(2)

Table 5.3: Partition of adjacencies from example 3 into groups. The adjacencies are noted with their frequency in parenthesis, and the groups are sorted by decreasing group frequency. Only groups with $u(g) > 1$ are represented.

5.3.2 Super-blocks and partial assemblies

Definition 45 *A* super-block *is a set S of $n \geq 1$ adjacencies such that $\forall a, b \in S$, a does not contradict b, and there exists an order over S such that $\forall i \in [1, n)$, a_i complements a_{i+1}, and a_1, a_n are either independent or $a_1 = a_n = 0$. A* partial assembly *$\mathcal{P} = \{S_k\}$ is a set of super-blocks such that $\forall k, l$ with $k \neq l$ if $S_k \cap S_l \neq \emptyset \Rightarrow S_k \cap S_l = \{0\}$.*

Lemma 8 *The adjacency graph $G = (V, E)$ of a partial assembly $\mathcal{P}$ is a graph such that (1) $\forall v \in V$, $d(v) \leq 2$, except for $v = 0$, and (2) any cycle in G contains* 0.

Proof: By construction from definition 45. □

Super-blocks, and thus partial assemblies, are formed by going through the groups of adjacencies in decreasing order of their frequencies. For a given partial assembly $\mathcal{P} = \{S_k\}$ and a current group g, any adjacency $b \in g$ is removed from it if there exists an adjacency $a \in S_k \in \mathcal{P}$ in contradiction with b. This operation is called *clean* and produces a $g_c \subseteq g$, $g_c = clean(g, \mathcal{P})$. However, when inspecting the current group g_c we do not have any means to prefer some of its adjacencies over the others.

The addition of all the adjacencies of $g_c = clean(g, \mathcal{P})$ to $\mathcal{P}$ is not always possible. It is clear why this is the case for any a, b of g_c in mutual contradiction. Nevertheless, the addition of several adjacencies of g_c that do not contradict each other can reveal contradictions. This situation arises since the assembly of non-contradicting adjacencies of g_c can form a cycle or since dependencies between adjacencies belonging to groups of different frequency can exist (see figure 5.3).

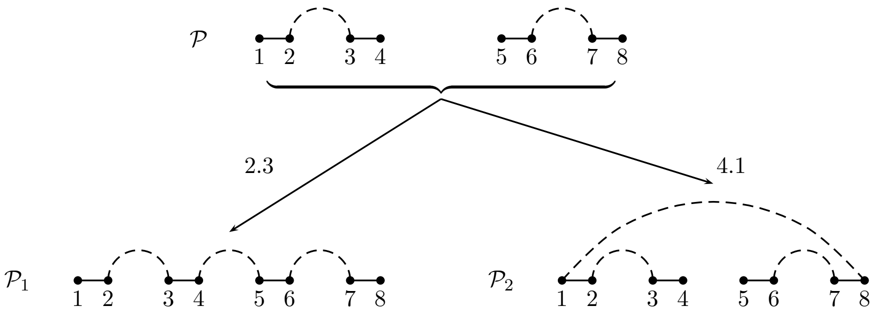


Figure 5.3: Two different adjacency graphs result from adding $g_c = \{2.3, 4.1\}$ to $\mathcal{P} = \{\{1.2\}, \{3.4\}\}$ depending on which adjacency between 2.3 and 4.1 is added. Adding both 2.3 and 4.1 creates a forbidden cycle.

The trivial way to exhaustively enumerate all the possibilities when adding g to $\mathcal{P}$, is to consider all possible orders over g_c, which is for a $|g_c| = n$ equal to $n!$. A less naive approach brings it down to a complexity of $O(2^{n/3})$ by considering maximal independent sets.

Definition 46 *A* maximal independent set of g_c *is a set of adjacencies μ such that (i) $\forall a, b \in \mu$, a and b do not contradict each other and (ii) $\forall a \in g_c \backslash \mu$, $\exists c \in \mu$ such that a and c are in contradiction.*

Let $\mathcal{M}$ be the set of all maximal independent sets for g_c.

Lemma 9 *For any maximal independent set $\mu \in \mathcal{M}$, its adjacency graph $G = (V, E)$ verifies $\forall v \in V$, $v \neq 0$, $d(v) \leq 2$.*

Proof: Suppose there exists a vertex $v \in V$ such that $v \neq 0$ and $d(v) \geq 3$. Let Y the neighbours of v in G with $|Y| = d(v)$. Then, there exists an unique vertex $y \in Y$ such that $(v\ y)$ is an element of the original permutations. Therefore, there exist at least two vertices y_1 and $y_2 \in Y$ such that $\{v, y_1\}$ and $\{v, y_2\}$ correspond to real adjacencies. So, these two adjacencies sharing v are in contradiction. This contradicts the fact that these two adjacencies belong to the same maximal independent set of g_c. Then, $\forall v \in V$, $d(v) \leq 2$, except for $v = 0$. □

Thus, we have to consider all maximal independent subsets of g_c. The problem is known to be NP-complete [GJ79] and of complexity $O(2^{n/3})$, where n is the number of elements in g_c [TT76].

Let $G_{\mathcal{P}}$ be the adjacency graph for $\mathcal{P}$ and let G_μ be the adjacency graph for a maximal independent set μ of $\mathcal{M}$. Let G^0_μ be a graph obtained by removing the vertex 0 and all of its incident edges from G_μ. Then, the connected components of G^0_μ can be either chains where all vertices have the degree equal to 2, except the two extremities which have the degree 1, or cycles where all vertices have the degree 2. Simply adding all the vertices and edges of G_μ to $G_{\mathcal{P}}$ may result in conflicts (see lemma 10). Let $G_\cup$ be the adjacency graph $G_\cup = \{V_{\mathcal{P}} \cup V_\mu, E_{\mathcal{P}} \cup E_\mu\}$.

Lemma 10 *The adjacency graph $G_\cup$ is a graph such that $\forall v \in V$, $v \neq 0$, $d(v) \leq 2$.*

Proof: Suppose there exists a vertex $v \in V$ such that $v \neq 0$ and $d(v) \geq 3$. Let Y the neighbours of v in G with $|Y| = d(v)$. Then, there exists an unique vertex $y \in Y$ such that $(v\ y)$ is an element of the original permutations. Therefore, there exist at least two vertices y_1 and $y_2 \in Y$ such that $\{v, y_1\}$ and $\{v, y_2\}$ correspond to real adjacencies that contradict each other and:

1. $\{v, y_1\}$ and $\{v, y_2\} \in G_{\mathcal{P}}$ or,

2. $\{v, y_1\}$ and $\{v, y_2\} \in G_\mu$ or,
3. $\{v, y_1\} \in G_\mu$ and $\{v, y_2\} \in G_\mathcal{P}$ or,
4. $\{v, y_1\} \in G_\mathcal{P}$ and $\{v, y_2\} \in G_\mu$.

Cases 1 and 2 are in contradiction with lemmas 8 and 9. For case 3 (case 4, respectively), the adjacency represented by the edge $\{v, y_1\}$ ($\{v, y_2\}$ respectively) was removed by *clean* from g to obtain g_c. So it is not possible to have these two cases. □
Forbidden cycles can appear in $G_\cup$. It is clear that G_μ can have cycles without 0. But cycles can also appear by closing chains of $G_\mathcal{P}$ by one or several chain(s) of G^0_μ. In order to obtain an adjacency graph of a partial assembly from $G_\cup$, we have to disconnect all existing cycles without 0 by deleting some adjacency from μ in each cycle (see lemma 11).

Let $C = \{c_1, c_2, .., c_m\}$ be a set of all cycles without 0 in $G_\cup$.

Lemma 11 *If $C \neq 0$, then $\forall i, j$ s.t. $i \neq j$, c_i and c_j are disjoint, and each cycle $c \in C$ has one or several adjacencies from μ.*

Proof: Let $c_i, c_j \in C$. Suppose that they are not disjoint. Then, there exists a vertex v such that $v \in c_i$ and $v \in c_j$. Thus, $d(v) \geq 3$, which contradicts lemma 10.

Let $c \in C$ such that for all adjacencies a of c, we have $a \notin \mu$. Then, $c \in G_\mathcal{P}$, which contradicts lemma 8. □
Let $\{G_\prec\}$ be the set of all graphs resulting from adding g_c to $\mathcal{P}$ for all possible orders $\prec$ over g_c. Let μ_j be the set of adjacencies from a maximal independent set μ that participate in a cycle $c_j \in C$. We denote by $S_\mu = \mu_1 \times \mu_2 \times ... \times \mu_m$ the Cartesian product of the sets of adjacencies from a maximal independent set μ participating in cycles $\{c_1, .., c_m\} \subset C$.

Lemma 12 *The following equality holds:*

$$\{G_\prec\} = \bigcup_{\mu \in \mathcal{M}} \bigcup_{\vec{a} \in S_\mu} \{G_\cup \setminus \{a_i\}\}$$

where $\vec{a}$ is composed of $\{a_i\}$ and $|\vec{a}| = m$, and $G_\cup \setminus \{a_i\}$ denotes the graph $G_\cup$ without the edges $\{a_i\}$.

Proof: Let us denote the right side of the equation by $\{G_\mathcal{M}\}$. The inclusion $\{G_\mathcal{M}\} \subseteq \{G_\prec\}$ is obvious. Let us suppose $\exists\, G \in \{G_\mathcal{M}\}$ such that $G \notin \{G_\prec\}$. This means that for some particular μ and $\vec{a}$ we have $\langle G_\cup \rangle \setminus \{a_1, a_2, .., a_m\} \notin \{G_\prec\}$. Which contradicts the fact that μ is maximal. □

Let us denote the operation of adding a group g to $\mathcal{P}$ by $\oplus$. This operation produces all possible partial assemblies $\{\mathcal{P}_i\} = \mathcal{P} \oplus g$ and can be realized by the algorithm *add_group* (algorithm 7). The complexity of this algorithm is bounded by the research of maximal independent sets over g_c.

Lemma 13 *Let $\mathcal{P}$ be a partial assembly and let g^1 and g^2 be two groups of same frequency $u(g^1) = u(g^2)$. Then, $\oplus$ is associative: $(\mathcal{P} \oplus g^1) \oplus g^2 = (\mathcal{P} \oplus g^2) \oplus g^1$.*

Proof: Suppose $\oplus$ to be not associative. Then, there exists an adjacency a in g^1 and b in g^2 such that a and b imply a contradiction in the constructed partial assembly $\mathcal{P} \oplus g^1 \oplus g^2$. Then, either:

Algorithm 7 add_group($g, \mathcal{P}$)

```
Require: a group $g$, a partial assembly $\mathcal{P}$
Ensure: $\mathbb{P}$ is a set of partial assemblies
let $G_{\mathcal{P}}$ be the adjacency graph for $\mathcal{P}$
let $\mathbb{P} = \emptyset$ and $g_c = clean(g, \mathcal{P})$
let $\mathcal{M}$ be the set of all maximal independent sets over $g_c$
for all $\mu \in \mathcal{M}$ do
    let $G_\mu$ be the adjacency graph for $\mu$
    let $T$ be the set of all connected components of $G_\mu^0$
    let $G_{new} = \{V_{\mathcal{P}} \cup V_\mu, E_{\mathcal{P}} \cup E_\mu\}$
    let $C = \emptyset$
    while $G_{new}^0$ has a cycle $c$ do
        let $V = \emptyset$ be the set of adjacencies from $\mu$ participating in $c$
        for all $t \in T$ do
            if $t \cap c \neq \emptyset$ then
                let $V = V \cup adjacencies(t)$
                let $G_{new} = \{V_{new} \backslash \{t[0]\}, E_{new}\}$
            end if
        end for
        let $C = C \cup \{V\}$
    end while
    let $G = \{V_{\mathcal{P}} \cup V_\mu, E_{\mathcal{P}} \cup E_\mu\}$
    let $\mathbb{G}_\mu = \{G\}$
    for all $c \in C$ do
        $\mathbb{G} = \emptyset$
        for all $a \in c$ do
            for all $G_\mu \in \mathbb{G}_\mu$ do
                $\mathbb{G} = \mathbb{G} \cup \{\{V_\mu, E_\mu \backslash \{a\}\}\}$
            end for
        end for
        $\mathbb{G}_\mu = \mathbb{G}$
    end for
    let $\mathbb{P} = \mathbb{P} \cup partial\_assemblies(\mathbb{G}_\mu)$
end for
return $\mathbb{P}$
```

1. a and b form a vertex contradiction or,
2. a and b form a cycle contradiction or,
3. a and b participate in a cycle without 0.

In case 1, if $u(a) = u(b)$ then a and b should be in the same group (see definition 44). If $u(a) \neq u(b)$ then $u(g^1) \neq u(g^2)$, which contradicts the hypothesis. In case 2, if a and b form a cycle contradiction then a and b should be in the same group (see definition 44), which contradicts the hypothesis. In case 3, all the vertices in the cycle have a frequency greater than $u(g^1) = u(g^2)$. Therefore, according to the definition 44, a and b should be in the same group. So, $\oplus$ is associative. □

Based on lemmas 12 and 13, and algorithm 7, the construction of all partial assemblies for genomes $G_1, ..., G_N$ proceeds as shown in algorithm 8. Notice that we do not consider groups where $u(g) = 1$ since these adjacencies do not have any additional support in any other genome.

5.3.3 Fusions of super-blocks

Algorithm 8 builds all partial assemblies by resolving conflicts between adjacencies up to group frequency 2. Groups of frequency 1 are excluded since there is no evidence if they are present by chance or not.

Algorithm 8 partial_assemblies($G_1, ..., G_N$)

```
Require: $G_1, ..., G_N$ genomes over the same set of gene markers
Ensure: $\mathbb{P}$ is a set of partial assemblies
let $\mathcal{A}$ be the set of all adjacencies for $G_1, ..., G_N$
let $G = \{g\}$ be the set of all groups for $\mathcal{A}$
let $n = max_G u(g)$
let $\mathbb{P} = \{\emptyset\}$
for all $g_i$ s.t. $n \geq i \geq 2$ do
    let $\mathbb{P}' = \emptyset$
    for all $\mathcal{P} \in \mathbb{P}$ do
        $\mathbb{P}_{\mathcal{P}} = \mathcal{P} \oplus g_i$
        $\mathbb{P}' = \mathbb{P}' \cup \mathbb{P}_{\mathcal{P}}$
    end for
    $\mathbb{P} = \mathbb{P}'$
end for
return $\mathbb{P}$
```

Definition 47 *A* fusion *of super-blocks $S_1 = (a_1, ..., a_n)$ and $S_2 = (b_1, ..., b_m)$ is a super-block S such that the order of definition 45 is either $S = (a_1, ..., a_n,\ b_1, ..., b_m)$, or $S = (a_1, ..., a_n, -b_m, ..., -b_1)$, or $S = (b_1, ..., b_m, a_1, ..., b_n)$, or $S = (b_1, ..., b_m,\ -a_n, ..., -a_1)$.*

This definition implies that a super-block S such that $a_1 = 0.\pi_i$ and $a_n = \pi_j.0$ cannot participate in a fusion. Indeed, such a super-block already forms a chromosome from telomere to telomere.

Let $\{\mathcal{P}\}$ be the set of all partial assemblies up to group frequency 2 for genomes $G_1, ..., G_N$ and $\mathcal{P} \in \{\mathcal{P}\}$ a partial assembly. The number of super-blocks in $\mathcal{P}$ can be relatively high. This is due to the fact that some elements cannot be inter-connected because of the low frequency (equal to 1) of corresponding adjacencies. Such elements are located at the extremities of the super-blocks. We connect them in order to form chromosomes by fusions of super-blocks without worsening the distance and breakpoint bounds (see theorem 8).

Theorem 8 *For any $\mathcal{P} \in \{\mathcal{P}\}$ of $G_1, ..., G_N$ such that $\mathcal{P} = \{S_k\}$, there exists a genome M such that for any chromosome π of M either $\exists\, S_k \in \mathcal{P}$ such that $\pi = S_k$, or $\exists\, \{S_k\} \subseteq \mathcal{P}$ such that π is formed by a series of fusions $\pi = S_1 ... S_k$. Moreover,*

$$\textstyle\sum_i^N d(M, G_i) - \sum_i^N d(P, G_i) \leq 0 \text{ and } \sum_i^N b(M, G_i) - \sum_i^N b(P, G_i) \leq 0.$$

Proof: By construction □

To find an optimal sequence of fusions, we classify them by their effect on the global rearrangement distance (the sum of rearrangement distances between median genome and $G_1 \dots G_N$). A greedy randomized algorithm is used to find ancestral candidates obtained after a limited number of fusions. By the parsimony criterion, solutions that minimize the global rearrangement distance are conserved.

Chapter 6

Applications

In this chapter, we propose application examples for our method for super-block construction presented in chapter 5. In section 6.1, we apply our method to a set of *non-WGD*[1] *Hemiascomycete* genomes in the *Kluyveromyces* and related clades provided by the Génolevures Consortium, with divergence similar to that of chordates [Duj06]. For this phylogenetic branch, our method shows a high convergence in the structure of different versions of super-blocks (16 in all), reinforcing the intuition that super-blocks encode the semantics of the ancestral genome. We can thus perform a reconstruction, despite extensive map reshuffling. In section 6.2, we show the pertinence of our method on theoretical test cases and comparisons to existing methods. Finally, section 6.3 provides a wider discussion about the super-block method. All this work is under revision in [JSN].

6.1 A Median Genome for non-WGD yeasts

We have applied our method to analyze ancestral architectures for the Génolevures project [DS$^+$04] in the case of non-WGD Hemiascomycete yeasts. The data consists in 5 completely sequenced yeasts from the *Saccharomycetacae* clades: *Kluyveromyces lactis*, *Saccharomyces kluyveri*, *Zygosaccharomyces rouxii*, *Ashbya (Eremothecium) gossypii* and *Kluyveromyces thermotolerans*[2]. These genomes have little genome redundancy and a relatively high (for yeasts) conservation of synteny.

Signed permutations representing each genome were computed using the SyDiG algorithm (see part II, page 59), using pairwise syntenic blocks obtained by the i-ADHoRE method [SVSP04] from orthology and synteny relations identified using Génolevures protein families [NS07]. These syntenic blocks contain 8–200 genes (mean size 14 genes) and cover roughly 60% of each genome. Basing these permutations only on protein-coding genes is sufficient, since yeast genomes are highly compact (protein-coding genes cover approximately 80% of the genome), and gene relics are quite rare (approximately 4%) [Duj06]. By combining pairwise syntenies, each genome was factored into a sequence of ordered syntenic blocks, from which a set of distinct blocks common to all genomes was determined. An arbitrary reference genome was chosen, and all the blocks forming this genome were numbered by unique sequential identifiers from 1 to n.

The permutations computed by this *in silico* chromosomal painting contained 487 blocks (mean

[1] Whole-Genome Duplication, an unique polyploidization event proposed in the ancestral *Saccharomyces* lineage; non-WGD yeasts from the other branches of the phylogenetic tree are not affected by this catastrophic event.

[2] Abbreviations: Klla, *K. lactis*; Sakl, *S. kluyveri*; Zyro, *Z. rouxii*; Ergo, *A. gossypii*; Klth, *K. thermotolerans.*

size 51 genes); keeping the longest blocks brought the permutations to 120 identifiers[3] (mean size, 94 genes). We were able to place active and inactive centromeres in each genome permutation by locating the flanking genes (personal communication from Jacky de Montigny). Each of 9 contemporary centromeres was encoded by two successive identifiers, resulting in 15 additional blocks. Thus, each genome was represented as a signed permutation of 135 elements (see figure 6.1) , in which chromosomal rearrangements (fusion, fission, translocation, inversion) were studied. The pairwise rearrangement distances between these genomes are shown in table 6.1.

```
Klth:
1a {} a2 3 4 5 6 7 8 $
9 10b {} b11 12 13 14 15 16 17 18 19 20 21 $
22 23 24 25 26 27 28 29 30 31 32 33c {} c34 35 36 $
37 38 39 40 41 42d {} d43 44 45 46 47 48 49 50 51 52 53 54 55 56i i57 58 59 60 61 $
62 63 64 65 66 67 68e {} e69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 $
84 85 86 87 88 89 90 91 92 93 94f {} f95 96 97 98 99 100 101 $
102 103 104 105 106 107 108 109 110 111 112 113 114g {} g115 116 117 118 $
119 120 121 122 123 124 125 126 127 128 129 130 131h {} h132 133 134 135 $
Ergo:
b11 -52 -51 -89 131h {} h132 96 -8 -65 103 -129 -128 $
-20 -19 -18 -17 61 40 -43d {} d-42 -41 45 46 92 100 -78 -29 118 $
116 117 9 31 32 33c {} c34 35 36 b-10 49 50 13 86 -38 $
93 83 91 -81 -48 23 -63 5 6 94f {} f95 -75 -74 119 120 12 -22 -104 99 135 -88 -37 $
-77 -76 85 62 -134 -133 -115g {} g-114 -113 58 59 60 80 82 -98 -26 -105 53 54 106 $
47 87 -7 121 122 123 124 125 126 127 67 68e {} e69 -16 -15 39 130 97 71 -30 -25 102 -24 66 70 4 55 56i i57 -84 $
-28 -27 -73 -108 -90 -44 -3 -2a {} a-1 109 110 111 72 79 64 -112 21 14 -107 101 $
Klla:
-103 -116 -86 85 -99 131h {} 68e e69 -67 21 133 -15 $
104 105 123 61 -60 117 -55 93 91 -74 -96 -81 40 -43d {} d-42 -41 -24 $
84 88 -82 -112 -129 31 102 -54 -53 -6 -8 -52 -51 30 -46 89 92 122 -119 -111 -70 {}$
-36 -35 -34c c-33 -32 -59 124 118 49 16 17 b-10 -83 -100 -73 121 -95f {} f -94 -65 -64 -63 -108 -90 19 20 $
-57i i-56 -101 -97 -58 27 -45 107 -66 26 -80 1a {} a2 -87 23 9 -115g g-114 -44 -125 13 -29 -28 98 -134 75 76 77 78 79 $
48 37 -106 120 -7 126 -135 38 -128 127 -50 b11 {} h-132 72 39 25 71 -110 -109 -18 -113 62 -130 47 14 -4 22 -12 3 -5 $
Sakl:
-8 -7 121 122 123 124 125 126 127 128 129 112 42d {} d43 44 $
-118 -4 -3 -2a {} a-1 -117 -116 80 81 -91 -93 -92 -38 -37 12 $
84 85 86 87 88 -82 30 113 114g {} g115 101 $
-83 26 39 130 -132h {} h-131 133 134 -25 -24 45 46 $
-36 -35 -34c {} c-33 -32 -31 -41 -40 -71 135 -107 -106 47 14 15 16 17 18 19 20 $
-96 -95f {} f-94 -65 -64 -63 5 6 -120 -119 13 -29 -28 -27 100 $
-99 -98 -97 102 103 104 105 62 89 10b {} b11 -9 -23 59 60 -61 -22 $
-21 -58 -57i {} i-56 -55 -54 -53 -52 -51 -50 -49 -48 -79 -78 -77 -76 -75 -74 -73 -72 90 108 109 110 111 66 67 68e e69 70 $
Zyro:
-83 106 107 -115g {} g-114 82 -54 62 -135 -71 72 -79 99 -117 -116 77 $
41 42d d43 44 13 -29 -28 -27 -88 -113 -30 -57i i-56 -55 -61 47 10b {} b11 -111 58 59 60 23 $
110 101 -90 129 -7 31 32 -34c {} c-33 35 6 17 18 19 -65 -89 -127 81 -91 -93 -92 -84 $
-100 86 87 -76 -37 38 109 -21 -8 94f {} f95 -97 102 103 104 105 -112 96 -64 -63 5 $
22 -12 128 -134 -133 -132h {} h-131 -130 45 46 -123 -122 -121 $
-36 -118 -4 -3 -2a {} a-1 -25 -120 49 50 51 73 74 75 53 -98 48 -119 52 -108 78 20 $
-24 40 -16 -15 -14 -125 -124 66 67 -69e {} e-68 70 9 80 -26 -39 -85 -126 $
```

Figure 6.1: Signed permutations on 135 elements for contemporary non-WGD Hemiascomycete yeasts (*Zyro*, *Ergo*, *Klla*, *Klth* and *Sakl*). *Klth* is taken as reference for the numbering. The character $ represent the end of a chromosome. The positions of the active centromeres are located by two embraces. A letter indicates the agreement of the flanking genes of a centromere across the five species.

Comparative genome maps were painted (see figure 6.2) with *K. thermotolerans* as reference. Active centromeres are represented by red ovals, telomeres are represented by triangles. The assigned letter indicates the agreement of this centromere across the five species. Markers are well distributed on the chromosomes, so the choice of these markers is representative of the architecture of the contemporary genomes. A high degree of synteny, and a limited number of large-scale rearrangements, is observed between *K. thermotolerans* and *S. Kluyveri*; they share many common adjacencies and their rearrangement distance is half of that seen between other pairs of genomes. Note that *K. lactis* presents two syntenic breaks in centromere areas: the

[3] The number of retained markers does not allow one to obtain an ancestral permutation candidate by using the public version of MGR.

	Zyro	Klth	Sakl	Klla	Ergo
Zyro	0	84	79	115	101
Klth		0	45	105	88
Sakl			0	98	85
Klla				0	109
Ergo					0

Table 6.1: Pairwise rearrangement distances between non-WGD Hemiascomycete genomes as calculated from common synteny blocks representing 135 major conserved segments. For abbreviations, see footnote on page 93.

centromere of *Klla0F* is located between the flanking genes of centromeres h and b, and the centromere of *Klla0A* is located between the flanking genes of centromeres h and e. Moreover, *S. kluyveri* has an active centromere (the centromere i), that was disabled in all the other studied genomes.

We computed 16 sets of super-blocks, each containing either 34 or 35 super-blocks. These super-block sets are highly similar. Indeed, 29 super-blocks are common among all of the sets, and there are only 4 conflicts (see figure 6.3). A given partial assembly of super-blocks $\mathcal{P}$ represents a potential structure of an ancestral architecture. Finally, it is possible to construct a final assembly from these super-blocks by successive fusions. Two sets of assemblies were computed: with and without the constraint on centromere position. For both of these cases 90 final assembly candidates were generated. In the first case the global sum of distances $\sum(M, G_i)$ varies between 281 and 285 (283,4 on average); in the second case it varies between 281 and 283 (282,2 on average). The latter represents biologically plausible architectures whose rearrangement distances are close to minimal. The whole set of solutions shows a high convergence in terms of rearrangement distances, reinforcing the intuition that the computation of ancestral architectures by super-blocks assembly results in a reduced neighborhood in the search space. Further filtering of the results was done by a plausibility metric p based on the chromosomal structure of the candidate solution (distributions of chromosome sizes and of centromere locations on the chromosome). Figure 6.2 shows the candidate for ancestral architecture which has the best compromise between a maximal value for p and minimal value for $\sum(M, G_i) = 284$.

6.2 Comparison to MGR

We compare our super-block algorithm to the software MGR-MEDIAN [BP02] developed to reconstruct ancestral gene orders according to rearrangement distance. MGR is not publicly available software, so we could only make comparisons to publicly available results, or to results that can be computed using the MGR demonstration web site[4]. This web site handles small instances; although it is not formally stated on the MGR webpage, it seems that this public version is limited to genomes of at most 30 markers.

6.2.1 Human, Cat, Mouse Instances

MGR constructs median genomes for the three-genome case only; if more are provided it computes the rearrangement tree. For this reason, we used the only available multi-chromosomal

[4] `http://nbcr.sdsc.edu/GRIMM/mgr.cgi`

data from the MGR webpage: that of human, cat and mouse (ancestral permutation available on-line). This dataset has 114 markers [BP02]. For this dataset, we obtain two versions $\mathcal{P}_1$ and $\mathcal{P}_2$ of 32 super-blocks that differ only in one super-block. The ancestral permutation obtained by MGR contains all of the super-blocks of one of the two sets $\mathcal{P}_2$ (see figures 6.4 and 6.5).

6.2.2 Simulated instances

In order to estimate the conservation of super-blocks, we generate simulated instances, where the distances between genomes are bounded. An arbitrary ancestral genome is generated from which a specified number of random rearrangements are applied to give three genomes. We specify the number of genes (n) and chromosomes (N), and the number of rearrangements done during the simulation (r); this parameter is an upper bound on the optimal median genome score. We generated 300 instances with parameters $n = 30$, N between 1 and 5, and $r = 50$. For all of these instances, we computed the sets of super-blocks, the median genome obtained by the public version of MGR, and the possible assemblies into median genomes.

The number of sets of super-blocks varies between 1 and 4, and the number of nontrivial super-blocks in a set varies between 2 and 10. This small number of partial assemblies and nontrivial super-blocks is due to the small number of identifiers with only 3 genomes. MGR does not provide an ancestral permutation for 60 of the 300 instances. For the 240 remaining instances, the median genome proposed by MGR conserves the totality of the super-blocks except for one instance (figure 6.6). For this instance, we find one partial assembly decomposed into 7 nontrivial super-blocks. The median solution A_MGR recovered by MGR has a global rearrangement distance of 41 and contains 6 of the 7 super-blocks. The super-block 23 24 25 is missing in A_MGR although it has support in two of the three genomes of the instance (G_2 and G_3). Nevertheless, it is possible to obtain better solutions in terms of super-block conservation, that present moreover a better global rearrangement score. Super-block assemblies return 10 different solutions that are equivalent in terms of global rearrangement distance and better than the one found by MGR (39 against 41). Moreover, all of these solutions contain the 7 super-blocks of the partial assembly.

The super-block fusion procedure generates medians that are competitive from the rearrangement distance point of view. Moreover, our method provides candidates that have better breakpoint characteristics than those obtained by MGR. Example 4 page 81 shows 3 genomes G_1, G_2 and G_3. For this dataset, we obtain two partial assemblies of super-blocks that lead to two optimal solutions M_1 and M_2 in terms of rearrangement distance. However, they have a different global number of breakpoints: 11 breakpoints for M_1 against 12 for M_2 (see example 6). Under a parsimonious criterion, M_1 appears as the best ancestral candidate for G_1, G_2 and G_3. MGR gives M_2 as ancestral gene order for this dataset. For the human, mouse and cat genomes, the fusion procedure provides the same result in terms of rearrangements, and a better compromise in terms of breakpoints (see figure 6.5).

Example 6 *We consider the three genomes from example 4. Super-blocks algorithm leads to two partial assemblies* $P_1 = \{1, 3\,2, 4\,5, 6\,7\}$ *and* $P_2 = \{1, 2\,3, 4\,5, 6\,7\}$. *From those two, two optimal (median) solutions* M_1 *and* M_2 *are possible:* $M_1 = \{1\,\text{-}\boldsymbol{2}\,\text{-}\boldsymbol{3}\,4\,5,\ 6\,7\}$ *and* $M_2 = \{1\,\text{-}\boldsymbol{3}\,\text{-}\boldsymbol{2}\,4\,5,\ 6\,7\}$. *The rearrangement distances (*d*) and the number of breakpoints (*b*) from* M_1 *and* M_2 *to* G_1, G_2 *and* G_3 *are shown below.*

	G_1		G_2		G_3	
	d	b	d	b	d	b
M_1	2	3	1	2	4	6
M_2	1	2	2	3	4	7

6.2.3 Instance with centromeres

When contemporary centromere positions are known, they can be used to constrain ancestor reconstructions: biologically plausible results must have one and only one centromere per reconstructed chromosome. These constraints are not taken into account in the MGR algorithm, which can consequently return mathematically optimal, but biologically absurd, results.

In the same way that it was explained in the last subsection, we generated one instance of 3 genomes with 30 markers. On these genomes, we placed active and inactive centromeres: each centromere is located between two identifiers. We computed super-block sets and assemblies for this instance, and compared our results with those returned by MGR (figure 6.7). For this instance, we obtain one partial assembly with 8 super-blocks. The solution recovered by MGR has a global rearrangement distance of 35 and contains all the super-blocks. Nevertheless, this solution is not viable due to the fact that the second chromosome of this median has no active centromere.

Viable solutions that respect this biological constraint may be non minimal in terms of rearrangement distance, so respecting this biological constraint can require exploration of solutions that are mathematically suboptimal. For this example, we find 10 solutions that respect super-blocks and where global rearrangement score varies between 34 and 36. All minimal solutions are absurd as they do not respect the centromere constraint, but we do find 3 viable solutions with a global distance equal to 35 (figure 6.7).

6.3 Discussion

Computing the median for a given set of genomes is informative when the sample set of genomes is carefully chosen and the interpretation of the common features that are so identified is performed with caution. As with any statistical study, if the sample is too small or not representative of the population under study, then the median may be biased. It is not the object of this work to provide guidance into sampling strategies for genome comparisons, but to provide robust mathematical tools for performing the comparisons. Practical studies ([Eri07], [GNS08], for example) concur that the set of plausible medians is quite large and that it is misleading to present just one as “the” ancestral architecture of a set of genomes (see section 2.4.4 page 51 for more details).

The focus of this work is on the identification of common structural features that are likely to be inherited from ancestral genomes. These super-blocks can be seen as complex traits in the sense of Dollo parsimony, whose conservation and possible loss from a common ancestor is more likely than independent gain in separate lineages. They are identified without use of a hypothesized phylogeny, and indeed nothing suggests that recombinatory evolution coincides with mutational evolution (see section 2.5.2 page 55).

This use of phylogeny is an important feature of the work of [MZS$^+$06] (see section 2.5.1 page 51 for the method of Ma et al.). Super-blocks share certain aspects of the motivation behind CARs: that is, assembling only adjacencies having sufficient support in contemporary genomes.

The sharing tree of super-blocks (such as seen in figure 6.3) encodes all the possibilities of ancestral genome architectures by including in the super-blocks the adjacencies common to at least 2 genomes, and leaving the super-block extremities as the only places where no semantically sound assembly is possible. This final assembly is then just a question of optimization under some metric, and in this work we use the Hannenhalli-Pevzner rearrangement distance.

The super-blocks themselves implement a compromise between the rearrangement and breakpoint distances, and thus, thanks to the latter, encode the ancestral semantics, while leaving room for optimization thanks to the former.

In practice, our method realizes two successive search-space reductions. First, the super-blocks themselves diminish the number of unresolved adjacencies (left for the optimization step). Second, we rely on the biological constraints for further search-space reduction, as well as solution filtering. In particular, in our application to the non-WGD yeasts we use the centromere positions, yielding biologically plausible solutions only.

6.3.1 Gene and Segmental Duplication

Accounting for gene and segmental duplication is an important challenge, that we do not address in this work. In [MRL$^+$07] Martin et al. use the interleaving patterns of gene orders to study rearrangements before and after the hypothesized whole genome duplication (WGD) event in the *Saccharomyces* lineage [WS97]. Interestingly, they claim that a series of partial genome duplications leads to more parsimonious rearrangement scenarios that does a single whole genome duplication in apparent contradiction of the widely accepted hypothesis [KBL04, DS$^+$04]. In their study they combined rearrangement events with duplication and deletion events; during a preprocessing step their method renumbers duplicated elements in gene orders to produce a permutation compatible with the Hannenhalli-Pevzner rearrangement algorithms that they use. For computational reasons, only a single chromosome of *A. gossypii* is studied in detail. For this example our results agree; indeed, the segments in their figure 5 (and supplemental file S1 provided by reviewer 2) are found in our adjacent markers 52 and 51 (Figure 6.2), conserved in our median and all genomes we consider except *Z. rouxii.* Our study is otherwise quite different. Since we deliberately only consider species outside of the WGD lineage, we are not concerned with the large-scale duplications and deletions that mask the underlying rearrangement events. Our method works efficiently on complete genomes, and is not reliant on the Hannenhalli-Pevzner method, but rather proposes a partial reconciliation between it and the breakpoint method. Our super-blocks method does not take duplications into account, since it is not obvious how to weigh duplicated adjacencies when counting their frequency. This is a direction for future work.

6.3.2 Towards Ancestor Construction in Yeasts

Comparative genomics in the hemiascomycete yeasts has proven extremely informative about the basic mechanisms of eukaryotic molecular evolution, both using genetic tools and computer analysis. These species represent a homogeneous phylogenetic group with small and compact genomes, but nonetheless a large diversity at the physiological and ecological levels, and an evolutionary range comparable to the Chordate phylum [DS$^+$04, Duj06]. They provide a kind of 'evolutionary playground' in which various genome-modifying mechanisms have been tested over and over. Building a mathematical description of this rich history will provide important insight.

In this work we have used our super-block method to construct a plausible ancestral architecture for a phylogenetically circumscribed group of non-WGD yeasts, using ordered markers derived from all-against-all search for conserved syntenic segments. Surprisingly, highly similar sets of super-blocks are constructed from these markers, reinforcing the idea that the ancestral semantics can be recovered using adjacencies observed in contemporary genomes. Final assemblies of these super-blocks were constructed by an optimization procedure using the Hannenhalli-Pevzner rearrangement distance as a metric. A strength of our method is that such final assemblies can be made to respect biological constraints on chromosome architecture, in this work centromere position.

Since our method can efficiently handle hundreds of markers in dozens of genomes simultaneously, these results open the way to a more in-depth study of the rearrangement history of the yeasts. This will require technical advances, for detecting synteny in the presence of segmental duplication, for masking the effects of highly mobile elements, and for improved respect of biological constraints.

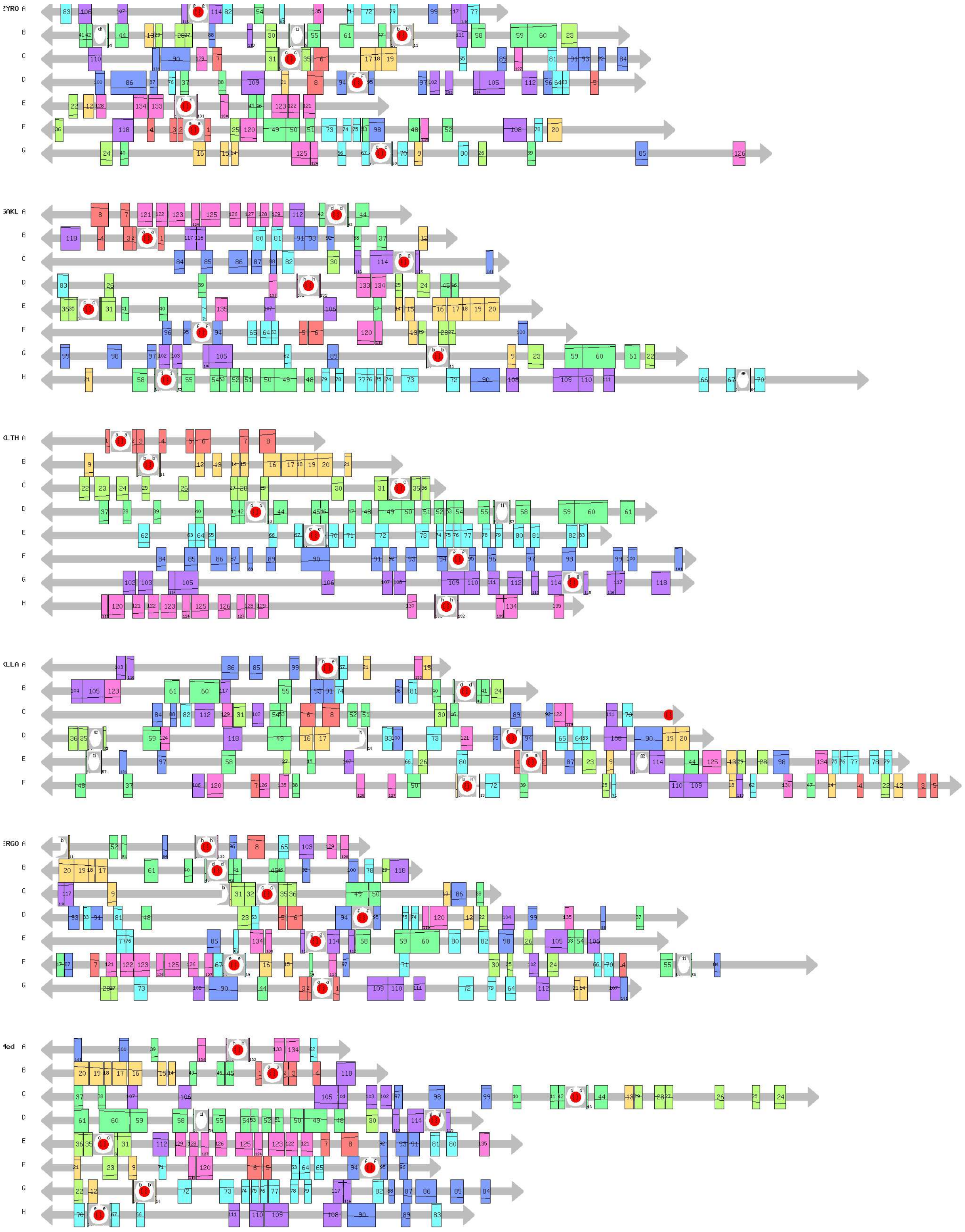


Figure 6.2: Reconstructions of genome-scale homology from common synteny blocks representing major conserved segments. Med is the proposed ancestral architecture with $\sum d(Med, G_i) = 284$. Each unique numbered synteny block is given a color indicating its chromosome in the reference genome ($Klth$), and a diagonal bar indicating its relative position on the chromosome. Other genomes are signed permutations of these colored blocks; a change of slope in the diagonal bar indicates an inversion. Block widths are to scale and the size of interleaving non-syntenic regions is shown by large grey lines. Red circles: centromeres; gray triangles: telomeres.

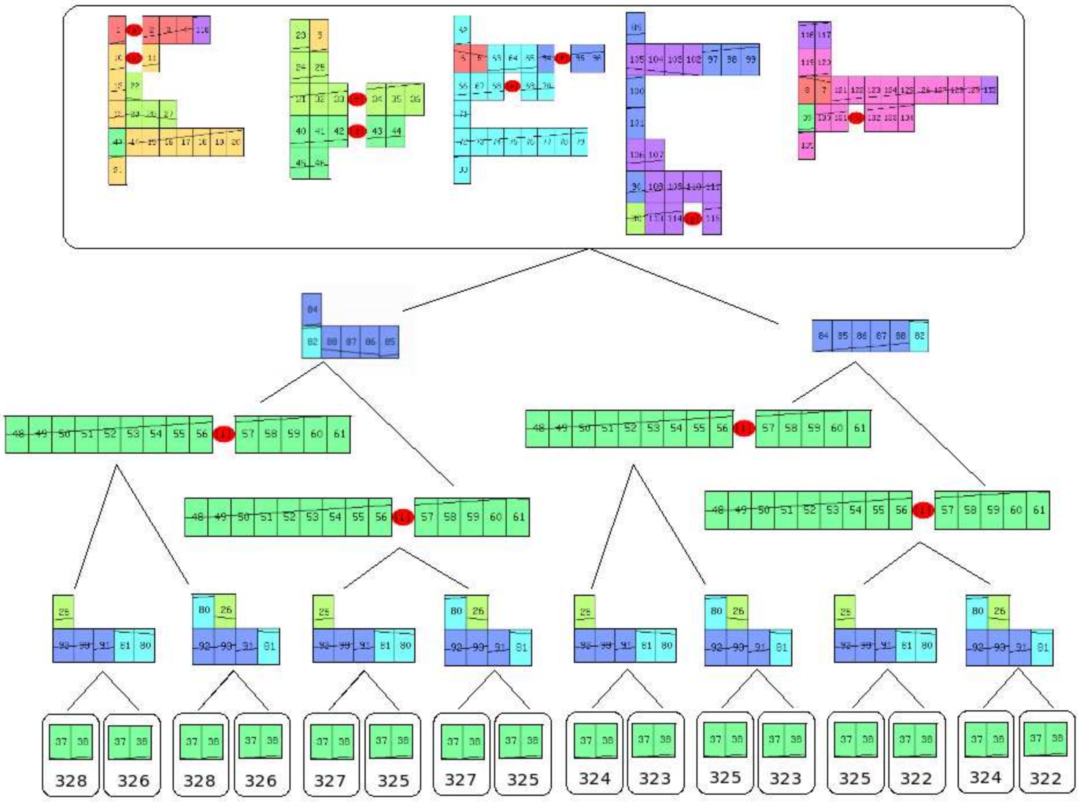


Figure 6.3: Sharing tree of super-blocks from the 16 sets of super-blocks obtained from non-WGD Hemiascomycete yeasts genomes. The root contains the super-blocks shared among all the 16 sets. Each path from the root to a leaf represents a set of super-blocks. The number inside the leaf nodes indicates the sum of the distance between this set of super-blocks and the contemporary genomes. Colors and marker numbers were chosen using *Klth* as a reference. The diagonal line in each box indicates the relative position and orientation of the marker on the reference genome.

```
HUMAN: 1 2 3 4 5 6 7 8,                    9 10 11 12 13 14,
       15 16 17 18 19 20,                  21 22 23 24,
       25 26 27 28 29 30 31 32 33,         34 35,
       36 37,                              38 39 40 41 42 43,
       44 45 46 47 48,                     49 50 51 52 53 54 55 56,
       57 58 59 60 61 62                   63 64 65 66 67 68,
       69 70,                              71 72 73 74 75 76 77 78 79,
       80 81,                              82 83,
       84 85 86 87,                        88,
       89 90,                              92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109,
       91,                                 110 111 112 113 114

CAT:   -22 -21 23 24,                      -70 -69,
       86 87 -85 -84,                      -37 -36,
       -26 -25 27 28 29 30 31 -33 -32,     106 107 -109 -108,
       44 45 -58                           -57 -60 -59 61 62 63 64 65 66 67 68 90 91,
       1 2 3 -10 -9 11 12 13 14,           -88 -20 -19 -18 -17 15,
       -50 -49 53 54 -52 -51 55 56,        46 47 48,
       -89 80 81,                          -38 39 40 41 42 43,
       -71,                                -75 -74 -73 -72 -77
       -76 78 79,                          -83 -82,
       34 35,                              -7 -6 8 -5 -4,
       16,                                 92 93 94 95 96 97 98 99 100 101 102 103 104 105
       110 111 112 113 114

MOUSE: 34 35,                              11 12 13 14 -8 -7 -6,
       32,                                 44 45 -43 -42 -41 -10 -9 -54 -53 84 85 -87 -86,
       -65,                                -20 -19 -5 -4 -3 -2 -1,
       33 38 39 40,                        57 58 59 60 61 62,
       36 37,                              82 83 -52 -51 48 49 50,
       15 16,                              -56 -55 -70 -69 -18 -17
       103 104 105 -93 -92,                -89 -88 -68 -67,
       -26 -25 -24,                        23 71 -75 -74 -73 -72 76 77 78 79,
       -47 -46,                            21 22 90 91 63 64 -66
       80 81,                              -97 -96 98 99 100 106 107 -109 -108 -95 -94 101 102
       -31 -30 -29 -28 -27,                110 111 112 113 114

A:     -8 -7 -6,                           71 -75 -74 -73 -72 76 77 78 79,
       80 81,                              -70 -69 -5 -4 -3 -2 -1,
       82 83,                              -12 -11 -52 -51 55 56,
       49 50,                              44 45 -43 -42 -41 -40 -39 -38,
       -14 -13                             92 93 94 95 96 97 98 99 100 101 102 103
       -37 -36,                            -30 -29 -28 -27 25 26,
       -35 -34,                            87 -85 -84 53 54 9 10,
       32 33 -31                           -48 -47 -46,
       21 22 88 89,                        104 105 106 107 -109 -108,
       86                                  -91 -90 -68 -67 -66 -65 -64 -63 -62 -61 -60 -59 -58 -57,
       -16 -15 17 18 19 20,                -24 -23,
       110 111 112 113 114
```

Figure 6.4: Human, mouse and cat permutations as well as the ancestral permutations (A) recovered by MGR-MEDIAN [BP02].

<table>
<tr><th colspan="2">Super-blocks common to all the partial assemblies</th></tr>
<tr><td colspan="2">-8 -7 -6
-71
72 73 74 75
76 77 78 79
80 81
82 83
-50 -49
-70 -69
-5 -4 -3 -2 -1
-14 -13 -12 -11
51 52
55 56
46 47 48
-45 -44
36 37
34 35
-33 -32
-31 -30 -29 -28 -27
25 26
-22 -21
88
-89
90 91
57 58 59 60 61 62 63 64 65 66 67 68
84 85 -87 -86
53 54
-10 -9
-16 -15 17 18 19 20
23 24
108 109 -107 -106 -105 -104 -103 -102 -101 -100 -99 -98 -97 -96 -95 -94 -93 -92 -114 -113 -112 -111 -110</td></tr>
<tr><th>Super-block specific to $\mathcal{P}_1$</th><th>Super-block specific to $\mathcal{P}_2$</th></tr>
<tr><td>-43 -42 -41 -40 -39 38</td><td>-43 -42 -41 -40 -39 -38</td></tr>
</table>

Figure 6.5: 2 sets of super-blocks obtained from the public dataset of Human, Cat and Mouse gene order (see figure 6.4).

(a) G_1: -27 16 1,
-8 30 14 15 -26 -21,
20 -22 -25 17 18 -24 3 -5 -4 6 9 -29 -28 -19 23 -2 7 10 11 12 13

G_2: 29 -20 16 -8 -7 15 -12 18,
19 13 14 9 -4 -21 30 28 -25 -24 -23 2 22,
-1 -11 -10 3 5 6 17 26 27

G_3: 9 10 19 -27 -26 -25 -24 -23 -16 -15,
-4 -3 -2 -13 -12 5 6 20,
14 18 -11 -28 -30 -29 -22 -21 -17,
-1 -8 -7

(b) F: -27 -26, 10 11, 23 24 25, -6 -5, -8 -7, -13 -12, -28 -30

(c) A_MGR: 10 11 2 3 4 7 8 23 24 -18 -17 25 22 -20 -6 -5 12 13,
9 15 16 1,
-27 -26 -21 30 28 29 -14 -19

(d) A_SB: 29 30 28 -8 -7,
21 26 27 -19 23 24 25 22 -20,
-18 -17 -6 -5 12 13 2 3 4 -11 -10 -9,
14 15 16 1

Figure 6.6: Results for (a) the simulated instance G_1, G_2 and G_3: (b) Super-block set F, (c) median genome A_MGR provided by the public version of MGR and (d) one of the 10 median solutions recovered by fusions of super-blocks.

(a) G_1: 1 $\{c\}$ 10 3 -30 -23 -19 22 14 15 27 28 -13 -12 -11 25 9,
26 $\{c\}$ 8,
20 21 -24 5 6 -2 7 -29 -18 $\{c\}$ -17 -4 -16

G_2: -7 5 11 12 13 14 23 24 -18 $\{c\}$ -17 -6,
1 $\{c\}$ 10,
26 $\{c\}$ 8 27 -3 21 22 -16 -4 28 -30 -29 15 19,
20 -2 -9 $\{c\}$ -25

G_3: -24 28 -18 $\{c\}$ -17 -13 4 -29 -16 -11 -7 23 -19 14 15 25 9,
26 $\{c\}$ 8 27 12 -3 -30 5 6,
-22 -21 -20 1 $\{c\}$ 10 2

(b) F: 25 9, -22 -21 -20, 11 12 13, 17 18, -27 -26 -8, -15 -14, -6 -5, -10 -1

(c) A_MGR: -9 $\{c\}$ -25 17 $\{c\}$ 18 -28 -27 -8 $\{c\}$ -26,
20 21 22 -16 7 11 12 13 -15 -14 19 23 24 -30 5 6,
29 -4 -3 -2 -10 $\{c\}$ -1

(d) A_SB: 1 $\{c\}$ 10 2 -7 -23 -19,
-9 $\{c\}$ -25 -15 -14 -22 -21 -20,
3 4 -29 -18 $\{c\}$ -17 -6 -5 24,
26 $\{c\}$ 8 27 28 -30 16 -13 -12 -11

Figure 6.7: (a) A simulated instance G_1, G_2 and G_3 with active centromere positions indicated by the letter c between embraces. (b) For this instance, a set of super-blocks F is obtained. (c) The median genome A_MGR provided by the public version of MGR presents a chromosome without active centromere.(d) A_SB is a sub-optimal median solution in terms of global rearrangement score which is plausible for centromere constraint.

Part IV

Optimal rearrangement scenarios

Chapter 7

Computing a correct optimal scenario

Analysis of genome rearrangements provides a measure for the evolutionary distance between species. Two closely related problems are considered in the study of genome rearrangements. The first problem is to find, by parsimony criteria and for a defined set of rearrangement operations, the exact number of such operations needed to rewrite one genome into another. The second problem is to compute a most parsimonious rearrangement scenario. Solving the latter would enable the understanding of evolutionary mechanisms.

In the considered model (see section 1.2), two genomes defined on the same set of gene markers without duplications, are represented by signed permutations. Thus, the analysis of genomes leads to a combinatorial problem of transforming one signed permutation into another. The theory proposed by Hannenhalli and Pevzner [HP95a, SM97] for unichromosomal genomes based on reversals only is presented in detail in chapter 2 (see section 2.2.1). Their main results consist in an exact formula for reversal distance, and the first polynomial time algorithm for computing a parsimonious reversal-based scenario between two signed permutations.

This theory was further adapted by the same authors to the multichromosomal case and is presented in the same chapter, section 2.2.2. For multichromosomal genomes, a larger set of rearrangement operations is considered: translocations, fusions and fissions as well as reversals. In [HP95b], Hannenhalli and Pevzner devise a method that mimics all multichromosomal rearrangements by reversals operating on an unique permutation. This is achieved by a conversion to the unichromosomal model, which requires an *optimal capping* to cleverly delineate chromosomes of a given genome, as well as an *optimal concatenate* in order to assemble them into a single permutation. The computed parsimonious scenario relies on the structure of this permutation.

However, both the formula for rearrangement distance and the algorithm for computing a parsimonious sequence of operations given by Hannenhalli and Pevzner [HP95b] present errors. Tesler in [Tes02a] partially corrected the rearrangement distance formula. In the same paper, the algorithm that leads to optimal concatenates was completed by a proper bonding step (for more details, readers are invited to refer section 2.3.2). Ozery-Flato and Shamir in turn redefined some notions and suggest further corrections essentially for the rearrangement distance formula [OFS03]. Nevertheless, the algorithm that is supposed to construct an optimal capping, fails.

Various definitions and their relationships presenting incoherences between papers by different authors, we first propose a single and coherent classification of interleaving graph components based on relevant literature in section 7.1. This classification permits a better understanding of what is wrong in the existing algorithm for determining optimal capping. In section 7.2, we present cases for which Ozery-Flato and Shamir's algorithm fails and provide a counterexample for each case. Finally, we introduce in section 7.3, a correct algorithm for optimal capping with a proof of its correction. This whole work was published in [JN07].

7.1 Double classification of connected components

Let Π and Γ be two multichromosomal genomes with respectively N_Π and N_Γ chromosomes defined over the same set of gene markers N_g. Two steps are needed to encode a multichromosomal genome as an unique permutation: *capping* and *concatenate*. $\hat{\Pi}$ and $\hat{\Gamma}$ represent a capping of Π and Γ and we denote by $\hat{\pi}$ and $\hat{\gamma}$ concatenates for $\hat{\Pi}$ and $\hat{\Gamma}$.

The notions of adjacencies and breakpoints are transferred to the *breakpoint graph* defined in [HP95a]. Denote by $G(\Pi, \Gamma)$ ($G(\hat{\Pi}, \hat{\Gamma})$, $G(\hat{\pi}, \hat{\gamma})$ respectively) the breakpoint graph constructed from permutations Π and Γ ($\hat{\Pi}$ and $\hat{\Gamma}$, and $\hat{\pi}$ and $\hat{\gamma}$ respectively).

The distance value is computed based on the breakpoint graph $G(\Pi, \Gamma)$, free of any capping and concatenate, in which we can distinguish three types of vertices: isolated vertices called *tails*, cap vertices of degree 1 called Π-*caps*, and other vertices of degree 1 called Γ-*tails*. The graph $G(\Pi, \Gamma)$ can be decomposed into cycles and paths that are characterized by their extremities ($\Pi\Pi$-*path*, $\Gamma\Gamma$-*path* and $\Pi\Gamma$-*path*).

Construction of the *interleaving graph* $I(G)$ (see section 2.2.2 page 32 for more details) is defined from non-trivial paths or cycles (with more than 2 edges) of the breakpoint graph $G = G(\Pi, \Gamma)$ and based on the notion of *edge interleaving*. We propose a coherent and unambiguous classification for the connected components of an interleaving graph that is the result of a synthesis of previously cited references. In fact, the components can be classified in two different and complementary ways, as shown in figure 7.1.

7.1.1 Intrinsic classification

We call *intrinsic classification* the way to discriminate between components based on the properties of their edges. It is represented by the vertical hierarchy of filled nodes in figure 7.1. A dashed edge (representing an adjacency in Γ) $\{\hat{\pi}_i, \hat{\pi}_j\}$ in $G(\Pi, \Gamma)$ is *oriented* if $|j - i|$ is even, otherwise it is *unoriented*. The same edge is *intrachromosomal* if the vertices $\hat{\pi}_i$ and $\hat{\pi}_j$ belong to the same chromosome, and *interchromosomal* otherwise. A connected component K of $I(G)$ is *oriented* (*interchromosomal*, respectively) if any cycle or path belonging to K has at least one oriented (interchromosomal, respectively) dashed edge, otherwise K is *unoriented* (*intrachromosomal*, respectively). Let $\mathcal{U}(G)$ be the set of unoriented components of $I(G)$, $\mathcal{IU}(G)$ the set of unoriented and intrachromosomal ones.

We have seen that the difficulty to compute the rearrangement distance comes from unoriented and intrachromosomal components (see the unichromosomal case, section 2.2.1 page 28). The intrinsic classification is then refined for this set of components: we distinguish *real* components from *unreal* components within unoriented and intrachromosomal components. As a reminder, a connected component K of $I(G)$ is *real* if K belongs to $\mathcal{IU}(G)$ and if it has no Π-cap or Γ-tail in its span. Let $\mathcal{RU}(G)$ be the set of real components.

Example 7 gives the intrinsic classifications for the breakpoint graph of the figure 7.2.

7.1.2 Extrinsic classification

We call *extrinsic classification* the way to describe a component by its relationship with surrounding components. It is represented horizontally by dashed lines in figure 7.1. This classification concerns the sets of unoriented components $\mathcal{U}(G)$, $\mathcal{IU}(G)$ and $\mathcal{RU}(G)$ that require a more detailed study in order to determine the rearrangement distance as well as the algorithms that lead

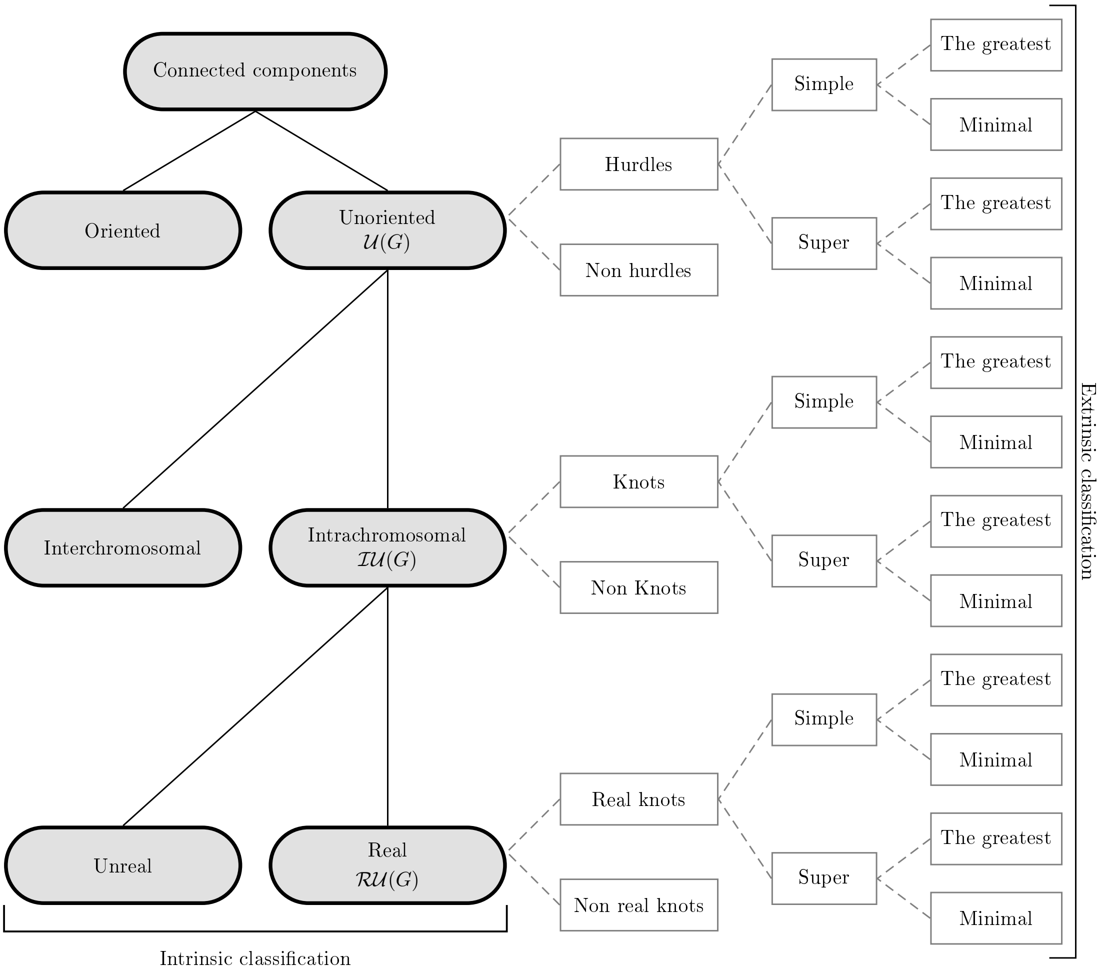


Figure 7.1: Double classification of connected components. The children nodes form a partition of the component set represented by their parent node. Intrinsic classification is read from top to bottom while extrinsic classification is read from left to right.

to parsimonious scenario computation.

The first partition for these sets relies on the notion of *component separation* (see definition 13 page 29). $\mathcal{U}(G)$ is partitioned into *non hurdles* and *hurdles*, where a hurdle is a component of $\mathcal{U}(G)$ that does not *separate* two other components in the same set. The notion of separation defines in the same way the partitions of $\mathcal{IU}(G)$ and $\mathcal{RU}(G)$: *knots* and *non knots* for the former, and *real knots* and *non-real knots* for the latter.

The second level of the extrinsic classification is based on *protection* notion (see definition 16 page 30). Within the hurdle set, we distinguish the *super hurdles* from the *simple* ones. A hurdle is *super* if it *protects* a non hurdle, otherwise it is *simple*. These notions are defined similarly for knots and real knots.

While protection notion characterizes hurdle (knots, real knots respectively) relationships with non hurdles (non knots, non real knots, respectively), the last level of classification is based on the relationships between hurdles themselves. A hurdle can be *the greatest* one if its span contains all the spans of the others hurdles, otherwise it is a *minimal hurdle*. These notions are defined similarly for knots and real knots.

Example 7 gives the extrinsic classifications for the breakpoint graph of the figure 7.2.

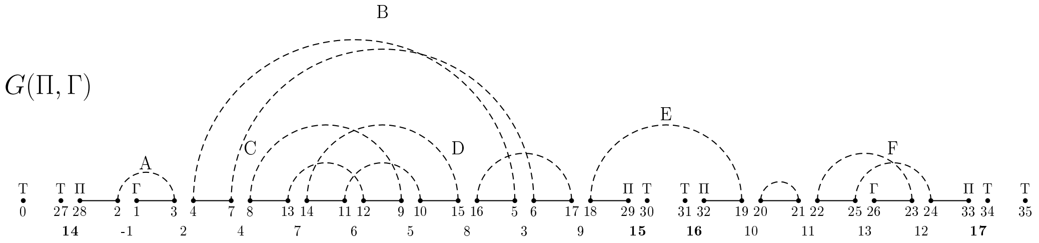


Figure 7.2: Breakpoint graph $G(\Pi,\Gamma)$ for $\Pi = \{-1\ 2\ 4\ 7\ 6\ 5\ 8\ 3\ 9, 10\ 11\ 13\ 12\}$ and $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\}$. Tails vertices are marked by T, Π-caps by Π and Γ-tails by Γ. Non trivial cycles and paths are denoted by letters from A to F. The interleaving graph $I(G)$ corresponding to $G(\Pi,\Gamma)$ is composed of 5 connected components: $K_1 = \{A\}$, $K_2 = \{B\}$, $K_3 = \{C, D\}$, $K_4 = \{E\}$ and $K_5 = \{F\}$.

Example 7 *Figure 7.2 presents a breakpoint graph $G(\Pi,\Gamma)$. The intrinsic classification is as follows: K_1 is intrachromosomal oriented, $\mathcal{U} = \{K_2, K_3, K_4,\ K_5\}$, $\mathcal{IU} = \{K_2, K_3, K_5\}$ and $\mathcal{RU} = \{K_2, K_3\}$. The extrinsic classification is: K_3 is a super hurdle while K_4 and K_5 are simple hurdles, and K_3 and K_5 are super knots. However, K_2 and K_3 are real knots (K_2 is the greatest one), while K_5 is a minimal semi-real knot and K_1 is a simple component.*

7.1.3 Particular structures and distance formula

Based on this classification, particular structures of the breakpoint graph are defined. Counting specific components (defined both by the nature of their edges and their relationships with other components) is required in order to compute the rearrangement distance. Within the set of unreal components we can distinguish those called *semi-real knots* that are characterized by their potential of becoming real knots (see definition 24 page 33). A *simple component* is defined as a component with at least one ΠΓ-path and which is not a semi-real knot.

From all these considerations, global specific structures for the breakpoint graph are defined. The breakpoint graph G is a *fortress* (*fortress of knots*, or *fortress of real knots*, respectively) if it contains an odd number of hurdles (knots, or real knots, respectively) that are all super. We say that a graph G is a *weak fortress of real knots* if (a) G has an odd number of real knots, (b) there exists the greatest real knot in G, (c) all real knots are super except the greatest one and (d) the number of semi-real knots in G is strictly greater than 0. Note that a weak fortress of real knots becomes a fortress of real knots by closing the ΠΓ-paths in a semi-real knot.

Denote by $\bar{G}(\Pi,\Gamma)$ the graph obtained by closing all the ΠΓ-paths in simple components of $G(\Pi,\Gamma)$. Ozery-Flato and Shamir [OFS03] give an exact formula for the distance between two multichromosomal genomes Π and Γ (see theorem 3 page 34): $d(\Pi,\Gamma) = b - c + p_{\Gamma\Gamma} + r + \lceil\frac{s'-gr'+fr'}{2}\rceil$ where b is the number of solid edges in $G(\Pi,\Gamma)$ ($b = N_g + max(N_\Pi, N_\Gamma)$), c is the number of cycles and paths, $p_{\Gamma\Gamma}$ is the number of ΓΓ-paths, r is the number of real knots, s' is the number of semi-real knots in $G(\Pi,\Gamma)$, gr' is equal to 1 if $\bar{G}$ has the greatest real-knot and $s' > 0$, and is 0 otherwise, fr' is equal to 1 if either (i) $\bar{G}$ is a fortress of real knots and the greatest semi-real knot does not exist in $\bar{G}$, or (ii) $\bar{G}$ is a weak fortress of real knots.

Computing the distance between two multichromosomal genomes is independent of capping and concatenation. However, computing a parsimonious scenario consists in finding a sequence of reversals mimicking multichromosomal rearrangements that satisfy the minimal distance. Thus, *optimal capping* and *optimal concatenate* are required to find a parsimonious scenario. Nevertheless, in spite of corrections brought by Tesler [Tes02a] and by Ozery-Flato and Shamir [OFS03], the algorithm for computing optimal capping remains incorrect.

7.2 Cases for which optimal capping algorithm fails

Optimal capping Π^* and Γ^* is finding positions and signs for caps in the genome Γ such that $d(\Pi^*, \Gamma^*) = d(\Pi, \Gamma)$ (see lemma 4 page 36). This is done for any arbitrary capping in Π. In the breakpoint graph, it consists in adding $2N_\Gamma$ edges linking a Π-cap to a Γ-tail and $N_\Pi - N_\Gamma$ edges between two Π-caps if $N_\Pi > N_\Gamma$.

The algorithm for construction of an optimal capping that takes into the account the last corrections for rearrangement distance is provided by Ozery-Flato and Shamir [OFS03] (see algorithm 1 page 37). However, this algorithm is incorrect. There are two cases for which their algorithm fails. In what follows, we describe each of these cases and provide a counterexample.

7.2.1 Difference in the number of chromosomes

Since the distance function is symmetric, we have $d(\Pi, \Gamma) = d(\Gamma, \Pi)$ and so Ozery-Flato and Shamir [OFS03] consider only the case where $N_\Gamma \leq N_\Pi$ without lost of generality. However, the proposed algorithm fails if $N_\Gamma < N_\Pi$. The number of Π-caps is equal to $2\max(N_\Pi, N_\Gamma)$ and the one of Γ-tails is $2N_\Gamma$. Clearly, the number of Π-caps is strictly greater that the number of Γ-tails if $N_\Gamma < N_\Pi$. Thus, $p_{\Pi\Pi} > p_{\Gamma\Gamma}$. Steps 2 and 3 of algorithm 1 consist in joining a $\Pi\Pi$-path with a $\Gamma\Gamma$-path to the point of $\Pi\Pi$-path exhaustion according to lemma 2 page 36. Consequently, the number of $\Gamma\Gamma$-paths is not sufficient to close all the $\Pi\Pi$-paths when $N_\Gamma < N_\Pi$. See figure 7.3 and example 8 for a counterexample.

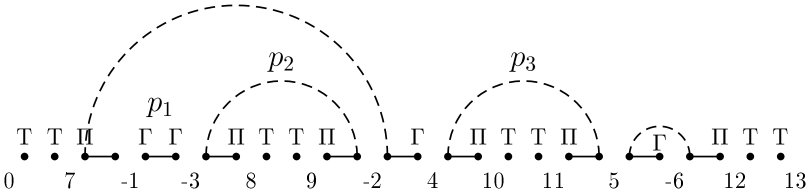


Figure 7.3: Counterexample to Ozery-Flato and Shamir's algorithm [OFS03] for building an optimal capping. Breakpoint graph $G(\Pi, \Gamma)$ with $\Pi = \{-1\ -3,\ -2\ 4,\ 5\ -6\}$ and $\Gamma = \{1\ 2\ 3,\ 4\ 5\ 6\}$.

Example 8 *The breakpoint graph $G = G(\Pi, \Gamma)$ in figure 7.3 has two $\Pi\Pi$-paths p_2 and p_3, and one $\Gamma\Gamma$-paths p_1. A first occurrence of steps 2 and 3 of Ozery-Flato and Shamir's algorithm joins p_2 or p_3 with p_1. An other one has to join the remaining $\Pi\Pi$-path with a $\Gamma\Gamma$-path but there is no $\Gamma\Gamma$-path left anymore.*

7.2.2 A specific breakpoint graph structure

Another case for which the algorithm fails can be described as follows: (i) s' is even and $s' > 2$, (ii) G is a fortress of real knots and (iii) G has the greatest semi-real knot. If G is a fortress of real knots and there exists the greatest semi-real knot then $fr' = 0$. Moreover, the greatest semi-real knot and the greatest real knot can not exist simultaneously, so $gr' = 0$. Hence, the genomic distance is $d = b - c + p_{\Gamma\Gamma} + r + \lceil \frac{s'}{2} \rceil = b - c + p_{\Gamma\Gamma} + r + \frac{s'}{2}$ since s' is even. The step 5 of the optimal capping algorithm in [OFS03] joins any two semi-real knots. Suppose that the greatest semi-real knot is joined by an interchromosomal or oriented edge to another semi-real knot. The obtained graph is still a fortress of real knots, but the greatest semi-real knot does not exist anymore, so $fr' = 1$. Thus, we get $d = b-(c-1)+p_{\Gamma\Gamma}+r+\lceil\frac{(s'-2)+1}{2}\rceil = b-c+1+p_{\Gamma\Gamma}+r+\frac{s'}{2}$. See figure 7.4 and example 9 for a counterexample.

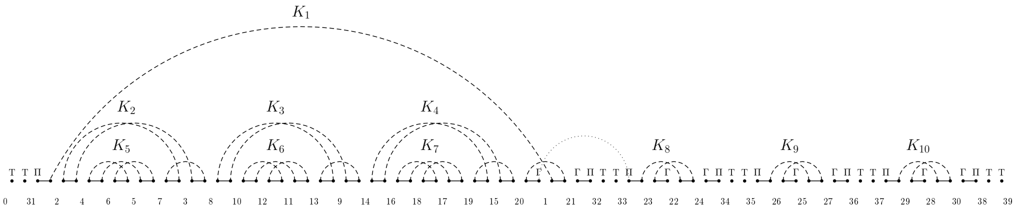


Figure 7.4: Counterexample to the Ozery-Flato and Shamir's algorithm [OFS03] for building an optimal capping. Breakpoint graph $G(\Pi,\Gamma)$ with $\Pi = \{2\ 4\ 6\ 5\ 7\ 3\ 8\ 10\ 12\ 11\ 13\ 9\ 14\ 16\ 18\ 17\ 19\ 15\ 20\ 1\ 21,\ 23\ 22\ 24,\ 26\ 25\ 27,\ 29\ 28\ 30\}$ and $\Gamma = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ 21,\ 22\ 23\ 24,\ 25\ 26\ 27,\ 28\ 29\ 30\}$. The connected components K_5, K_6 and K_7 are super real knots that protect respectively K_2, K_3 and K_4. G has the greatest semi-real knot K_1 and three minimal semi-real knots K_8, K_9 and K_{10}.

Example 9 *The breakpoint graph $G = G(\Pi,\Gamma)$ in figure 7.4 is a fortress of real knots with $fr' = 0$. The distance is $d(\Pi,\Gamma) = 34 - 14 + 0 + 3 + \lceil\frac{4-0+0}{2}\rceil = 25$. Step 5 of Ozery-Flato and Shamir's algorithm allows joining the greatest semi-real knot K_1 to K_8 by an interchromosomal edge (dashed line), which results in a new graph G'. G' is still a fortress of real knots, but $fr' = 1$. So $d = 34 - 13 + 0 + 3 + \lceil\frac{2-0+1}{2}\rceil = 26$, which does not respect the minimal distance.*

7.3 A correct algorithm for optimal capping

In what follows we propose a new algorithm for optimal capping (algorithm 9) and the proof of its correction (theorem 9). The proof is based on two technical lemmas from [HP95b] (lemmas 2 and 3 page 36) and possible configurations for pertinent parameters of the breakpoint graph presented by figure 7.5.

Theorem 9 (Jean and Nikolski [JN07]) *Let $d = d(\Pi,\Gamma)$ be the distance between two multichromosomal genomes Π and Γ. Algorithm 9 constructs an optimal capping $\hat{\Gamma}$ for any arbitrary capping $\hat{\Pi}$, such that $d(\hat{\Pi},\hat{\Gamma}) = d$.*

Proof: Let M be the total number of edges needed to close all the paths. If $N_\Pi > N_\Gamma$, then $M = 2N_\Gamma + N_\Pi - N_\Gamma$, otherwise $M = 2N_\Gamma$. Building a capping $\hat{\Gamma}$ involves adding M edges e_i to $G(\Pi,\Gamma)$. This process defines a new graph G_i for the ith addition of an edge. It results after M

Algorithm 9 Correct_Optimal_Capping

```
 1: Construct the graph $G = G(\Pi, \Gamma)$
 2: while there is a $\Gamma\Gamma$-path in $G$ do
 3:    Find an interchromosomal or oriented edge joining this $\Gamma\Gamma$-path with a $\Pi\Pi$-path (lemma
       2) and add it to $G$
 4: end while
 5: Close all remaining $\Pi\Pi$-paths in $G$
 6: Close all $\Pi\Gamma$-paths in simple components in $G$
 7: if  $s'$ is even and $s' \geq 2$ and $G$ is a fortress of real knots then
 8:    if $G$ has the greatest semi-real knot then
 9:       Close all $\Pi\Gamma$-paths in the greatest semi-real knot
10:    else
11:       Close all $\Pi\Gamma$-paths in any one semi-real knot
12:    end if
13: end if
14: while $G$ has more than one semi-real knot do
15:    Find an interchromosomal or oriented edge joining $\Pi\Gamma$-paths in any two semi-real knot
       (lemma 3) and add it to $G$
16: end while
17: Close all remaining $\Pi\Gamma$-paths in $G$
18: Find a capping $\hat{\Gamma}$ defined by the graph $G(\hat{\Pi}, \hat{\Gamma})$
```

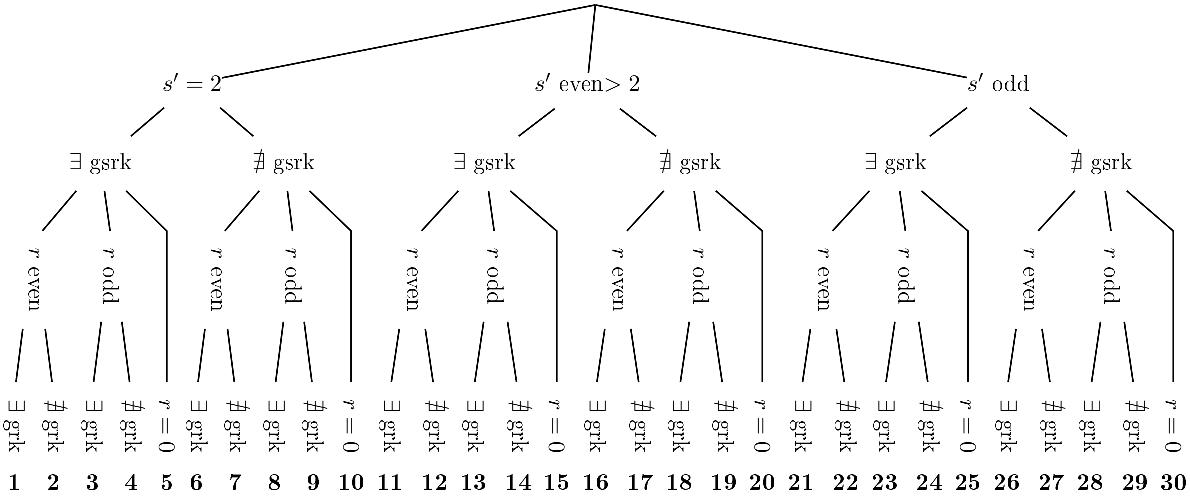


Figure 7.5: Possible configurations for pertinent parameters of the breakpoint graph G: the parity and value of s', the presence of the greatest semi-real knot (gsrk), the parity and value of r and the presence of the greatest real knot (grk). Configurations are numbered from 1 to 30.

additions in $G(\hat{\Pi}, \hat{\Gamma})$: $G(\Pi, \Gamma) = G_0 \xrightarrow{e_1} G_1 ... \xrightarrow{e_M} G_M = G(\hat{\Pi}, \hat{\Gamma})$. We denote by d_i the distance computed on the graph G_i, and we index by i all the distance formula parameters.

For each parameter p we denote by Δ_p the difference of its values for successive graphs $p_i - p_{i-1}$. Then $\Delta = d_i - d_{i-1}$. In what follows, we prove that for each added edge $\Delta = 0$ and consequently $d_M - d_0 = 0$.

The first *while* loop (lines 2-4 in algorithm 9 results in $\Delta = 0$. Indeed, if G_{i-1} has a $\Gamma\Gamma$-path then there also exists a $\Pi\Pi$-path. Connecting a $\Gamma\Gamma$-path with a $\Pi\Pi$-path via an interchromosomal or an oriented edge results in $\Delta_{p_{\Gamma\Gamma}} = -1$, $\Delta_c = -1$, and hence in $\Delta = 0$. The graph structure modifications in lines 5 and 6 do not affect any parameter value, and thus we still have $\Delta = 0$.

Starting from line 5, what remains is to close $\Pi\Gamma$-paths in semi-real knots. The proof for this part of the algorithm is based on a case analysis. The last part of distance formula, $\lceil \frac{s'-gr'+fr'}{2} \rceil$, depends on the parity and value of s'. Moreover, semi-real knots can become real knots and then modify the values of gr' and fr'. That is why, we have also to consider the parity and value of r. The greatest semi-real knot (the semi-real knot, respectively) does not have the same behavior as the minimal ones: we have to take into the account the presence or absence of these particular components. All the possible graph configurations are shown in figure 7.5. We show that for all of them $\Delta = 0$. Notice that configurations 1, 3, 11, 13, 21 and 23 in figure 7.5 are impossible since the greatest semi-real knot and the greatest real knot can not exist simultaneously.

The *then* part of the *if* statement (lines 7 through 13 in algorithm 9) concerns three possible cases:

1. the greatest semi-real knot exists (configurations 4 and 14),
2. the greatest semi-real knot does not exist, but the greatest real knot exist (configurations 8 and 18),
3. the greatest semi-real knot and the the greatest real knot do not exist (configurations 9 and 19).

For these 6 configurations we have $\Delta_c = \Delta_{p_{\Gamma\Gamma}} = 0$ and $\Delta_{s'} = -1$. The values of $\Delta_{fr'}$, Δ_r and $\Delta_{gr'}$ vary between the three cases.

1. We are in the *then* part at line 9, $fr'_{i-1} = 0$ and $fr'_i = 0$ since the number of real-knots becomes even. So $\Delta_{fr'} = 0$, $\Delta_r = 1$ and $\Delta_{gr'} = 1$.
2. We are and the *else* part at line 11, $fr'_{i-1} = 1$ and $fr'_{i-1} = 1$. Closing all the $\Pi\Gamma$-paths in a minimal semi-real knot does not modify the number of real knots: the greatest real knot becomes an unreal component. Thus, $\Delta_r = 0$, $\Delta_{gr'} = -1$ and $\Delta_{fr'} = 0$
3. We have $gr'_{i-1} = gr'_i = 0$. Therefore $\Delta_r = 1$, $\Delta_{gr'} = 0$ and $\Delta_{fr'} = -1$ since the number of real knots becomes even.

Thus in all the possible cases before line 14 we have $\Delta = 0$.

The second *while* loop (line 14 through 16) is entered in three cases:

1. $s'_{i-1} = 2$ (configurations from 2 to 10 except 3),
2. $s'_{i-1} > 2$ is even (configurations from 12 to 20 except 13),
3. s'_{i-1} is odd (configurations from 22 to 30 except 23).

In all of these configurations $\Delta_c = -1$, $\Delta_{p_{\Gamma\Gamma}} = \Delta_r = 0$ and $\Delta_{s'} = -2$. The values $\Delta_{gr'}$ and $\Delta_{fr'}$ depend on the configuration.

1. For all configurations, except 6 and 8, we have. $\Delta_{gr'} = \Delta_{fr'} = 0$. For configurations 6 and 8, $\Delta_{gr'} = -1$. For configuration 6, $fr'_{i-1} = fr'_i = 0$. For configuration 8, G_{i-1} can be a weak fortress of real knots, and so $fr'_{i-1} = 1$ or 0 but $fr'_i = 0$ since $s'_i = 0$. Thus, $\Delta_{fr'}$ is either 0 or -1.

2. In all configurations $\Delta_{gr'} = 0$. For all configurations except 18, $\Delta_{fr'} = 0$ since $fr'_{i-1} = fr'_i = 0$. For 18, if G_{i-1} is a weak fortress of real knots then G_i is one too, and $fr'_{i-1} = fr'_i = 1$, otherwise $fr'_{i-1} = fr'_i = 0$, and so $\Delta_{fr'} = 0$.

3. Two cases are possible: (a) one of the two semi-real knots is the greatest semi-real knot or (b) the two semi-real knots are minimal. For (a) $gr'_{i-1} = gr'_i = 0$ and $fr'_{i-1} = 0$, but fr'_i is either 1 or 0 depending on whether G_{i-1} is a fortress of real knots. For (b) $\Delta_{gr'} = \Delta_{fr'} = 0$.

Applying the distance formula from theorem 3, we obtain $\Delta = 0$ in all cases.

If at this point (line 17) there still remains a semi-real knot and one of the following conditions holds

1. either G_{i-1} has the greatest real knot (configurations 26 and 28),

2. or G_{i-1} has the greatest semi-real knot (configurations 22, 24 and 25),

3. or G_{i-1} has neither one nor the other (configurations 27, 29 and 30),

then we have to close the $\Pi\Gamma$ paths.

For all these cases, we have $\Delta_c = \Delta_{p_{\Gamma\Gamma}} = 0$ and $\Delta_{s'} = -1$. The values of Δ_r, $\Delta_{gr'}$ and $\Delta_{fr'}$ depend on the particular configuration.

1. $\Delta_r = 0$ and $\Delta_{gr'} = -1$ since $s'_i = 0$. As for the value of fr', consider that G_{i-1} can be either a weak fortress of real knots, or a fortress of real knots, or none. In all of these cases the value of fr' does not change.

2. $\Delta_r = 1$, $\Delta_{gr'} = 0$, and $\Delta_{fr'} = 0$ since the greatest semi-real knot becomes the greatest simple real knot.

3. $\Delta_r = 1$ and $\Delta_{gr'} = 0$. As for the value of fr', fr'_{i-1} is either 1 or 0 depending on whether G_{i-1} is a fortress of real knots or not, and $fr'_i = 0$.

Applying the distance formula from theorem 3, we obtain $\Delta = 0$ in all cases.

We see then, that in all possible cases, graph modifications $G(\Pi, \Gamma) = G_0 \overset{e_1}{\rightarrow} G_1 ... \overset{e_M}{\rightarrow} G_M = G(\hat{\Pi}, \hat{\Gamma})$ by our algorithm are neutral with respect to the distance formula. □

Chapter 8

VIRAGE: an interactive tool for the visualization of rearrangement scenarios

Efficient algorithms exist to compute rearrangement scenarios between two genomes. In particular, chapters 2 and 7 present algorithms based on the Hannenhalli and Pevzner theory for the computation of a rearrangement scenario between two signed multichromosomal genomes in terms of reversals, translocations, fusions and fissions. The first implementation that made it possible to analyze rearrangements in multichromosomal genomes was realized in GRIMM [Tes02b]. However, the resulting rearrangement scenario is visualized as a static, and possibly quite long, sequence of permutations. Genome modeling in the form of signed permutations makes the analysis and comparison of scenarios difficult. Hence, a challenge lies in the visualization of plausible results in order to facilitate their interpretation by expert biologists.

We developed a new tool called VIRAGE for VIsualization of ReArrangement within GEnomes, which permits the interactive and animated visualization of several rearrangement scenarios. Rearrangements taken into the account are reversals, translocations, fusions and fissions. VIRAGE is divided in two main parts: the generator of the visualization document and the visualizer of rearrangement scenarios.

In this chapter, we first present the generator of the visualization document. This generator is strongly based on the *genome graph*, a common structure to all of the scenarios. The obtained document contains information relative to scenarios under study and also includes modules required for the visualizer of rearrangements. A second section is dedicated to the visualizer, which is built of two main parts: the sequencing module that manages the course of scenarios according to users' instructions and the animating module that enables the animation of rearrangements.

8.1 Generator of the visualization document

The generator of the visualization document is the static part of VIRAGE, which consists in producing an SVG (Scalable Vector Graphics [SVG01]) document from a set of scenarios provided as parameters. The code of the generator is written in Python.

8.1.1 Syntax of input files

VIRAGE requires as many input files as there are different scenarios to visualize. The chosen syntax for a scenario is similar to the one of GRIMM results [Tes02b].

A scenario is a sequence of genomes where two consecutive genomes differ by one transformation among reversals, translocations, fusions and fissions. In a scenario file, each line corresponds to a step in the scenario, i.e to one genome.

A multichromosomal genome is represented by a signed permutation where elements are separated by space character and delimiters '\$' are inserted after chromosomes. If centromere positions are known, it is possible to add this information in the scenario file by indicating each centromere by a letter framed by two braces. See figure 8.1 for an example.

```
1 2 3 4 {a} 90 $ 5 6 {b} 91
1 2 -4 -3 {a} 90 $ 5 6 {b} 91
-1 2 -4 -3 {a} 90 $ 5 6 {b} 91
-1 2 -4 -3 {a} 90 $ -5 6 {b} 91
-1 2 4 -3 {a} 90 $ -5 6 {b} 91
-1 2 4 3 {a} 90 $ -5 6 {b} 91
```

Figure 8.1: Example of a scenario file between two multichromosomal genomes. The first line represents the source genome, the last, the target genome and all the lines except for the first are intermediate genomes obtained from the previous one by a reversal in this example. Genomes have two chromosomes delimited by the character '\$' and two centromeres located by letters *a* and *b* between braces.

We consider three different configurations for the set of input files:

- $1-1$ case: all of the input files start and end by the two same genomes;
- $1-n$ case: all of the first lines of input files correspond to the same genome;
- $n-1$ case: all of the last lines of input files correspond to the same genome.

A syntactic analysis of scenario files is realized in order to verify that files are well formed.

8.1.2 Genome graph and nearly genome graph

VIRAGE was developed to ease the visualization of one or several rearrangement scenarios between species. In the case of multiple scenarios, we group the different scenarios together into a common data structure: the *genome graph*. This graph is the basis for the rest of the implementation. Moreover, this structure is quite useful for the end users. In fact, it makes it possible to quickly visualize the mutual organization of scenarios and, during the animated phase, to understand the current step in the scenarios' progress.

Vertex hierarchy

A scenario is a sequence of genomes that represent intermediate states during evolution. Hence, we can associate to each genome its index within a scenario, and genomes are ordered according to their indices. The notion of order between genomes must be conserved in the genome graph. That is why the genome graph is a directed graph where vertices represent genomes while each edge represents a transformation between two consecutive genomes in a scenario. However, it is possible that intermediate genomes are identical within several scenarios. The genome graph takes into account these common points between the scenarios by modeling the equivalent genomes by an unique vertex. Nevertheless, in order to facilitate the reading of graphs by users, the depth position of a vertex in the genome graph must be equal to the index of corresponding genomes in the scenarios. However, according to the case under study, equivalent genomes may have different indices:

- the $1-1$ *case* is the case where if the provided scenarios are parsimonious then intermediate genomes that are identical have necessarily the same index in their corresponding scenarios. Otherwise, identical genomes may have different indices,
- the $1-n$ *case* concerns evolution from a common ancestral genome towards n of its descendants. The n scenarios under study may have different lengths. Hence, identical genomes can occur at different indices in the scenarios,
- the $n-1$ *case* is the mirror of the $1-n$ case. It is treated in the same manner that the $1-n$ case.

Considering these different cases, one genome present in two scenarios is represented by only one vertex in the genome graph if it appears at the same index in two scenarios. Let $\mathcal{S} = \{s\}$ be the set of the scenarios to visualize and $s = (g_1, g_2, ..., g_m)$ a scenario of $\mathcal{S}$ where g_1 is the source genome, g_m is the target genome and the others are intermediate ones.

Definition 48 *A* genome graph *is a directed acyclic graph $G = (V, E)$ such that:*

- $V = \{(g, i) \mid \exists\ g_i \in s \in \mathcal{S}$ *such that* $g = g_i\}$,
- $E = \{((g_1, i), (g_2, i+1)) \mid \exists\ g_i$ *and* $g_{i+1} \in s \in \mathcal{S}$ *such that* $g_1 = g_i$ *and* $g_2 = g_{i+1}\}$.

The genome graph is constructed by scanning through all of the scenarios. At the k^{th} step of the algorithm, genomes at index k are compared in order to create corresponding vertices.

In the $n-1$ case, scenarios are preprocessed: all of them are inverted in order to simulate this case by an equivalent $1-n$ case. Next, the direction of all the edges of the obtained genome graph is inverted. The final graph is a directed acyclic graph but no longer a genome graph, since indices considered to construct the initial graph are those of scenarios from the common genome to its n descendants. This graph is called a *nearly genome graph.* See example 10 for the construction of a nearly genome graph.

Example 10 *Let us consider* 4 *scenarios from* 4 *distinct genomes to a common one. Table 8.1 shows these scenarios and table 8.2 presents the same scenarios but inverted. g_5 is present at different indices in scenarios 1 and 2 while its index is the same in the inverted scenarios. Thus, genome g_5 is represented by an unique vertex in the genome graph of figure 8.2 and the nearly genome graph presented in figure 8.3. On the other hand, genome g_6, which has the same index in initial scenarios but not in their inverse is represented by two distinct vertices in the (nearly) genome graph.*

Edge labeling

Once the (nearly) genome graph is obtained, we can associate a rearrangement to each edge. The supported rearrangements are reversals, translocations, fusions and fissions. All of the other transformations are defined as unknown rearrangements. Algorithm 10 specifies the kind of rearrangement that transforms genome g_i into genome g_{i+1}.

Next, specific information for each rearrangement is defined:

index	scenario 1	scenario 2	scenario 3	scenario 4
1	g_1	g_2	g_3	g_4
2	g_7	g_5	g_6	g_6
3	g_5	g_9	g_{11}	g_{10}
4	g_8	g_{11}		g_{11}
5	g_{11}			

Table 8.1: 4 scenarios from 4 distinct genomes to the common genome g_{11}.

index	scenario 1	scenario 2	scenario 3	scenario 4
1	g_{11}	g_{11}	g_{11}	g_{11}
2	g_8	g_9	g_6	g_{10}
3	g_5	g_5	g_3	g_6
4	g_7	g_2		g_4
5	g_1			

Table 8.2: Inverted scenarios of table 8.1.

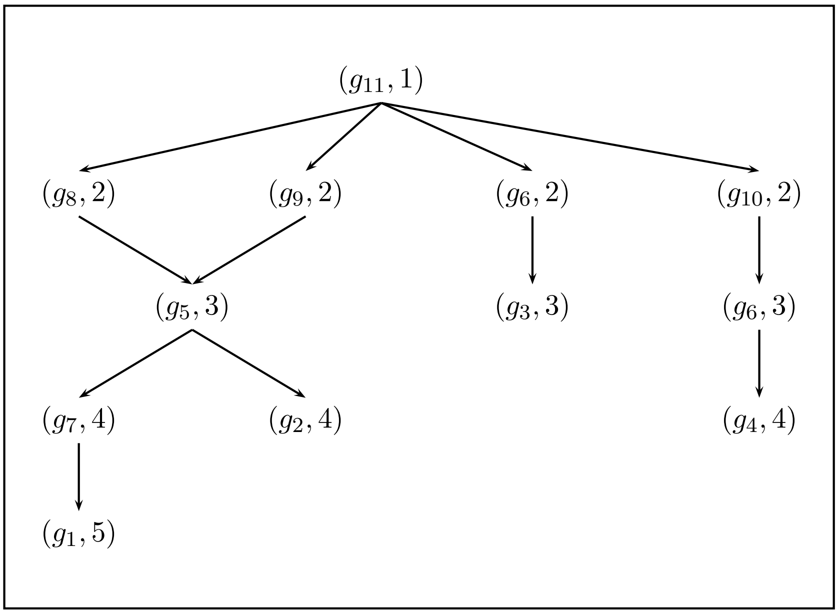


Figure 8.2: Genome graph obtained from the scenarios of table 8.2.

- reversal: the sequence of markers within a chromosome of g_i that are reversed within the same chromosome in g_{i+1},

- translocation: two sequence extremities in two distinct chromosomes of g_i that are reversed and exchanged between the two same chromosomes in g_{i+1},

- fusion: two extremity markers of two distinct chromosomes of g_i that are consecutive in an unique chromosome in g_{i+1},

- fission: two consecutive markers within a chromosome of g_i that are extremities of two

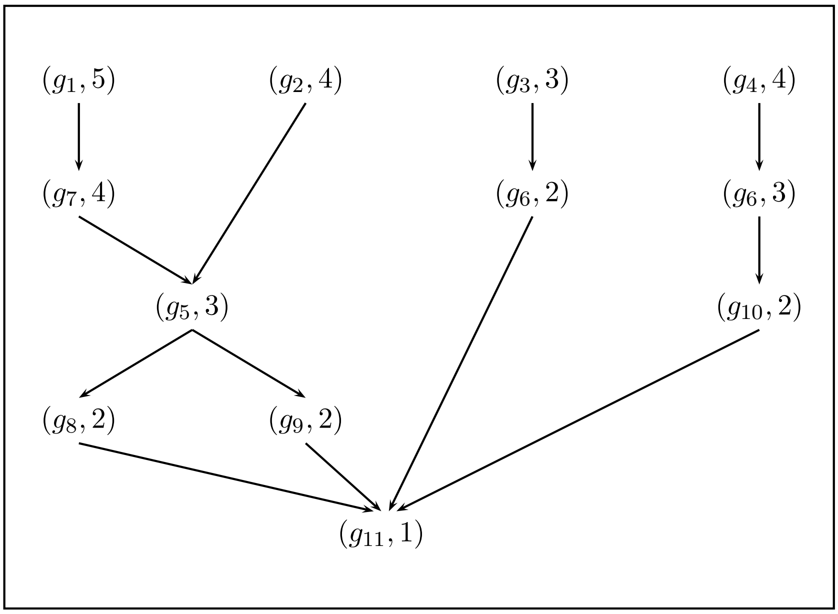


Figure 8.3: Nearly genome graph for scenarios of table 8.1.

distinct genomes in g_{i+1},

- unknown rearrangement: no specific information is required, since this kind of transformation is not animated.

The search of rearrangements is realized through a semantic analysis of genomes in order to verify that a given transformation between two genomes is interpretable by only one rearrangement. Otherwise, the transformation will be considered as an unknown rearrangement.

8.1.3 SVG document generation

After the syntactic analysis of scenarios and the construction of the (nearly) genome graph labeled by rearrangements, all of this information is registered in graphic form in an SVG document. In particular, a graphic version of the (nearly) genome graph and the genomes is generated in the document. The document also registers spatial positions of genomes as well as all the steps of transformations. Finally, sequencing and animating modules (explained in sections 8.2.2 and 8.2.3) are included in the document.

8.2 Rearrangement visualizer

The visualizer is the dynamic part of VIRAGE, which enables users to observe rearrangements as animations thanks to a browser. It is divided in two modules: the sequencing and the animating modules. The associated code is written in javascript.

Algorithm 10 Type of a rearrangement that transforms g_i into g_{i+1}

```
if $g_i$ and $g_{i+1}$ have the same number of chromosomes then
  if $g_i$ and $g_{i+1}$ differ from one chromosome then
    it is a reversal
  else
    if $g_i$ and $g_{i+1}$ differ from two chromosomes then
      it is a translocation
    else
      it is an unknown rearrangement
    end if
  end if
else
  if $g_i$ has one chromosome more than in $g_{i+1}$ then
    it is a fusion
  else
    if $g_i$ has one chromosome less than in $g_{i+1}$ then
      it is a fission
    else
      it is an unknown rearrangement
    end if
  end if
end if
```

8.2.1 Interface

Description

The graphic interface includes a global control bar, the (nearly) genome graph and a space for the representation of genomes. This space is divided in three parts: start and target genomes are respectively represented at the left hand side and at the right hand side while middle space is reserved for animated genomes. Figures 8.4, 8.5 and 8.6 show the three possible configurations (cases $1-1$, $1-n$ and n-n) of the graphic interface.

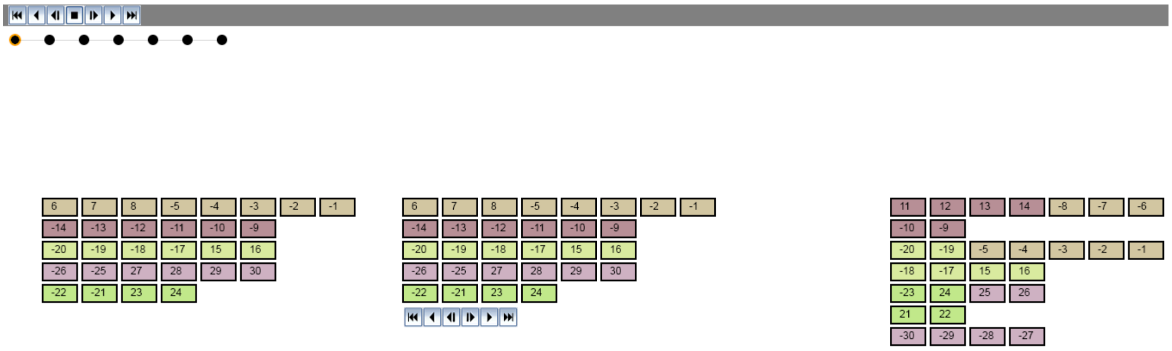


Figure 8.4: Graphic interface for a $1-1$ case.

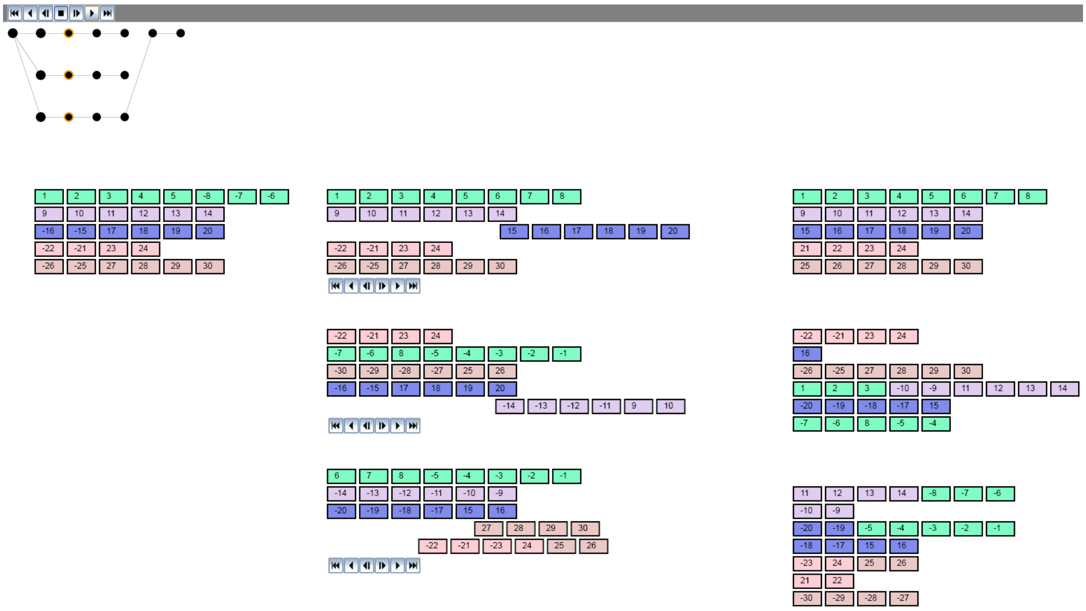


Figure 8.5: Graphic interface for a $1 - n$ case.

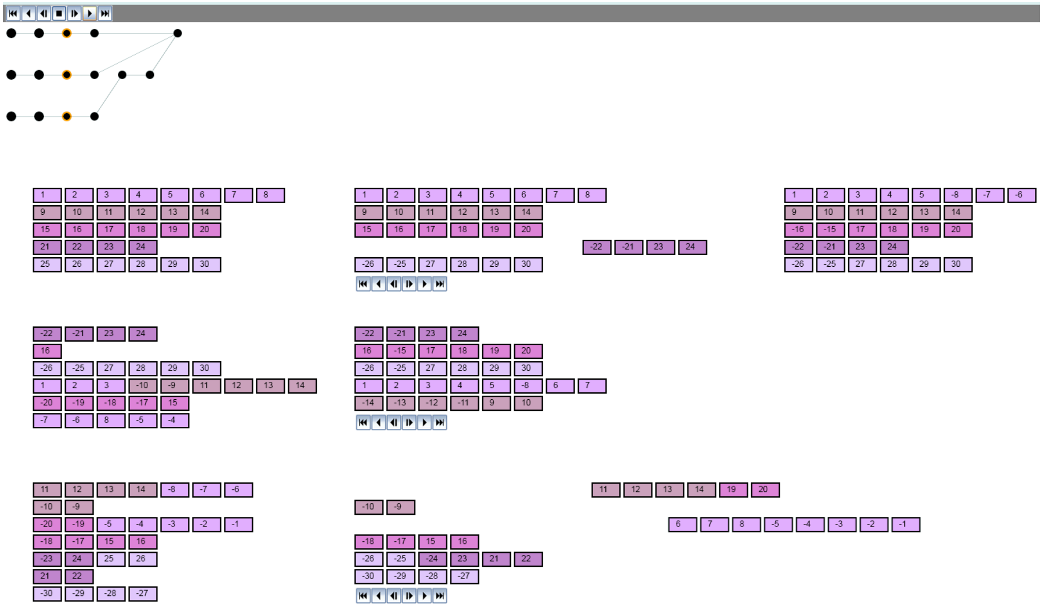
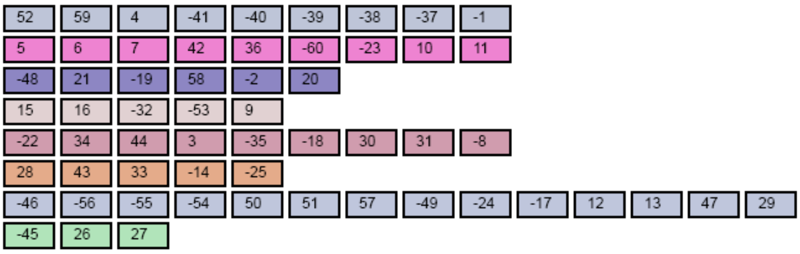


Figure 8.6: Graphic interface for a $n - 1$ case.

Genome representation

A genome is visualized as a set of lines that correspond to distinct chromosomes. Each genome marker is represented by a box colored according to its chromosome in the first starting genome. The box contains the number and the sign of the marker. If centromere positions are known, they are indicated by an ellipse shape, which contains the corresponding letter inside. Figure 8.7 shows an example of a starting genome without a centromere.



52 59 4 -41 -40 -39 -38 -37 -1

5 6 7 42 36 -60 -23 10 11

-48 21 -19 58 -2 20

15 16 -32 -53 9

-22 34 44 3 -35 -18 30 31 -8

28 43 33 -14 -25

-46 -56 -55 -54 50 51 57 -49 -24 -17 12 13 47 29

-45 26 27

Figure 8.7: Graphic representation of a genome.

Control bar

The control bar is used to progress through scenarios. Various functionalities are available: step by step or continuous reading, forward or backward; stopping; and directly going to start or end genome(s). A graphic representation of the (nearly) genome graph is presented below the control bar. The direction of edges are represented by the spatial position of their vertices: the graph is read from left to right. The current displayed states of scenarios are indicated by vertices framed in red circles. This graph is given as an informative guide and cannot be modified. An example is presented figure 8.8.

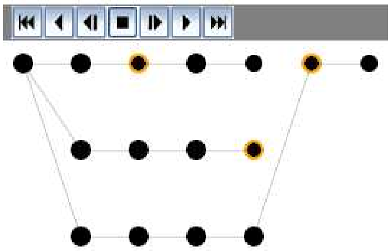


Figure 8.8: Control bar and graphical representation of a genome graph.

8.2.2 Sequencing module

The sequencing module is a set of javascript functions that assures the running of scenarios according to users' instructions. In particular, this module permits:

- to update the genome graph display,
- to launch the animations,
- to control the dependency relationships between steps of scenarios: a transformation that leads to a vertex can be realized only if all of its predecessor vertices are already reached.

8.2.3 Animating module

This module generates animations appropriate for each kind of rearrangements. The principle is the same for all of the rearrangements: affected chromosomes are "extracted" from their initial position and aligned according to exchanged markers if necessary. Finally, after the modification, chromosomes are replaced to their initial position. Figures 8.9 and 8.10 show animations for each kind of rearrangements.

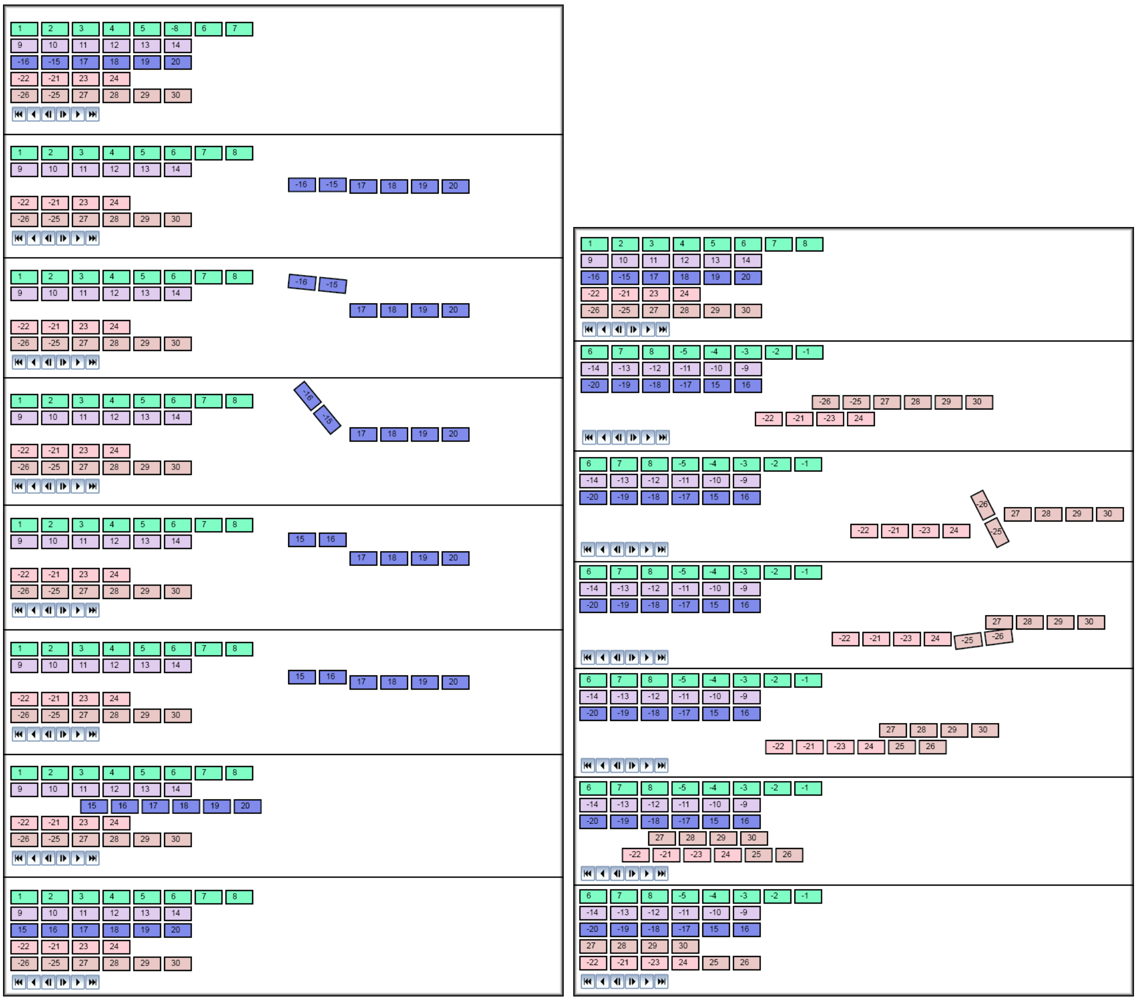


Figure 8.9: Animations of a reversal (left) and a translocation (right).

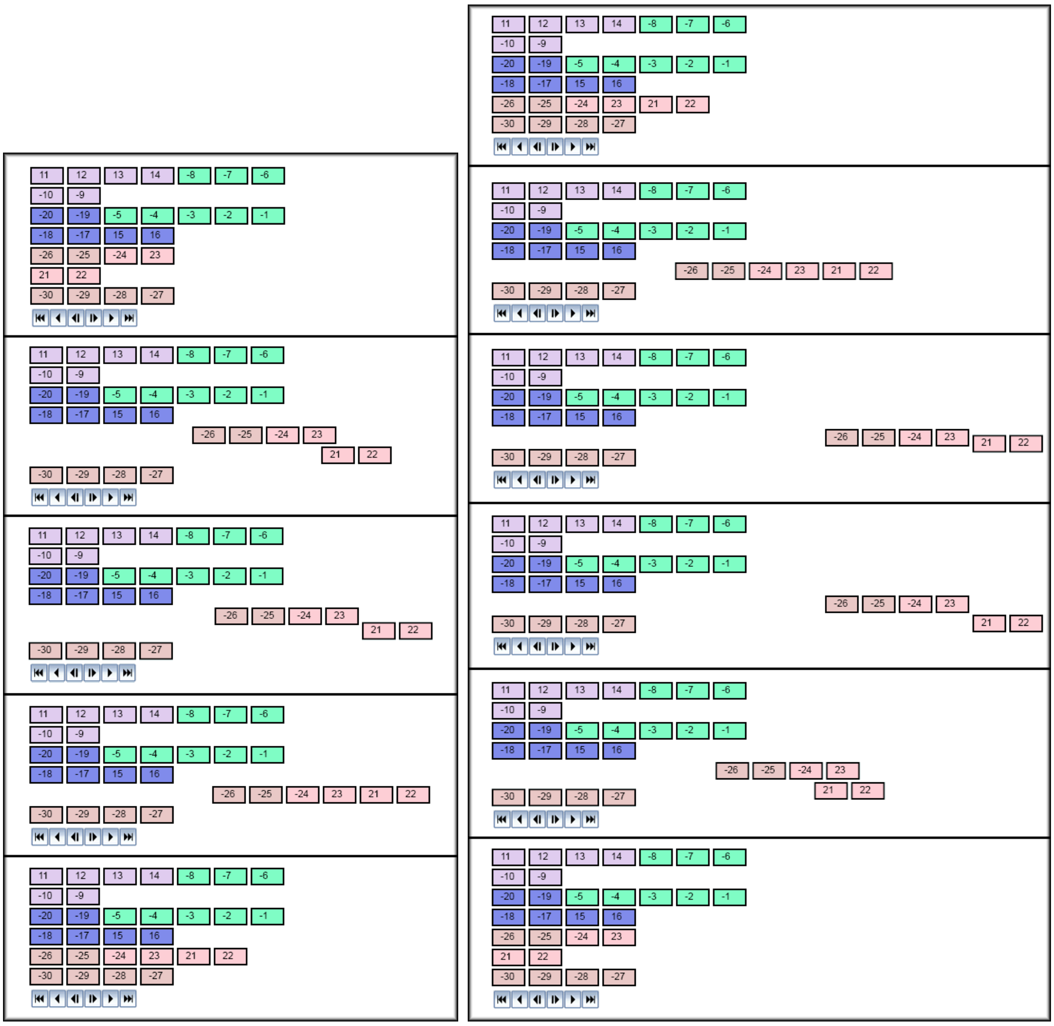


Figure 8.10: Animations of a fusion (left) and a fission (right).

Conclusion

The subject of this thesis is in the general research domain of *comparative genomics*. More particularly, we were interested in the study of evolutionary events through *genomic rearrangements* based on a combinatorial and algorithmic comparison of genomes. We developed original computational methods, that advance the state of the art by, on one hand, overcoming limitations of existing approaches and, on the other hand, by providing a complete and adapted framework for a rearrangement study in distant genomes.

Theoretical contributions

Analyzing and understanding evolutionary events is a long and complex process. It first starts with the identification of common markers between species, second requires the formulation of hypothesis about ancestral genomes and third uncovers rearrangement scenarios. In this thesis, we have contributed to these three questions in a computational way.

In an applicative framework, we were interested in distant genomes, for which existing methods for identification of common markers do not perform well. In fact, a certain number of computational methods already exist for identifying common markers, that can be either *conserved segments* or *synteny blocks*. However, these methods, which are efficient for some genomes, do not preserve sufficient signal for others so that a rearrangement study can be done. Thus, we were led to develop a new method called SyDiG -Synteny in Distant Genomes- , which can be equally applied to both close and distant genomes. Based on pairwise chromosomal homologies (i.e multiplicons) provided by AdHoRe [VSS^{+}02], SyDiG algorithm conserves all of the information contained within the multiplicons in a graph and, from it, infers new homology relationships by transitivity. Contrary to other approaches such as GRIMM-Synteny [PT03a, BPT04, BZB^{+}05], SyDiG algorithm does not filter input data but solves potential conflicts at the very end.

We also introduced the notion of *super-blocks* for identifying common ancestral features for the general N-genome instance ($N \geq 3$). We started from the observation that, given the very large number of equivalent solutions, providing one global architecture is misleading. That is why, based on adjacency and rearrangement analysis under the signed permutation model of genomes, we developed a new method that builds the *sharing tree of super-blocks* representing all the possible sets of super-blocks. Each set of super-blocks is a set of reliable ancestral chromosomal fragments whose extremities are unsolved adjacencies due to the lack of information. This approach makes it possible to constitute the basics of the putative ancestral architecture and, by combining super-blocks of a same set, to provide a global solution to the problem without any phylogenetic consideration.

This thesis started by the detailed study of Hannenhalli and Pevzner theory [HP95a, HP95b] and all the peripheral works on the computations of the rearrangement distance and parsimonious scenarios. This study led us to propose a clear view of the main notions by providing a single and coherent classification of *interleaving graph* components. This classification highlighted errors in the algorithm for *optimal capping* proposed by Ozery-Flato and Shamir [OFS03], that it itself part of the recovery of a parsimonious scenario in terms of reversals, translocations, fusions and fissions. We thus pinpointed cases for which their algorithm fails and provided a new algorithm for this step with a proof of its correction.

We were confronted with the fact that analyzing scenarios by reading successive permutations is a quite laborious task. This kind of output data does not possess a high case of use for biologist experts. We thus developed a new tool called VIRAGE -VIsualization of ReArrangements within GEnomes- that permits the interactive exploration of one or several scenario(s) between two species or between one common ancestor and its descendants thanks to the *genome graph*. Visually, each rearrangement mechanism among reversals, translocations, fusions and fissions is clearly shown by isolating chromosomes on which it occurs and by dynamically applying it to them.

Applicative contributions

Throughout this thesis, we were involved in Génolevures project [SDI$^+$06], a large-scale comparative genomics project studying species in the *Hemiascomycetous yeast phylum*. Génolevures provided an ideal application domain, since the clade of species under study presented enough synteny in order to identify common markers and therefore to apply computational methods for ancestral analysis.

At the beginning of our work, we first attempted to use existing methods, in particular, for the detection of common markers. However, current methods either revealed themselves to be not suitable to this type of genomes, or were not available. Therefore, we had to go back to basis and reconsider certain theoretical foundations. We thus have developed a complete framework for genome rearrangement analysis starting with SyDiG for the identification of common markers, through the construction of super-blocks, up to the visualization of obtained scenarios by VIRAGE.

All of the developed approaches were validated on a set of five completely sequenced yeasts from the *Saccharomycetacae* clades: *Kluyveromyces lactis*, *Saccharomyces kluyveri*, *Zygosaccharomyces rouxii*, *Ashbya (Eremothecium) gossypii* and *Kluyveromyces thermotolerans*.

Perspectives and future work

From the theoretical point of view, organisms represent very complex machineries that computational models do not totally manage yet to simulate. It is hence still required to refine existing models by adding new biological constraints in order to provide more biologically realistic results.

SyDiG algorithm developed in this thesis computes synteny blocks that contain exactly one segment per genome by avoiding groups of homologous segments non-representative of all genomes and by keeping only the longest segment in the case where more than one segment belongs to the same genome. These filters are applied in order to obtain common markers that can easily be translated in the usual model for genomes to perform current rearrangement methods. In fact, two limitations are implicitly considered in a large part of the literature on rearrangements:

- duplication events are not taken into the account: each gene marker is present exactly once in each genome;

- genomes have exactly the same gene content: insertions and deletions of genes are avoided.

In the same way, super-block construction leans on this standard genome model, that does not take into account duplication, insertion and deletion events. Nevertheless, this model is not appropriate for most genomes. In fact, while small genomes such as viruses or organelles may be simulated by this model, divergent species notably those under study present different copies of the same gene. Thus, it would be interesting to consider duplication events on one hand, and to allow genomes with different gene contents on the other hand.

Some of current methods for ancestral reconstruction or distance computation have been already extended for taking into the account these biological considerations. Sankoff [San99] introduced the *exemplar distance* between two genomes based on the hypothesis that their common ancestor has only one copy per family. Thus, the idea of the method consists in getting back the best ancestral position of each gene by removing all but one member of each marker in each genome, its *exemplar*, so as to minimize some rearrangement distance (breakpoint or reversal) between the two reduced genomes. Another approach proposed by El-Mabrouk [EM02] consists in finding, for one genome with multigene families, its ancestral genome without duplicates such that the distance between them in terms of duplication transpositions and reversals is minimized. These two approaches were used to recover ancestral nodes of a species tree [EM00a, EM02] and we can imagine applying a similar approach during super-block construction.

El-Mabrouk and Sankoff were also interested in comparing genomes with different gene contents. The former in [EM00b] extended the Hannenhalli and Pevzner theory [HP95a] by including insertions and deletions of gene blocks in the computation of rearrangement distance. As for Sankoff and colleagues [SB97], they adapted the TSP resolution of the median problem for genomes for which sets of genes differ in very few genes. Our super-block construction builds a bridge between breakpoint and rearrangement distances and methods proposed by El-Mabrouk and Sankoff may provide a strong basis in order to extend our algorithms.

Finally, we propose an approach for identifying common ancestral features for the general, N-genome instance, through the computation of super-blocks. This computation is a particular instance of species tree reconstruction by considering a N-star as the target tree. The continuation of our work is to solve, for a set of modern genomes, the whole reconstruction of the species tree by recovering the root and internal nodes. Two approaches can be considered.

(1) Without phylogenetic consideration: computational inference of species trees can be done through the resolution of the well-studied *multiple genome rearrangement problem* [SSK96, HCKP95] by optimizing Steiner points [HRW92];

(2) With phylogenetic consideration: the root node of the species tree is initialized to super-blocks of the N-genome instance resolution. Then given a phylogenetic tree, super-block inference of internal nodes is solved by combining information from leaves that correspond to modern genomes and root node. The bias potentially induced by allowing phylogenetic considerations in species tree reconstruction is reduced by the fact that root node is initially computed without this kind of information.

Biological applications of this work can be extended to other clades. In fact, although the SyDiG and super-blocks algorithms were developed in the context of the Génolevures project, these methods are general enough to be applied to other species. From the applicative point of view, it is important to apply these methods to various types of genomes. For example, it would be pertinent to test the scalability of our methods on the *Drosophila* twelve [SLK^{+}07]. Moreover, the sequences of five species phylogenetically close to the yeast *Candida glabrata* will be soon available in the Génolevures project. Other than the scientific interest in the validation of our methods on other species, a complete rearrangement study for these organisms would be of medical interest, since *Candida glabrata* is a human pathogen, that is at the origin of diseases such as Candidemia when it infects the bloodstream.

Bibliography

[AGM$^+$90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, October 1990.

[AJD$^+$06] Jean-Marc Aury, Olivier Jaillon, Laurent Duret, Benjamin Noel, Claire Jubin, Betina M. Porcel, Béatrice Ségurens, Vincent Daubin, Véronique Anthouard, Nathalie Aiach, Olivier Arnaiz, Alain Billaut, Janine Beisson, Isabelle Blanc, Khaled Bouhouche, Francisco Câmara, Sandra Duharcourt, Roderic Guigo, Delphine Gogendeau, Michael Katinka, Anne-Marie Keller, Roland Kissmehl, Catherine Klotz, France Koll, Anne Le Mouël, Gersende Lepère, Sophie Malinsky, Mariusz Nowacki, Jacek K. Nowak, Helmut Plattner, Julie Poulain, Fran'c'coise Ruiz, Vincent Serrano, Marek Zagulski, Philippe Dessen, Mireille Bétermier, Jean Weissenbach, Claude Scarpelli, Vincent Schächter, Linda Sperling, Eric Meyer, Jean Cohen, and Patrick Wincker. Global trends of whole-genome duplications revealed by the ciliate paramecium tetraurelia. *Nature*, 444:171–178, November 2006.

[AS08] Z. Adam and D. Sankoff. The ABCs of MGR with DCJ. *Evolutionary Bioinformatics*, 4:69–74, 2008.

[BBCP07] S. Berard, A. Bergeron, C. Chauve, and C. Paul. Perfect Sorting by Reversals Is Not Always Difficult. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(1):4–16, 2007.

[BBS97] M. Blanchette, G. Bourque, and D. Sankoff. Breakpoint phylogenies. In *Genome Informatics*, pages 25–34. Univ. Academy Press, 1997.

[BCHSO02] A. Bergeron, C. Chauve, T. Hartman, and K. Saint-Onge. On the properties of sequences of reversals that sort a signed permutation. *JOBIM*, pages 99–108, 2002.

[Ber01] A. Bergeron. A Very Elementary Presentation of the Hannenhalli-Pevzner Theory. *Lecture Notes in Computer Science*, 2089:106–117, 2001.

[BH96] P. Berman and S. Hannenhalli. Fast Sorting by Reversal. In D. S. Hirschberg and E. W. Myers, editors, *Proceedings of the7th Annual Symposium on Combinatorial Pattern Matching,* pages 168–185, Laguna Beach, CA, 1996. Springer-Verlag, Berlin.

[BLW76] N.L. Biggs, E.K. LLoyd, and R.J. Wilson. *Graph Theory 1736-1936.* Clarendon Press, 1976.

[BMW$^+$] D.A. Bader, B.M.E. Moret, T. Warnow, S.K. Wyman, and M. Yan. GRAPPA (Genome Rearrangements Analysis under Parsimony and other Phylogenetic Algorithms). `www.cs.unm.edu/~moret/GRAPPA/`.

[BMY01] D. A. Bader, B. M.E. Moret, and M. Yan. A Linear-Time Algorithm for Computing Inversion Distance between Signed Permutations with an Experimental Study. *Journal of Comp. Biol.*, 8(5):483–491, 2001.

[BP93] V. Bafna and P. Pevzner. Genome Rearrangements and Sorting by Reversals. *34th IEEE Symp. on foundations of Computer Science*, pages 148–157, 1993.

[BP98] V. Bafna and P. Pevzner. Sorting by Transpositions. *SIAM J. Discret. Math.*, 11(2):224–240, 1998.

[BP02] G. Bourque and P. Pevzner. Genome-Scale Evolution: Reconstructing Gene Orders in the Ancestral Species. *Genome Research*, 12:26–36, 2002.

[BPT04] G. Bourque, P. Pevzner, and G. Tesler. Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse and rat genomes. *Genome Research*, 2004.

[Bry98] D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM2579, Centre de Recherches Mathematiques, Universite de Montreal, 1998.

[BS01] A. Bergeron and F. Strasbourg. Experiments in Computing Sequences of Reversals. In *WABI '01: Proceedings of the First International Workshop on Algorithms in Bioinformatics*, pages 164–174, London, UK, 2001. Springer-Verlag.

[BSR$^+$08] A. Bhutkar, S.W.W. Schaeffer, S.M.M. Russo, M. Xu, T.F.F. Smith, and W.M.M. Gelbart. Chromosomal rearrangement inferred from comparisons of twelve Drosophila genomes. *Genetics*, July 2008.

[BTB$^+$02] Viacheslav N. Bolshakov, Pantelis Topalis, Claudia Blass, Elena Kokoza, Alessandra della Torre, Fotis C. Kafatos, and Christos Louis. A comparative genomic analysis of two distant diptera, the fruit fly, drosophila melanogaster, and the malaria mosquito, anopheles gambiae. *Genome Research*, 12:57–66, 2002.

[BTP06] G. Bourque, G. Tesler, and P. Pevzner. The convergence of cytogenetics and rearrangement-based models for ancestral genome reconstruction. *Genome Research*, 16(3):311–313, 2006.

[BZB$^+$05] G. Bourque, E.M. Zdobnov, P. Bork, P. Pevzner, and G. Tesler. Comparative architectures of mammalian and chicken genomes reveal highly variable rates of genomic rearrangements across different lineages. *Genome Research*, 15(1):98–110, 2005.

[Cap99] A. Caprara. Formulations and Complexity of Multiple Sorting by Reversals. In S. Istrail, P. Pevzner, and M. Waterman, editors, *In Proc. of RECOMB99*, pages 84–93, Lyon, 1999. acmp.

[Cap03] A. Caprara. The Reversal Median Problem. *INFORMS Journal on Computing*, 15(1):93–113, 2003.

[CNN$^+$00] Seoighe C., Federspiel N.J.T., Hansen N., Bivolarovic V., Surzycki R., Tamse R., Komp C., Huizar L., Davis R.W., Scherer S., Tait E., Shaw D.J., Harris D., Murphy L., Oliver K., Taylor K., Rajandream M.A., Barrell B.G., and Wolfe K.H.

Prevalence of small inversions in yeast gene order evolution. In *Proc Natl Acad Sci*, volume 97, pages 14433–14437, 2000.

[Con08] The UniProt Consortium. The Universal Protein Resource (UniProt). *Nucleic Acids Research Database Issue*, 2008.

[CPB$^+$03] O. Couronne, A. Poliakov, N. Bray, T. Ishkhanov, D. Ryaboy, E. Rubin, L. Pachter, and I. Dubchak. Strategies and tools for whole-genome alignments. *Genome Res.*, 13(1):73–80, 2003.

[DMBP04] A.C.E. Darling, B. Mau, F.R. Blattner, and N.T Perna. Mauve: Multiple alignment of conserved genomic sequence with rearrangements. *Genome Res.*, 14(7):1394–1403, 2004.

[Doo90] R.F. Doolittle. Molecular evolution: computer analysis of protein and nucleic acid sequences. *Meth Enzymol*, 183, 1990.

[DS38] T. Dobzhansky and A. H. Sturtevant. Inversions in the chromosomes of drosophila pseudoobscura. *Genetics*, 23(1):28–64, 1938.

[DS$^+$04] B. Dujon, D. Sherman, et al. Genome evolution in yeasts. *Nature*, 430(6995):35–44, 2004.

[Duj06] B. Dujon. Yeasts illustrate the molecular mechanisms of eukaryotic genome evolution. *Trends in Genetics*, 22:375–387, 2006.

[EDO02] A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30:1575–1584, 2002.

[EH05] I. Elias and T. Hartman. A 1.375-Approximation Algorithm for Sorting by Transpositions. In *Proc. of the 5th International Workshop on Algorithms in Bioinformatics (WABI'05)*, volume 3692 of *Lecture Notes in Computer Science*, pages 204–214. Springer-Verlag, October 2005.

[EM00a] N. El-Mabrouk. Duplication, rearrangement and reconciliation. *Computational Biology Series*, 1:537–550, 2000.

[EM00b] N. El-Mabrouk. Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *J. Discrete Algorithms*, 1(1):105–122, 2000.

[EM02] N. El-Mabrouk. Reconstructing an ancestral genome using minimum segments duplications and reversals. *J. Comput. Syst. Sci.*, 65(3):442–464, 2002.

[EMS03] N. El-Mabrouk and D. Sankoff. The construction of doubled genomes. *SIAM J. Comput.*, 32(3):754–792, 2003.

[Eri07] N. Eriksen. Reversal and transposition medians. *Theoretical Computer Science*, 374(1-3):111–126, 2007.

[FCG$^+$06] L. Froenicke, M.G. Caldés, A. Graphodatsky, S. Müller, L.A. Lyons, T.J. Robinson, M. Volleth, F. Yang, and J. Wienberg. Are molecular cytogenetics and bioinformatics suggesting diverging models of ancestral mammalian genomes? *Genome Research*, 16(3):306–310, 2006.

[Fig04] M. Figeac. *Étude de l'ordre des gènes : clusters de gènes et algorithmique des réarrangements.* Thèse en informatique, Université des Sciences et Technologies de Lille, 2004.

[Fit71] W.M. Fitch. Toward defining the course of evolution: Minimum change for a specified tree topology. *Syst Zool*, 20:406–416, 1971.

[FNS96] V. Ferretti, J.H. Nadeau, and D. Sankoff. Original synteny. In *CPM '96: Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, pages 159–167, London, UK, 1996. Springer-Verlag.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W.H. Freeman & Co, 1979.

[GNS08] A. Goëffon, M. Nikolski, and D. Sherman. An Efficient Probabilistic Population-Based Descent for the Median Genome Problem. In *Proceedings of GECCO 2008*, pages 315–321, 2008.

[HAB$^+$07] T.J.P. Hubbard, B.L. Aken, K. Beal, B. Ballester, M. Caccamo, Y. Chen, L. Clarke, G. Coates, F. Cunningham, T. Cutts, et al. Ensembl 2007. *Nucleic Acids Res.*, 35, Database issue:D610–D617, 2007.

[Han96] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics 71(1-3)*, pages 137–151, 1996.

[HCKP95] S. Hannenhalli, C. Chappey, E. Koonin, and P. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics*, 30(2):299–311, 1995.

[HP95a] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). *Proceedings of twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 178–189, 1995.

[HP95b] S. Hannenhalli and P. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. The Steiner Tree Problem. *Annals of Discrete Mathematics*, 53, 1992.

[HS03] T. Hartman and R. Shamir. A Simpler 1.5-Approximation Algorithm for Sorting by Transpositions. In *CPM*, pages 156–169, 2003.

[J$^+$04] O. Jaillon et al. Analysis of the tetraodon nigroviridis genome reveals the protokaryotype of bony vertebrates ans its duplication in teleost fish. *Nature*, 431:946–957, 2004.

[JN07] G. Jean and M. Nikolski. Genome rearrangements: a correct algorithm for optimal capping. *Inf. Process. Lett.*, 104(1):14–20, 2007.

[JSN] G. Jean, D. Sherman, and M. Nikolski. Super-blocks or mining the semantics of ancestral genome architectures. *JCB.* submitted.

[KBH$^+$03] W. J. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc Natl Acad Sci U S A*, 100(20):11484–11489, September 2003.

[KBH$^+$04] W.J. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolution's cauldron: Duplication, deletion and rearrangement in the mouse and human genomes. *Proc. Nat. Acad. Sci.*, 100(20):11484–9, 2004.

[KBL04] M. Kellis, B.W. Birren, and E.S. Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast Saccharomyces cerevisiae. *Nature*, 428:617–624, 2004.

[KR95] J. Kececioglu and R. Ravi. Of Mice and Men: Algorithms for Evolutionary Distances Between Genomes with Translocation. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1995.

[KS93] J.D. Kececioglu and D. Sankoff. Exact and Approximation Algorithms for the Inversion Distance Between Two Chromosomes. In *CPM*, pages 87–105, 1993.

[KST97] H. Kaplan, R. Shamir, and R.E. Tarjan. Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997.

[LEMTS03] J.F. Lefebvre, N. El-Mabrouk, E. Tillier, and D. Sankoff. Detection and validation of single gene inversions. In *ISMB (Supplement of Bioinformatics)*, pages 190–196, 2003.

[LQWZ04] G. Li, X. Qi, X. Wang, and B. Zhu. A Linear-Time Algorithm for Computing Translocation Distance between Signed Genomes. In *CPM*, pages 323–332, 2004.

[LS08] C. Lemaitre and M.-F. Sagot. A small trip in the untranquil world of genomes. *Theor. Comput. Sci.*, 395(2-3):171–192, 2008.

[MRL$^+$07] N. Martin, E. Ruedi, R. LeDuc, F.-J. Sun, and G. Caetano-Anollés. Gene-interleaving patterns of synteny in the saccharomyces cerevisiae genome: are they proof of an ancient genome duplication event? *Biology Direct*, 2(1):23, 2007.

[MRR$^+$08] J. Ma, A. Ratan, B.J.J. Raney, B.B.B. Suh, L. Zhang, W. Miller, and D. Haussler. DUPCAR: Reconstructing Contiguous Ancestral Regions with Duplications. *Journal of computational biology : a journal of computational molecular cell biology*, September 2008.

[MSTL02] B.M.E. Moret, A.C. Siepel, J. Tang, and T. Liu. Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In *WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, pages 521–536. Springer-Verlag, 2002.

[MTL02] B. Ma, J. Tromp, and M. Li. Patternhunter: Faster and more sensitive homology search. *Bioinformatics*, 18(3):440–5, 2002.

[MTWW02] B.M.E. Moret, J. Tang, L. Wang, and Y. Warnow. Steps toward accurate reconstructions of phylogenies from gene-order data. *J. Comput. Syst. Sci*, 65:508–525, 2002.

[MWB$^+$01] B.M.E. Moret, S. Wyman, D.A. Bader, T. Warnow, and M. Yan. A new implementation and detailed study of breakpoint analysis. In *Proc. 6th Pacific Symp. on Biocomputing (PSB 2001)*, pages 583–594. World Scientific Pub., 2001.

[MZS$^+$06] J. Ma, L. Zhang, B. Suh, B. Raney, R. Burhans, W.J. Kent, and M. Blanchette. Reconstructing Contiguous Regions of an Ancestral Genome. *Genome Research*, 16(12):1557–1565, December 2006.

[NS07] M. Nikolski and D. Sherman. Family relationships: should consensus reign? - consensus clustering for protein families. *Bioinformatics*, 23(2):71–76, 2007.

[NT84] 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 of the United States of America, Vol. 81, No. 3, [Part 1: Biological Sciences]*, pages 814–818, 1984.

[NW70] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, March 1970.

[OF03] M. Ozery-Flato. A Correction to the Theory of Sorting Genomes by Reversals and Translocations. Master's thesis, Tel-Aviv university, June 2003.

[OFS03] M. Ozery-Flato and R. Shamir. Two Notes On Genome Rearrangement. *J. Bioinformatics Comput. Biol.*, 1(1):71–94, 2003.

[OFS06] M. Ozery-Flato and R. Shamir. Sorting by Translocations Via Reversals Theory. In *Comparative Genomics*, pages 87–98, 2006.

[PH88] J.D. Palmer and L.A. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 28:87–97, 1988.

[PPT06] Q. Peng, P. Pevzner, and G. Tesler. The fragile breakage versus random breakage models of chromosome evolution. *PLoS Comp. Bio.*, 2006.

[PS98] I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(071), 1998.

[PT03a] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research*, 2003.

[PT03b] P. Pevzner and G. Tesler. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *PNAS*, 100(13):7672–7677, June 2003.

[PT03c] P. Pevzner and G. Tesler. Transforming men into mice: the Nadeau-Taylor chromosomal breakage model revisited. In *RECOMB*, pages 247–256, 2003.

[RAS06] M. Rocchi, N. Archidiacono, and R. Stanyon. Ancestral genomes reconstruction : An integrated, multi-disciplinary approach is needed. *Genome Research*, In Press, 2006.

[San92] D. Sankoff. Edit Distance for Genome Comparison Based on Non-Local Operations. In *Proc. Third Ann. Symp. Combinatorial Pattern Matching (CPM'92)*, pages 121–135, 1992.

[San99] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999.

[San06] D. Sankoff. The signal in the genomes. *PLoS Comp. Biol.*, 2(4):e25, 2006.

[SB97] D. Sankoff and M. Blanchette. The Median Problem for Breakpoints in Comparative Genomics. In *COCOON '97: Proceedings of the Third Annual International Conference on Computing and Combinatorics*, pages 251–264, London, UK, 1997. Springer-Verlag.

[SB98] D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *Journal of Computational Biology*, 5(3):555–560, 1998.

[SCL76] D. Sankoff, R.J. Cedergren, and G. Lapalme. Frequency of insertion-deletion, transversion, and transition in the evolution of 5s ribosomal RNA. *J. Mol. Evol.*, 7:133–149, 1976.

[SDI$^+$06] D. Sherman, P. Durrens, F. Iragne, E. Beyne, M. Nikolski, and J.-L. Souciet. Génolevures complete genomes provide data and tools for comparative genomics of hemiascomycetous yeasts. *Nucleic Acids Research*, 34:D432–D435, January 2006.

[Sel74] P.H. Sellers. An Algorithm for the Distance Between Two Finite Sequences. *J. Comb. Theory, Ser. A*, 16(2):253–258, 1974.

[SHG04] A.F.A. Smit, R. Hubley, and P. Green. RepeatMasker open-3.0. `http://www.repeatmasker.org`, 1996-2004.

[Sie02] A. Siepel. An algorithm to find all sorting reversals. *RECOMB'02*, pages 281–290, 2002.

[SJSV08] C. Simillion, K. Janssens, L. Sterck, and Y. Van de Peer. i-ADHoRe 2.0: An improved tool to detect degenerated genomic homology using genomic profiles. *Bioinformatics*, 24:127–8, 2008.

[SKS$^+$04] S. Schwartz, W.J. Kent, A. Smit, Z. Zhang, R. Baertsch, R.C. Hardison, D. Haussler, and W. Miller W. Human-mouse alignments with BLASTZ. *Genome Res.*, 14(4), 2004.

[SLK$^+$07] A. Stark, M.F. Lin, P. Kheradpour, J.S. Pedersen, L. Parts, J.W. Carlson, M.A. Crosby, M.D. Rasmussen, S. Roy, A.N. Deoras, J.G. Ruby, J. Brennecke, Harvard FlyBase curators, Berkeley Drosophila Genome Project, E. Hodges, A.S. Hinrichs, A. Caspi, B. Paten, S.-W. Park, M.V. Han, M.L. Maeder, B.J. Polansky, B.E. Robson, S. Aerts, J. van Helden, B. Hassan, D.G. Gilbert, D.A. Eastman, M. Rice, M. Weir, M.W. Hahn, Y. Park, C.N. Dewey, L. Pachter, W.J. Kent, D. Haussler, E.C. Lai, D.P. Bartel, G.J. Hannon, T.C. Kaufman, M.B. Eisen, A.G. Clark, D. Smith, S.E. Celniker, W.M. Gelbart, and M. Kellis. Discovery of functional elements in 12 Drosophila genomes using evolutionary signatures. *Nature*, 450(7167):219–232, 2007.

[SM97] J.S. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing, 1997.

[SM01] A.C. Siepel and B.M.E. Moret. Finding an optimal inversion median: experimental results. In *In Proc. 1st Workshop on Algs. in Bioinformatics WABI 2001*, pages 189–203. Springer-Verlag, 2001.

[SMN+09] David Sherman, Tiphaine Martin, Macha Nikolski, Cyril Cayla, Jean-Luc Souciet, and Pascal Durrens. Génolevures: protein families and synteny among complete hemiascomycetous yeast proteomes and genomes. *Nucleic Acids Research*, 37(Database-Issue):550–554, 2009.

[SSK96] D. Sankoff, G. Sundaram, and J. D. Kececioglu. Steiner points in the space of genome rearrangements. *International Journal of Foundations of Computer Science*, 7(1):1–9, 1996.

[ST05] D. Sankoff and P. Trinh. Chromosomal breakpoint reuse in genome sequence rearrangement. *J Comput Biol*, 12(6):812–821, 2005.

[SVG01] Scalable Vector Graphics (SVG) 1.0 Specification, September 2001. `http://www.w3.org/TR/2001/REC-SVG-20010904/`.

[SVSP04] C. Simillion, K. Vandepoele, Y. Saeys, and Y.V.D. Peer. Building genomic profiles for uncovering segmental homology in the twilight zone. *Genome Res.*, 14(6):1095–106, 2004.

[SW81] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, March 1981.

[Tes02a] G. Tesler. Efficient algorithms for multichromosomal genome rearrangements. *J. Comput. Syst. Sci.*, 65(3):587–609, 2002.

[Tes02b] G. Tesler. GRIMM: genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.

[Tes04] G. Tesler. Human-mouse-rat alignments. `http://nbcr.sdsc.edu/GRIMM/HMR_Aug2003/`, 2004.

[TMS04] P. Trinh, A. Mclysaght, and D. Sankoff. Genomic features in the breakpoint regions between syntenic blocks. *Bioinformatics*, 20(1):318–325, 2004.

[TS04] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Proceedings of CPM*, pages 1–13, 2004.

[TT76] R. Tarjan and A. Trojanowski. Finding a maximum independent set. Technical Report CS-TR-76-550, Stanford University, USA, 1976.

[VSS+02] K. Vandepoele, Y. Saeys, C. Simillion, J. Raes, and Y. Van de Peer. The Automatic Detection of Homologous Regions (ADHoRe) and Its Application to Microcolinearity Between Arabidopsis and Rice. *Genome Research*, 12(11):1792–1801, 2002.

[Wil93] D. G. Wilkinson. *In Situ Hybridization: A Practical Approach.* IRL Press, 1993.

[WLTB$^+$02] R. H. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J. F. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, S. E. Antonarakis, J. Attwood, R. Baertsch, J. Bailey, K. Barlow, S. Beck, E. Berry, B. Birren, T. Bloom, P. Bork, M. Botcherby, N. Bray, M. R. Brent, D. G. Brown, S. D. Brown, C. Bult, J. Burton, J. Butler, R. D. Campbell, P. Carninci, S. Cawley, F. Chiaromonte, A. T. Chinwalla, D. M. Church, M. Clamp, C. Clee, F. S. Collins, L. L. Cook, R. R. Copley, A. Coulson, O. Couronne, J. Cuff, V. Curwen, T. Cutts, M. Daly, R. David, J. Davies, K. D. Delehaunty, J. Deri, E. T. Dermitzakis, C. Dewey, N. J. Dickens, M. Diekhans, S. Dodge, I. Dubchak, D. M. Dunn, S. R. Eddy, L. Elnitski, R. D. Emes, P. Eswara, E. Eyras, A. Felsenfeld, G. A. Fewell, P. Flicek, K. Foley, W. N. Frankel, L. A. Fulton, R. S. Fulton, T. S. Furey, D. Gage, R. A. Gibbs, G. Glusman, S. Gnerre, N. Goldman, L. Goodstadt, D. Grafham, T. A. Graves, E. D. Green, S. Gregory, R. Guigó, M. Guyer, R. C. Hardison, D. Haussler, Y. Hayashizaki, L. W. Hillier, A. Hinrichs, W. Hlavina, T. Holzer, F. Hsu, A. Hua, T. Hubbard, A. Hunt, I. Jackson, D. B. Jaffe, L. S. Johnson, M. Jones, T. A. Jones, A. Joy, M. Kamal, E. K. Karlsson, D. Karolchik, A. Kasprzyk, J. Kawai, E. Keibler, C. Kells, W. J. Kent, A. Kirby, D. L. Kolbe, I. Korf, R. S. Kucherlapati, E. J. Kulbokas, D. Kulp, T. Landers, J. P. Leger, S. Leonard, I. Letunic, R. Levine, J. Li, M. Li, C. Lloyd, S. Lucas, B. Ma, D. R. Maglott, E. R. Mardis, L. Matthews, E. Mauceli, J. H. Mayer, M. McCarthy, W. R. McCombie, S. McLaren, K. McLay, J. D. McPherson, J. Meldrim, B. Meredith, J. P. Mesirov, W. Miller, T. L. Miner, E. Mongin, K. T. Montgomery, M. Morgan, R. Mott, J. C. Mullikin, D. M. Muzny, W. E. Nash, J. O. Nelson, M. N. Nhan, R. Nicol, Z. Ning, C. Nusbaum, M. J. O'Connor, Y. Okazaki, K. Oliver, E. Overton-Larty, L. Pachter, G. Parra, K. H. Pepin, J. Peterson, P. Pevzner, R. Plumb, C. S. Pohl, A. Poliakov, T. C. Ponce, C. P. Ponting, S. Potter, M. Quail, A. Reymond, B. A. Roe, K. M. Roskin, E. M. Rubin, A. G. Rust, R. Santos, V. Sapojnikov, B. Schultz, J. Schultz, M. S. Schwartz, S. Schwartz, C. Scott, S. Seaman, S. Searle, T. Sharpe, A. Sheridan, R. Shownkeen, S. Sims, J. B. Singer, G. Slater, A. Smit, D. R. Smith, B. Spencer, A. Stabenau, N. Stange-Thomann, C. Sugnet, M. Suyama, G. Tesler, J. Thompson, D. Torrents, E. Trevaskis, J. Tromp, C. Ucla, A. Ureta-Vidal, J. P. Vinson, A. C. Von Niederhausern, C. M. Wade, M. Wall, R. J. Weber, R. B. Weiss, M. C. Wendl, A. P. West, K. Wetterstrand, R. Wheeler, S. Whelan, J. Wierzbowski, D. Willey, S. Williams, R. K. Wilson, E. Winter, K. C. Worley, D. Wyman, S. Yang, S. P. Yang, E. M. Zdobnov, M. C. Zody, and E. S. Lander. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562, December 2002.

[WS97] K. Wolfe and D. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387(6634):708–713, 1997.

[WSO$^+$05] M.T. Walter, C. Sobrinho, T.G. Oliveira, S. Soares, G. Oliveira, E.S. Martins, and M. Fonseca. Improving the algorithm of Bafna and Pevzner for the problem of sorting by transpositions : a practical approach. *Journal of Discrete Algorithms*, 3(2-4):342–361, 2005.

[WZLM05] L. Wang, D. Zhu, X. Liu, and S. Ma. An $O(N^2)$ algorithm for signed translocation problem. In *APBC*, pages 349–358, 2005.

[YAF05] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

[ZZAS08] C. Zheng, Q. Zhu, Z. Adam, and D. Sankoff. Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes. *Bioinformatics*, 24:96–104, 2008.