Algorithms and methods for large-scale genome rearrangements identification

Jose Antonio Arjona Medina

Department of Computer Architecture
University of Málaga

This dissertation is submitted for the degree of
Doctor of Philosophy in Computer Science

Artificial Intelligence and Software
Engineering Doctorate Program

June 2017

UNIVERSIDAD DE MÁLAGA

AUTOR: José Antonio Arjona Medina

http://orcid.org/0000-0002-5033-4725

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

To my parents Julia and Jose Antonio and my sister Julia María …

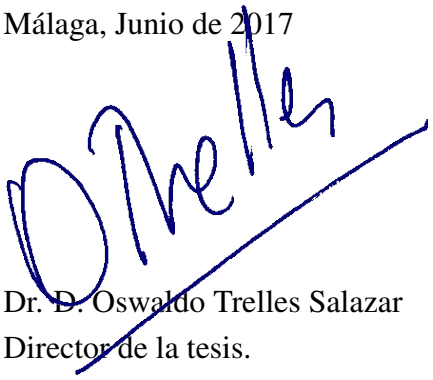
Declaration

Dr. D. Oswaldo Trelles Salazar.
Profesor Catedrático del Departamento de
Arquitectura de Computadores de la
Universidad de Málaga.

CERTIFICA:

Que la memoria titulada “Algorithms and methods for large-scale genome rearrangements identification”, ha sido realizada por D. Jose Antonio Arjona Medina bajo mi dirección en el Departamento de Arquitectura de Computadores de la Universidad de Málaga y constituye la Tesis que presenta para optar al grado de Doctor en Ingeniería Informática.

Málaga, Junio de 2017



Dr. D. Oswaldo Trelles Salazar
Director de la tesis.

Jose Antonio Arjona Medina
June 2017

Acknowledgements

This work is the result of many years of work and the outcome of a learning process that I went through with the help of many people that I want to thank.

First, to my parents for providing me education and their guidance, encouragement and great example throughout my whole life made me develop my potential and made this thesis possible in the first place. My sister, whose love and support sustained me throughout this work, also deserves my wholehearted thanks..

Also to my teachers in high school, especially my math teacher Paco Laure who woke up my attention to science. I would like to thank my biologist friends Victor Ruiz and Paulina Flores for many hours of inspiring conversations, Daniel Cebrián for his support and advices and Elke Strobl for her valuable corrections and unconditional moral support.

I would like to express my gratitude to my old colleges at RISC Software, especially to Michael Krieger and Paul Heinzlreiter for their corrections, and my new colleagues at the Institute of Bioinformatics at the Johannes Kepler University in Linz, Austria, especially Gundula Povysil and Andreas Mayr for their comments and suggestions in some sections of this thesis.

Over the last four years I have spent the most productive time at the BITLAB research group. I am very grateful to all of them: Antonio Muñoz, Eugenia Ulzurrun, Esteban Pérez, Sergio Díaz and especially to Oscar Torreño who I spent the most time working with and who was always there to offer his help and valuable advices.

I would like to express my special thanks of gratitude to my professor Dr. Oswaldo Trelles, who offered me the opportunity to work in his group and learn from his invaluable experience. I owe him a very special gratitude for being my advisor and a continuous supporter of my studies and related research. I am very grateful for his patience, motivation and immense knowledge. His guidance helped me all the time in planning, organizing and writing this thesis. I could not have imagined a better advisor and mentor for my Ph.D. study.

Finally, must thank all the financial support offer by the Instituto de Salud Carlos III ("Instituto Nacional de Bioinformática" (INB-GN5), RIRAAF (RD07/0064/0017 and RD12/0013/0006) and Plataforma tecnológica de Recursos Biomoleculares y Bioinformáti-

cos (PT13/0001/0012)); Junta de Andalucía (Plataforma computacional de alto rendimiento para la gestión y análisis de datos clínico-genéticos (P10-TIC-6108)) and the European Commision funded project Mr.Symbiomath, under the 7th framework programme grant agreement no. 324554, whose funds made me enjoy a two year investigation stay at RISC Software GmbH company part of the Johannes Kepler University in Linz.

Abstract

DNA rearrangements are one of the main causes of evolution and their effects can be observed on new species, new biological functions, etc. Short-scale genome rearrangements such as insertions, deletions or substitutions have been profusely studied and there are accepted models to detect them.

However, methods to identify large-scale genome rearrangements (LSGR) still suffer from limitations and lack of precision mostly because an accepted formal definition of Synteny Block (SB) is still missing. The SB concept refers to conserved regions that share the same order and strand between two genomes. There exist some methods to detect SBs, but they avoid dealing with repetitions or restrict the search just for the coding part in order to keep the model simple. The refinement of SBs' edges is also an open problem.

This thesis by compendium addresses the formal definition of SBs starting from High-Scoring Segments Pairs (HSPs), which are accepted and well known. The first target was focused to the SB detection as a combination of HSPs, including repetitions, which increased the model complexity. As a result, a more precise method came up improving the state-of-art quality performance [6].

This method applies rules based on SB adjacency and also allows detecting LSGR and categorizes them as inversions, translocations or duplications. As a consequence, a framework to deal with LSGR for organisms with one chromosome was developed.

Afterwards, in a second article [5], the framework is applied to refine the SBs' edges. In a novel approach, repetitions flanking the SBs are used to exploit sequence redundancy in order to refine SB boundaries. Performing a multiple alignment of those repetitions, an identity vector of the consensus sequence and the identity vector for SBs are calculated. A Finite State Machine is designed to detect transition points in the difference between such vectors. As a result, transition points demarcate the beginning and ending of SBs [5]. The method is also shown to be helpful for detecting breakpoints (BP). The BP appears as the region (or point) between two adjacent SBs. The method does not force the BP to be a region or a point but depends on the alignment transitions within the SBs and repetitions.

The method is applied in a third manuscript, which faces a metagenome analysis use case [76]. It is well known that the information stored in current databases does not necessarily correspond to the uncultured samples contained in a metagenome, and it is possible to imagine a LSGR occurring in such organisms making difficult the mapping of reads. It shows that metagenome reads mapping over exclusive regions (regions that are not share with other genomes) from a certain genome strongly support the genome presence in the metagenome. Exclusive regions are easily derived from multiple genome comparison (MGC), as the regions not being part of any SB.

A SB definition under a MGC environment is more precise than a pairwise definition, since it allows for a SB refinement following a similar approach described in the second publication (using SBs in different genomes instead of aligning repetitions). This SB definition also solves the contradiction of the BPs definition mentioned in the second publication [5], which states that under pairwise SB definition, a region detected as BP in one comparison can be part of a SB in another comparison.

The SB definition under MGC environment also provides accurate information for the rearrangement reconstruction towards an approximation of the true last common ancestor. In addition, it provides a solution for the granularity problem in the SB detection: starting with small and well-conserved SBs and through rearrangement reconstruction gradually increasing the length of the SB.

Expected results from this line of work point towards a definition of a metric aimed to obtain a more accurate inter-genome distance, combining similarity between sequences and rearrangements frequency.

Table of contents

List of figures **xiii**

1 Introduction **1**
- 1.1 Research objectives . . . 3
- 1.2 Results . . . 3
- 1.3 Contribution of this thesis . . . 4
- 1.4 Outline of the Thesis . . . 5

2 Background **7**
- 2.1 Synteny Blocks and Break Points . . . 7
- 2.2 The granularity problem in Synteny Block detection . . . 8
- 2.3 The Break Point definition . . . 9
- 2.4 Sorting Permutation Problem . . . 10
- 2.5 Some remarks regarding current Sorting Permutations methods approaches 12

3 Systems and Methods **13**
- 3.1 Introduction . . . 13
- 3.2 Synteny Block definition . . . 14
- 3.3 The Unitary Conserved Element problem . . . 19
- 3.4 Transitivity property of Synteny Blocks: Inferring less conserved HSPs . . . 22
- 3.5 Rearrangements detection and reconstruction via Unitary Synteny Block . . . 24
  - 3.5.1 Synteny Block concatenation . . . 24
  - 3.5.2 Insertions and deletions . . . 25
  - 3.5.3 Duplications . . . 25
  - 3.5.4 Inversions . . . 26
  - 3.5.5 Transpositions . . . 27

4 Results and discussion **29**

5 Conclusions and future work **37**
5.1 Conclusions . . . 37
5.2 Future work . . . 38
5.2.1 Detecting Break Points using a Machine Learning approach . . . 39

References **41**

Appendix A Sequence comparison algorithms **49**
A.1 Pairwise sequence alignments . . . 49
A.2 Multiple sequence alignment algorithms . . . 50
A.3 New strategies: homology search methods . . . 51
A.4 Statistical Significance . . . 51
A.5 Dealing with repetitions . . . 52

Appendix B Methods in the State of art for Synteny Block detection **55**

Appendix C Sorting permutation problem state of art **59**
C.1 Sorting by reversals . . . 59
C.2 Sorting by transpositions . . . 60
C.3 Weighted operations and other evolutionary events . . . 60
C.4 DCJ . . . 61

Appendix D Publications **63**
D.1 Publication 1 . . . 63
D.1.1 Summary . . . 63
D.2 Publication 2 . . . 65
D.2.1 Summary . . . 65
D.3 Publication 3 . . . 67
D.3.1 Summary . . . 67

Appendix E List of 68 Mycoplasmas **69**

Appendix F Resumen en español **71**

List of figures

3.1 Comparisons (A,B) and (B,C) show a conserved pair that is not detected in comparison (A,C). 13
3.2 Three comparisons involving A,B and C sequences. (A,C) and (A,B) show an inversion that is not present in (B,C). 14
3.3 Representation of Block Element. α^h and α^t represent the head and the tail positions of the block in the full sequence. 15
3.4 Representation of the set of Unitary Block Elements A_{Φ_A} in the sequence Φ_A. 16
3.5 A graphic representation of Conserved Elements from HSPs. 17
3.6 Graphic representation of three Synteny Elements. Synteny Element π_1 links α_1,β_1 and γ_1 Unitary Conserved Elements. 18
3.7 Representation of Break Point. α_1, α_2 and α_3 are Unitary Conserved Elements. 18
3.8 Representation of the trimming process. A) Two overlapped HSPs. B) Result of the trimming process. The two HSPs have been split into four pairs of Conserved Elements. Two of them are still overlapped. C) New overlapped Conserved Elements trigger a new trimming process. D) Final result of the recursive trimming process. The final pairs of Conserved Elements do not overlap. 21
3.9 Representation of the trimming process in a multiple comparison. In the comparison *AB* there is an inversion, that triggers a trimming process in the comparison *BC*. As a result, another trimming process is triggered in comparison *DC*. 22

4.1 Average length, average percentage of identity, and coverage from all against all comparison of 68 mycoplasmas. Grouped by closely, remote and poorly related species. The X and Y axis represent coverage (as percentages) in the sequences. Each point represents a comparison. The color represents the average identity in the comparison. The shape represents the average length of the detected blocks. On the top, results from our method, Gecko-CSB. On the bottom, results from progressiveMauve. In the image it can be observed that Gecko-CSB works better in terms of getting more coverage over the sequences at the similar level of identity, especially in those comparisons of poorly related species. 31
4.2 Frequency distribution of Breakpoint length. 32
4.3 CSBs before and after the refinement. A) Selection of the Region of Interest (ROI), between two Computational Synteny Blocks (CSB). B) Representation of the virtual CSBs. C) Result after the refinement process. We also detect BPs and extract PRASB and GAP sequences to analyse the accuracy of the method. PRASB and BP have the same length. For more details of the refinement process visit the second publication [5]. 33
4.4 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects one SB. (b) progressiveMauve detects three SBs (B,C and D). The reasons of this difference are explained in the main text. 34
4.5 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects three SBs (A,B and C). (b) progressiveMauve detects one large SB. 35

5.1 GenomeA and GenomeB are included in the public database. Genome C is not in the database. If a certain metagenome read comes from Genome C specie, it might match better in the Last Common Ancestor-ABC than in the genomes A or B, because the genome C is not present in the database. . . . 39
5.2 The input is encoded as one-hot vector. After the LSTM layer, we use a fully connected layer to combine all the LSTM cells outputs to produce a single output. 40

Section 1

Introduction

New, massively parallel data acquisition technologies in many fields of science are producing huge quantities of data that needs to be processed and analysed, in the context of all the challenges that this situation entails. In many cases, current formal models are not valid anymore or they are not robust enough to deal with the new challenges that the massive available data and the type of the data bring out. The problem is not only the amount of data, but also the diversity of such data (i.e. DNA, RNA or amino acid sequences) that needs to be tackled with new methods and strategies adapted to get satisfactory results.

The field of sequence analysis has traditionally been working with short sequences such as genes and proteins producing widely accepted and profusely used and stable models. Nowadays, the availability of full genomes have risen the area of Comparative Genomics, but the simple translation of methods from sequence to genome analysis seems not to be valid for whole genome analysis. Besides base changes, short insertions and deletions in genes, the full-genome analysis involves large rearrangements, where the statistical models developed for proteins under assumptions of a general scoring system do not apply.

Algorithms and models designed for short sequences cannot be directly exported to deal with long sequences like full genomes. Genomes are not only much longer than genes or proteins, but are also more complex. New aspects to take into account emerge, such as Synteny Blocks (SBs) or large-scale genome rearrangement (LSGR) events, which are only observable working with whole or large parts of genomes. Therefore, these new challenges are not just only at technical level but also at conceptual level, which implies a higher level of complexity and understanding.

To illustrate that, let's think the genome as a library where all the information that an organism needs can be found. Over evolution, some shelves, books, chapters, paragraphs, sentences, words or letters (units of information) have been lost, duplicated, merged, split or

shifted (operations). As we can see, different levels of information and operations come into play.

This thesis is a compendium of three articles recently published in high impact journals, in which we show the process that led us to propose the definition of the Elementary Unit of conservation (conserved regions in the genome that are detected after a multi comparison), as well as some basic operations (inversion, transposition and duplication). The three articles are transversely connected by the detection of SBs and rearrangement events (see background in section 2), and strongly support the necessity of the framework which is described in section 3. Indeed, the intellectual work carried out in this thesis and the conclusions provided by these publications have been essential to fully understand that a proper definition of SB is the keystone of the success in many Comparative Genomics methods.

The first publication proposes a framework to detect SBs and (LSGR) from a pairwise point of view. In this framework, SBs are detected dealing with repetitions and small fragments whilst other methods generally do not consider them for the sake of simplicity (see section 2 for more details). We also propose a set of rules to identify LSGRs based on some properties that SB must to fulfill.

Under the framework defined in the first publication, the refinement of the SB borders using repetitions is addressed in a second work. Indeed, the method takes advantage of the repetitions, since all of them together constitute a reliable source of information to determine the exact point where the repetition area ends. This information was also useful to determine SB boundaries and therefore Break Point (BP) regions.

The third publication addresses the metagenome analysis problem using the previous findings. In more details, we proposed a method to estimate differences between genome abundances given two metagenomes. To achieve that goal, reads contained in the metagenome must be correctly assigned to a certain genome. Most probably, genomes in databases are not exactly the same that the genomes contained in the sample from which reads are extracted. Therefore, a perfect match read-genome is highly unlikely. Moreover, one read usually matches in several genomes, what increases the complexity of choosing the true set of genomes contained in a metagenome. However, after detecting SBs over the collection of genomes, those reads that match in non-SB regions at specific genome regions (regions that are not in other genomes) would be considered as true matches, a strong evidence of the presence of such genome in the metagenome. For more details, see Genome-specific experiments section in [76].

Currently, we are extending the framework from pairwise to multiple comparison. In a multiple comparison scenario, SBs can be refined with much more precision because the number of sub-sequences used for the alignment increases as the number of compared

genomes does. The BP detection also improves as a consequence of a better SBs refinement process, but also because under a multiple comparison environment it is possible to detect more BPs than in a pairwise comparison (for more details, see section 3). Additionally, the new framework allows the reconstruction to the last common ancestor.

1.1 Research objectives

This thesis is aimed to propose solutions for the following problems:

- Definition and detection of SBs
- Detection and identification of large scale genome rearrangements
- Detection of Break Points
- Refinement of the SBs borders
- Application of the SB detection for metagenome analysis.

1.2 Results

The global results achieved in this thesis can be summarised as the design and implementation of a pairwise framework that:

- SBs detected after the comparison and the refinement process have more coverage and enhance the quality than state-of-the-art methods.
- It is designed for dealing with overlapped HSPs, one of the main drawbacks in current software tools.
- It is able to detect and organise repetitions. For instance, interspersed repeats or tandem repeats.
- It is able to work in much complex environments than state-of-the-art methods (i.e. overlapped fragments, small fragments, highly repetitive fragments)
- It is able to work with HSPs collections provided by other programs. It is not necessary to apply any previous filtering process to simplify the input of these problematic fragments.

- It is non-parametric in the sense that it does not need parameters to detect SBs or repetitions. In our case, all parameters are internally estimated based on distributions. We also use some formulas to suggest parameter values to be used in the process.
- It is demonstrated to be a robust method able to deal with genomes related at different levels of similarity.
- It produces a more accurate refinement using repetitions flanking the SBs.
- It enables a more precise identification of genomes in metagenome samples using SBs and BPs.

A prototype for multiple comparison framework is also proposed in this thesis. This framework enables a better detection and refinement of SBs and BPs. We also propose a set of rules to identify genome rearrangements and operations to perform the reconstruction of the rearrangements history in order to estimate the last common ancestor.

1.3 Contribution of this thesis

The main contribution of this thesis is the proposal of a novel framework to handle properly large scale rearrangement events in a multiple genome comparison, combining homology search methods, sequence alignment and sorting permutation methods. This framework includes a refined definition of the SB concept (what we have defined as Unitary Synteny Block Element), and algorithms to detect them and identify SBs rearrangements. In addition, a definition of Break Point is provided.

The results of this work reinforced the idea of trimming fragments to properly identify repetitions and building SBs afterwards in terms of certain previously defined properties. The idea of using repetitions to refine SBs brought us to think that this strategy could be also applied in multiple comparison scenarios.

The sum of the work brings us to the following conclusions:

1. The SB concept has a dual nature: the block content (the sub-sequence) and the relation with other blocks (the synteny) from different genomes.
2. Both parts (the block and the synteny) are equally necessary for the rearrangement reconstruction.
3. The SB concept should be redefined in a N-dimensional space rather than pairwise (2-dimensional space) because 1 and 2.

1.4 Outline of the Thesis

This thesis is structured as follows:

- Section 1 provides a high-level description of the work carried out in this thesis.
- Section 2 provides the needed background to understand the following sections and reviews state-of-the-art methods.
- Section 3 describes the proposal of the new framework with the SB definition and algorithms in a multiple comparison environment.
- Section 4 describes the global results of this thesis in more detail.
- Section 5 lists the conclusions derived from this work and sets future work lines.

This thesis also contains four Appendixes, which go in depth in some topics that might be interesting for the reader:

- Appendix A contains a brief introduction to sequence alignment methods.
- Appendix B provides a summary about methods to detect SBs.
- Appendix C reviews sorting permutations problems.
- Appendix D contains a copy of the three publications.
- Appendix E contains the list of the 68 Mycoplasmas used in the experiments.
- Appendix F contains a summary in Spanish

Section 2

Background

This section coarsely describes the problematic regarding SBs detection and rearrangement reconstruction (in sections 2.1 and 2.2 respectively) that motivated the submitted publications in appendix D.3.1. In order to keep the section fluid and linear for the reader, some specific sections that might be familiar to the reader have been included as separate Appendixes at the end of the thesis.

- Appendix A contains a brief description about Sequence Comparison methods, statistical significance and methods to deal with sequence repetitions.
- Appendix B reviews the main methods in the state-of-the-art for SB detection.
- Appendix C provides a review in sorting permutation problems.

2.1 Synteny Blocks and Break Points

The possibility to work with full genome sequences made possible an abstraction jump from nucleotides level of information to higher units of information like genes, blocks of genes or chromosomes.

As it is commented in Appendix A, comparative methods had to implement strategies to reduce the space search due to the increase in sequence length. One of these solutions is based on finding K-mers present in the sequences under comparison, as perfect matches for a given length (K)[74], or allowing some mismatches [3]. Using such matching K-mers as seed points, the HSPs are extended. Increasing the value of k, and relaxing the parameters that control the HSP extension – e.g. decreasing the similarity threshold - we will observe that it is possible to detect more HSPs. In addition, some of these conserved regions could

be grouped under the form of blocks. This evidence is more obvious when we represent graphically the results of HSPs calculation changing the similarity threshold.

Another important issue is the presence of sequence repetitions, which increases the complexity of methods to detect such blocks. For more details, see Appendix A section 4. These blocks that we are referring to are known as SBs. The SBs concept allows us to describe Large Scale Genome Rearrangements (LSGR) between sequences, and therefore design methods to detect them. There is still not an accepted formal definition of SB. Some authors base the definition on genes, whilst others define it based on homologous markers.

Nadeau and Taylor [68] first introduced the notion of conserved segment. In their work, they stated that conserved segments are regions where genes content is the same and gene order is conserved. However, many other definitions can be found in the literature:

- "A set of equal to or larger than a minimum number of gene pairs" [73],
- "Segments of chromosomes containing orthologous markers in the same or reverse order in the two genomes" [25],
- "Conserved blocks of genes on chromosomes of related species" [97],
- "Conserved regions corresponded to pairs of segments, one in each genome, that are orthologous and have not been rearranged in either lineage" [57],
- "A maximal sequence of genes on a chromosome of genome A, occurring unchanged in genome B" (assuming that genomes A and B have the same gene content) [96],
- "Segments that can be converted into conserved segments by micro rearrangements. The Synteny Blocks do not necessarily represent areas of continuous similarity between two genomes. Instead, they usually consist of short regions of similarity that may be interrupted by dissimilar regions and gaps" [77].

This lack of a formal definition is a clear influencing factor of the different results produced by available software, and questions the robustness of one of the basic units of information in Comparative Genomics, as it was stated in Ghiurcuta's work [39]. In addition, without a formal definition, comparison between methods is extremely difficult to carry out.

2.2 The granularity problem in Synteny Block detection

The granularity problem appears when the detection of SBs depends somehow on a block-length or similarity threshold. There is a compromise between calculating small well-conserved blocks, and large blocks by relaxing the percentage of identity.

The first approach is useful to analyse particular regions in the genome and to detect small evolutionary events such as insertions, deletions and mutations, whereas the second approach is useful to get the whole picture of big evolutionary events in a comparison (i.e transpositions, inversions or translocations). Ghiurcuta et al. mentioned this paradox as the granularity attribute of SBs [39].

This problem evidences the different levels of information that SBs definition has been trying to cover with unsatisfactory results. As consequence, most methods include user parameters to solve the granularity problem (see Appendix A.3 for more details).

2.3 The Break Point definition

A SB is defined as a relation between two conserved regions in the sequence of two different species, in terms of homology or similarity. A BP is usually known as the region in between two SBs that have suffered a rearrangement due to a LSGR [57, 29, 18].

Many studies support that rearrangements do not happen randomly but follow an unknown model [82, 63, 70]. Some regions of the sequence seem to be more fragile [17] or predispose to suffer a LSGR (*hotspots*) [11, 2]. Indeed, these BPs can be reused [77, 80] and their reuse rate is strongly linked with the resolution in which SBs are detected [7]. Therefore, if a BP seems to depend on the "fragility" of the specific regions in the sequence, then it should not be defined as a relation between two specific regions of two sequences (as a SB is defined), although so far a comparison method is needed to detect them.

Current methods based on sequence comparison, detect SBs by joining or chaining High Score Segment Pairs, and when they refine their borders, they try to expand the SB borders by maximizing a target score function. This would mean that the BP region is a region without similarity. However, following the previous reasoning about BP definition, it implies that BPs regions do not have to be necessarily regions with almost no similarity. Two species could share the same BP and therefore, the sequences would have some level of similarity. We think that when refining SBs, they can be trimmed as well as expanded after the refinement process.

This reasoning would be a contradiction if we base the SB definition over a pairwise comparison scope, and BP as the region in between two SBs. For example, a BP might be between two SBs in one pairwise comparison i.e A,B and included within a SB in other comparison A,C, which would lead us to an incoherence. As soon as we incorporate a multiple comparison environment in the definition of SB, this inconsistency disappears (see section 3).

2.4 Sorting Permutation Problem

Sorting permutations problems face the challenge to transform one sequence into another by permutations [1]. Usually, these methods have been proofed to be NP-hard [21]. The reconstruction of the history of rearrangements can be viewed as a sorting permutation problem where one sequence is transformed into another by certain operations. Methods to sort permutations have been widely applied to sort genome rearrangements in a genome comparison context [60]. For a deep review, A. Christie wrote in 1998 a convenient thesis about *genome rearrangement problems*, in which methods to date were analyzed [1]. A more recent review was written by Li *et al.* [60].

Generally, these methods aim to calculate a sequence distance based on the parsimony criterion: find the minimum number of operations to transform one sequence into another. However, in this section we will focus on the algorithmic part of these methods, leaving the sequence-distance problem.

Most methods assume that the sequences to sort are a list of integers, in which each number represents regions between sequences. These regions can represent homologous genes, homologous markers, HSPs, etc. In some cases, existing methods allow signed numbers to include the representation of synteny regions that have been found in the reverse complementary sequence.

The set of available operations, or better said, defined in the model description, also varies from one method to another. In a chronological overview, early approaches only considered one type of operation (either reversals or transpositions) [56, 10]. In dribs and drabs, they started to design methods using different operations (such as block interchange) or combining them [87].

The limitations in the model from the features' input point of view have also changed over time. First methods were designed to work only with linear genomes, without signed elements. Allowing signed permutations was one of the first model limitations solved. Later on, new methods were developed taking into account different topographies, such as circular genomes or chromosomes, which involve more than one sequence and new operations and were improving the state of art in this field of study (see Appendix C).

However, most methods – even the latest ones – keep trying to sort permutations in a pairwise way. They assume that one is the reference and all the permutations take place over one sequence. From a logical point of view this does not make sense. Firstly, because in the genome evolution process, both species suffer rearrangements in parallel. Secondly, because even in the case that there is only one rearrangement to revert, there is no way to know –

from a pairwise comparison – in which specie the rearrangement has happened. Therefore, a multiple comparison is needed to support the hypothesis.

Cross-sectional at all methods is the fact that they do not use any inside-block information from the permutations they try to sort. Traditionally, the sorting permutation problem in a Comparative Genomics context has been faced as a "pure" sorting permutation problem. That is to say, methods to solve this problem are easy to adapt to solve the same problem in other context since there is no specific information at sequence level (i.e. similarity between blocks, or length) included in the model description. Although this approach yielded to solve many theoretical problems in the combinatorial analysis, their solutions come into conflict with other approaches based on inside-block information, like multiple sequence alignment.[1]

In section 3, we describe the definition of SB in which our framework is based on. We argue that SBs notion has two concepts closely linked: *block* and *synteny*. Although it might seem a useless tautology, SB definitions in other methods ignore the dual nature of the concept. As a consequence, methods to solve sorting permutation problems in a comparative genomics context just look at one part of the *synteny* side of the SB, obviating the *block* features. In a roughly simplification, they sort the "synteny" permutations forgetting the *block* content.

On the other hand, most accepted methods to calculate inter-genome distance are based on block content information [58], by aligning sequences or extracting features (alignment-free methods [46, 20]) overlooking any information extracted from rearrangement reconstruction. In this case, they look just at the *block* side (the sequence content) forgetting one part of the *synteny* dimension.

Rearrangements reconstruction (or sorting permutations) in comparative genomics is extremely connected with relation at the sequence level. Indeed, both approaches share the same goal – explaining the evolution by phylogeny – and therefore, both should be complementary and coherent (see foot note).

Our proposal for sorting the permutation problem is based on the SB definition provided in section 3. It combines block content with synteny relation to perform a coherent operation in the reconstruction process: Synteny relation determines where there is a rearrangement to be sorted; and Block content determines which block (or blocks) must be permuted.

[1] When there is a high rate of rearrangements, and these rearrangements are defined in the model, both methods tend to converge [93].

2.5 Some remarks regarding current Sorting Permutations methods approaches

Methods that only take into account reversals or transpositions cannot be applied in real comparison problems where we can find both. Most methods, which combine them, do not take into account other kind of rearrangements, for instance, duplications.

Most methods sort permutations taking one genome as a reference and the other one as the "unsorted". Then, making operations, they transform one genome into the reference. This approach would assume that one sequence evolved from the reference. Evolutionary events are relationships between two species, but we cannot assume that all this evolutionary events have happened exclusively in one species.

Generally, these methods are based on the order that common genes appear in two species. They were not designed for working with SBs from sequences. Sorting permutations problems do not take into account any information extracted from sequences, for example, similarity between SBs. Most of them are leaded by minimizing the sum of weight of operations, which in most cases, are poorly justified [60].

The new framework that we propose enables to reorder these permutations in both sequences, based on breakpoints and similarity between blocks. At every step it is made, one sequence is transformed into the intermediate sequence. At the end of the process, both original sequences are transformed into a different sequence, which would be close to the "last common ancestor" one.

Section 3

Systems and Methods

3.1 Introduction

As described in section 2, there is still no formal accepted definition about Synteny Block (SB). SB concept refers to conserved block that maintain the same order, and it only has a meaning in a comparison environment. As a relation, SBs depend on the sequences, for instance: single nucleotides, genes or other kind of unit of information to be compared within the sequence. Methods to detect SBs are generally based on similarity between regions, controlled by user-defined parameters, where length and statistical significance play an important role. Due to these constrains, SBs identified in sequence A might be different depending on the other sequence we are comparing with. This is to say, SBs detected through comparison of (A,B) and (A,C) may be dissimilar.

For instance, in a multiple comparison, less conserved pairs might not be detected due to similarity constraints as illustrated in Figure 3.1. These pairs, which have not been detected by conventional methods, could be inferred from (B,C) and (A,B) comparisons to (A,C). The method to infer less conserved pairs (similar to calculate intermediate sequences [4]) is described in section 3.4.

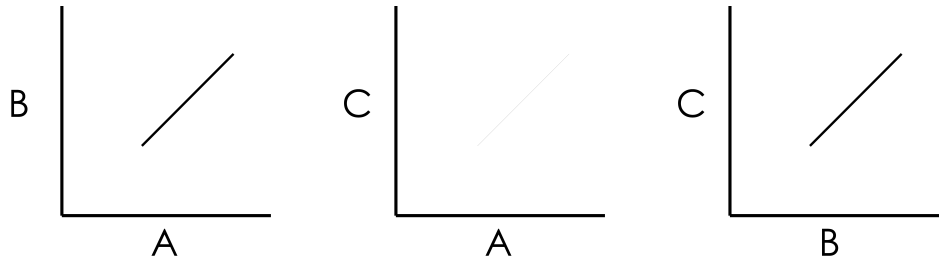


Fig. 3.1 Comparisons (A,B) and (B,C) show a conserved pair that is not detected in comparison (A,C).

Furthermore, when rearrangements happen (or Large Scale Genome Rearrangements (LSGR)), genomes also change the number of conserved pairs that comparison methods are able to detect. Figure 3.2 shows three comparisons involving A, B and C sequences. Let's assume that between C and B there are no rearrangements, and on the contrary there is an inversion between (A,C) and (A,B). Therefore, comparison methods would detect three different conserved pairs in the comparisons (A,C) and (A,B) and only one in (B,C). This also would lead us to the BP contradiction explained in section 2.3.

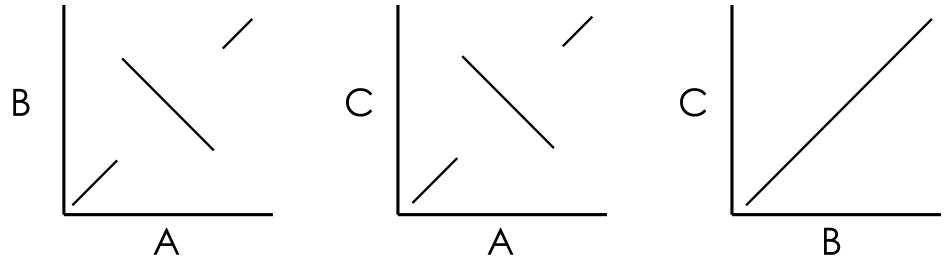


Fig. 3.2 Three comparisons involving A, B and C sequences. (A,C) and (A,B) show an inversion that is not present in (B,C).

Hence, SB definition must be built from the multiple comparisons perspective, rather than pairwise comparison.

As it has been described in section 2, there are many aspects to take into account when facing the SB detection, like repetitions between and within sequences or the granularity problem. The proposed definition of SB not only models repetitions between sequences, but it also includes repetitions within the sequence. Regarding the granularity problem, our strategy relies in the rearrangements history reconstruction, instead of hyper parameter to fine-tuning the SB size. Nevertheless, hyper parameter can play an important role when computational resources are limited. As we reviewed in section 2, rearrangement history reconstruction has been addressed from the sorting permutation point of view, mostly in pairwise comparison. In our approach, a solid definition of SB lets us design operations that will progressively increase the length of SBs while at the same time rearrangements are detected. Thus, SB could be the unit to detect rearrangements at invariant scale, which in turn solve the granularity problem.

3.2 Synteny Block definition

There are two categories strongly related to each other in the SB concept, that cannot be explained one without the other. The first one is concerning the sequence: there are certain regions in one sequence that appear in other sequences. This category concerns to the *Block*

essence. The second one is the relation among these regions: what kind of rearrangement they have suffered and what degree of conservation they share. This second category concerns to the *synteny* relation.

However, in SB definitions, these categories are not yet separated. One of its consequences is that methods to detect SBs produce widely different results and it is therefore extremely difficult to compare them (see section 2.1 for more details). Another consequence is that methods to trace back rearrangements generally do not use any inside-block information (see section 2.2). And finally, methods to estimate distances between sequences either align sequences or use some metrics related to the rearrangement reconstruction, but generally they do not combine them.

In this subsection, these two categories that have appeared together in all existing definitions are differentiated in order to build up a simple yet solid definition of Synteny Block. This definition uses the concepts of Block Element, Unitary Block Element, Unitary Conserved Element, Unitary Synteny Element and Break Point.

Definition 1. ***Block Element***

A Block Element is an arbitrary subsequence in the sequence, formally defined as:

- $\alpha = (\alpha^h, \alpha^t)$
- *where* $\alpha^h, \alpha^t \in \mathbb{N}$ *and represent the head and the tail absolute positions of the block in the full sequence (see figure 3.3).*

And it has the following properties:

1. $\alpha^h < \alpha^t$
2. $|\alpha| = \alpha^t - \alpha^h$
3. $|\alpha| \geq 0$ *(As a consequence of 1 and 2)*

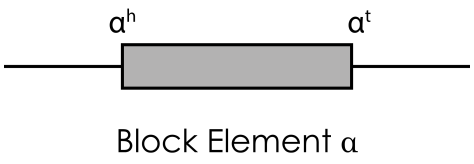


Fig. 3.3 Representation of Block Element. α^h and α^t represent the head and the tail positions of the block in the full sequence.

Definition 2. ***Unitary Block Element***

Let $\Phi = \{\phi_A, \phi_B, \phi_\Gamma, ..., \phi_{N_\Phi}\}$ be a set of N_Φ sequences. Let $A_{\Phi_A} = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_{N_A}\}$ be a set of N_A Block Elements in the sequence ϕ_A. Then, the set A_{Φ_A} is a ordered set of Unitary Block Elements if the following property is fulfilled:

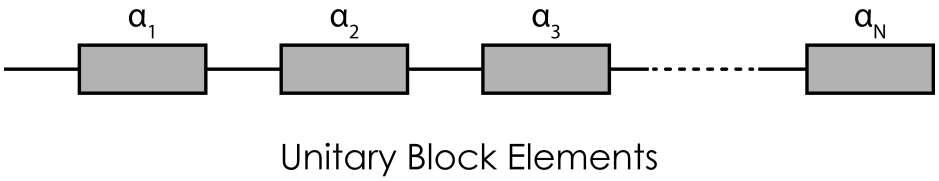


Fig. 3.4 Representation of the set of Unitary Block Elements A_{Φ_A} in the sequence Φ_A.

$$\forall \alpha_i, \alpha_{i+1} \in A_{\Phi_A} : \alpha_i^h < \alpha_i^t < \alpha_{i+1}^h \tag{3.1}$$

In other words, a Unitary Block Element is a Block Element that does not overlap with others Unitary Block Elements (see figure 3.4).

Definition 3. ***Unitary Conserved Element***

A Unitary Conserved Element is a Unitary Block Element originate from comparison[1] *like High-Score Segment Pairs, homology pairs, homologous genes...*

What makes this part of the definition different to other SB definitions is the constraint imposed by the Unitary Block Element. Therefore, the key-point of this definition is how to transform the blocks coming from comparison methods in a way that fulfill the property 3.1. Section 3.3 addresses the problem of the transformation from HSPs to the set of Unitary Conserved Elements.

The second part of the SB definition regards the relation of the Unitary Conserved Elements among them: the *synteny* part. The *synteny* plays as the link between two regions detected as similar regions. For example, conserved block elements α and β are detected by a comparison method (α is *similar* to β). The way we define to be *similar* depends on the comparison method. However, we will say that α and β has the same *synteny* (π).

Definition 4. ***Unitary Synteny Element***

A Unitary Synteny Element is a set of Unitary Conserved Elements from different sequences that share the same synteny (π), and fulfill the following properties:

- *More than one Unitary Block Element can be present in the same sequence.*

$$\pi = \{\alpha, \alpha', \alpha'', ..., \beta, \beta', \beta'', ..., \gamma, \gamma', \gamma'', ..., \omega''\} \tag{3.2}$$

[1] See figure 3.5

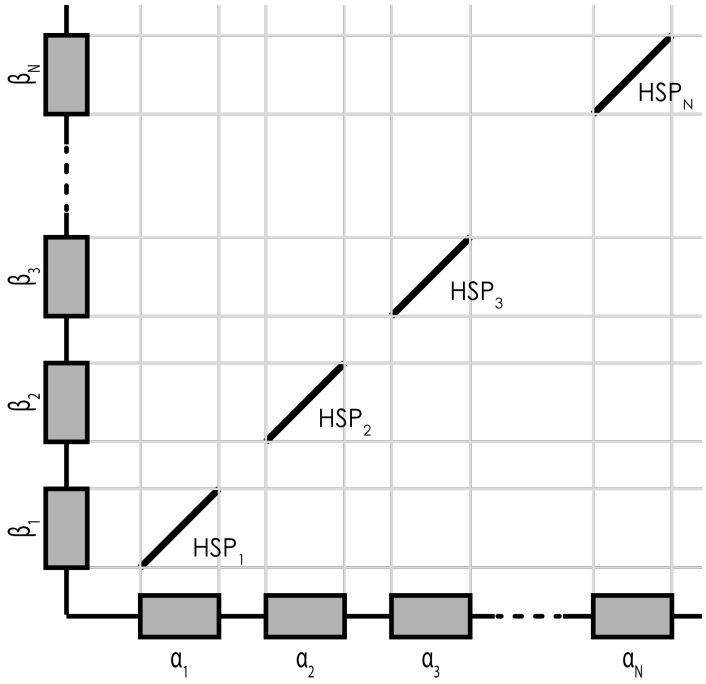


Fig. 3.5 A graphic representation of Conserved Elements from HSPs.

- *Unitary Block Elements in the Unitary Synteny Elements have the same module*

$$|\alpha| = |\alpha'| = |\alpha''| = ... = |\beta| = |\beta'| = |\beta''| = ... = |\gamma| = |\gamma'| = |\gamma''| = ... = |\omega''| \quad (3.3)$$

- *The relation between Unitary Conserved Elements and Unitary Synteny Elements is bijective. Let* $\Pi = \{\pi_1, \pi_2, ..., \pi_{N_\Pi}\}$ *be a set of* N_Π *Unitary Synteny Elements. Then,*

$$\forall \pi_i, \pi_j \in \Pi, j \neq i : \pi_i \cap \pi_j = \emptyset \quad (3.4)$$

and

$$\pi_1 \cup \pi_2 \cup \pi_3 \cup ... \cup \pi_{N_\Pi} = A_{\Phi_A} \cup B_{\Phi_B} \cup \Gamma_{\Phi_\Gamma} \cup ... \cup \Omega_{\Phi_\Omega} \quad (3.5)$$

This is to say, every Unitary Conserved Element belongs to one and only one Unitary Synteny Element. See figure 3.6.

$$\Pi(\alpha) = \pi \quad (3.6)$$

The Unitary Synteny Elements are the smallest unit from which SBs are detected. Through rearrangement operations, two or more SBs can be concatenated into a single one, increasing its size and detecting SBs and rearrangements at different scales. Section 3.5 defines the set of scale invariant operations to build SBs.

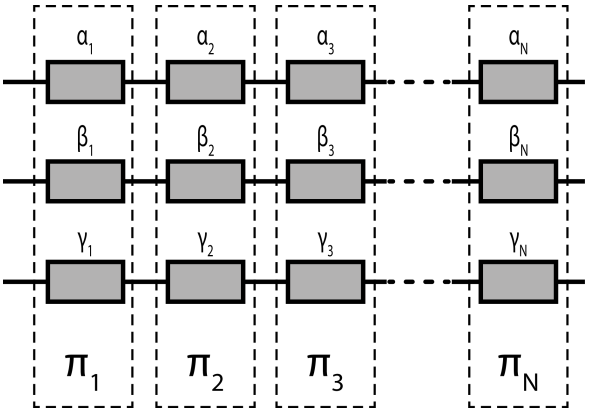


Fig. 3.6 Graphic representation of three Synteny Elements. Synteny Element π_1 links α_1,β_1 and γ_1 Unitary Conserved Elements.

Notice that this definition ensures that Unitary Synteny Elements are free of internal rearrangements because of the property 3.1 in Unitary Conserved Elements. Furthermore, notice that under this definition of SB, there is no reference genome.

Finally, in a multiple genome comparison, Unitary Synteny Elements can represent different levels of synteny: a Unitary Synteny Element can hold Unitary Block Elements from different genomes. We name *Synteny Level* ($SL(\pi)$) as the number of different genomes involved in the Unitary Block Elements linked by the Unitary Synteny Element.

Definition 5. ***Break Point***

Let α_i and α_{i+1} be two adjacent Unitary Conserved Elements that belong to the set A_{Φ_A}. Then, a Break Point ($BP_{\alpha i,\alpha i+1}$) is defined as the region between α_i^{f+1} and α_{i+1}^{o-1} (see figure 3.7). If $\alpha_i^{f+1} = \alpha_{i+1}^{o}$, then the Break Point is consider as a point.

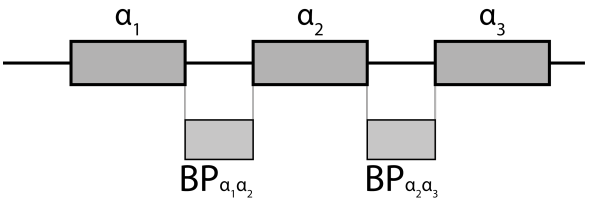


Fig. 3.7 Representation of Break Point. α_1, α_2 and α_3 are Unitary Conserved Elements.

Notice that under this definition, a BP is defined in the sequence, and not as a relation between sequences, although a comparison method is still needed to detect it. This implies that BPs in the sequence Φ_A and BPs in the sequence Φ_B originated by Unitary Conserved Elements that share the same *synteny*, might share high similarity, avoiding the contradiction described in section 2.

3.3 The Unitary Conserved Element problem

The Unitary Block Elements can be calculated from any pair of markers in the sequence detected by a proper comparison method. To illustrate the method to transform conserved pairs into Unitary Conserved Elements, we will assume that those pairs are calculated using a method whose output are ungapped High-Score Segment Pairs (HSPs). The set of all HSPs from all versus all pairwise sequence comparison does not fulfill the properties we have defined before in section 3.1. In order to provide a formal description of the method, a definition of HSPs under our framework is needed. Afterwards, we will describe the problem and we will propose a method to transform HSPs to a set of Unitary Conserved Elements.

Definition 6. ***High-Score Segment Pair***

A High-Score Segment Pair (HSP) is a vector in $\mathbb{N}^2$ *that represents a similarity between two subsequences from two sequences, not necessarily different. Formally, and using the previous notation, we can define a HSP (*$\mathscr{H}$*) from sequences* Φ_A *and* Φ_B *as:*

$$\mathscr{H} = (\alpha^h, \beta^h, \alpha^t, \beta^t, sim(\mathscr{H}), sign(\mathscr{H}))$$

Where $\alpha^h, \beta^h, \alpha^t, \beta^t$ *represent the coordinates in the sequences,* $sim(\mathscr{H})$ *measures the similarity between these subsequences and* $sign(H)$ *indicates whether the relation is found in the forward sequence (positive sign) or in the reverse complement (negative sign).*

We can split the HSP in two one-dimension vectors according the sequence in which they are extracted. They would represent Block Elements in the sequence Φ_A and Φ_B.

$$\begin{gathered}\mathscr{H} = (\mathfrak{H}_A, \mathfrak{H}_B) \\ \mathfrak{H}_A = \alpha = (\alpha^h, \alpha^t) \\ \mathfrak{H}_B = \beta = (\beta^h, \beta^t)\end{gathered}$$

Since HSPs are ungapped, the magnitude for both Block Elements are the same:

$$|\mathscr{H}| = |\mathfrak{H}_A| = |\mathfrak{H}_B| = (\alpha^t - \alpha^h) = (\beta^t - \beta^h)$$

As a consequence, the value of the direction is always the same. However, as we already commented, the sign can be positive or negative.

$$sign(\mathscr{H}) = \begin{cases} +1 & \text{if } forward \\ -1 & \text{if } reverse\ complement \end{cases} \tag{3.7}$$

The set of HSPs (conserved Block Elements) originate from (A,B) comparison must be transform into Unitary Conserved Elements. After this transformation, property 3.1 must be fulfilled. This means that Block Elements in the sequence cannot overlap. Formally, we define the concept of overlapping as follows:

Definition 7. ***Overlap***

Let $\mathscr{H}_1 = (\alpha_1^h, \beta_1^h, \alpha_1^t, \beta_1^t)$ and $\mathscr{H}_2 = (\alpha_2^h, \beta_2^h, \alpha_2^t, \beta_2^t)$ be two HSPs. Then, $\mathscr{H}_1$ and $\mathscr{H}_2$ overlap if:

$$max(\alpha_1^h, \alpha_2^h) < min(\alpha_1^t, \alpha_2^t) \tag{3.8}$$

and / or

$$max(\beta_1^h, \beta_2^h) < min(\beta_1^t, \beta_2^t) \tag{3.9}$$

In the particular case in which $\alpha_1^h = \alpha_2^h$ and $\alpha_1^t = \alpha_2^t$ (or $\beta_1^h = \beta_2^h$ and $\beta_1^t = \beta_2^t$) there is no need to split the Block Element since α_1 and α_2 (or β_1 and β_2) would be the same Block Element. In that case we will say that α_1 and α_2 (or β_1 and β_2) fully overlap, in one or both sequences.

If any of these conditions are true, (and they are not the same Block Element), then $\mathscr{H}_1$ and $\mathscr{H}_2$ shall be split. In order to avoid losing information, we will split them in four HSPs as illustrated in Figure 3.8 B. According to property 1 of Unitary Synteny Element, we will split the HSPs following the equations:

$$\begin{aligned} H_1 &= (\alpha_1^h, \beta_1^h, \Delta_1, \beta_2^h) \\ H_2 &= (\Delta_1, \beta_2^h, \alpha_1^t, \beta_1^t) \\ H_3 &= (\alpha_2^h, \beta_2^h, \Delta_2, \beta_1^t) \\ H_4 &= (\Delta_2, \beta_1^t, \alpha_2^t, \beta_2^t) \end{aligned}$$

where

$$\begin{aligned} \Delta_1 &= \alpha_1^h + \beta_1^t - \beta_1^h \\ \Delta_2 &= \alpha_2^h + \beta_2^t - \beta_2^h \end{aligned}$$

At this point is worth to point out that under this framework, all HSPs are pairs of Conserved Blocks (PCBs) but not all PCBs are HSPs. When we split one HSP, as a result we obtain two pairs of Conserved Blocks that are not longer HSPs.

Notice that every new PCB might trigger a new overlapping conflict that is solved in a recursive way (see Figure 3.8 C and D).

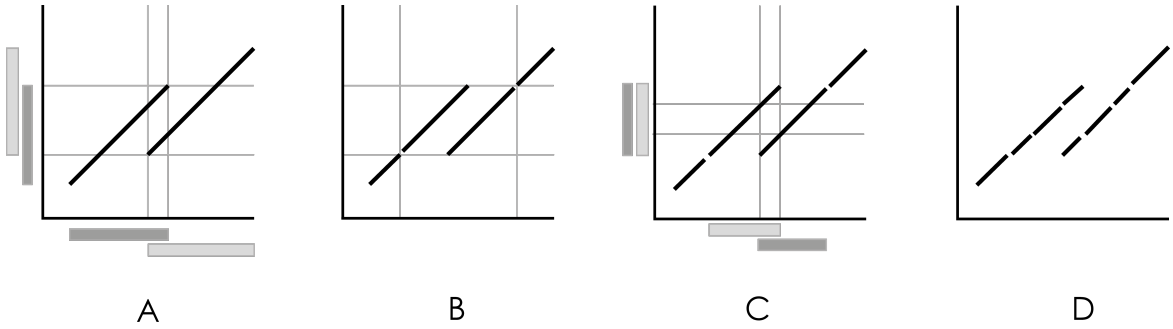


Fig. 3.8 Representation of the trimming process. A) Two overlapped HSPs. B) Result of the trimming process. The two HSPs have been split into four pairs of Conserved Elements. Two of them are still overlapped. C) New overlapped Conserved Elements trigger a new trimming process. D) Final result of the recursive trimming process. The final pairs of Conserved Elements do not overlap.

However, this pairwise overlapping conflict is just the beginning of the transformation problem. It is just solving the conflict for one plane. In a multiple comparison, we must ensure that the overlapping conflict is solved for all the planes:

Let $H_{AB} = \{\mathscr{H}_1, \mathscr{H}_2, ..., \mathscr{H}_{N_{AB}}\}$be the set of N_{AB} PCBs from the comparison of sequences Φ_A and Φ_B. Then, $H_\Phi = \{H_{AB}, H_{A\Gamma}, H_{A\Delta}, \ldots, H_{B\Gamma}, H_{B\Delta}, ..., H_{\Psi\Omega}\}$ is the set of all PCBs from the multiple comparison of sequences contained in Φ and $H_A = \{H_{AB}, H_{A\Gamma}, H_{A\Delta}, ..., H_{A\Omega}\}$ the set of all the PCBs that share the dimension A meaning that every PCB in this set can overlap in the A dimension. Therefore, if one PCB in the AB plane is overlapping, after the split process the new PCBs must not overlap in the H_A and H_B set. This triggers a recursive operation since modifications of HSPs in $H_{A\Delta}$ leads to modifications in the set H_Δ, which in turn might causes modifications in other sets. Figure 3.9 illustrates the problem.

As we said before, Unitary Synteny Elements represent the relation among Unitary Block Elements in a multi-dimension space. Therefore, it also includes the signed pairwise relation. Every pair of Unitary Conserved Element within the same Unitary Synteny Element conforms a vector, and its sign represents the strand.

After the process, the sets of Unitary Conserved Elements holds the property 3.1 and therefore constitute sets of Unitary Conserved Elements, linked by the set of Unitary Synteny Elements. The process does not change the sequences or the relation between them. It just transforms the set of all HSPs into a set of Unitary Conserved Elements and Unitary Synteny Elements.

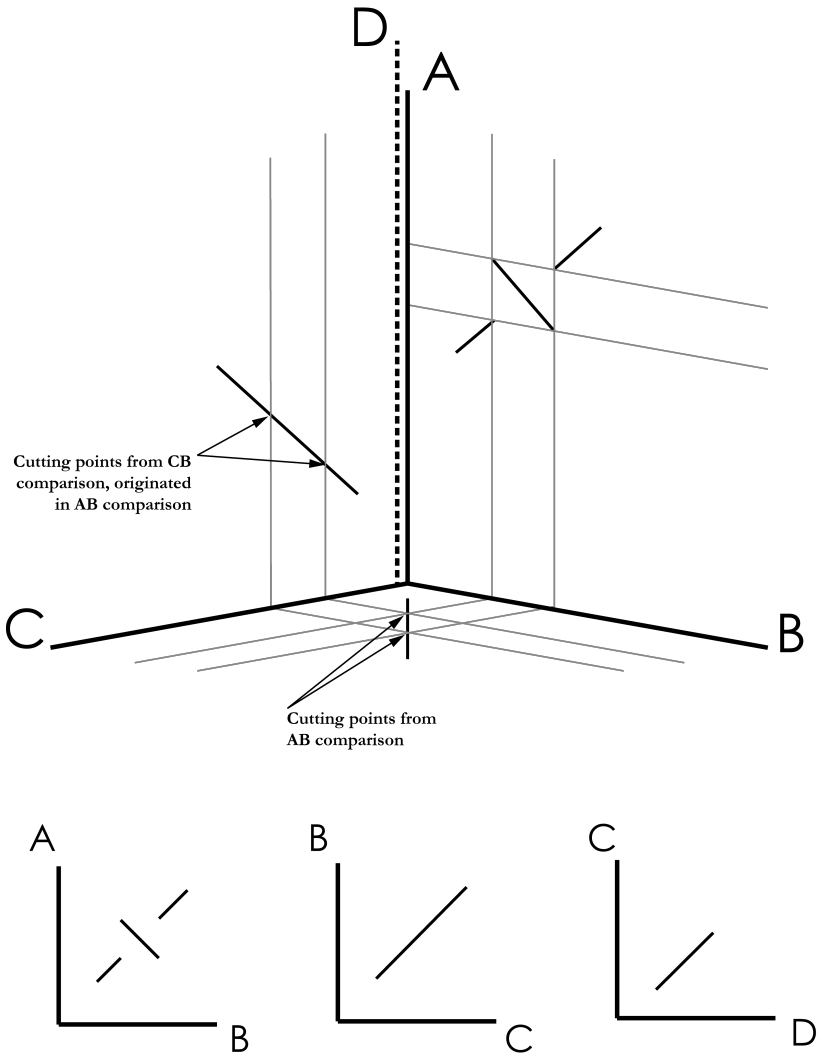


Fig. 3.9 Representation of the trimming process in a multiple comparison. In the comparison *AB* there is an inversion, that triggers a trimming process in the comparison *BC*. As a result, another trimming process is triggered in comparison *DC*.

3.4 Transitivity property of Synteny Blocks: Inferring less conserved HSPs

As it was commented previously in the introduction of this section, in a multiple comparison scenario, HSPs in a sequence can be different depending on the other sequence it is compared with. Thanks to the multi-dimensional SB definition, it is possible to infer by transitivity. HSPs that have not been able to be detected using traditional methods, generally due to such regions in the sequences under comparison, might have not reached certain user-defined thresholds.

Methods to detect HSPs (or homology markers, homologous genes, etc) between sequences are based, to some extent, on similarity between regions. Other methods like PSI-Blast [4] goes further and calculate a profile of conserved regions, in order to find less conserved regions in other sequences. However, a recursive search over the database is needed.

The method we describe in this subsection infers HSPs between sequences without an explicit comparison between the subsequences that the inferred HSP represent. Hence, this method is not similarity-dependent, what opens the possibility to detect less conserved HSPs that might have not reached the similarity or statistical significance thresholds. It is just a necessary consequence of the SB definition in a N-dimensional space.

In this subsection we provide the inferred HSP definition as well as an algorithm to infer them.

Definition 8. ***Inferred HSP***

A Inferred HSP is a pair of Conserved Blocks that is obtained from a Unitary Synteny Element. Let H_{AB}, H_{AC} be and H_{BC} the set of all HSPs detected in the pairwise comparison of (A,B), (A,C) and (C,B). Let $\pi = \{\alpha, \beta, \gamma\}$ be a Unitary Synteny Element obtained by two arbitrary HSPs, $\mathscr{H}_{AB} = (\alpha^h, \beta^h, \alpha^t, \beta^t) \in H_{AB}$ and $\mathscr{H}_{AC} = (\alpha^h, \gamma^h, \alpha^t, \gamma^t) \in H_{AC}$. Then, a necessary HSP is inferred by $\mathscr{H}_{BC} = (\beta^h, \gamma^h, \beta^t, \gamma^t)$.

If $\mathscr{H}_{BC}$ (the inferred HSP) is not in the set of detected HSPs H_{BC} is due to parameter configuration in the method to calculate HSPs in the comparison, because conceptually it should appear. Notice that this method also allows inferring HSPs within the same sequence.

This method does not increase the number of Unitary Conserved Blocks, it just reveals *synteny* relations that have not been detected by the chosen comparison method. Hence, this supports the evidence why SBs must be defined in a N-dimensional space.

Sign of the inferred HSP

The HSP sign is allocated depending on which sequence we have used for the comparison (forward or reverse complement). Since the inferred HSP has not been detected explicitly by sequence comparison, the sign of the new HSP is unknown and must be inferred as well.

It is interesting to see how for the sign of the HSP, the sign rule applied to scalar multiplication in maths works. This is to say, if both HSPs have a forward relation (positive) or reverse (negative), then the related fragment will have a positive relation (both sequences to detect the HSP are the same strand). Otherwise, the sign will be negative because for one sequence it is necessary the reverse complementary whilst for the other it is the forward.

3.5 Rearrangements detection and reconstruction via Unitary Synteny Block

Through reconstruction operations, SBs increase the length by the concatenation of Unitary Conserved Elements. At the same time, new rearrangements can be detected as a consequence of these operations aimed to reconstruct the rearrangement history.

3.5.1 Synteny Block concatenation

After a rearrangement operation, a new SB is detected as a consequence.

Let $\{\alpha_{a-1}, \alpha_a, \alpha_{a+1}\}$ be three Conserved Blocks that belong to A_{Φ_A}; and $\{\beta_{b-1}, \beta_b, \beta_{b+1}\} \in B_{\Phi_B}$, $\{\gamma_{g-1}, \gamma_g, \gamma_{g+1}\} \in \Gamma_{\Phi_\Gamma}$ and so on.

- if the succession of Unitary Synteny Elements for adjacent Unitary Conserved Elements at each sequence is the same

$$\Pi(\alpha_{a+i}) = \Pi(\beta_{b+i}) = \Pi(\gamma_{g+i}) = ... = \Pi(\omega_{o+i}) = \pi_i : i = \{-1, 0, +1\} \tag{3.10}$$

- all these Unitary Conserved Elements conform each a Unitary Synteny Element:

$$\begin{array}{lllllll} \pi_{-1} & = \alpha_{a-1} & \cup\beta_{b-1} & \cup\gamma_{g-1} & \cup... & \cup\omega_{o-1} \\ \pi & = \alpha_a & \cup\beta_b & \cup\gamma_g & \cup... & \cup\omega_o \\ \pi_{+1} & = \alpha_{a+1} & \cup\beta_{b+1} & \cup\gamma_{g+1} & \cup... & \cup\omega_{o+1} \end{array} \tag{3.11}$$

- Synteny Elements have the same *Synteny Level*:

$$SL(\pi_{-1}) = SL(\pi) = SL(\pi_{+1})$$

- and the sign relation between them is the same along adjacent Elementary Conserved Blocks

$$\begin{array}{lll} sign(\alpha_{a-1}, \beta_{b-1}) & = sign(\alpha_a, \beta_b) & = sign(\alpha_{a+1}, \beta_{b+1}) \\ sign(\alpha_{a-1}, \gamma_{g-1}) & = sign(\alpha_a, \gamma_g) & = sign(\alpha_{a+1}, \gamma_{g+1}) \\ sign(\beta_{b-1}, \gamma_{g-1}) & = sign(\beta_b, \gamma_g) & = sign(\beta_{b+1}, \gamma_{g+1}) \\ \multicolumn{3}{c}{...} \\ sign(\psi_{p-1}, \omega_{o-1}) & = sign(\psi_p, \omega_o) & = sign(\psi_{p+1}, \omega_{o+1}) \end{array} \tag{3.12}$$

Then, Unitary Synteny Elements π_{-1}, π and π_{+1} can be merged into a single one by concatenating their Unitary Conserved Elements as follows:

$$\begin{array}{rl} & \pi_{new} = \{\alpha_{new}, \beta_{new}, ..., \omega_{new}\} \\ where & \\ & \alpha_{new} = (\alpha^h_{-1}, \alpha^t_{+1}) \\ & \beta_{new} = (\beta^h_{-1}, \beta^t_{+1}) \\ & ... \\ & \omega_{new} = (\omega^h_{-1}, \omega^t_{+1}) \end{array} \tag{3.13}$$

3.5.2 Insertions and deletions

When concatenating SBs, it might happen that BPs between Unitary Conserved Blocks have not the same length. In this case, a DNA insertion (or deletion) can be detected. A multiple alignment of Unitary Conserved Elements and BPs might help to set the boundaries of the insertion(s).

However, this is not the only way to detect insertions or deletions. If two or more sequences share the same insertion, with a certain level of conservation enough to be detected by a comparison method, then an insertion can be detected as follows:

- if

$$\begin{array}{lllllll} \Pi(\alpha_{a-1}) & = & \Pi(\beta_{b-1}) & = \Pi(\gamma_{g-1}) & = ... = & \Pi(\omega_{o-1}) & = \pi_{-1} \\ \Pi(\alpha_a) & = & \Pi(\beta_b) & = \Pi(\gamma_g) & = ... = & \Pi(\omega_o) & = \pi \\ & & \Pi(\beta_{b+1}) & = \Pi(\gamma_{g+1}) & & & = \pi_{in} \\ \Pi(\alpha_{a+1}) & = & \Pi(\beta_{b+2}) & = \Pi(\gamma_{g+2}) & = ... = & \Pi(\omega_{o+1}) & = \pi_{+1} \end{array} \tag{3.14}$$

- and equation 3.11 is fulfilled,
- then, $\Pi(\beta_{b+1}), \Pi(\gamma_{g+1})$ are detected as insertions.

After detecting an insertion, Elementary Synteny Units $\{\pi_{-1}, \pi, \pi_{+1}\}$ can be merged following the process described above in section 3.5.1.

3.5.3 Duplications

Duplications are one of the most important rearrangement events in evolution. A duplication is detected within the same Synteny Unit, when we find more than one Conserved Block that belongs to the same sequence. Therefore:

- if

$$\pi = \{\alpha_1, \beta_2, \gamma_3, ..., \alpha_4\} \tag{3.15}$$

- then, either α_1 or α_4 is a duplication.

Thanks to the multiple comparison environment, we can detect which Conserved Block was duplicated by the following rule:

- if

$$\begin{array}{lllllll} \Pi(\alpha_{a-1}) & = \Pi(\beta_{b-1}) & = \Pi(\gamma_{g-1}) & = ... = \Pi(\omega_{o-1}) & = \pi_{-1} & \neq \Pi(\alpha'_{d-1}) \\ \Pi(\alpha_a) & = \Pi(\beta_b) & = \Pi(\gamma_g) & = ... = \Pi(\omega_o) & = \pi & = \Pi(\alpha'_d) \\ \Pi(\alpha_{a+1}) & = \Pi(\beta_{b+1}) & = \Pi(\gamma_{g+1}) & = ... = \Pi(\omega_{o+1}) & = \pi_{+1} & \neq \Pi(\alpha'_{d+1}) \end{array} \tag{3.16}$$

- then, α'_d is a duplication.

3.5.4 Inversions

Inversions are easy to detect in a pairwise comparison by just looking at the strand in which the fragment (or de HSP) was detected. However, from a pairwise point of view is not possible to detect in which sequence the event was produced. In a multiple comparison environment, it is possible to detect which sequence was reverted, and therefore, revert the event and restore the former SBs.

A inversion can be detected as follows:

- if

$$\begin{array}{lllllll} \Pi(\alpha_{a-1}) & = \Pi(\beta_{b-1}) & = \Pi(\gamma_{g-1}) & = ... = \Pi(\omega_{o-1}) & = \pi_{-1} \\ \Pi(\alpha_a) & = \Pi(\beta_b) & = \Pi(\gamma_g) & = ... = \Pi(\omega_o) & = \pi \\ \Pi(\alpha_{a+1}) & = \Pi(\beta_{b+1}) & = \Pi(\gamma_{g+1}) & = ... = \Pi(\omega_{o+1}) & = \pi_{+1} \end{array} \tag{3.17}$$

- and Synteny Elements π_{-1}, π and π_{+1} have the same *Synteny Level*

- and

$$\begin{array}{lll} sign(\alpha_{a-1}, \beta_{b-1}) & = sign(\alpha_{a+1}, \beta_{b+1}) & = -sign(\alpha_a, \beta_b) \\ sign(\alpha_{a-1}, \gamma_{g-1}) & = sign(\alpha_{a+1}, \gamma_{g+1}) & = -sign(\alpha_a, \gamma_g) \\ & ... & \\ sign(\beta_{b-1}, \gamma_{g-1}) & = sign(\beta_{b+1}, \gamma_{g+1}) & = sign(\beta_b, \gamma_g) \\ & ... & \\ sign(\psi_{p-1}, \omega_{o-1}) & = sign(\psi_{p+1}, \omega_{o+1}) & = sign(\psi_p, \omega_o) \end{array} \tag{3.18}$$

- then, α_a can be considered as a candidate for reversion.

3.5.5 Transpositions

A transposition is an operation that cuts one block in the genome and moves into another place in the genome. In our framework a transposition is detected as follows:

- if

$$\begin{array}{llllllll} \Pi(\alpha_{a-1}) & = & \Pi(\beta_{b-1}) & = \Pi(\gamma_{g-1}) & = ... = & \Pi(\omega_{o-1}) & = \pi_{-1} \\ \Pi(\alpha_a) & = & \Pi(\beta_{b+1}) & = \Pi(\gamma_{g+1}) & = ... = & \Pi(\omega_{o+1}) & = \pi_{+1} \end{array} \tag{3.19}$$

- and

$$\begin{array}{llllllll} \Pi(\alpha_{i-1}) = & \Pi(\beta_{j-1}) & = \Pi(\gamma_{k-1}) & = ... = & \Pi(\omega_{l-1}) & = \pi_{m-1} \\ \Pi(\alpha_i) = & \Pi(\beta_b) & = \Pi(\gamma_g) & = ... = & \Pi(\omega_o) & = \pi \\ \Pi(\alpha_{i+1}) = & \Pi(\beta_{j+1}) & = \Pi(\gamma_{k+1}) & = ... = & \Pi(\omega_{l+1}) & = \pi_{m+1} \end{array} \tag{3.20}$$

- and Synteny Elements π_{-1}, π and π_{+1} have the same *Synteny Level*

- then, α_i is a transposition.

Section 4

Results and discussion

This section summarises the results of the three publications presented in this thesis. Our results are compared with progressiveMauve [30] , GRIM-Synteny [87] and CASSIS [13]. Since our method works with pure sequence data, we discarded all methods based on gene annotation, protein information, or methods that are able to identify SBs only in coding regions.

The dataset that have been used for the experiments is a collection of 68 Mycoplasma genomes. This dataset contains genomes with different level of similarity. The list of species included in the dataset is available in the appendix E. With regards to the infrastructure, the tests reported in the publications were performed in the Picasso multiprocessor located at the University of Málaga, Spain.[1]

In our first paper, we conducted three experiments to validate the framework, using progressiveMauve and GRIMM-Synteny to compare our results.

- The first experiment is aimed to illustrate the algorithm using a simple pairwise comparison.
- In a second experiment, we compared our method Gecko-CSB against GRIMM-Synteny for a mammalian genome comparison (chromosome 18 of human and mouse). GRIMM-Synteny detects duplications in an early step before detecting SBs. As a consequence, duplications do not break collinearity of other SBs, and breakpoints are lost in the process.
- A massive comparison was carried out in a third experiment. We perform $2,278$ pairwise sequence comparisons using Gecko-CSB and progressiveMauve. Results

[1] http://www.scbi.uma.es

show the same tendency shown in the first experiment: better coverage at all levels of orthology, especially in the less related genomes.

In the second publication, three additional experiments were conducted to validate the method to refine SBs (Gecko-refined-CSB):

- In a first experiment, a simple case illustrates the algorithm behavior in the SB border-refinement method using two close related species of Mycoplasma genus in which an inversion is detected. We focus in the results of Gecko-refined-CSB for the inverted SB.

- In a second experiment, we use CASSIS to refine the same simple example in order to compare it with Gecko-refined-CSB. Results are widely different, mostly because CASSIS does not consider repetitions.

- Finally, a massive comparison is carried out in the third experiment to avoid the bias that a selection of two particular genomes could introduce. Our method refined $2,213$ SBs, 829 were trimmed after the refined process and $1,384$ were extended. To analyse the results, BPs sequences were extracted. We also extracted the adjacent regions of the BPs (located at the SBs beginnings and ends), which we named PRASB (proportional regions of the adjacent SB) to compare with BPs sequences. For more details about PRASB, see figure 4.3.

Additionally, several tools and databases have been used as reference to test accuracy and validate our arguments. For example, NCBI BLASTn has been used for database search to prove that certain sequences that Gecko-CSB report and others not, are present in other species, supporting the significance of these sequences in the rearrangement event history. SMA3s [67] and blast2GO [27] have been used to find biological annotations of sequences, to compare annotations in BPs and PRASB sequences. Regarding the databases, we have used the Uniprot bacteria (ftp://ftp.ebi.ac.uk) and NCBI Non-Redundant databases.

Now, we are going to discuss the main results from the publications:

- The framework is able to work in complex environments (i.e., overlapped fragments, small fragments, highly repeated fragments) and with all HSPs collections provided by other programs. It is not necessary to apply any previous filtering process to clean the input of these problematic fragments. The method is automatic in the sense that it does not need parameters to detect SBs or repetitions. In our case, all parameters are internally estimated based on distributions. Also we use some formulas to estimate values to be used in the process.

- The framework is designed to deal with overlapped HSPs, one of the main limitations in current software tools. As a consequence, the method is able to detect repetitions and organize them in interspersed repeats, tandem repeats or duplications. Dealing with repetitions also allows to detect more BPs because repetitions break the collinearity between SBs.

- Since the method is able to detect repetitions, it allows having more coverage in the results and enhances the quality outperforming state-of-the-art methods. Repetitions are used to refine the SBs borders according to the consensus alignment of the repetitions.

- The results show that Gecko-CSB is robust and able to deal with genomes related at different levels of similarity. This means that genomes under comparison can be closely related to each other, poorly related, or mixed in a heterogeneous dataset where genomes have different level of relatedness among them. See figure 4.1.

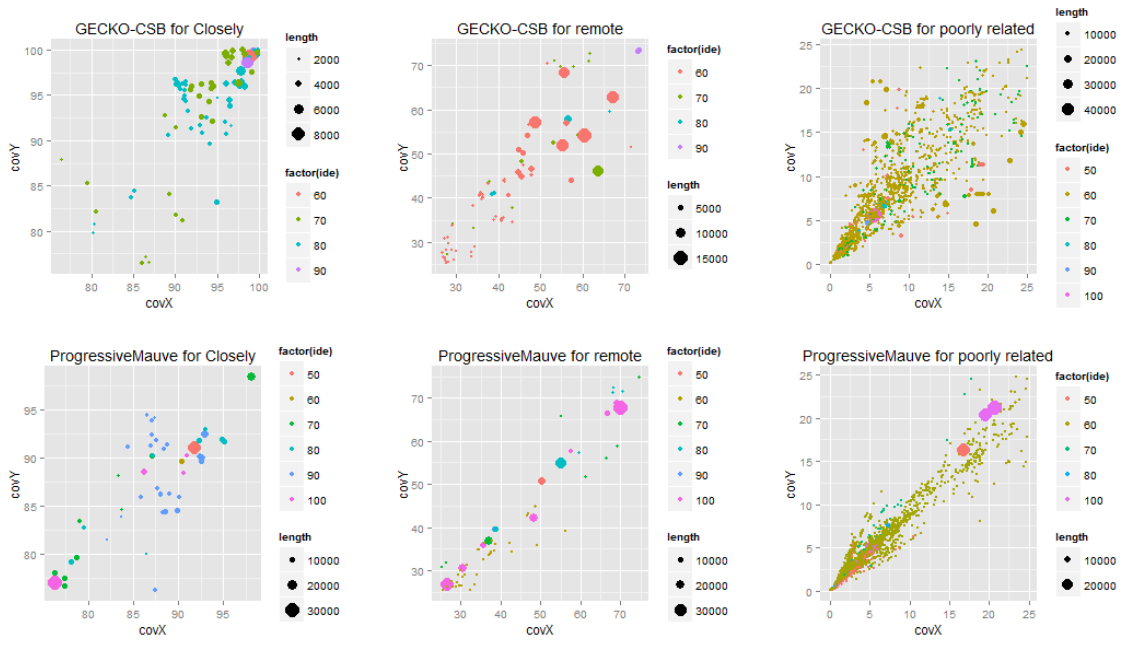


Fig. 4.1 Average length, average percentage of identity, and coverage from all against all comparison of 68 mycoplasmas. Grouped by closely, remote and poorly related species. The X and Y axis represent coverage (as percentages) in the sequences. Each point represents a comparison. The color represents the average identity in the comparison. The shape represents the average length of the detected blocks. On the top, results from our method, Gecko-CSB. On the bottom, results from progressiveMauve. In the image it can be observed that Gecko-CSB works better in terms of getting more coverage over the sequences at the similar level of identity, especially in those comparisons of poorly related species.

- Our method has more coverage over sequences and over both types of regions (coding and non-coding regions) than progressiveMauve and GRIMM-Synteny. In our experiments Gecko-CSB performed around 90% of coverage whilst progressiveMauve and GRIMM-Synteny performed 70% and 80% respectively. For non-coding regions Gecko-CSB achieved 76% against 60% and 75%.

- In a massive comparison, around 70% of the BPs detected by Gecko-CSB are sized below 100 bps and 95% below 300 bps (see figure 4.2). In a particular example of two genomes highly related, Gecko-CSB reports BPs sized below 100bps whereas CASSIS reports BPs sized up to 86.000 bps, which seems to be excessive for a BP. A BLAST search over the CASSIS's BPs showed that those regions are found in several other species with high values of identity and coverage, which point out that the sequences are part of conserved regions. Also, the SMA3s annotation process was carried out to collect biological annotations over the CASSIS's BPs sequences, finding several annotations for that sequences. The same tests were performed over the BPs detected by Gecko-CSB. In this case, the BPs sequences were not found in other species and no annotation was found either, supporting that these BPs were not conserved regions.

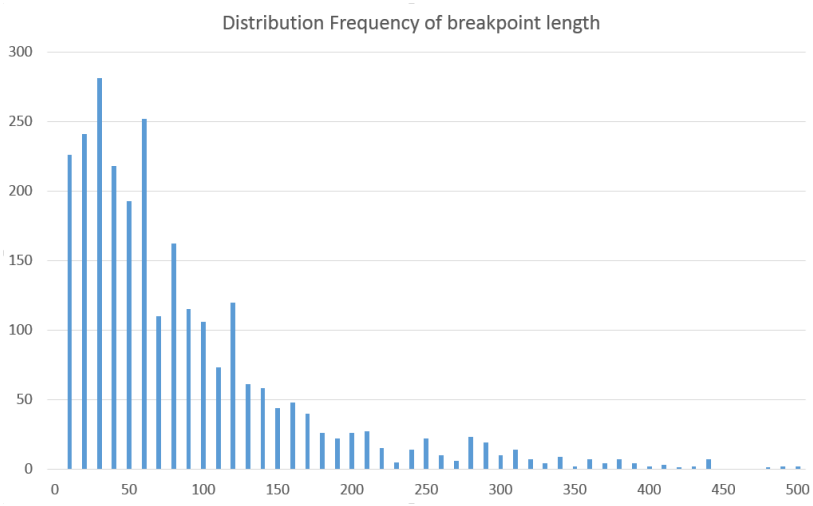


Fig. 4.2 Frequency distribution of Breakpoint length.

- We also observed that annotations in BPs seem to depend on the relatedness between genomes under comparison. The sequences were compared against the NCBI non-redundant protein database, filtered by bacteria taxa. After that, sequences were mapped and annotated using blast2GO. In poorly related species, we found that BPs sequences have more biological annotations (27%) than BPs from highly related genomes (17%).

- Regarding the content of the annotations, we found several differences in the biological process and molecular function categories. Stress response, DNA topological change and DNA replication were more present in BPs sequences than in PRASB sequences. On the other hand, DNA damage, SOS response and DNA integration are found in more proportion in PRASB than BPs sequences.

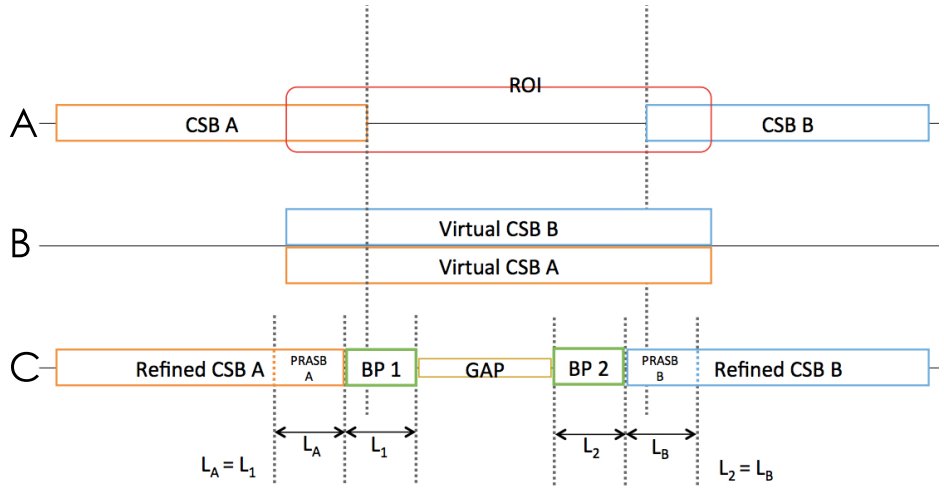


Fig. 4.3 CSBs before and after the refinement. A) Selection of the Region of Interest (ROI), between two Computational Synteny Blocks (CSB). B) Representation of the virtual CSBs. C) Result after the refinement process. We also detect BPs and extract PRASB and GAP sequences to analyse the accuracy of the method. PRASB and BP have the same length. For more details of the refinement process visit the second publication [5].

- Results show that regions that are not reported in other state-of-the-art methods but are detected by Gecko-CSB are coding regions and potential SBs, which can explain LSGR.

In some cases, Gecko-CSB reports longer fragments than other methods for the same region of the genomes under comparison. In other situations, Gecko-CSB reports shorter fragments. In the manuscript [6] we explain the reason of both situations, which can be summarised as follows:

Gecko-CSB reports longer SBs

If the method detects two SBs that fulfill complete collinearity property, then these two SBs are concatenated in a single one. Complete collinearity is described in the first publication, and is based principally on adjacency between SBs. The reason why other methods might not report this SB is because the final alignment could have less similarity than the two SBs separately. However, since Gecko-CSB is designed to analyse LSGR, we allow SBs

with less similarity if it helps to understand LSGR events. In order to test the accuracy of this interpretation, we illustrate one example (see figure 4.4) in which Gecko-CSB reports a longer SB than progressiveMauve (one of the most used state-of-the-art methods). In the example, for the same regions in the comparison, progressiveMauve reports three SBs (B, C and D) whereas Gecko-CSB reports one SB (SB A, which corresponds with the concatenation of the three SBs, B, C and D that progressiveMauve detects). The reason why progressiveMauve does not concatenate these three SBs is because the objective function for the "greedy breakpoint elimination" heuristic process (described in progressiveMauve's paper) does not improve when these three SBs are concatenated. A closer inspection of the region between SBs (E and F) shows a poor conservation (30% and 40% of identity respectively), and this is the main reason why the progressiveMauve's objective function is not improved. However, an annotation process using SMA3s showed that these regions share the same functionality (DNA restriction-modification system for the first regions E and site specific DNA-methyltransferase activity for the second region F); supporting that although conservation is poor, functionality is the same, and therefore the concatenation to report one single SB makes sense.

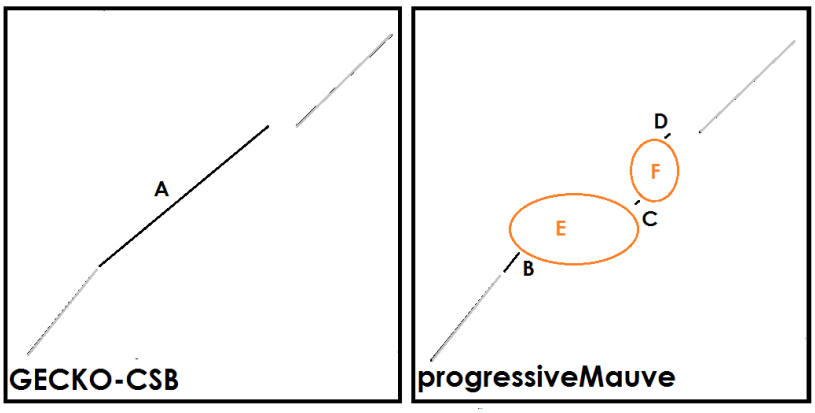


Fig. 4.4 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects one SB. (b) progressiveMauve detects three SBs (B,C and D). The reasons of this difference are explained in the main text.

Gecko-CSB reports shorter SBs

In other cases, Gecko-CSB reports a shorter SB than progressiveMauve. The main reason is because Gecko-CSB detects repetitions, and they break collinearity in SBs, producing more SBs. However, this is not the only reason, since Gecko-CSB detects smaller regions, they can also be responsible of breaking the collinearity between SBs. In the fist manuscript we

illustrate one example in which for the same region in a genome comparison, progressiveMauve reports one SB whereas Gecko-CSB reports three SBs (see figure 4.5). In this case, Gecko-CSB takes into account a small SB (SB C, around 600 bps of length and 60% of identity), which breaks the collinearity between the two main SBs A and B. progressiveMauve concatenates these two SBs because after the concatenation, the resulting SB D has 91% of identity (before 87% for A and 89% for B of identity). However, a database search using NCBi BLASTn was carried out using the sequence of this region (the small SB) against the NR database. The results showed that the main feature of this region is related with ATPase enzymes (around 35% of all the results) and if we exclude the Mycoplasma Taxa, the same sequence is found in other species like Plasmodium, Zebrafish or Vitis, supporting the significance of this "small" SB in order to understand rearrangement events.

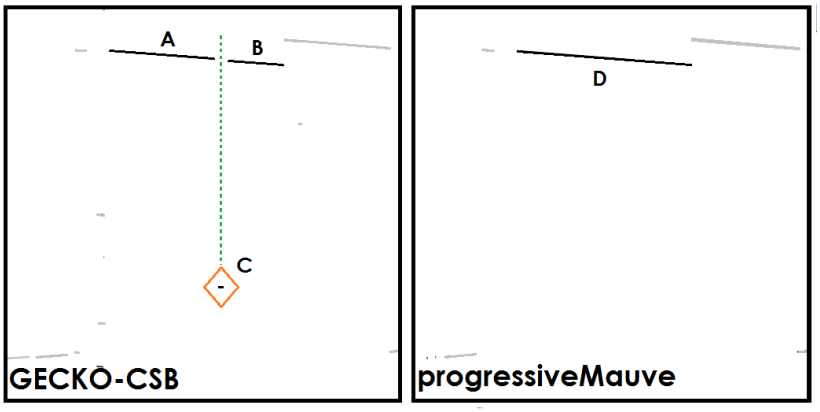


Fig. 4.5 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects three SBs (A,B and C). (b) progressiveMauve detects one large SB.

Section 5

Conclusions and future work

5.1 Conclusions

In this thesis we present three publications aimed to the SB detection, refinement and their applications. A framework for a pairwise SB processing is presented in the first two publications; an application to the metagenome analysis in the third publication; and we introduce the basis for a multi comparison SB framework in section 3 (providing definitions, rules and algorithms).

In a first work, we introduce a parameter-free and robust method aimed to the automatic SB detection, able to work in complex environments (short repeats, overlapped fragments and small fragments). This method outperforms current software tools both in number and quality of the detected SBs. In the publication, a set of definitions is presented to formalize linearity and collinearity properties in SBs. These properties are useful to detect LSGR such as inversion, transpositions or duplications.

To validate the results, two different applications were used: progressiveMauve and GRIMM-Synteny in three experiments. Parameters in those applications were set to produce comparable results.

In all cases:

- Our method obtains more coverage and better quality
- Our method is designed for dealing with overlapped HSPs and detect repeats, one of the main drawbacks in current software.
- Our method works in complex environments (small fragments and repeats) and with HSPs collections provided by other programs.
- Our method is automatic in the sense that it does not need parameters to detect SBs.

- Our method is designed to detect SBs that can explain LSGR.

This method is the starting point for SBs and BP refinement that was carried out in a second work. In the second publication, we developed a method to refine the borders of SB taking into account repetitions and using them to improve the accuracy of the refinement. The method uses a Finite State Machine (FSM) to find the transition point, instead of maximizing a target function like other methods. This FSM is designed to detect transitions in the difference between repeats and SBs alignments. Due to the methods' features, BPs are detected as regions or points, depending on the specific case. The FSM needs two thresholds to detect the transition points. Although so far these parameters are fixed we will work on a dynamic configuration of them based on SB similarity.

Several analyses were carried out in order to find biological differences between BPs and SBs borders with satisfactory results. BPs sequences are biologically richer than the SB borders (corresponding with the Proportional Region of the Adjacent Syntney Block, PRASB, which is defined in the second publication). Both searches using Uniprot and NCBI databases reported more results in BPs sequences than the PRASB sequences. However, PRASB sequences showed more diversity in annotations than BP sequences.

Our experiments also revealed that there might be a correlation between the number of sequences annotated in BPs and PRASB; and the relatedness of the species from which those sequences were extracted. This is to say, BPs detected in poorly related species were biologically richer than BPs detected in close species in our experiments. We also found that there are biological differences between what we consider as BPs and the regions between BPs, whereas other methods just consider the whole region as BP.

Detection of SBs in a multiple genome comparison was useful to face the metagenome analysis described in the third publication. Metagenome reads mapping in non-SB regions from certain genome strongly support that such genome is present in the metagenome.

5.2 Future work

The proposed framework for multiple comparisons SB detection provides the basis for a better understanding of genome evolution and its applications go beyond a precise SB detection:

- Refining SBs also provides refined BPs, which can be used as input to find hidden patterns or extract features in order to set up a formal definition of BP, which could enable their detection. The detection of BPs in a genome sequence may help the understanding of LSGR and the prediction of future LSGRs.

- Frequencies of LSGR occurrences can be used for a LSGR matrix penalization in a refined inter-genome distance measure, which could penalise LSGRs in the same way that alignment methods penalise nucleotide or aminoacid substitutions or gaps.
- The rearrangement history reconstruction could also be helpful for the phylogenetic organizations, especially in those cases in which other methods based on sequence similarity generate contradictory results.
- The proposed framework enables the rearrangement history reconstruction between species towards the last common ancestor (LCA). In the process, intermediate virtual species can be calculated. A database of intermediate virtual species can be generated, and it could help to the metagenome mapping analysis since the read we want to map belongs to a different strain than genomes registered in public databases. For instance, if we have a read that belongs to Genome C (see figure 5.1) that are not registered in the database, the read could map better in the LCA-ABC than in genomes A or B.

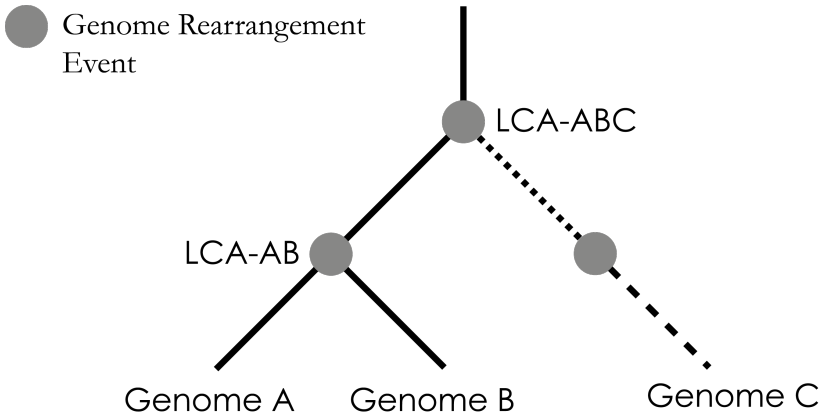


Fig. 5.1 GenomeA and GenomeB are included in the public database. Genome C is not in the database. If a certain metagenome read comes from Genome C specie, it might match better in the Last Common Ancestor-ABC than in the genomes A or B, because the genome C is not present in the database.

5.2.1 Detecting Break Points using a Machine Learning approach

As we deduced in section 3, Break Points (BP) should be defined as a region in the sequence, instead as a relation among sequences (the region between two SBs) although so far, the way to detect them is through sequence comparison. Machine Learning methods have successfully solved high complex problems specially when hidden patterns are involved. Many authors have suggested that BP might be *weak* regions in the genome more likely to break. If this

hypothesis were true, there would be a chance to predict Break Points just analysing the sequence.

Long-Short Term Memory cells (LSTM) [50] have shown to be powerful to discover hidden patterns in sequences. In comparative genomics field it has been used for searching homology in sequences [49].

As a future work, we will use LSTMs to train a network to detect BPs. Labels for BP positive and negative class can be taken based on SBs detected by our method (or any other method that detects SBs). The positive class will be the collection of BPs. For the negative class, we will extract subsequences from SBs following the length distribution of the BP collection, in order to avoid any length bias. Our input could be coded following the approach used in [49]. We could train a classifier by gradient descent to minimize cross entropy loss function. In figure 5.2 we draft the proposed architecture.

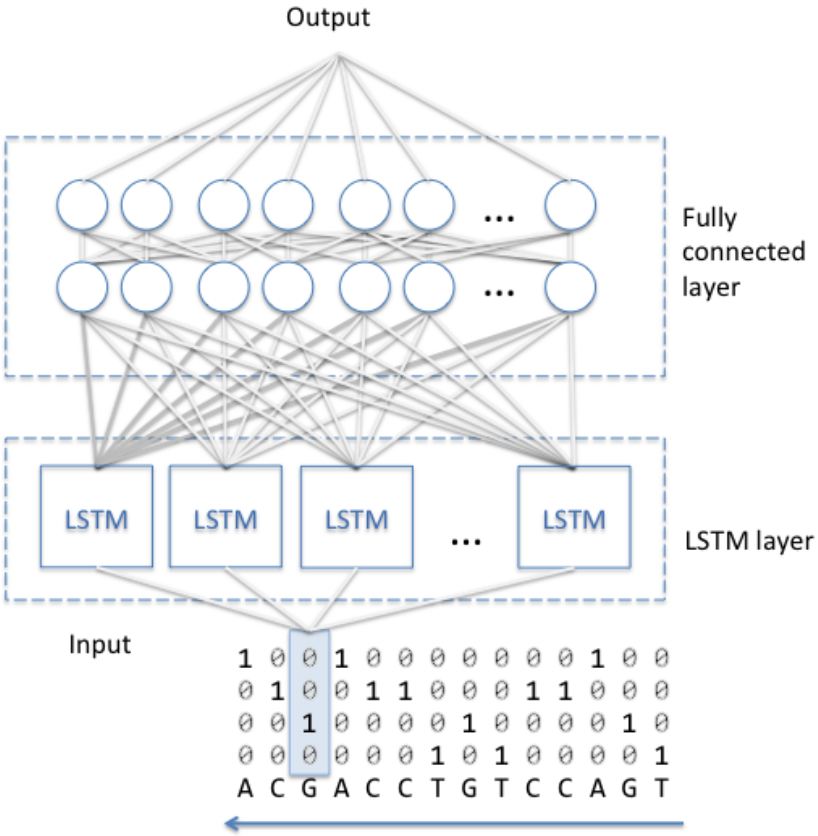


Fig. 5.2 The input is encoded as one-hot vector. After the LSTM layer, we use a fully connected layer to combine all the LSTM cells outputs to produce a single output.

References

[1] Alan Christie, D. (1998). *Genome Rearrangement Problems*. PhD thesis, University of Glasgow.

[2] Alekseyev, M. A. and Pevzner, P. A. (2007). Are there rearrangement hotspots in the human genome? *PLoS Computational Biology*, 3(11):2111–2121.

[3] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–10.

[4] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–402.

[5] Arjona-Medina, J. and Trelles, O. (2016a). Refining borders of genome-rearrangements including repetitions. *BMC Genomics*, 17(S8):804.

[6] Arjona-Medina, J. A. and Trelles, O. (2016b). Computational Synteny Block: A Framework to Identify Evolutionary Events. *IEEE Transactions on NanoBioscience*, 15(4):1–11.

[7] Attie, O., Darling, A. E., and Yancopoulos, S. (2011). The rise and fall of breakpoint reuse depending on genome resolution. *BMC Bioinformatics*, 12(Suppl 9):S1.

[8] Bader, D. a., Moret, B. M., and Yan, M. (2001). A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of computational biology : a journal of computational molecular cell biology*, 8(5):483–91.

[9] Bader, M. and Ohlebusch, E. (2007). Sorting by weighted reversals, transpositions, and inverted transpositions. *Journal of computational biology : a journal of computational molecular cell biology*, 14(5):615–36.

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

[11] Bailey, J. a., Baertsch, R., Kent, W. J., Haussler, D., and Eichler, E. E. (2004). Hotspots of mammalian chromosomal evolution. *Genome biology*, 5(4):R23.

[12] Bairoch, A. and Bucher, P. (1994). PROSITE: recent developments. *Nucleic acids research*, 22(17):3583–9.

[13] Baudet, C., Lemaitre, C., Dias, Z., Gautier, C., Tannier, E., and Sagot, M. F. (2010). Cassis: Detection of genomic rearrangement breakpoints. *Bioinformatics*, 26(15):1897–1898.

[14] Bedell, J., Korf, I., and Yandell, M. (2003). *Blast*.

[15] Berman, P. and Hannenhalli, S. (1996). Fast sorting by reversal. *Combinatorial Pattern Matching*.

[16] Berman, P. and Karpinski, M. (1999). On some tighter inapproximability results. *Proceedings of the 26th international Conference on Automata, Languages and Programming*, 1644:200–209.

[17] Biller, P., Guéguen, L., Knibbe, C., and Tannier, E. (2016). Breaking good: accounting for fragility of genomic regions in rearrangement distance estimation. *Genome Biology and Evolution*, 8(5):evw083.

[18] Blanchette, M., Bourque, G., and Sankoff, D. (1997). Breakpoint Phylogenies. *Genome Inform Ser Workshop Genome Inform*, 8:25–34.

[19] Blanchette, M., Kunisawa, T., and Sankoff, D. (1996). Parametric genome rearrangement. *Gene*, 172(1):GC11–7.

[20] Bonham-Carter, O., Steele, J., and Bastola, D. (2013). Alignment-free genetic sequence comparisons: A review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905.

[21] Caprara, A. (1997). Sorting by reversals is difficult. *Proceedings of the first annual international conference ...*.

[22] Capy, P., Langin, T., Anxolabehere, D., and Bazin, C. (1998). Dynamics and evolution of transposable elements. *Molecular Biology Intelligence Unit*, page 197.

[23] Carrillo, H. and Lipman, D. (1988). The Multiple Sequence Alignment Problem in Biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082.

[24] Chao, K.-M. and Zhang, L. (2009). *Sequence comparison : theory and methods*. Springer.

[25] Choi, V., Zheng, C., Zhu, Q., and Sankoff, D. (2007). Algorithms for the extraction of synteny blocks from comparative maps. *Algorithms in Bioinformatics*, pages 277–288.

[26] Chu, T. C., Liu, T., Lee, D. T., Lee, G. C., and Shih, A. C. C. (2009). GR-Aligner: An algorithm for aligning pairwise genomic sequences containing rearrangement events. *Bioinformatics*, 25(17):2188–2193.

[27] Conesa, A., Götz, S., García-Gómez, J. M., Terol, J., Talón, M., and Robles, M. (2005). Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics (Oxford, England)*, 21(18):3674–6.

[28] Corpet, F. (1988). Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 16(22):10881–10890.

[29] Darling, A. C. E., Mau, B., Blattner, F. R., and Perna, N. T. (2004). Mauve : Multiple Alignment of Conserved Genomic Sequence With Rearrangements Mauve : Multiple Alignment of Conserved Genomic Sequence With Rearrangements. pages 1394–1403.

[30] Darling, A. E., Mau, B., and Perna, N. T. (2010). Progressivemauve: Multiple genome alignment with gene gain, loss and rearrangement. *PLoS ONE*, 5(6).

[31] Dayhoff, M. O. (1978). *Atlas of Protein Sequence and Structure*, volume 3. Silver Spring.

[32] Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. (1999). Alignment of whole genomes. *Nucleic acids research*, 27(11):2369–76.

[33] Dias, Z. and Meidanis, J. (2001). Genome rearrangements distance by fusion, fission, and transposition is easy. *Proceedings Eighth Symposium on String Processing and Information Retrieval*, pages 250–253.

[34] Doolittle, W. F. and Sapienza, C. (1980). Selfish genes, the phenotype paradigm and genome evolution. *Nature*, 284(5757):601–603.

[35] El-Mabrouk, N. (2000). Genome rearrangement by reversals and insertions/deletions of contiguous segments. In *Combinatorial Pattern Matching*, pages 222–234.

[36] Eriksen, N. (2002). (1 +) -Approximation of sorting by reversals and transpositions. 289:517–529.

[37] Feijão, P. and Meidanis, J. (2013). Extending the algebraic formalism for genome rearrangements to include linear chromosomes. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 10(4):819–831.

[38] Fostier, J., Proost, S., Dhoedt, B., Saeys, Y., Demeester, P., van de Peer, Y., and Vandepoele, K. (2011). A greedy, graph-based algorithm for the alignment of multiple homologous gene lists. *Bioinformatics*, 27(6):749–756.

[39] Ghiurcuta, C. G. and Moret, B. M. E. (2014). Evaluating synteny for improved comparative studies. *Bioinformatics*, 30:9–18.

[40] Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708.

[41] Gotoh, O. (1996). Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinement as Assessed by Reference to Structural Alignments. *Journal of Molecular Biology*, 264(4):823–838.

[42] Gu, Q., Peng, S., and Sudborough, H. (1999). A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theoretical Computer Science*, 3975(98).

[43] GUPTA, S. K., KECECIOGLU, J. D., and SCHÄFFER, A. A. (1995). Improving the Practical Space and Time Efficiency of the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment. *Journal of Computational Biology*, 2(3):459–472.

[44] Hannenhalli, S. and Pevzner, P. (1996). To Cut... or Not to Cut (Applications of Comparative Physical Maps in Molecular Evolution). *SODA*, pages 304–313.

[45] Hartman, T. and Sharan, R. (2005). A 1.5-approximation algorithm for sorting by transpositions and transreversals. *Journal of Computer and System Sciences*, 70(3):300–320.

[46] Haubold, B., Klotzl, F., and Pfaffelhuber, P. (2014). andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, (December 2014):1–7.

[47] Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919.

[48] Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343.

[49] Hochreiter, S., Heusel, M., and Obermayer, K. (2007). Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736.

[50] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

[51] John D. Storey (2002). A direct approach approach to false discovery rates. *Journal of the Royal Statistical Society*, 64(3):479–498.

[52] Kaminker, J. S., Bergman, C. M., Kronmiller, B., Carlson, J., Svirskas, R., Patel, S., Frise, E., Wheeler, D. A., Lewis, S. E., Rubin, G. M., Ashburner, M., and Celniker, S. E. (2002). The transposable elements of the Drosophila melanogaster euchromatin: a genomics perspective. *Genome biology*, 3(12):RESEARCH0084.

[53] Kaplan, H., Shamir, R., and Tarjan, R. E. (2000). A Faster and Simpler Algorithm for Sorting Signed Permutations by Reversals. *SIAM Journal on Computing*, 29(3):880–892.

[54] Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the United States of America*, 87(6):2264–8.

[55] Kazazian, H. H. (2004). Mobile elements: drivers of genome evolution. *Science (New York, N.Y.)*, 303(5664):1626–1632.

[56] Kececioglu, J. and Sankoff, D. (1995). Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*.

[57] Lemaitre, C., Tannier, E., Gautier, C., and Sagot, M.-F. (2008). Precise detection of rearrangement breakpoints in mammalian chromosomes. *BMC bioinformatics*, 9:286.

[58] Lemey, P., Salemi, M., and Vandamme, A.-M. (2009). *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press.

[59] Lerat, E. (2010). Identifying repeats and transposable elements in sequenced genomes: how to find your way through the dense forest of programs. *Heredity*, 104(6):520–533.

[60] Li, Z., Wang, L., and Zhang, K. (2006). Algorithmic approaches for genome rearrangement: a review. *... , Part C: Applications and Reviews, ...*, 36(5):636–647.

[61] Lin, G. H. and Xue, G. (2001). Signed genome rearrangement by reversals and transpositions: Models and approximations. *Theoretical Computer Science*, 259(1-2):513–531.

[62] Lipman, D. J., Altschul, S. F., and Kececioglu, J. D. (1989). A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415.

[63] Mani, R.-S. and Chinnaiyan, A. M. (2010). Triggers for genomic rearrangements :. *Nature Publishing Group*, 11(12):819–829.

[64] Mccammon, J. A. and Wolynes, P. G. (1998). Highly specific protein sequence motifs for genome analysis. *Computational Biomolecular Science*, 95(May):5865–5871.

[65] Miklós, I. and Tannier, E. (2010). Bayesian sampling of genomic rearrangement scenarios via double cut and join. *Bioinformatics*, 26(24):3012–3019.

[66] Morgenstern, B. (2004). DIALIGN: Multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Research*, 32(WEB SERVER ISS.):W33–6.

[67] Muñoz-Mérida, A., Viguera, E., Claros, M. G., Trelles, O., and Pérez-Pulido, A. J. (2014). Sma3s: A Three-Step Modular Annotator for Large Sequence Datasets. *DNA research : an international journal for rapid publication of reports on genes and genomes*, (February):1–13.

[68] Nadeau, J. H. and Taylor, B. a. (1984). Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences of the United States of America*, 81(February):814–818.

[69] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–53.

[70] Nicolas, S. D., Monod, H., Eber, F., Chèvre, A.-M., and Jenczewski, E. (2012). Non-random distribution of extensive chromosome rearrangements in Brassica napus depends on genome organization. *The Plant Journal*, 70(4):691–703.

[71] Notredame, C. and Higgins, D. G. (1996). SAGA: Sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524.

[72] Notredame, C., Higgins, D. G., & Heringa, J., Notredame, C., Higgins, D. G., and Heringa, J. (2000). T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217.

[73] Pan, X., Stein, L., and Brendel, V. (2005). SynBrowse: A synteny browser for comparative sequence analysis. *Bioinformatics*, 21(17):3461–3468.

[74] Pearson, W. R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in enzymology*, 183(1988):63–98.

[75] Pellicer, J., Fay, M. F., and Leitch, I. J. (2010). The largest eukaryotic genome of them all? *Botanical Journal of the Linnean Society*, 164(1):10–15.

[76] Pérez-Wohlfeil, E., Arjona-Medina, J. A., Torreno, O., Ulzurrun, E., and Trelles, O. (2016). Computational workflow for the fine-grained analysis of metagenomic samples. *BMC Genomics*, 17(S8):802.

[77] Pevzner, P. and Tesler, G. (2003). Genome Rearrangements in Mammalian Evolution : Lessons From Human and Mouse. *Genome Research*, 13(1):37–45.

[78] Pham, S. and Pevzner, P. (2010). DRIMM-Synteny: decomposing genomes into evolutionary conserved segments. *Bioinformatics*, 26(20):2509–16.

[79] Saha, S., Bridges, S., Magbanua, Z. V., and Peterson, D. G. (2008). Empirical comparison of ab initio repeat finding programs. *Nucleic Acids Research*, 36(7):2284–2294.

[80] Sankoff, D. and Trinh, P. (2005). Chromosomal Breakpoint Reuse in Genome Sequence Rearrangement. *Journal of Computational Biology*, 12(6):812–821.

[81] SanMiguel, P., Tikhonov, a., Jin, Y. K., Motchoulskaia, N., Zakharov, D., Melake-Berhan, a., Springer, P. S., Edwards, K. J., Lee, M., Avramova, Z., and Bennetzen, J. L. (1996). Nested retrotransposons in the intergenic regions of the maize genome. *Science (New York, N.Y.)*, 274(5288):765–768.

[82] Schnepf, N., Deback, C., Dehee, A., Gault, E., Parez, N., and Garbarg-Chenon, A. (2008). Rearrangements of rotavirus genomic segment 11 are generated during acute infection of immunocompetent children and do not occur at random. *Journal of virology*, 82(7):3689–96.

[83] Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Soding, J., Thompson, J. D., and Higgins, D. G. (2014). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1):539–539.

[84] Simossis, V. A. and Heringa, J. (2005). PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. *Nucleic Acids Research*, 33(Web Server):W289–W294.

[85] Skovgaard, M., Jensen, L. J., Brunak, S., Ussery, D., and Krogh, A. (2001). On the total number of genes and their length distribution in complete microbial genomes. *Trends in Genetics*, 17(8):425–428.

[86] Smith, C. D., Edgar, R. C., Yandell, M. D., Smith, D. R., Celniker, S. E., Myers, E. W., and Karpen, G. H. (2007). Improved repeat identification and masking in Dipterans. *Gene*, 389(1):1–9.

[87] Tesler, G. (2002). GRIMM: genome rearrangements web server. *Bioinformatics (Oxford, England)*, 18(3):492–493.

[88] Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680.

[89] Torreno, O. and Trelles, O. (2015). Breaking the computational barriers of pairwise genome comparison. *BMC Bioinformatics*, 16(1):250.

[Walter et al.] Walter, M., Dias, Z., and Meidanis, J. Reversal and transposition distance of linear chromosomes. *Proceedings. String Processing and Information Retrieval: A South American Symposium (Cat. No.98EX207)*, pages 96–102.

[91] WANG, L. and JIANG, T. (1994). On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology*, 1(4):337–348.

[92] Wang, L.-S. (2001). {Exact-IEBP}: A New Technique for Estimating Evolutionary Distances Between Whole Genomes. *Proc.\ 1st Workshop Algs.\ in Bioinformatics ({WABI'01})*, 2149:175–188.

[93] Wang, L.-S., Warnow, T., Moret, B. M. E., Jansen, R. K., and Raubeson, L. a. (2006). Distance-based genome rearrangement phylogeny. *Journal of molecular evolution*, 63(4):473–83.

[94] Yakir, B. and Siegmund, D. (2000). Approximate {$}p{$}-values for local sequence alignments. *The Annals of Statistics*, 28(3):657–680.

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

[96] Yancopoulos, S. and Friedberg, R. (2008). Sorting genomes with insertions, deletions and duplications by DCJ. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5267 LNBI, pages 170–183.

[97] Zeng, X., Nesbitt, M. J., Pei, J., Wang, K., Vergara, I. a., and Chen, N. (2008). OrthoCluster: a new tool for mining synteny blocks and applications in comparative genomics. *EDBT '08: Proceedings of the 11th international conference on Extending database technology*, pages 656–667.

Section A

Sequence comparison algorithms

A DNA sequence is a string of symbols from an alphabet. Formally, DNA can be described as a sequence $S = s_1, s_2, s_3, \ldots, s_n$, where $s_i \in SA, C, G, T$; being A for Adenine, C for Cytosine, G for Guanine and T for Thymine. The length of sequences can vary between $1e04$ (Human mitochondrion) and $1,3e14$ (Paris Japonica) [75], which involves a computational challenge to store, process, and mining information, keeping in mind the scalability of the problem.

A.1 Pairwise sequence alignments

Dot-plot matrix methods were used for visualizing similarities between sequences with a length less than 1 Mb. In this method, all possible matches are taken into account. This way, diagonals represent similarity regions of the sequence while isolated points represent random matches. This visualization can be improved by filtering random matches using a sliding window of length W and allowing for some mistmatches, or stringency threshold. For longer sequences, long execution time and computational memory requirements make this method not feasible.

In 1970 Needleman and Wunsch proposed a global alignment method base on dynamic programming [69]. This approach ensures the best possible alignment given a substitution matrix, such a PAM[31] or BLOSUM[47], and other parameters to penalise gaps in the alignment. This method is $O(mn)$ complexity both in memory and time, which could be prohibitive in long sequences like genomes. An optimization of this method was carried out by Dan Hirschberg, using less memory $O(m+n)$, but still requiring $O(mn)$ time [48].

Later on, Smith and Waterman developed a local alignment method for sequences[86]. Actually, it was a variation of Needleman and Wunsch method, keeping the substitution matrix and the gap-scoring scheme but setting to zero those cells in the similarity matrix with

negative score. The complexity for this algorithm is $o(n^2M)$. Osamu Gotoh published an optimization of this method, running in $o(mn)$ time [40].

The main difference between both methods could be resumed as followed:

- Needleman and Wunsch method aligns the sequences fixing the first and the last position of both sequences. It attempts to align every symbol in the sequence, allowing some gaps, but the main purpose is to get a global alignment. This is especially useful when the two sequences to compare are highly similar. For instance:

```
ATCGGATCGACTGGCTAGATCATCGCTGG
CGAGCATC-ACTGTCT-GATCGACCTTAG
* *** **** ** ****  *  * *
```

- As an alternative to global methods, Smith and Waterman local method align the sequences with a bigger degree of freedom, allowing the alignment to start or end with gaps. This is extremely useful when the two sequences are substantially dissimilar in general but suspected of having a highly related sub region.

```
ATCAAGGAGATCATCGCTGGACTGAGTGGCT----ACGTGGTATGT
ATC----CGATCATCGCTGG-CTGATCGACCTTCTACGT-------
***     ************ ****  * *     ****
```

Back then, available sequences were proteins or genes, with a length between 20 and 400 amino acids (375 in average for proteins in humans) [85] and around 1.5Kb for genes in average(http://www.gencodegenes.org/). Thus, methods previously described had no major problem regarding computational effort. However, when full genomes came up, new methods appeared to solve memory and running limitations.

A.2 Multiple sequence alignment algorithms

Those methods opened a new window to address the multiple sequence alignment domain. For many of the algorithms in this field, the complexity is NP-hard. There are many different methods for multiple sequence alignment. They can be classified in four categories:

- exact methods: Proved to NP-Hard by Wang and Jiang. [91].Some MSA methods[23, 62, 43],
- progressive methods: CLUSTALW[88],Clustal Omega [83], PRALINE [84], T-Coffee [72].

- iterative and search algorithms: DIALIGN[66], MultiAlign[28], PRRP[41], SAGA[71];
- local methods: eMOTIF[64], PROSITE [12].

For more details, visit *Sequence Comparison: Theory and methods* [24], Chapter 5 or *The Phylogenetic Handbook* [58], Chapter 3.

A.3 New strategies: homology search methods

Many methods started to reduce the space search of the problem by finding small words of length k, also called k-mers, that two or more sequences share. Once these shared k-mers are calculated in the sequence to compare, they represent a hit or match, and can be represented as points (or seeds) in the dot-matrix. Then, the algorithm tries to extend the alignment at this point. This idea was introduced first by FASTA[74] and later by BLAST[3]. The main difference between them resides in the way to calculate the hits (or seeds). For FASTA method, there is a match (seed) if the k-mers are exactly the same. For BLAST method, they allow some degree of dissimilarity.

New methods adopted this computational space reduction strategy and kept working to improve sensitivity, speed and memory limitations. For example, Psi-BLAST[4], MUMmer [32] or progressiveMauve[30].

In order to solve memory limitation, other methods explored different strategies. For example, Gecko [89] follows the out-of-core strategy, using external memory as a support for intermediate results when it is necessary. As a result, multiple genome comparison can be done in a reasonable time.

Output of these methods are generally a collection of segment pairs that reach some level of score, meaning some degree of similarity. They are called High Scoring Segment Pair (HSP).

A.4 Statistical Significance

A necessary post-processing step when calculating HSPs is to assign some value that measures how likely or unlikely a specific alignment is to be found. This value is calculated under a statistical model in which the HSPs are distributed.

In 1990, Karlin and Altshul published a theory of local alignment statistics [54]. This equation states that the number of alignment expected by chance E during a sequence database search is a function of the size of the sear space $(m*n)$, the normalized score (λS)

and a minor constant (k). The size of the search space is simply the product of the size of the the query (m) and the size of database (n) [14].

$$E = kmne^{-(\lambda Score)} \tag{A.1}$$

K and λ are calculated using the nucleotide frequencies and the scoring matrix.

Using this equation a significance value can be assigned to a HSP. By choosing a threshold the sensibility of the results can be controlled.

Recently, the distribution of the maximum score in gappd alignment was deducted[94, 51]

A.5 Dealing with repetitions

Repeats, tandem repeats and duplications make the detection of SBs extremely difficult. Thus, in order to make Synteny Blocks (SBs) detection easier, many of the methods used to calculate SBs avoid these events, such as DRIMM-Synteny [78], GRIMM-Synteny [77] or GRAligner [26]. However, a considerable part of genome is repetitive. Human genome is almost 50% repetitive DNA and 80% for maize genome [81]. It is also know that repetitions -mostly associated with mobile elements- have been driven evolution in many ways [55], playing an interesting role [34].

Repeats can be classified in two main groups. Tandem repeats and dispersed repeats. Tandem repeats are nucleotide patterns that are repeated in an adyancent way. Depending on the number of repetitions, and the pattern size, they can be classified in satellites (from 1 to 200 nucleotides); microsatellite, or simple sequence repeat, small patterns up to 6 nucleotides; minisatellites (from 10 to 60 nucleotides) and rDNA repeats. Dispersed repeats, mainly constitute by transposable elements, can be classified according to the intermediate element that help them to move. Connection between them could be used to make a reconstruction of the evolution history [22].

Detection of these elements is complicated due to punctual mutations, insertions, deletions and rearrangements. Moreover, some times these elements are combined creating nested elements. [81, 52]. Methods to detect repetitions can be classified in two main groups: database search and de novo approaches.

The first ones use a database containing known repetitions such as RepeatMasker, Censor, Maskeraid, Plotrep or Greedier. The main limitation of these methods is that they are not capable to identify repetitions that are not catalogued in the database in which we are searching. A variation of this method is to use the repeats signature in which we are interested. This approach allows finding repetitions that are not catalogued but we have to know in

advance the features of the sequence we expect to find. (RTAnalyzer, TSDfinder, SINEDR, FINDMITE...)

De novo approaches try to find any kind of repeats in the sequence. In this group we can differentiate self-comparison methods like Repeat Pattern toolkit, RECON, PILER; and method based in k-mers and spaced seed approaches like Reputer, Repeat-Match, RepeatScout, Repseek. A comparative study showed that RepeatScout performed the best results [79].

Due to difference between methods and diversity of purpose, it would be possible to create pipelines to exploit the advantages of all of them. However, software dependencies, updates involving input and output format changes, lack of maintenance, lack of documentation and other problems described by Lerat [59] make extremely difficult in practice to utilize these methods, specially the combination of some of them in a pipe line. As a consequence, there is a big amount of software that, in broad strokes, share the same purpose.

All this methods are designed to find repeats in the sequence, but they are not designed in a rearrangement framework. Their only purpose is just to identify them.

Section B

Methods in the State of art for Synteny Block detection

Orthocluster

Orthocluster (Zeng et al. 2008) assumes that a mapping between genes in genomes under comparison is given. They use several user-defined parameter: lower and upper bound on the number of genes in each cluster, maximal percentage of mismatched in map genes, synteny block size and whether gene ordering or strandedness is preserved or not.

Cyntenator

(Rödelsperger , Dieterich 2010). They use BLASTP to extract fragments but instead of using nucleotids or aminoacids, they use an alphabet of genes. Then they extract alignments with a score higher than a predefined threshold and then they implement several filters. For example, to compare rat/mouse with human they only use sequence regions shared by mouse-rat. All other sequences regions from mouse or rat are discarded. Then they define a threshold for total number of alignments, and the times that they can occurred.

Cassis

(Baudet et al. 2010) Cassis receives as an input a list of pairs of one2one orthologous genes. Genes which have same order and direction in both genomes are merged. Overlapping genes that do not respect this criteria are discarded. To create synteny blocks, they use the algorithm described by Lemaitre (sagot), using k=2. This parameter enables individual isolated genes to be out of order without disrupting a synteny block because all synteny blocks must contain at least two genes.

DRIMM-Synteny

DRIMM-Synteny (Pham , Pevzner 2010) is based on de Bruijn graphs. DRIMM-Synteny takes a set of anchors without overlaps that can be local alignments or pairs of similar genes. (but they use gene order in their results).

MCScan

MCScan combines information of gene position and protein sequences to perform an all-against all using BLASTP. To search for homology MCSscan (Wang et al. 2012) compares protein-coding genes from each genome and itself. To avoid local collinear gene pairs, if consecutive matches have a common gene and its paired genes are separated by fewer than five genes, these matches are collapsed using a representative pair with the smallest BLASTP E-value. Then they use a scoring schema assuming that two genes are collinear if the number of intervening genes between them is fewer than 25. Finally, non-overlapping chains with scores over 250 (involving at least 5 collinear genes) are reported.

i-ADHoRe v3.0

(Proost et al. 2012) i-ADHoRe needs two user-defined parameters, the gap size and q_value, and they warn that this selection have a direct impact on the accuracy and sensitivity of the collinearity detection. To calculate Synteny Blocks, first they get gene family information, then they build a gene homology matrix (GHM). Significant collinear regions founded in GHM are aligned using a novel alignment algorithm (GG2)[38] based on protein-Needledman and Wunsch algorithm.

GR-Aligner

(Chu et al. 2009) GR-Aligner searches only non-overlapping matches, with a certain *pvalue* and score higher than a given threshold from a BLAST comparison and collected them as elements for a candidate of SB. Two fragment from BLAST are then merged into a Synteny Block if 1) They are adjacent, 2) the space between fragment is smaller than the minimum of the fragment length, and 3) if the candidate SB final score is higher than the minimum score of the fragments. They cannot deal with duplications. They cannot treat in inverted transpositions o transpositions. Only block interchange (they call it simple translocations).

ProgressiveMauve

(Darling et al. 2010) ProgressiveMauve uses HOXD matrix to discriminate well between homologous and unrelated sequence in a variety of organism. To minimize compute time and focus only on anchoring coverage on single-copy regions, their method only extends seeds that are unique in two or more genomes. They define LMA as local multiple alignments. Is a generalization of Maximal-Unique-Matches (MUM) but including multiple genomes. They define a pairwise locally collinear block (LCB) as a subset of local alignments (LMA) in a genome that occur in the same order and orientation in a pair of genomes and they are free from internal rearrangements. After transform LMAs into local pairwise alignments, they apply well-known breakpoint analyses procedure [29, 18], to minimally partition into pairwise LCB. They use a breakpoint penalty which is a user controlled parameter (they are working on a parameter which take into account how related species are). Then they remove breakpoints by greedy breakpoint elimination to make LCBs bigger. Then use a recursive anchoring to improve alignments. They use a smaller k to find new alignments and they incorporate them into the main alignment. After that they calculate again the score. This process is recursive and it stops when difference of score cannot improve more than a given parameter. They apply a Hidden Markov Chain to predict pairwise homology, to avoid align non related regions in genome.

Shuffle-LAGAN

(Brudno et al. 2003) Shuffle-LAGAN uses CHAOS to generate local alignments between two sequences. Given a word length k and degeneracy c it reports words of k length that match with a c differences. Then, given a distance d and maximum shift s, two letters in different sequences are joined if they are lower than d and their difference is less than s. Chain are extended using ungapped BLAST until the score drops below a certain threshold. After computing the chains, the program CHAOS scores each chain and insert gaps. All local alignments scoring above 2000 are returned and used to create the 1-monotonic conservation map. To build it they use different gap penalties with different thresholds. In their paper they work with rearrangement events longer than 100 bp and shorter than 100Kbp for reasons of efficiency. Since they have to split genomes (por problemas de memoria, pero tengo q revisarlo), translocations and duplications that have been split between two contigs are not detected. Only alignment that covers at least 70% of another is reported as a duplication. Because Shuffle-LAGAN is not symmetric only duplications in one genome are found.

Sibelia

(Minkin et al. 2013) Sibelia, which is prepared only for bacterias, is based on Brujin graph algorithm and uses LAGAN for aligning synteny blocks. It uses as an input the whole sequence.

MUMmer

(Kurtz et al. 2004) MUMmer finds local alignments of highly identical sequence, then aggregates them into one that cover collinear regions. Each groupr is free from rearrangements. MUMer can identify and align genomes with rearrangements. They have (at least) 3 user parameters: length of exact matches, distances between two matches to be aligned, a parameter to decide if a collinear chain is extracted and processed.

Section C

Sorting permutation problem state of art

Reconstruct the history of evolutionary events can be viewed as a sorting permutation problem where we can transform the order by evolutionary events operations. Zimao Li and his colleagues made in 2006 a wide review of different methods of sorting permutation [60].

C.1 Sorting by reversals

The first serious strike for sorting permutations was made by Kececioglu and Sankoff [56]. They developed a method for sorting unsigned permutations by reversals. The method was based in two conjetures:

- There exists an optimal series of reversals that does not cut stripes other than at their first or last element.
- There exists an optimal series of reversals that never increases the number of breakpoints.

Later on, Bafna and Pevner [10][comprobar 1-Genome Rearrangements and sorting by reversals] improved it for signed and unsigned permutations. One year later Hannenhalli and Pevner proved those conjectures [44]. They also develop an exact $O(n^4)$ algorithm to sort permutations by reversals [15] which was later improved by Kaplan, designing an algorithm in $O(n^2)$ [53]. Later on, Bader presented an algorithm in a linear time for signed permutations [8].

Sorting unsigned permutations by reversals was proved to be NP-hard problem by Caprara in 1997 [21] and reduced to MAX SNP-hard by Berman and Karpinsky [16]. El-Mabrouk also studied the problem of sorting permutations by reversals but also included insertions and deletions as operations. She extended Hannenhalli and Pevzner's polynomial-time approach

[28] and develop a new algorithm in $O(n^2)$ for the sorting signed permutation by reversals with insertions and deletions problem [35].

C.2 Sorting by transpositions

Inversions, -or reversals-, are not the only operation why biological sequences have evolved and other methods included this operation in the model. At the beginning, to solve the problem just using them instead of reversals; and later on combining reversals with transpositions.

Bafna and Pevner first studied transposition in 1998[10]. Walter et al [Walter et al.] presented a ratio-3 approximation algorithm for computing the unsigned reversal and transposition distance running in time $O(n^2)$. Gu et al proposed a greedy heuristic $O(n^2)$ algorithm for signed permutations [42]. Lin and Xue developed a method for sorting signed permutations by combined operations [61]. Wang and Warnow [Wang et al. 2006] developed a technique called the inverse of the expected number of breakpoints (IEBP) to estimate the "true evolutionary distance" [93], which was later on refined with a more accurate method, the Exact-IEBP [92][Wang 2001].

C.3 Weighted operations and other evolutionary events

Scientist have observed that in practice, transposition occur with about half the frequency of reversals [19]. This leaded to develop new methods where reversals and transpositions are weighted.

Eriksen [36] presented PTAS under the restriction that the given permutations are signed and circular. Dias and Meidanis studied another weighted problem of sorting by fusion, fission and transposition simultaneously [33]. Bader et al [9], presented a fast algorithm (heuristic) for the multiple genome rearrangement problem with weighted reversals and transposition. They did not consider unsigned permutations. Given such weights, the weighted genome rearrangement problem asks for a sorted sequence of rearrangement operations such that the sum of the weights of the operations in the sequence is minimal. Under this criterion, a shortest sequence of operations is not necessarily optimal, unlike all the methods developed to date. The complexity of a method that combines transposition, reversal and reverted transposition is still unknown. Hartman and Sharan provided a very efficient 1.5-approximation algorithm for this case [45].

C.4 DCJ

Sophia Yancopoulos proposed DCJ in 2005 [95] and it was extended in [96]. DCJ include duplications. DCJ approach [42] tries to minimize the number of DCJs required to sort the graph. Many studies has used this methods for calculate distances between genomes [65].

Feijao and Meidanis proposed an extension of algebraic formalism for rearrangements that includes linear chromosomes The main difference with DCJ model is that they change the weight for some operations and they do not consider duplications in their model [37].

Section D

Publications

D.1 Publication 1

Computational Synteny Blocks: A framework to identify Evolutionary Events

Arjona-Medina, J. A., & Trelles, O. (2016). Computational Synteny Block: A Framework to Identify Evolutionary Events. IEEE Transactions on NanoBioscience, 15(4), 1–11. http://doi.org/10.1109/TNB.2016.2554150

D.1.1 Summary

The identification and accurate description of large genomic rearrangements is crucial for the study of Evolutionary Events among species and implicitly defining breakpoints. Although there is a number of software tools available to perform this task, they usually either a) require a collection of pre-computed non-conflicting High-scoring Segment Pairs (HSPs) and gene annotations; or b) involve working at protein level (what excludes non-coding regions) ; or c) need many parameters to adjust the software behavior and performance; or d) imply working with duplications, repeats and tandem repeats, which complicates the identification of rearrangements task. Although there are many programs specialized in the detection of these repetitions, they are not designed for the identification of main genomics rearrangements.

The methodology we envisage starts with the detection of all HSPs by pairwise genome comparison. The second step involves solving conflicts generated by fragments that overlap in both sequences (double overlapped fragments) to end yielding a collection of gapped fragments. In the third step, the quality measures (length, score, identities) of the gapped fragment are refined by using a modified dynamic programming approach. This collection of refined gapped fragments represents the input of a recursive process in which we identify

blocks of gapped fragments that maintain co-localization, regardless of them occurring in coding or non-coding regions. The identification of repeats is an important step in the subsequent refinement of these blocks. This step allows for the separation of repeats and the correct identification in turn of longer blocks. Finally, groups of repeats, duplications, inversions and translocations are identified.

The set of algorithms presented in this manuscript is able to detect and identify blocks of large rearrangements-taking into account repeats, tandem repeats and duplications-starting with the simple collection of ungapped local alignments. To the best of our knowledge, this is the first method to approach the whole process as a coherent workflow -thus outperforming current state-of-the-art software tools-and additionally allowing to classify the type of rearrangement. The results obtained are an important source of information for breakpoints refinement and featuring, as well as for the estimation of the Evolutionary Events frequencies to be used in inter-genome distance proposals, etc.

D.2 Publication 2

Refining borders of genome-rearrangements including repetitions

Arjona-Medina, J., & Trelles, O. (2016). Refining borders of genome-rearrangements including repetitions. BMC Genomics, 17(S8), 804. http://doi.org/10.1186/s12864-016-3069-4

D.2.1 Summary

DNA rearrangement events have been widely studied in comparative genomics for many years. The importance of these events resides not only in the study about relatedness among different species, but also to determine the mechanisms behind evolution. Although there are many methods to identify genome-rearrangements (GR), the refinement of their borders has become a huge challenge. Until now no accepted method exists to achieve accurate fine-tuning: i.e. the notion of breakpoint (BP) is still an open issue, and despite repeated regions are vital to understand evolution they are not taken into account in most of the GR detection and refinement methods.

We propose a method to refine the borders of GR including repeated regions. Instead of removing these repetitions to facilitate computation, we take advantage of them using a consensus alignment sequence of the repeated region in between two blocks. Using the concept of identity vectors for Synteny Blocks (SB) and repetitions, a Finite State Machine is designed to detect transition points in the difference between such vectors. The method does not force the BP to be a region or a point but depends on the alignment transitions within the SBs and repetitions.

The accurate definition of the borders of SB and repeated genomics regions and consequently the detection of BP might help to understand the evolutionary model of species. In this manuscript we present a new proposal for such a refinement. Features of the SBs borders and BPs are different and fit with what is expected. SBs with more diversity in annotations and BPs short and richer in DNA replication and stress response, which are strongly linked with rearrangements.

D.3 Publication 3

Computational workflow for the fine-grained analysis of metagenomic samples Pérez-Wohlfeil, E., Arjona-Medina, J. A., Torreno, O., Ulzurrun, E., & Trelles, O. (2016). Computational workflow for the fine-grained analysis of metagenomic samples. BMC Genomics, 17(S8), 802. http://doi.org/10.1186/s12864-016-3063-x

D.3.1 Summary

The field of metagenomics, defined as the direct genetic analysis of uncultured samples of genomes contained within an environmental sample, is gaining increasing popularity. The aim of studies of metagenomics is to determine the species present in an environmental community and identify changes in the abundance of species under different conditions. Current metagenomic analysis software faces bottlenecks due to the high computational load required to analyze complex samples.

A computational open-source workflow has been developed for the detailed analysis of metagenomes. This workflow provides new tools and datafile specifications that facilitate the identification of differences in abundance of reads assigned to taxa (mapping), enables the detection of reads of low-abundance bacteria (producing evidence of their presence), provides new concepts for filtering spurious matches, etc. Innovative visualization ideas for improved display of metagenomic diversity are also proposed to better understand how reads are mapped to taxa. Illustrative examples are provided based on the study of two collections of metagenomes from faecal microbial communities of adult female monozygotic and dizygotic twin pairs concordant for leanness or obesity and their mothers.

The proposed workflow provides an open environment that offers the opportunity to perform the mapping process using different reference databases. Additionally, this workflow shows the specifications of the mapping process and datafile formats to facilitate the development of new plugins for further post-processing. This open and extensible platform has been designed with the aim of enabling in-depth analysis of metagenomic samples and better understanding of the underlying biological processes.

Section E

List of 68 Mycoplasmas

NC_000908.2	M. *genitalium G37*
NC_000912.1	M. *pneumoniae M129*
NC_002771.1	M. *pulmonis UAB CTIP*
NC_004432.1	M. *penetrans HF-2 DNA*
NC_004829.2	M. *gallisepticum str. R(low)*
NC_005364.2	M. *mycoides subsp. mycoides SC str. PG1 chromosome*
NC_006360.1	M. *hyopneumoniae 232*
NC_006908.1	M. *mobile 163K*
NC_007294.1	M. *synoviae 53*
NC_007295.1	M. *hyopneumoniae J*
NC_007332.1	M. *hyopneumoniae 7448*
NC_007633.1	M. *capricolum subsp. capricolum ATCC 27343*
NC_009497.1	M. *agalactiae PG2 chromosome*
NC_011025.1	M. *arthritidis 158L3-1*
NC_012806.1	M. *conjunctivae HRC/581T*
NC_013511.1	M. *hominis ATCC 23114 chromosome*
NC_013948.1	M. *agalactiae 5632 chromosome*
NC_014014.1	M. *crocodyli MP145*
NC_014448.1	M. *hyorhinis HUB-1*
NC_014552.1	M. *fermentans JER*
NC_014751.1	M. *leachii PG50 clone MU clone A8*
NC_014760.1	M. *bovis PG45 clone MU clone A2*
NC_014921.1	M. *fermentans M64*
NC_014970.1	M. *haemofelis str. Langford 1*

NC_015153.1 M. *suis KI3806*
NC_015155.1 M. *suis str. Illinois*
NC_015431.1 M. *mycoides subsp. capri LC str. 95010*
NC_015725.1 M. *bovis Hubei-1*
NC_015946.1 M. *putrefaciens KS1*
NC_016638.1 M. *haemocanis str. Illinois*
NC_016807.1 M. *pneumoniae 309 DNA*
NC_016829.1 M. *hyorhinis GDL-1*
NC_017502.1 M. *gallisepticum str. R(high)*
NC_017503.1 M. *gallisepticum str. F*
NC_017504.1 M. *pneumoniae FH*
NC_017509.1 M. *hyopneumoniae 168*
NC_017519.1 M. *hyorhinis MCLD*
NC_017520.1 M. *haemofelis Ohio2*
NC_017521.1 M. *leachii 99/014/6*
NC_018077.1 M. *bovis HB0801*
NC_018149.1 M. *wenyonii str. Massachusetts*
NC_018406.1 M. *gallisepticum VA94_7994-1-7P*
NC_018407.1 M. *gallisepticum NC95_13295-2-2P*
NC_018408.1 M. *gallisepticum NC96_1596-4-2P*
NC_018409.1 M. *gallisepticum NY01_2001.047-5-1P*
NC_018410.1 M. *gallisepticum WI01_2001.043-13-2P*
NC_018411.1 M. *gallisepticum NC06_2006.080-5-2P*
NC_018412.1 M. *gallisepticum CA06_2006.052-5-2P*
NC_018413.1 M. *gallisepticum NC08_2008.031-4-3P*
NC_018495.1 M. *genitalium M2321*
NC_018496.1 M. *genitalium M6282*
NC_018497.1 M. *genitalium M6320*
NC_018498.1 M. *genitalium M2288*
NC_019552.1 M. *hyorhinis SK76*
NC_019949.1 M. *cynos C142*
NC_021002.1 M. *fermentans PG18 DNA nearly*
NC_021025.1 M. *mycoides subsp. mycoides SC str. Gladysdale MU clone*
NC_021083.1 M. *putrefaciens Mput9231*
NC_021283.1 M. *hyopneumoniae 168-L*
NC_021831.1 M. *hyopneumoniae 7422*
NC_022575.1 M. *parvum str. Indiana*
NC_022807.1 M. *hyorhinis DBS 1050*
NC_023062.1 M. *ovis str. Michigan*

Section F

Resumen en español

Esta tesis es un compendio de tres artículos recientemente publicados en revistas de alto impacto, en los cuales mostramos el proceso que nos ha llevado a proponer la definición de *Unidades Elementales de conservación* (regiones conservadas entre genomas que son detectadas después de una comparación múltiple), así como algunas operaciones básicas como inversiones, transposiciones y duplicaciones. Los tres artículos están transversalmente conectados por la detección de Bloques de Sintenia (SB) y reorganizaciones genómicas de gran escala (LSGR) (consultar sección 2), y respaldan la necesidad de elaborar el *framework* que se describe en la sección 3. De hecho, el trabajo intelectual llevado a cabo en esta tesis y las conclusiones aportadas por las publicaciones han sido esenciales para entender que una definición de SB apropiada es la clave para muchos de los métodos de comparativa genómica.

Los eventos de reorganización del ADN son una de las principales causas de evolución y sus efectos pueden ser observados en nuevas especies, nuevas funciones biológicas etc. Las reorganizaciones a pequeña escala como inserciones, deleciones o substituciones han sido ampliamente estudiadas y existen modelos aceptados para detectarlas.

Sin embargo, los métodos para identificar reorganizaciones a gran escala aún sufren de limitaciones y falta de precisión, debido principalmente a que no existe todavía una definición de SB aceptada. El concepto de SB hace referencia a regiones conservadas entre dos genomas que guardan el mismo orden y *strand*. A pesar de que existen métodos para detectarlos, éstos evitan tratar con repeticiones o restringen la búsqueda centrándose solamente en las regiones codificantes en aras de un modelo más simple. El refinamiento de los bordes de estos bloques es a día de hoy un problema aún por solucionar.

Esta tesis por compendio aborda la definición formal de SB, empezando por Pares de Segmentos de alta puntuación (HSP), los cuales son bien conocidos y aceptados. El primer objetivo se centró en la detección de SB como una combinación de HSPs incluyendo

repeticiones lo cual incrementó la complejidad del modelo. Como resultado, se obtuvo un método más preciso y que mejora la calidad de los resultados del estado del arte [6].

Este método aplica reglas basadas en la adyacencia de SBs, permitiendo además detectar LSGR e identificarlos como inversiones, translocaciones o duplicaciones, constituyendo un *framework* capaz de trabajar con LSGR para organismos de un solo cromosoma.

Más tarde en un segundo artículo, se utilizó este *framework* para refinar los bordes de los SBs. En nuestra novedosa propuesta, las repeticiones que flanquean los SB se utilizaron para refinar los bordes explotando la redundancia introducida por dichas repeticiones. Mediante un alineamiento múltiple de estas repeticiones se calculan los vectores de identidad del SB y de la secuencia consenso de las repeticiones alineadas. Posteriormente, una máquina de estados finitos diseñada para detectar los puntos de transición en la diferencia de ambos vectores determina los puntos de inicio y fin de los SB refinados [5]. Este método también se mostró útil a la hora de detectar *puntos de ruptura* (conocidos como break points (BP)). Estos puntos aparecen como la región entre dos SBs adyacentes. El método no fuerza a que el BP sea una región o un punto, sino que depende de los alineamientos de las repeticiones y del SB en cuestión.

El método es aplicado en un tercer trabajo, donde se afronta un caso de uso de análisis de metagenomas [76]. Es bien sabido que la información almacenada en las bases de datos no corresponde necesariamente a las muestras no cultivadas contenidas en un metagenoma, y es posible imaginar que la asignación de una muestra de un metagenoma se vea dificultada por un evento reorganizativo. En el articulo se muestra que las muestras de un metagenoma que mapean sobre las regiones exclusivas de un genoma (aquellas que no comparte con otros genomas) respaldan la presencia de ese genoma en el metagenoma. Estas regiones exclusivas son fácilmente derivadas a partir de una comparación múltiple de genomas, como aquellas regiones que no forman parte de ningún SB.

Una definición bajo un espacio de comparación múltiple de genomas es más precisa que las definiciones construidas a partir de una comparación de pares, ya que entre otras cosas, permite un refinamiento siguiendo un procedimiento similar al descrito en el segundo artículo (usando SBs, en vez de repeticiones). Esta definición también resuelve la contradicción existente en la definición de puntos de BPs (mencionado en la segunda publicación), por la cual una misma región de un genoma puede ser detectada como BP o formar parte de un SB dependiendo del genoma con el que se compare.

Esta definición de SB en comparación múltiple proporciona además información precisa para la reconstrucción de LSGR, con vistas a obtener una aproximación del verdadero ancestro común entre especies. Además, proporciona una solución para el problema de la granularidad en la detección de SBs: comenzamos por SBs pequeños y bien conservados y a

través de la reconstrucción de LSGR se va aumentando gradualmente el tamaño de dichos bloques.

Los resultados que se esperan de esta línea de trabajo apuntan a una definición de una métrica destinada a obtener distancias inter genómicas más precisas, combinando similaridad entre secuencias y frecuencias de LSGR.