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

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

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. 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.

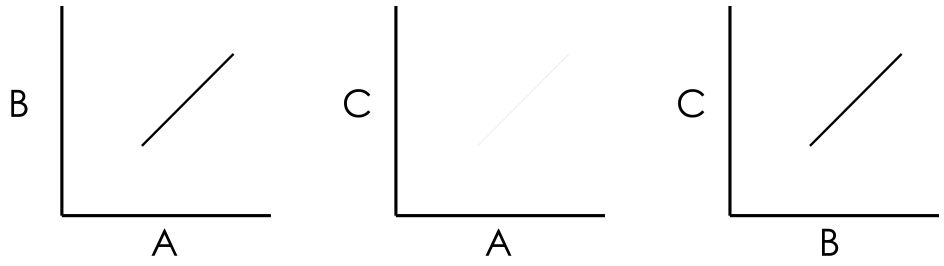


Fig. 3.1 Comparisons (A,B) and (B,C) show a conserved pair that is not detected in comparison (A,C).

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.

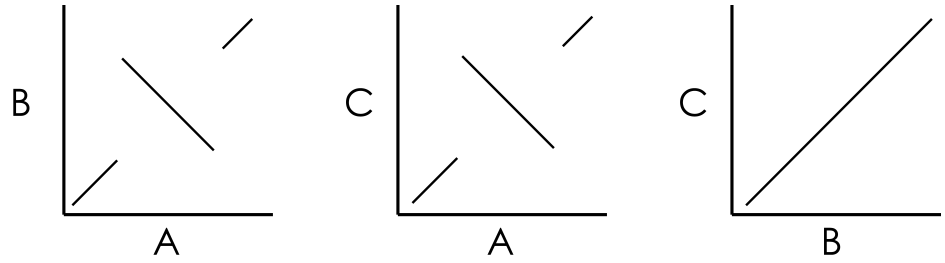


Fig. 3.2 Three comparisons involving A, B and C sequences. (A,C) and (A,B) show an inversion that is not present in (B,C).

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)*

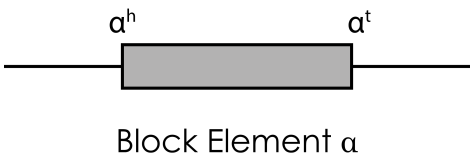


Fig. 3.3 Representation of Block Element. α^h and α^t represent the head and the tail positions of the block in the full sequence.

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:

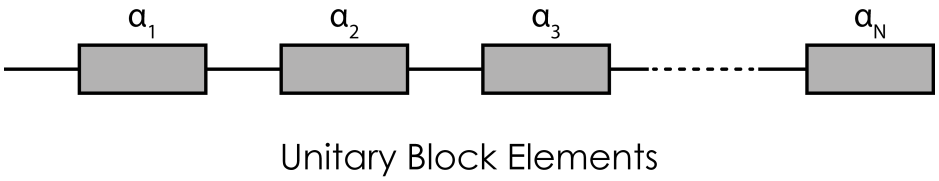


Fig. 3.4 Representation of the set of Unitary Block Elements A_{Φ_A} in the sequence Φ_A.

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

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

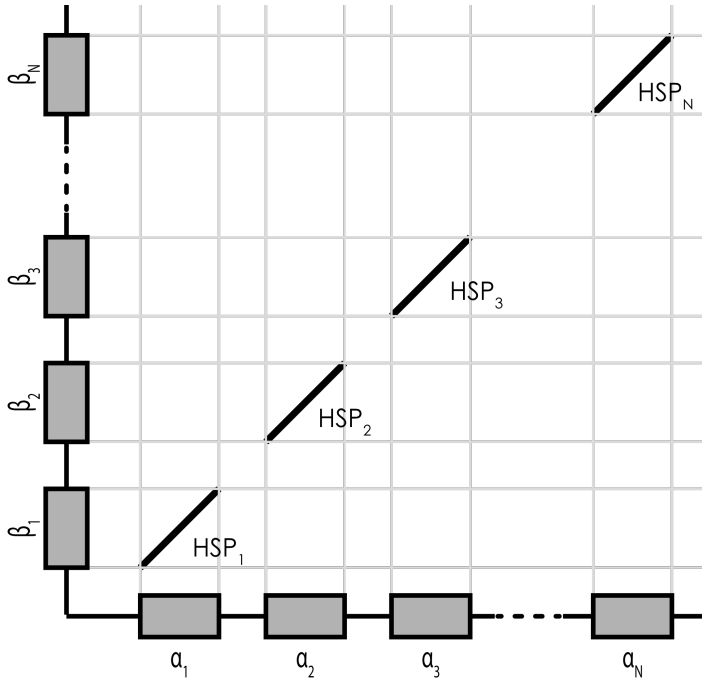


Fig. 3.5 A graphic representation of Conserved Elements from HSPs.

- *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.

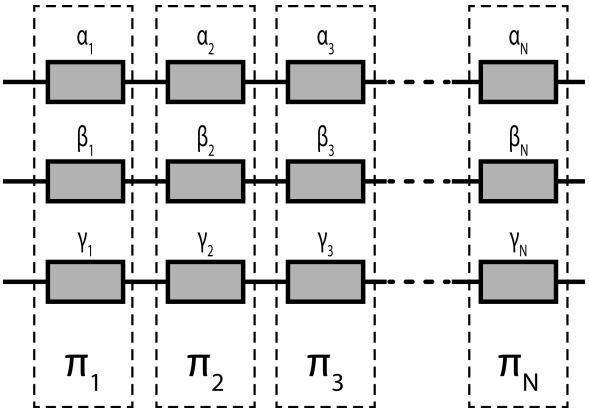


Fig. 3.6 Graphic representation of three Synteny Elements. Synteny Element π_1 links α_1,β_1 and γ_1 Unitary Conserved Elements.

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.

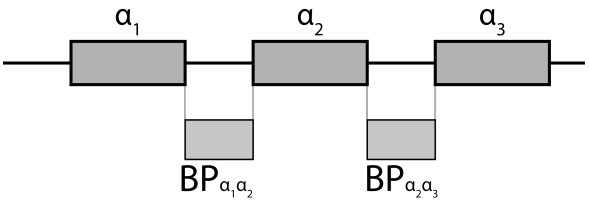


Fig. 3.7 Representation of Break Point. α_1, α_2 and α_3 are Unitary Conserved Elements.

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).

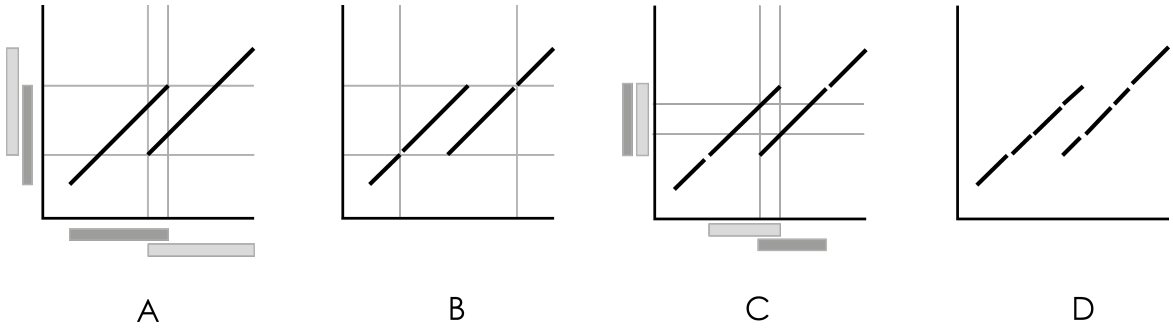


Fig. 3.8 Representation of the trimming process. A) Two overlapped HSPs. B) Result of the trimming process. The two HSPs have been split into four pairs of Conserved Elements. Two of them are still overlapped. C) New overlapped Conserved Elements trigger a new trimming process. D) Final result of the 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.

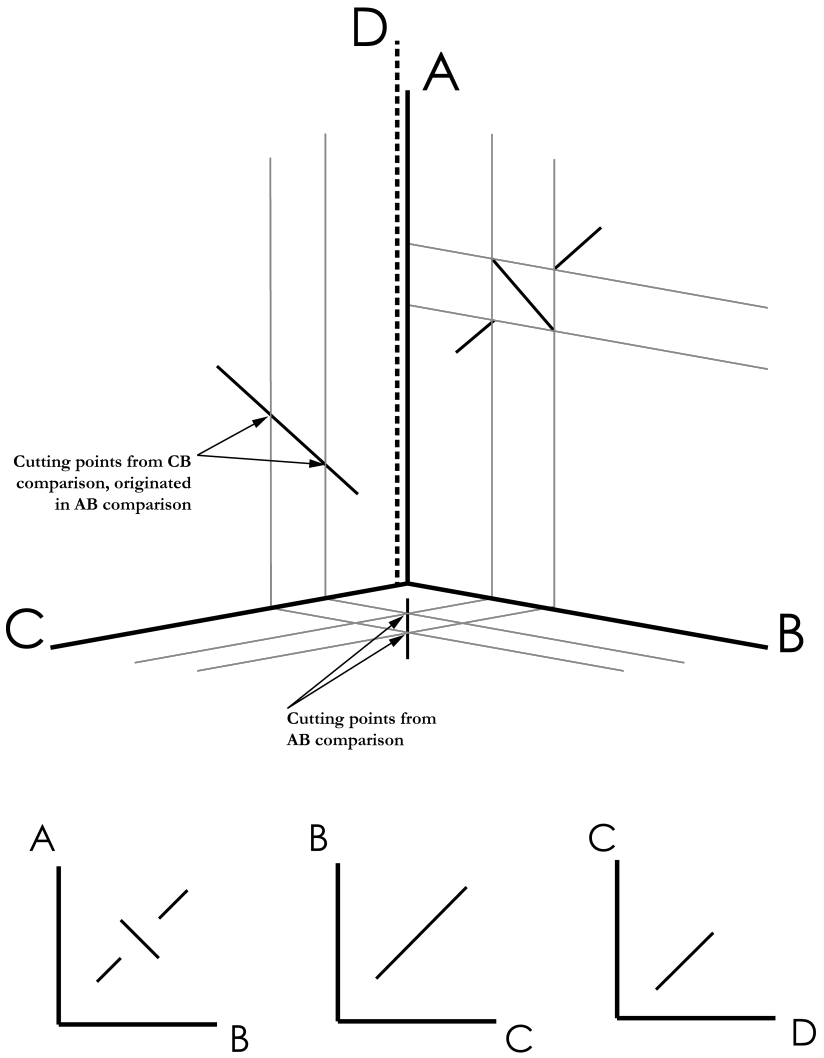


Fig. 3.9 Representation of the trimming process in a multiple comparison. In the comparison *AB* there is an inversion, that triggers a trimming process in the comparison *BC*. As a result, another trimming process is triggered in comparison *DC*.

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

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

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

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

- and

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

- 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.

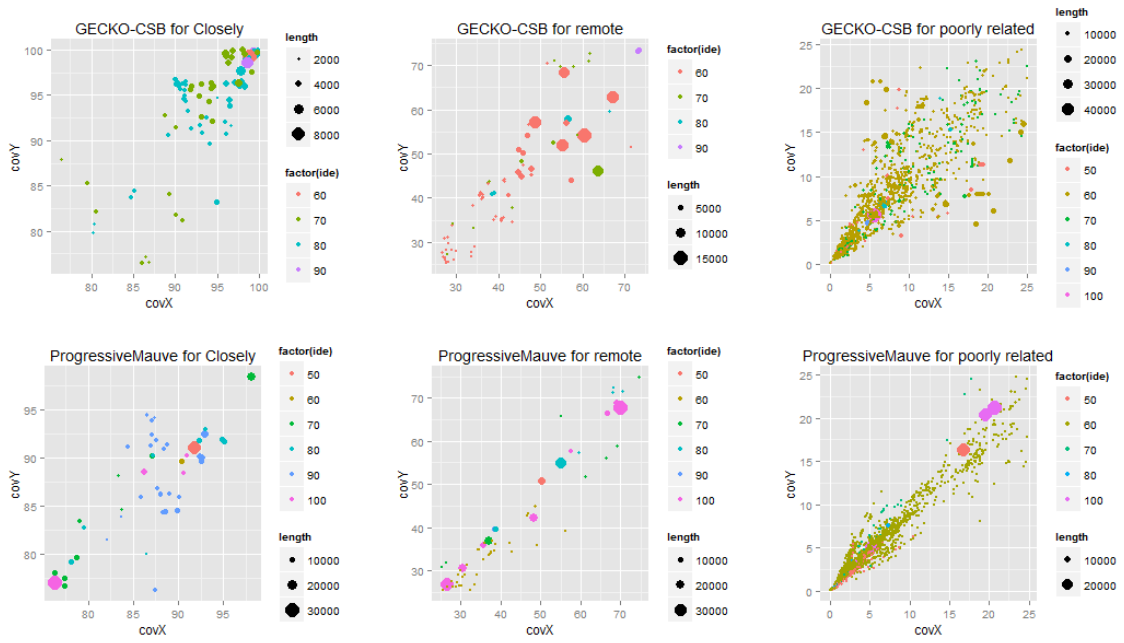


Fig. 4.1 Average length, average percentage of identity, and coverage from all against all comparison of 68 mycoplasmas. Grouped by closely, remote and poorly related species. The X and Y axis represent coverage (as percentages) in the sequences. Each point represents a comparison. The color represents 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.

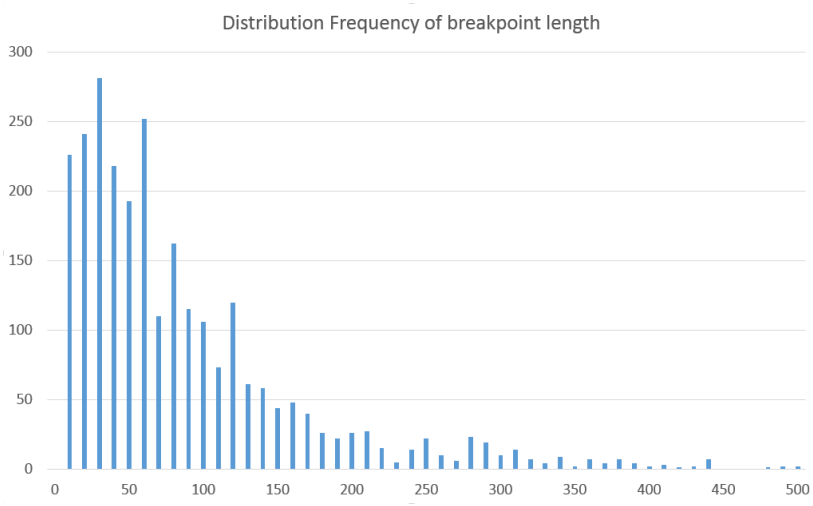


Fig. 4.2 Frequency distribution of Breakpoint length.

- 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.

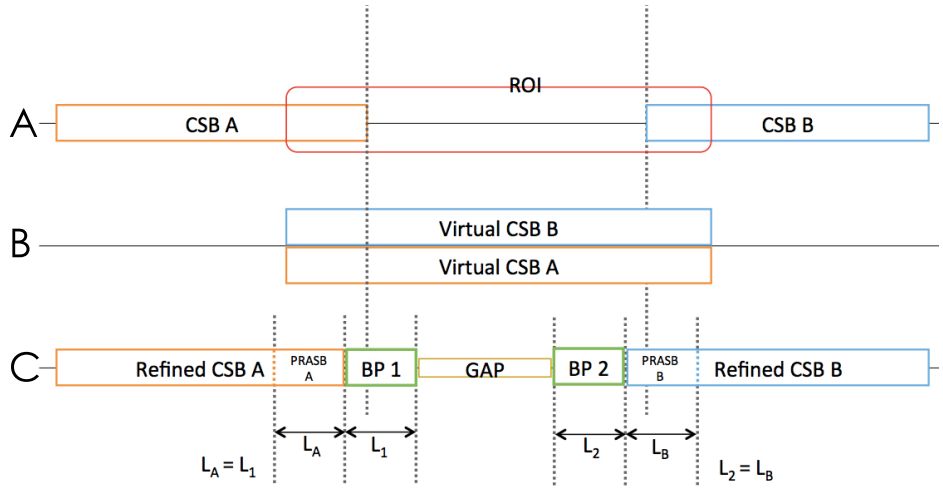


Fig. 4.3 CSBs before and after the refinement. A) Selection of the Region of Interest (ROI), between two Computational Synteny Blocks (CSB). B) Representation of the virtual CSBs. C) Result after the refinement process. We also detect BPs and extract PRASB and GAP sequences to analyse the accuracy of 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.

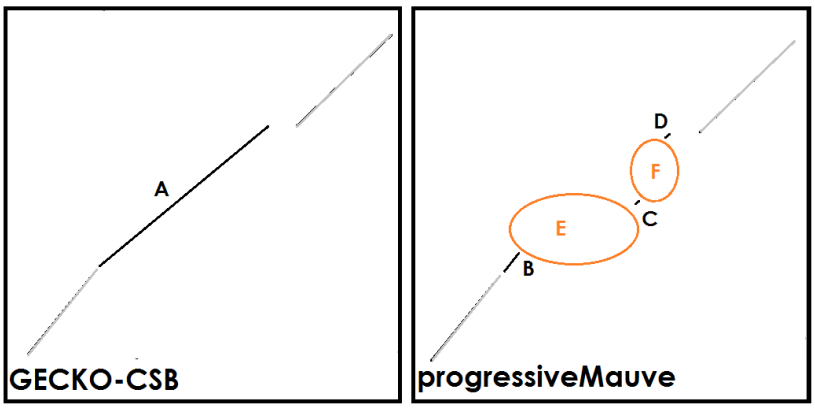


Fig. 4.4 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects one SB. (b) progressiveMauve detects three SBs (B,C and D). The reasons of this difference are explained in the main text.

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, progressive-Mauve 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.

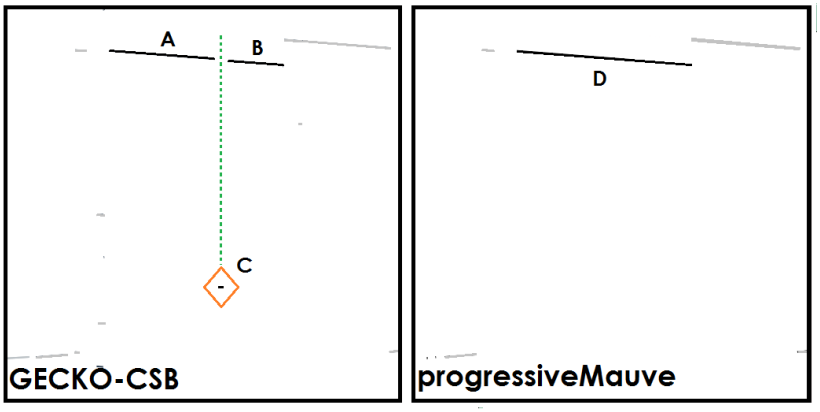


Fig. 4.5 Differences of SB detection for a certain region in the genomes using Gecko-CSB and progressiveMauve methods. (a) Gecko-CSB detects three SBs (A,B and C). (b) progressiveMauve detects one large SB.

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.

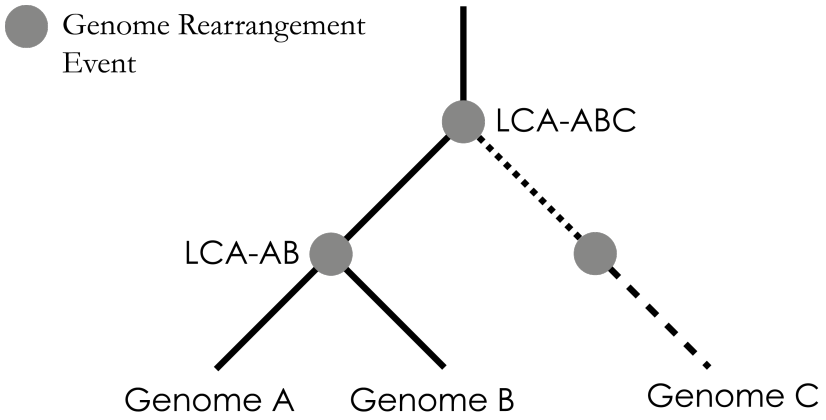


Fig. 5.1 GenomeA and GenomeB are included in the public database. Genome C is not in the database. If a certain metagenome read comes from Genome C specie, it might match better in the Last Common Ancestor-ABC than in the genomes A or B, because the genome C is not present in the database.

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.

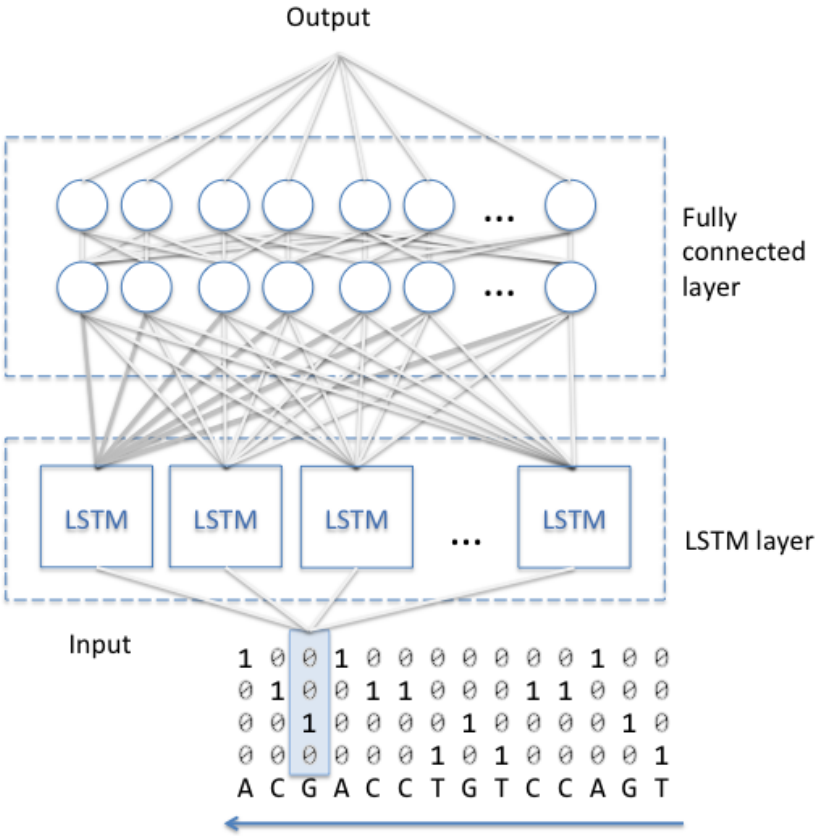


Fig. 5.2 The input is encoded as one-hot vector. After the LSTM layer, we use a fully connected layer to combine all the LSTM cells outputs to produce a single output.

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):819–31.

[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 break-points.

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

Computational Synteny Block: A Framework to Identify Evolutionary Events

Jose A. Arjona-Medina* and Oswaldo Trelles

Abstract— **Motivation: 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 genomic rearrangements. Methods: 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. Results: 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. Data sets and Supplementary Material are available at: http://bitlab-es.com/gecko-csb/.**

Manuscript received April 6, 2016; accepted April 9, 2016. Date of publication April 20, 2016; date of current version August 12, 2016. This work was partially supported by the Mr.SymBioMath IAPP (Project code: 324554), the 'Plataforma de Recursos Biomoleculares y Bioinformaticos (ISCIII-PT13.0001.0012)', 'Proyecto de Excelencia Junta de Andalucia (P10-TIC-6108), the Health Government of Andalucia (PI-0279-2012), and the RIRAAF network (RD12/0013/0006). *Asterisk indicates corresponding author.*

*J. A. Arjona-Medina is with the Advanced Computing Technologies Unit, RISC Software GmbH, Hagenberg, Upper Austria 4232. (e-mail: arjona@uma.es; jose_arjona@risc-software.at).

O. Trelles is with the Department of Computer Architecture, University of Malaga, Campus de Teatinos, Malaga 29071. (e-mail: ortrelles@uma.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNB.2016.2554150

Index Terms— **Breakpoints, computational synteny blocks, duplications, repeats, synteny blocks, tandem repeats.**

I. Introduction

EVOLUTIONARY Events (EE)—or large genomic rearrangements—play a crucial role in the evolution of species. Evolutionary events-based studies focused on molecular evolution are aimed at finding series of genomic rearrangements that transformed a genome into a different one. To perform this task, other tasks must be conducted before, ranging from detecting HSPs to grouping them into synteny blocks (SBs) [1], refining SBs boundaries [2], searching and refining breakpoints [3], identifying the type of rearrangement, and establishing a sequence of rearrangements.

A number of current software tools start with anchors that represent conserved regions, to end with a set of SBs. Some tools, such as Cassis [3] or Orthocluster [4], use a list of orthologous genes. On the other hand, tools as Cyntenator [5] represent genes as alphabetic letters and perform a BLAST-like comparison. Other programs (e.g., MCScan [6] or i-AdHore 3.0 [7]) combine information on gene position and protein sequences to perform an all-against-all comparison by using BLASTP. In all these approaches, the obtained results depend on annotated information that is not available for non-coding DNA.

Some alignment tools like MUMmer [8] are able to detect blocks of rearrangements. Shuffle-LAGAN [9], for example, uses the glocal alignment algorithm to detect rearrangements, whereas progressiveMauve [10] uses a recursive approach to find conserved collinear regions. GR-Aligner [11] can find and refine breakpoints of rearrangement events. Although these methods were not designed for identifying SBs, they are able to find rearrangement events.

All these software tools share a common feature: they need several parameters and restrictions in order to compute SBs (i.e., minimum SBs lengths, minimum number of blocks, non-overlapped SBs, etc.). According to Ghiurcuta and Moret [12], the definition of SBs does not seem to be strict enough, which might explain why current tools yield widely different results. A comparative table on the different methods for computing SBs has been included in Supplementary Material, Section I.

Repeats, tandem repeats and duplications make difficult the detection of SBs. Thus, in order to make SBs detection easier, many of the methods used to calculate SBs—such as DRIMM-Synteny [13], GRIMM-Synteny [1], or GRAligner [11]—avoid these events. There are also parameter-dependent methods, which manage larger blocks in the presence of repeats by filtering small blocks; for example,

MUMmer [8] needs a minimum span of block per species and a maximum gap, whereas progressiveMauve [10] and GRIMM-Synteny [1] use a recursive approach to remove the small blocks that break collinearity.

However, a considerable part of genomes are repetitive. Thus, almost 50% of the genome in humans and 80% in maize is repetitive DNA [14]. Although repetitions are not that common in bacteria, they play an important role in regulating expression patterns, and they can potentially destabilize the genome [15]. It is also widely known that repetitions drive and play an interesting role in evolution in many ways [16], [17].

Many programs have been developed to identify repeats. A 2008 study revealed that there are profound differences among these different methods [18]. For example, RepeatMasker,[1] RepeatScout [19], or ReAS use the RepBase database [20] to find repeats and low complexity DNA sequences in a genome. Recon [21] and PILER [22] use sequence self-comparison to detect repeats. However, they were not designed to detect rearrangements.

On this background, we present an automatic and precise method for identifying computational synteny blocks (CSB), which are equivalent to synteny blocks. The method takes into consideration coding, non-coding regions, and repeats.

II. Materials and Methods

In this section, the term "tandem repeats" refers to a pattern of one or more nucleotides that are repeated adjacent to each other[2]; and "repeats" are patterns of nucleotides that occur in multiple copies along the genome.

Our method starts with the collection of HSPs (a.k.a. ungapped local alignments or fragments). Small evolutionary events (EE) —in particular indels and duplications— can produce overlapped HSPs at a short distance from each other. A number of these HSPs are overlapped in both sequences. In the second step, these double-overlapped HSPs are merged into a single gapped fragment. Coarsely speaking, two fragments combine to form a new gapped fragment characterized by their minimum start-coordinates and maximum end-coordinates. In this process, tandem repeats are detected.

The precise site where gaps are incorporated into the gapped fragments is determined by using a modified dynamic programming approach, with a bounded window). This step also includes refining the quality measures of the collected gapped fragments (identity, length, etc.).

In the third step, a recursive method is applied to compute CSBs composed by a collection of fragments that maintain co-localization in the sequences. Once CSBs are computed, repeats—group of fragments that overlap in only one sequence—are identified. This process allows computing larger CSBs, since repeats spuriously break collinearity in CSBs. Finally, EEs are identified. Next sections describe the method in more detail.

A. Detection of HSPs

HSPs are extracted using Gecko [23]. Gecko is an application composed of a set of modules organized as a workflow to compute the set of fragments between two biological sequences. This software has demonstrated its capacity to yield HSPs of high-quality beating reference software.

However, any other program able to identify ungapped local alignments can also be used, provided that their output is converted into Gecko output format. The sensitivity of these programs generally depends on certain parameters such as the length of the K-mers employed to identify seed points to accelerate the process, or p *-value* thresholds. The resulting HSPs are described as an 8-tuple $f = xStart, yStart, xEnd, yEnd, length, strand, score, identities$ where $xStart, xEnd$, and $yStart, yEnd$ represent anchoring in genome X and Y respectively, and $strand$ could be forward or reverse. Reverse fragments are found by comparing genome X and the reverse complement of genome Y. Notice that $xEnd$ and $yEnd$ are redundant for ungapped fragments, but they are necessary for gapped fragments.

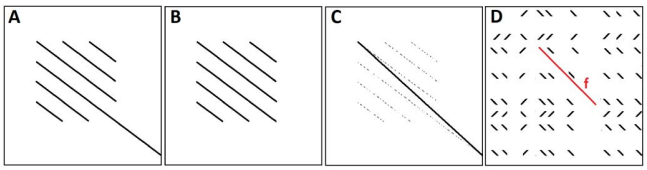


Fig. 1. Tandem repeats and repeats detection. This figure represents a scenario where tandem repeats are detected. (A) shows fragments that overlap both in X and Y sequences (double overlapped. (B) shows the tandem repeats that will be reported following the "tandem repeats detection" procedure. (C) shows the result of the "merging double overlapped" process. The new fragment is the result of connecting the extreme values of the set of fragments represented in (A) (dotted lines are shown for reference only). (D) Detection of repeats. The figure illustrates a set of candidate fragments to conform a repeat group. The difference between the length of fragment f and the length of any other fragment is greater than their minimum length. The difference in length between any fragment (excluding f) is less than their minimum length. Therefore, all fragments—f fragment excluded—form a group of repeats.

B. Merging Double-Overlapped Fragments and Tandem Repeat Detection

Two fragments that overlap partially or totally in the X and Y sequence are called "double-overlapped." They merge into a single fragment that connects their ends. Formally speaking, two fragments f_i, f_j are double-overlapped if

$$\begin{aligned} &\min(f_i \cdot xEnd, f_j \cdot xEnd) \\ &\quad - \max(f_i \cdot xStart, f_j \cdot xStart) > 0 \\ &\min(f_i \cdot yEnd, f_j \cdot yEnd) \\ &\quad - \max(f_i \cdot yStart, f_j \cdot yStart) > 0 \\ &\qquad f_i \cdot strand == f_j \cdot strand. \end{aligned} \quad (1)$$

This function is called $Overlap(f_i, f_j)$. The coordinates of the new fragment are computed as follow:

$$\begin{aligned} f_{new} \cdot Start &= \min(f_i \cdot xStart, f_j \cdot xStart) \\ f_{new} \cdot End &= \max(f_i \cdot xEnd, f_j \cdot xEnd) \\ f_{new} \cdot yStart &= \min(f_i \cdot yStart, f_j \cdot yStart) \\ f_{new} \cdot yEnd &= \max(f_i \cdot yEnd, f_j \cdot yEnd). \end{aligned} \quad (2)$$

At the same time that double-overlapped fragments are merged into a single fragment, tandem repeats are found because the overlapped section defines a tandem repeat (see Fig. 1(a)). Formally speaking, given

[1] http://www.repeatmasker.org

[2] U.S. National Library of Medicine Medical Subject Headings (MeSH).

two double-overlapped fragments f_i, f_j a tandem repeat is defined by:

$$f_{tan} \cdot Start = \max(f_i \cdot Start, f_j \cdot Start)$$
$$f_{tan} \cdot End = \min(f_i \cdot End, f_j \cdot End). \quad (3)$$

Noteworthy, this formula is valid both for X and Y sequence coordinates. Additionally, this formulation is also valid when a fragment is fully overlapped by the other.

C. Refining Quality Measures of Gapped Fragments

By definition, HSPs are highly conserved fragments between genome X and Y, but they do not necessarily represent the best possible alignment between their anchors, since they do not consider indels. Moreover, the previous step converts the collection of ungapped HSPs into gapped fragments, therefore, the $score$, $identities$, and $length$ must be recalculated.

1) Dynamic Programing Algorithm for Global Alignment (Needleman and Wunsch) to Analyze Only a "Subset" of Solutions: We use the Needleman and Wunsch algorithm [24] to calculate the best possible alignment for each fragment. We use a customized dynamic programming approach based on a limited computational space bounded by the diagonals of the joined fragments.

The full dynamic programming algorithm described by Needleman and Wunsch to find the best score with gaps (and the best alignment can be derived from this finding) has $O(N^2)$ complexity. This complexity means not only CPU time consumption but also memory allocation for a big matrix that growths with the length $(N \times M)$ of the sequences.

In Gecko-CSB program we use this expensive dynamic programing algorithm but with a reduction of the computational space. The rationale is that we apply this procedure only for fine-tuning the solution. In fact we already know the set of ungapped fragments and therefore we need only to explore a partial computational space. Reasons to reduce the space arise because of the long length of the sequences. Some HSPs can have more than 1 Mbp, therefore we would need approx. 1 Mbp $\times$ 1 Mbp $\times sizeOfDataStructure$, which is clearly unaffordable even running in multiprocessors machines.

Therefore, customization is necessary due to the large computational space defined by the coordinates of the involved fragments.

2) Length of Gapped Fragments: A fragment is defined by two subsequences. When a fragment is reported, it usually has start and end positions in both sequences, which define a subsequence. The length of the subsequences can be computed by subtracting end and start positions.

If the fragment is ungapped, the length in both subsequences is the same. If the fragment is gapped, sequence X length and sequence Y length might not be the same. In this case, an alignment between subsequence X and Y is needed in order to get the length of the fragment. Once the alignment is performed, both subsequences (which include gaps) will have the same length. Therefore, the length of the fragment is the length reported in the alignment including *gaps*.

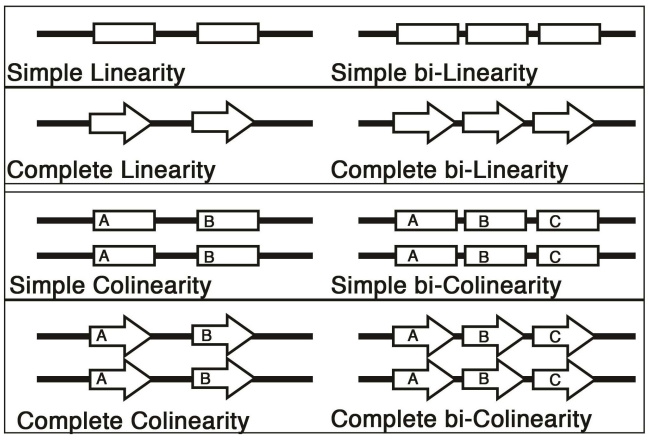


Fig. 2. Linearity and collinearity properties. This figure shows the properties of linearity and collinearity. Two horizontal lines represent the sequences under comparison or the sequence that meets a given property. The rectangles represent blocks where the strand is not relevant, whereas the arrows represent blocks in a given orientation. Similar blocks share the same marker (A, B, or C).

D. Definition of Computational Synteny Blocks

In this section we will start with the definition of computational synteny blocks (CSBs) and some EEs. A CSB is defined as a set of fragments that conserve strand and collinearity in both sequences. Notice that this definition presumes that these fragments are free of internal rearrangements. At this point a reduced set of N gapped-fragments is available $F = f_1, f_2, \ldots, f_N$ where the i^{th} element of the collection F is represented by f_i. The cardinality of the ordered fragments in X and Y is contained by O_X, and O_Y respectively.

P_X and P_Y represent the inverse index of O_X and O_Y. For example, O_X^i represents the order of $f_{P_x^i}$ in the genome X. Since duplications are allowed, if two or more fragments are overlapped in X or Y, they will share the same order. In this case, if f_{P^i} and f_{P^j} are duplicated in the genome X, then O_X^i is equal to O_X^j. Note that both in O_X and O_Y cases, $O^i \leq O^{i+1}$.

E. Linearity and Collinearity

This section includes a set of formal definitions describing different levels of linearity and collinearity between fragments that are employed to detect CSBs. The function $ST(f_i)$ returns $+1$ if $f_i.strand$ is forward and -1 if $f_i.strand$ is reverse.

1) Simple and Complete Linearity: Two fragments f_{P^i} and f_{P^j} fulfill $SimpleLinearity$ (SL) in a sequence if $abs(O^i - O^j)$ is equal to 1. In SL the strand of CSB is not taken into account. In $CompleteLinearity$ (CL) CSBs must have the same strand. Two fragments f_{P^i} and f_{P^j} satisfy CL in a sequence if $(O^i - ST(f_{P^i}))$ is equal to $(O^j - 2 \cdot ST(f_{P^j}))$.

2) Simple and Complete Collinearity: Two fragments f_{P^i} and f_{P^j} fulfill $SimpleCollinearity$ (SC) if they satisfy SL in the sequences X and Y, and they fulfill $CompleteCollinearity$ (CC) if they satisfy CL in both sequences. At this point, the linearity and collinearity between two fragments has been analyzed. However, the relationship among fragments can involve three or more fragments taken

in pairs. The concepts of bi-linearity and bi-collinearity are described below.

3) Simple and Complete Bi-Linearity: Three fragments, f_{Pi}, f_{Pk} and f_{Pj} fulfill $SimpleBi\text{-}Linearity$ (SbL) in a sequence if they fulfill SL taken in pairs in the sequence: $SbL(i,k,j) = SL(i,k) \cdot SL(k,j)$. Three fragments, f_{Pi}, f_{Pk} and f_{Pj} fulfill $CompleteBi\text{-}Linearity$ (CbL) in a sequence if they fulfill CL taken in pairs in the sequence: $CbL(i,k,j) = CL(i,k) \cdot CL(k,j)$

4) Simple and Complete Bi-Collinearity: Three fragments, f_{Pi}, f_{Pk} and f_{Pj} fulfill $SimpleBi\text{-}Collinearity$ (SbC) if they fulfill SC taken in pairs. $SbC(i,k,j) = SC(i,k) \cdot SC(k,j)$. Three fragments, f_{Pi}, f_{Pk}, and f_{Pj} fulfill $CompleteBi\text{-}Collinearity$ (CbC) if they fulfill CC taken in pairs. $CbC(i,k,j) = CC(i,k) \cdot CC(k,j)$

F. Automatic CSB Detection

According to the definitions of linearity and collinearity, if two fragments belong to the same CSB, then they fulfill $CompleteCollinearity$. During CSB calculation, if two or more fragments belong to the same CSB, they can be merged into a new fragment by using the same procedure $MergeCSBfunction$ described in the step "Merging double-overlapped fragments." Since a CSB is represented as a fragment that connects the heads and tails of the fragments, all functions available for fragments can also be applied to CSBs. This process repeats until no new CSB is detected. After the Merging process, the quality measure of the new CSB must be recalculated using the same approach as that described in the section *Refining quality measure of gapped HSPs*.

The algorithm to calculate CSB can be summarized as follows:

```
1: while REPEAT is true do
2:   REPEAT = false
3:   for f_Pi ∈ F do
4:     if CL(i, i+1) then
5:       mergeCSB(i, i+1)
6:       refiningQualityMeasure(i, i+1)
7:       REPEAT = true
8:     end if
9:   end for
10: end while
```

G. Detecting Repeats

Once CSBs are obtained, repeats can be found. In order to speed the process up, all overlapped CSBs are selected to be processed in the next step (see Fig. 1(d)). In order to avoid long fragments in the repeats selection, the following condition was set: two fragments that are overlapped can be part of a group of repeats if their length difference is less than the minimum of its length: $abs(f_i{\cdot}length - f_j{\cdot}length) < \min(f_i{\cdot}length, f_j{\cdot}length)$

Once a group of possible repeats is found, the order of each fragment in X and Y is recorded in O_x and O_y. Groups of duplicated CSBs in Y and X are recorded in D_x and D_y, respectively. In the Fig. 1(d) Dx values range up to 9 (number of columns) and Dy values range up to 8 (number of rows). This information is combined by spreading (i.e., if the first row is marked as group 1, all columns related to these fragments will be marked as group 1). At the end of the process, all fragments of a grid will be assigned to the same group.

The algorithm to detect these groups is summarized below:

```
1: F_dup = selectAllPossibleRepeats(F)
2: for f_Pi ∈ F_dup do
3:   if Overlap_X(f_Pi, f_Pi+1) then
4:     D_X^i = D_x^{i+1} = block_X
5:   else
6:     block_X = block_X + 1
7:   end if
8:   if Overlap_Y(f_Pi, f_Pi+1) then
9:     D_Y^i = D_Y^{i+1} = block_Y
10:  else
11:    block_Y = block_Y + 1
12:  end if
13:  CombineInformation(D_X, D_Y)
14: end for
```

H. Identifying Evolutionary Events

In this section a set of rules is defined in order to identify large EEs. An Inversion is an EE that changes the strand of a region in a sequence. A transposition is an EE that moves the region of a sequence from its original position to another position. Fig. 3 illustrates these EEs.

1) Detecting Inversions: Given a CSB A, the reversion of A is noted as $\overline{A}$. This operator changes the CSB's strand. Given three CSBs A, B, and C, B is reverted if and only if $A, \overline{B}, C$ fulfill CbC.

2) Detecting Transposition: Given five CSBs A, B, C, D and E, B is transposed if and only if: A, B, C fulfill CbL in X (or Y); D, B, E fulfill CbL in Y (or X); A, C fulfill CL in Y (or X); and D, E fulfill CL in X (or Y).

3) Detecting Reverted Transposition: In the case that $A, \overline{B}, C, D, E$ satisfy the rule for transpositions, then B has undergone a reverted transposition.

4) Detecting Duplications: Since the order of CSBs in X and Y are stored—excluding the groups of IRs—the CSBs that share the same order are considered duplications.

III. Results and Discussion

In this section, we present three different experiments that were conducted to validate the proposed set of algorithms (Gecko-CSB) at specific and global level: in the first experiment simple cases were used to dive into the algorithms, in the second experiment we compare Gecko-CSB against GRIMM-Syntenty while the third was a massive experiment that covered aspects such as algorithm performance associated with closeness in evolution (closely- and remotely-related genomes). The tests reported in this document were performed

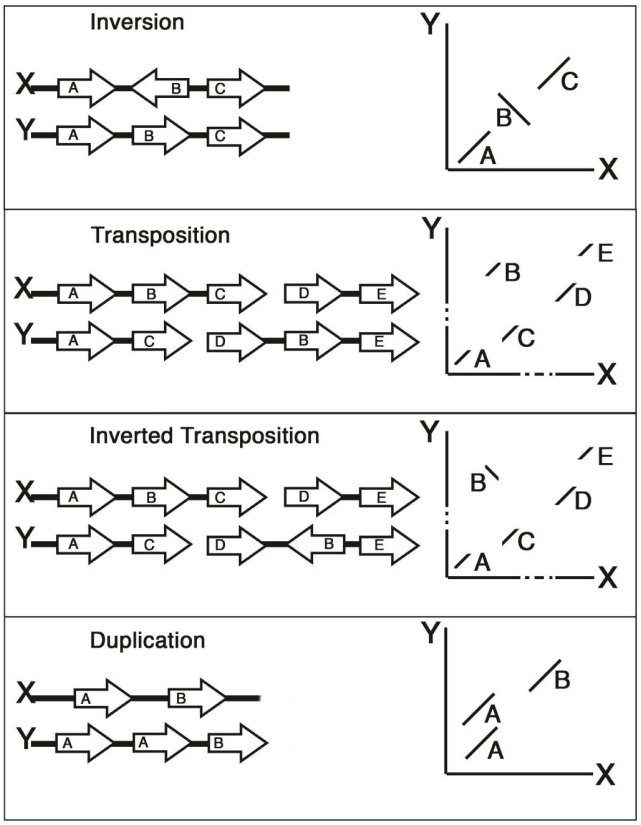


Fig. 3. Identification of evolutionary events. This figure shows four different evolutionary events that are detected by using linearity and collinearity properties. On the left are shown block positions in the sequences, whereas a dotplot-like representation is displayed on the right. Similar blocks share the same marker (A, B, C, D, or E). From top to bottom are represented: (a) Inversions: Blocks A, B, and C fulfill simple bi-collinearity but not complete bi-collinearity because block B does not share the same orientation in both sequences. If block B were reverted (in one sequence), A, B, and C would fulfill complete bi-collinearity. (b) Transpositions: blocks A, B and C fulfill Complete Linearity in sequence X. Blocks D, B, and E fulfill Complete Linearity in Y. Blocks A and C fulfill Complete Linearity in Y, and blocks C and D fulfill Complete Linearity in sequence X; (c) Inverted transpositions: blocks A, C, D, E, and reverted B fulfill the conditions explained in the section "Transposition"; and finally (d) duplications: in this example, A is duplicated in sequence Y. Notice that cases C and D are not linear neither in X nor in Y.

in the Picasso multiprocessor located at the University of Malaga.

A. Infrastructure

The tests reported in this document were performed in the Picasso multiprocessor located at the University of Malaga, Spain.[3] We used the same machine in the three experiments to avoid any biases associated with computational resources. Picasso is composed of shared and distributed-memory machines with up to 984 nodes. The distributed-memory part contains 48 nodes, each with two Intel E5–2670 processors with a performance of 160 Gflop/s. The peak performance of the distributed-memory part is 16 Tflop/s. Each node or pair of processors has 64 GB of RAM giving an aggregate memory of 3 TB. Interconnection is achieved through an Infiniband FDR network at 54.54 Gbit/s.

[3]http://www.scbi.uma.es/site/

B. Software

We discarded the large set of algorithms available for the identification of rearrangements because the vast majority are based on gene annotations or protein information and they are only able to identify SBs in regions containing genes but not in non-coding regions. The selected algorithms —which have a wider scope— were expected to outperform the other algorithms available. Consequently, we focused on GRIMM-Synteny and progressiveMauve.

The results from Gecko-CSB were compared against those reported by GRIMM-Synteny 2.02 [25] and progressiveMauve [10]. These software packages were downloaded from the official websites and installed in the Picasso multiprocessor following the provided instructions.

GRIMM-Synteny starts with HSPs provided by Gecko application, although any other software able to identify HSPs can be used (i.e., Blast-like programs). progressiveMauve starts with sequences since it can detect HSPs by using its own method to detect locally collinear blocks (LCBs) [10]. Strictly speaking, progressiveMauve is used for genome alignment based on LCBs. However, in this document we define CSBs as a set of fragments that conserve order and strand in both CSBs sequences and are free of internal rearrangements. It is noticeable that under this definition both, LCBs and CSBs are equivalent to some extent; however, our method admits overlapping among CSBs to allow for the detection of repeats and duplication events, elements avoided by progressiveMauve.

C. Quality of the Results

In this section we explain how we evaluated the quality of the results yielded by Gecko-CSB versus other applications using the same (or equivalent) parameters.

The rationale for this evaluation is to compare the coverage of the CSBs detected by each algorithm accounted as the number of positions covered by CSBs, HSPs, or LCBs (according to the nomenclature used in each software tool) and the percentage of identities (alignment quality). Coverage of coding and non-coding regions are reported in additional columns, since they are not equally relevant. The positions of coding regions were obtained from NCBI. In Gecko-CSB we included a further column to show the features of repeats, which are not detected by the other methods. Certainly, there must be more sophisticated ways of comparing results, such as qualifying and weighing matches depending on the closeness of the sequences under comparison. However, we decided not to use these methods as they may incorporate noise or biases in the evaluation.

It is important to observe some details on coverage computing. To compute coverage we map on the corresponding sequence only the ungapped part of the fragment. As a result, long blocks could produce lesser coverage because they could incorporate more gaps in their alignment than other equivalent blocks. A simple example of coverage in sequence X and Y for a set of 3 fragments is shown in Fig. 4

D. Test-I: Pairwise Comparison (Simple Use Case)

We start with a simple case termed Test-I to illustrate algorithm performance in synteny blocks and the detection

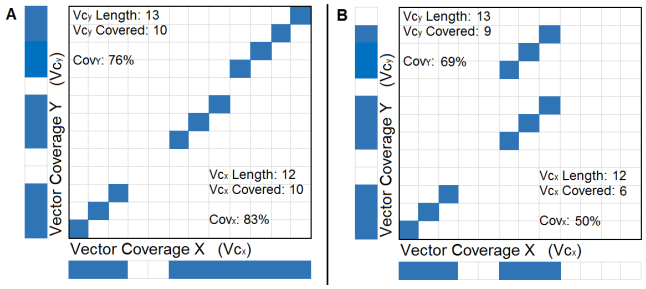


Fig. 4. Coverage of a fragment. This is an illustrative example to explain how the coverage of a set of fragments is calculated. (A) In this example we observe 3 fragments. Vc_x and Vc_y represent the coverage vector in the sequence X and Y. The set of fragments cover the same number of positions (10) in Vc_x and Vc_y. However, since sequence X is shorter in length than sequence Y, this set of fragments have better coverage in sequence X (83%) than in sequence Y (76%). (B) An example with 3 fragments where 2 of them are duplicated.

TABLE I

COMPARATIVE RESULTS OBTAINED FROM GECKO-CSB, PROGRESSIVE-MAUVE, AND GRIMM-SYNTENY ON TEST-I

(A) HSPs from Gecko are used as input for Gecko-CSB and GS. PM uses the sequences to produce its own HSPs. (B) CSBs and (C) Repeats are reported for Gecko-CSB and the aggregated value (B)+(C), (D) results from PM and (E) and (F) results from GS. Parameter settings are described in the main text.

	Gecko	Gecko-CSB			progressive Mauve	GRIMM Synteny
Values	HSP*	CSB	R	B+C	LCB	SB
	(A)	(B)	(C)	(D)	(E)	(F)
Number of fragments	1551	355	395	750	209	25
Length average	853	3083	602	1776	4142	40224
% of identities average	62.98	63.82	59.89	61.75	64.32	60.68
Coverage in sequence X	86.41	87.97	7.22	90.94	77.34	84.13
Coding regions	90.50	90.26	7.81	93.39	80.29	85.54
Non-coding	61.86	74.28	3.64	76.24	59.63	75.65
Coverage in sequence Y	77.79	79.15	7.82	82.30	67.60	78.33
Coding regions	81.65	81.54	8.31	84.80	70.45	79.14
Non-codign	47.59	60.56	3.98	62.81	45.32	72.04

*HSPs do not contains gaps, however to refine alignment we incorporate gaps producing fragments (to avoid incoherent gapped-HSP name)

of repeats using *Mycoplasma leachii* PG50 (Accession code NC-014751.1) and *Mycoplasma mycoides* subsp. *capri* (Accession code NC-015431.1) genome sequences with a length of 1008951 bp and 1153998 bp, respectively.

progressiveMauve software runs with default parameters. GRIMM-Synteny runs with the minimum-cluster-size parameter set to 1 to allow blocks made of one HSP in order to get comparable results (Gecko-CSB allows CSB made of only one HSP). GRIMM-Synteny were performed with gap threshold parameter set to 0. This parameter specifies the number of allowed blocks (clusters) between two blocks that will be merged into a single block. The equivalence in Gecko-CSB would be 0, since Gecko-CSB does not join CSBs that do not satisfy $CompleteCollinearity$.

To ensure comparability of the methods by the quality indices (in particular percentage of identity values), we compute these values using our modified version of Needleman and Wunsch [24] algorithm. Coverage values were computed as it was explained in *Quality of the results*.

Table I displays results for Test-I. progressiveMauve reports LCBs, GRIMM-Synteny SBs and Gecko-CSB reports results

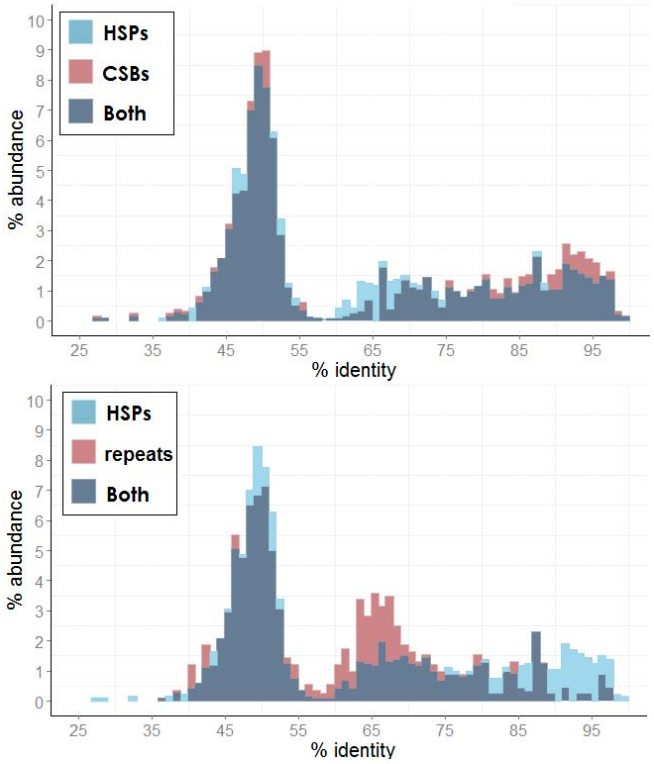


Fig. 5. Identity histogram for HSPs, CSBs, and repeats. This figure shows the histograms for HSPs versus CSBs (a) and HSPs versus repeats (b) by level of identity. The X axis represents percentages of identity values, whereas the Y axis represents normalized frequencies.

for CSBs (column B) after detection of repeats (column C). As it can be observed in Table I, Gecko-CSB has a better performance measured as coverage over both sequences and over both types of regions (coding and non-coding). Since progressiveMauve and GRIMM-Synteny remove overlapped segments, they have a lower coverage value.

Coverage obtained in coding regions is greater than that observed in non-coding regions. The main reason can be found in the fact that coding regions are better conserved and therefore it is easier to identify HSPs in those regions. For instance, when Gecko-CSB finds two blocks in coding regions separated by regions that do not contain HSPs but satisfy $CompleteCollinearity$, these two CSBs are merged, covering a greater region of the sequence, but with a collateral effect: the average identity is smaller (see Table I; column B+C) than Gecko (A) or progressiveMauve (E).

In order to better understand the identity values (associated with quality) obtained for HSPs, CSBs and repeats, Fig. 5 shows two histograms comparing HSPs versus CSBs (Fig. 5(a)) and HSP versus repeats (Fig. 5(b)), both distributed by identity level. In Fig. 5(a) the concentration of CSBs increases in the range of 90–100% as compared to HSPs. In the same range but in Fig. 5(b) we can observe, in the range of 60–70% of identity, a higher concentration of repeats as compared to HSPs. This explains why once repeats have been extracted, CSBs identity average is greater than HSPs identity.

TABLE II
COMPARATIVE RESULTS OBTAINED BY GRIMM-SYNTENY AND GECKO-CSB

	Anchors	Gecko CSB	Unique Coords	GRIMM-Synteny
Number of fragments	108	20	72	5
Length Average	99,891	1,549,855	94,885	6,302,395
Coverage in sequence X	10.31	38.78	8.75	40.36
Coverage in sequence Y	4.44	33.69	4.13	34.97

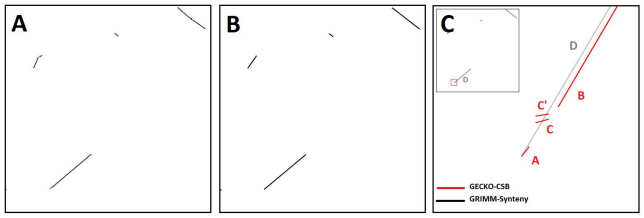


Fig. 6. Results of comparisons. This figure shows the different results for (A) Gecko-CSB result, (B) result from the GRIMM-Synteny, and (C) a detailed image of overlapped region. In red: Gecko-CSB's result. In gray: GRIMM-Synteny's result.

progressiveMauve and GRIMM-Synteny fragments are in general longer than Gecko-CSB's results because they do not take into account short fragments and remove all overlapped fragments. Indeed, this short repeats lower average length value.

E. Test-II. Grimsynteny vs Gecko-CSB

We performed a similar comparison using the dataset used by GRIMM-Synteny in their original document to verify that Gecko-CSB can deal also with anchors coming from others sources. Anchors from Human and Mouse in chromosome 18 were selected as input for the method presented in this work. Chromosome 18 was chosen due to the variety of evolutionary events presented in this comparison.

The minimum-cluster size parameter for GRIMM-Synteny was set to 1 to allow clusters made up of one block. The gap threshold parameter was set to 0 (the most restrictive).

GRIMM-Synteny performs a GRIMM-Anchors procedure to filter out anchors with conflicting coordinates. This process yields a set of unique coordinates (see Table II column Unique Coords). During GRIMM-Anchor process, overlapped anchors are extracted. This process prepares the ground for GRIMM-Synteny to detect fewer long blocks than Gecko-CSB. However, Gecko-CSB is able to work in the presence of such overlapped blocks and reports these repetitions as CSBs or IRs, if any. As a result, the coverage in both sequences is lower compared to GRIMM-Synteny, the number of CSB increases and the length average turns six times smaller.

Overlapped CSBs that break linearity between other CSBs are much smaller than them. For example, in Fig. 6(c) we observe two overlapped CSBs (C and C') that are breaking linearity between CSBs A and B. The length of CSB C is around 266 Kbp whereas the length of CSB B is around 15 Mbp. GRIMM-Synteny reports C and C' as repetitions and reports D as a Synteny Block. Gecko-CSB reports A, B, C, and C' as CSBs. Following the rules for detecting Evolutionary Events described in section *H*, C and C' are detected as duplications because they share the same cardinality order in sequence *X*.

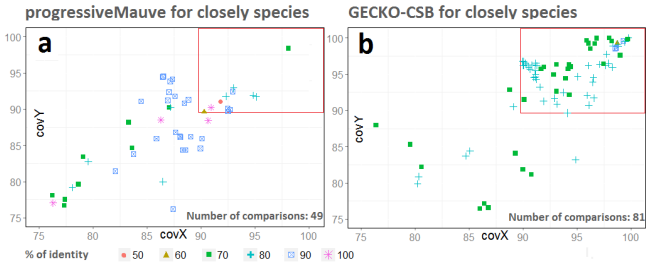


Fig. 7. Results from all-against-all comparison of 68 mycoplasmas. Closely related sequences. The *X* and *Y* axis represent coverage (as percentages) in the sequences. Each point represents a comparison. The color and shape of points represents the average identity in the comparison. This figure only shows those comparisons with a coverage greater than 75%. On the left (a) are shown 49 comparisons performed by progressiveMauve, whereas on the right (b) are shown 86 comparisons performed by Gecko-CSB. Figure (a) shows a higher concentration of comparisons that cover more than 90% (red square) in both sequences than in figure (b). Identities in (b) are smaller in general than (a) as we explained in subsection "Test-I."

F. Test-III. Comparison of 68 Mycoplasmas

In "Test-III" an all-against-all genome comparison of 68 *Mycoplasmas* was performed. This test was intended to avoid bias in the analysis that could arise from selecting two particular genomes. This large data set was prepared to contain different levels of orthology between genomes. The information needed to construct the data sets is available in the Supplementary Material, Section IV. Gecko-CSB and progressiveMauve were performed over the resulting 2.278 comparisons. Four main groups of comparisons were intuitively separated based on the coverage obtained: a comparison between two closely related sequences (more than 75% of coverage in both sequences), remotely related sequences (between 40% and 75% of coverage), poorly related (between 40% and 10% of coverage) and non-related (less than 10%). Fig. 7 shows the results in Gecko-CSB and progressiveMauve for closely-related species.

Gecko-CSB detected 81 comparisons with a coverage greater than 75%, 36 comparisons between 40% and 75% of coverage, 412 between 40% and 10%, and 1,749 under 10%. progressiveMauve reported 49 with coverage greater than 75%, 21 between 40% and 75%, 294 between 40% and 10%, and 1,914 below 10% (see Fig. 8 for a graphic representation). As it was discussed in Test-I, Gecko-CSB gets more coverage than progressiveMauve since Gecko-CSB does not remove any block and can handle overlapped fragments, which was confirmed in this massive experiment.

Gecko-CSB identified a total of 4,101 groups of repeats in 789 out of 2278 comparisons. These repeats cover up to 15% of the sequences in some comparisons, but they have in general less than 80% of identity. More details in Supplementary Material, section 8b. Fig. 9(a) shows a histogram of the average length of each group grouped by their identity average.

Repeats can be highly represented in comparisons. Fig. 9(b) shows a histogram displaying repeats length histogram but

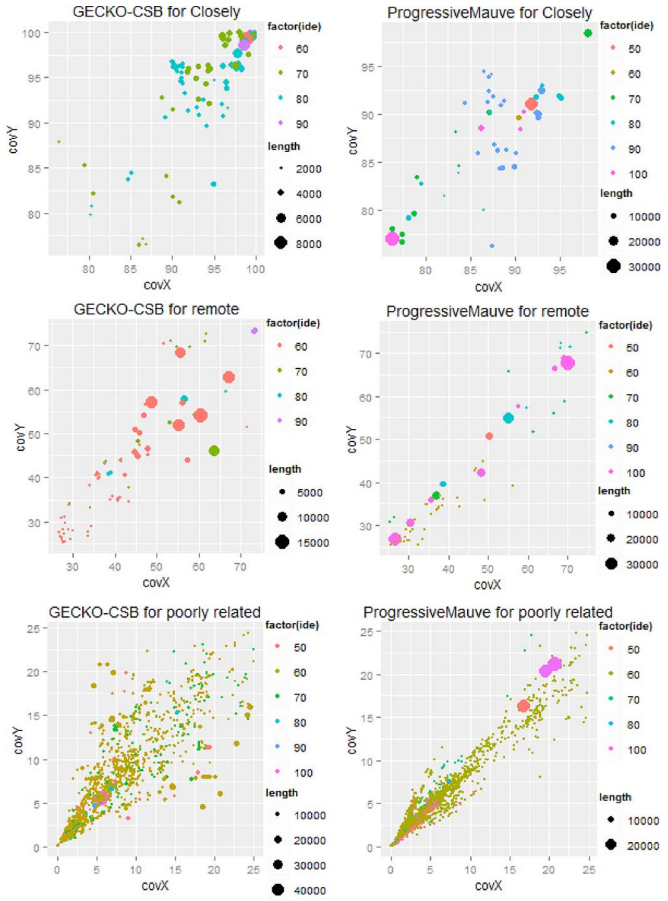


Fig. 8. Average length, average percentage of identity, and coverage from all-against-all comparison of 68 mycoplasmas. Grouped by closely, remote and poorly related species. The *X* and *Y* axis represent coverage (as percentages) in the sequences. Each point represents a comparison. The color represents the average identity in the comparison. The shape represents the average length of the blocks detected. On the left results from Gecko-CSB. On the right 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, specially in those comparisons of poorly related species.

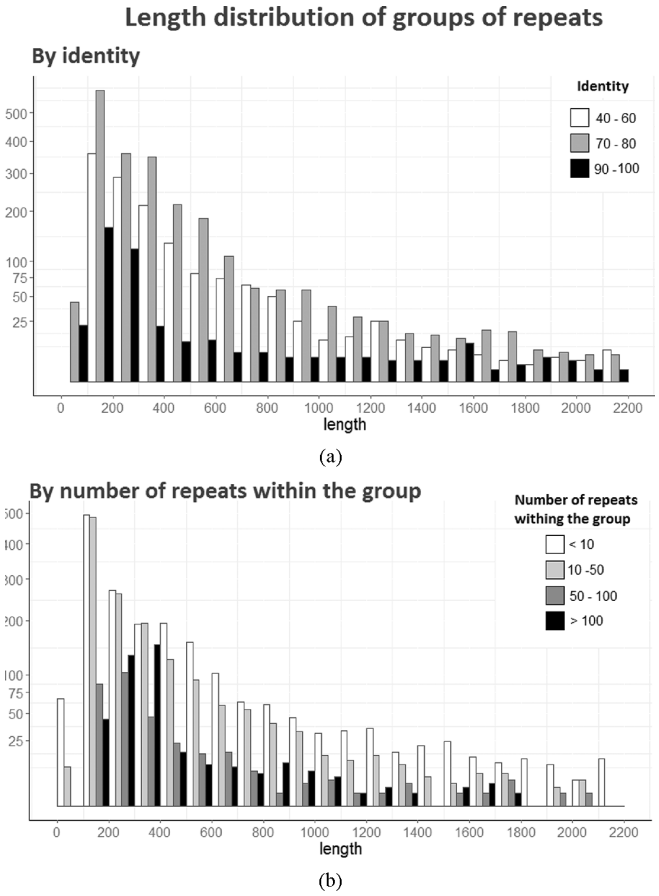


Fig. 9. Histogram of repeats lengths (a) grouped by identity average and (b) grouped by number of repeats within the groups. These repeats were identified in 2278 comparisons. Results are grouped by average identity. For example, the largest amount of groups of repeats appears at 200 of length. According to identity (a) the set of repeats with a length of 200 bp is made up of 286 groups with a 40% to 60% identity, 359 groups between 70–80% and 122 groups with more than 90% of identity. According to the number of repeats within the group (b) the set of repeats with a length of 200 bp is made up of 269 groups that have less than 10 repeats, 244 groups that have between 10 and 50 repeats, 102 groups that have between 50 and 100 repeats, and 132 groups that have more than 100 repeats.

showing the composition according to the number of repeats within the groups.

G. Locally Collinear Blocks in progressiveMauve vs Computational Synteny Blocks in Gecko-CSB

progressiveMauve calculates locally collinear blocks (LCB). A LCB is a subset of local alignments in a set of local multiple alignments that occur in the same order and orientation in a pair of genomes, free from internal rearrangements [10].

In Gecko-CSB, a computational synteny block (CSB) is defined as a set of similar regions (local alignments) that conserve strand and collinearity in both sequences.

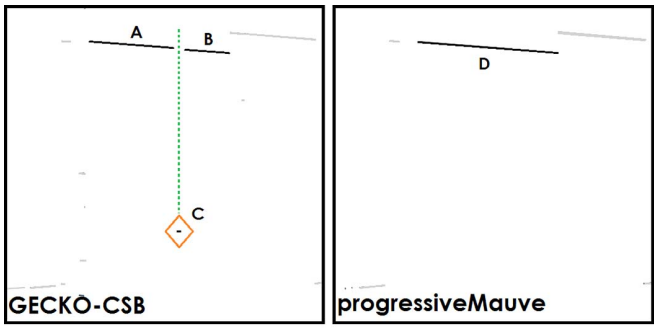
The difference between LCB and CSB definitions is that local multiple alignments in progressiveMauve are ungapped whereas similar regions in Gecko-CSB are gapped.

But there is one main difference in the way that LCB are built. While progressiveMauve is computing anchors, overlapping matches are trimmed. As a result, there is no overlapped LCBs in the process, whereas Gecko-CSB can report overlapped CSB (like repeats or duplications). Therefore, LCBs are a subset of local non-overlapping alignments.

In addition, in the xtit "Recursive anchor search" process, a heuristic *"greedy breakpoint elimination"* is performed in order to optimize the *"sum-of-pairs LCB anchoring objective function"*. As a result, progressiveMauve removes LCBs, decreasing the total number of LCBs in the solution. The anchoring function has two components, a breakpoint penalty and the sum of scores for each pair. The sum-of-pairs increases only if the sum of scores for each pair has a sufficiently small total score, *"favouring the deletion of small LCBs that interrupt large LCBs"*.

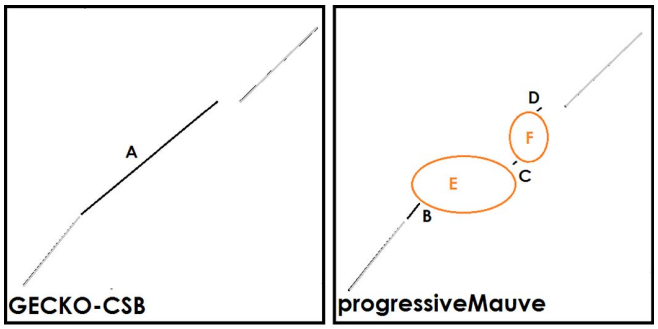
Gecko-CSB does not remove any pair from the set of local alignments (or CSBs in its recursive process), because it would break its definition. Only repeats are separated from the CSBs collection to be processed in a following step.

This main difference implies that progressiveMauve has in average longer fragments than Gecko-CSB. However, in



(a) (b)

Fig. 10. Gecko-CSB and progressiveMauve. (a) Region from Gecko-CSB. (b) Same region from progressiveMauve.



(a) (b)

Fig. 11. Gecko-CSB and progressiveMauve. (a) Region from Gecko-CSB. (b) Same region from progressiveMauve.

TABLE III
BLOCKS *A*, *B*, *C*, AND *D* COORDINATES IN FIG. 10

Block	X start	X end	Y start	Y end	length	%identity
A	142,085	148,863	996,171	989,340	6,862	87.12%
B	152,155	161,773	984,798	975,102	9,741	89.74%
C	149,959	150,495	329,085	329,681	622	60.45%
D	125,171	151,811	986,384	1,013,098	26,821	91.09%

TABLE IV
BLOCKS AND REGIONS COORDINATES IN FIG. 11

Block	X start	X end	Y start	Y end	length	%identity
A	98,017	107,018	120,789	127,736	9,512	50.93%
B	98,028	98,982	120,800	121,753	958	87.06%
C	104,840	105,043	124,239	124,431	220	58.18%
D	106,776	106,916	127,044	127,185	146	73.29%
E	98,982	104,840	121,753	124,239	5,963	30.71%
F	105,043	106,776	124,431	127,044	2,730	43.41%

some special cases, Gecko-CSB reports longer fragments than progressiveMauve.

We are going to analyze two kind of cases: CASE A: when progressiveMauve reports longer fragment than Gecko-CSB CASE B: when Gecko-CSB reports a fragment longer than progressiveMauve. Examples are extracted from Test-I, Mycoplasmas NC-014751 and NC-015431.

1) Case A: Progressivemauve Reports Longer Fragments Than Gecko-CSB: Fig. 10 shows a region where progressiveMauve reports a longer fragment (D) than Gecko-CSB (Fragment A, marked with an oval and fragment B). CSBs A and B have 6862 bp and 9741 bp of length respectively (see Table III for more details), and the shorter CSB C has only 622 bp of length and 60% of identity. This small fragment is removed by progressiveMauve because fragment D (A joined with B) improves the score (91%). However, Gecko-CSB does not merge fragment A with B (like progressiveMauve does) and detects the CSB C because it is breaking the linearity property.

To test the accuracy of this interpretation, we performed a database search using the subsequences in X and Y that CSB c defines. This search was performed by NCBI BLASTn using low parameters (wordsize 7, match 1, mismatch -1, gap cost 1 and 2 for existence and extension respectively) against the NR database. BLAST's results showed 1390 and 1846 highly similar sequences in X and Y respectively. The main feature reported for these subsequences were magnesium ABC transporter ATPase (25% and 3.6%), magnesium translocating P-type ATPase (6.8% and 5.3%), calcium translocating P-type ATPase (2.5% and 3.3%) or APTase (1.6% in X and 6.1% in Y).

A second search was carried out by megablast excluding *Mycoplasma* taxa (wordsize 16, match 1, mismatch -1, gap cost 1 and 2 for existence and extension respectively). BLAST's results showed 11994 and 17126 similar sequences in X and Y respectively (p *-value* up to 3.6). Prevalent species in the results were Plasmodium (6.7% in X and 3.3% in Y), Zebrafish (6.7% and 3.9%) and Vitis (6.5% and 4.4%).

A third search was carried out by using SMA3s [26] to find associations between biological information and CSB C. We performed the search over UniProt Swiss-Prot database from EBI. Both sequences were annotated as GO:0000166, GO:0046872, GO:0016021, P-type (p *-value* threshold 0.1, identity percentage cutoffs 90% and 70%, rost parameter 20).

More details can be found in Supplementary Material, section 5a.

2) Case B: Gecko-CSB Reports Longer Fragments Than Progressivemauve: However, in some cases Gecko-CSB can also identify longer fragments as compared to similar LCBs reported by progressiveMauve. This situation arises because Gecko-CSB is able to merge fragments that fulfil $CompleteLinearity$. progressiveMauve merges LCB only when the objective function (for "greedy breakpoint elimination" heuristic process) improved, which is not the case in this example.

Since Gecko-CSB is designed to detect and identify evolutionary events, it is able to detect potential SBs which can explain big evolutionary events. It means that even if the region is poorly conserved (and that is the reason why other programs cannot identify them) Gecko-CSB is able to find them.

In Fig. 11 we present an example. Gecko-CSB obtains a bigger block (A) than progressiveMauve which obtains 3 fragments (B, C, D) separated by two regions (E and F). Coordinates of blocks and regions are shown in Table IV.

The CSB A covers LCBs B, C, and D, covering as well regions E and F. These regions are poorly conserved (30% and 43% of identity) and that is the main reason why progressiveMauve does not join B, C, and D into a single LCB. A database search using SMA3s was carried

out over the sequences in the region E and F. The results shown that in the region E, both sequences X and Y share the same functionality: DNA restriction-modification system, site-specific DNA-methyltransferase (adenine-specific) activity.

IV. Conclusions

In this paper we introduce a method named Gecko-CSB aimed to the automatic, parameters free and robust CSBs detection. It is able to work in complex environments (e.g., highly-interspersed short repeats, overlapped fragments, small fragments). Gecko-CSB results outperform those of current software tools both in the number and quality of the detected signals that correspond to evolutionary events.

Most methods start with a collection of particular HSPs, in some cases setting up some previous requirements such as non-overlapping HSPs or even more, they start one step forward using a collection of SBs. Gecko-CSB methodology starts with simple HSPs or ungapped fragments, one of the classical outputs from pairwise sequence comparison algorithms.

In this paper, a set of definitions is presented in order to formalize linearity and collinearity properties in a CSB framework. These properties are useful not only to detect CSBs as it is shown in the results section but also to detect and identify evolutionary events. We would like to propose CSBs as a basic unit of large evolutionary events detection and the starting point for breakpoint refinement.

Since a HSP is defined as a maximum ungapped similar segment shared by the sequences under comparison, it is—by definition—computationally hard to extend the orthology beyond their extremities [2]. Some attempts for this extensions are based on the short fragments shared by the sequences in the region between two adjacent SBs [3] or using less strict parameters for the detection of SBs, which can drive to obtain spurious extensions. We propose to use a more robust method based on the information stored in sequence databases, which provide much more information to delineate the borders of SBs. In this scenario, small blocks and IRs that break collinearity are essential to determine the appropriate boundaries where CSBs extension must be analyzed. In general, these small blocks are quite important for the determination of the boundaries of Evolutionary Events, regardless of these blocks having not been taken into account by state-of-the-art software. We are working in this refinement with promising preliminary results.

To validate results, two different applications were used: progressiveMauve and GRIMM-Synteny. Parameters in those applications were set to produce comparable results. In all cases,

- Gecko-CSB obtains more coverage and similar quality.
- Gecko-CSB is designed for dealing with overlapped HSPs, one of the main drawbacks in current software tools. Due to the previous capability, Gecko-CSB can detect and organize IRs.
- Gecko-CSB works in complex environments (i.e., overlapped fragments, small fragments, highly repeats 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 troubling fragments. However, Gecko-CSB is able to incorporate statistically significant short fragments into CSBs or report groups of IRs.
- Gecko-CSB is automatic in the sense that it does not need parameters to detect CSBs or IRs. In our case, all parameters are internally estimated based on distributions. Also we use some formulas to suggest values to be used in the process. Gecko-CSB has also demonstrated to be a robust method able to deal with genomes related at different levels of orthology.

Finally, the results provided by Gecko-CSB represent the starting point for SBs and breakpoints refinement and they are the last step when addressing the correct reversion of evolutionary events in order to produce better inter-genome distances based on the number of rearrangements.

References

[1] P. Pevzner and G. Tesler, "Genome rearrangements in mammalian evolution : Lessons from human and mouse," *Genome Res.*, vol. 13, no. 1, pp. 37–45, 2003.

[2] C. Lemaitre, E. Tannier, C. Gautier, and M.-F. Sagot, "Precise detection of rearrangement breakpoints in mammalian chromosomes," *BMC Bioinformat.*, vol. 9, p. 286, 2008.

[3] C. Baudet, C. Lemaitre, Z. Dias, C. Gautier, E. Tannier, and M. F. Sagot, "Cassis: Detection of genomic rearrangement breakpoints," *Bioinformatics*, vol. 26, no. 15, pp. 1897–1898, 2010.

[4] X. Zeng, M. J. Nesbitt, J. Pei, K. Wang, I. a. Vergara, and N. Chen, "OrthoCluster: A new tool for mining synteny blocks and applications in comparative genomics," in *Proc. 11th Int. Conf. Extending Database Technol. (EDBT '08)*, pp. 656–667.

[5] C. Rödelsperger and C. Dieterich, "CYNTENATOR: Progressive gene order alignment of 17 vertebrate genomes," *PLoS ONE*, vol. 5, no. 1, 2010.

[6] Y. Wang, H. Tang, J. DeBarry, and X. Tan, "MCScanX: A toolkit for detection and evolutionary analysis of gene synteny and collinearity," *Nucleic Acids Res.*, vol. 40, no. 7, p. e49, Apr. 2012.

[7] S. Proost, J. Fostier, D. De Witte, B. Dhoedt, P. Demeester, Y. Van De Peer, and K. Vandepoele, "i-ADHoRe 3.0-fast and sensitive detection of genomic homology in extremely large data sets," *Nucleic Acids Res.*, vol. 40, no. 2, 2012.

[8] S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg, "Versatile and open software for comparing large genomes," *Genome Biol.*, vol. 5, no. 2, p. R12, 2004.

[9] M. Brudno, S. Malde, A. Poliakov, C. B. Do, O. Couronne, I. Dubchak, and S. Batzoglou, "Glocal alignment: Finding rearrangements during alignment," *Bioinformatics*, vol. 19, no. 1, 2003.

[10] A. E. Darling, B. Mau, and N. T. Perna, "Progressivemauve: Multiple genome alignment with gene gain, loss and rearrangement," *PLoS ONE*, vol. 5, no. 6, 2010.

[11] T. C. Chu, T. Liu, D. T. Lee, G. C. Lee, and A. C. C. Shih, "GR-Aligner: An algorithm for aligning pairwise genomic sequences containing rearrangement events," *Bioinformatics*, vol. 25, no. 17, pp. 2188–2193, 2009.

[12] C. G. Ghiurcuta and B. M. E. Moret, "Evaluating synteny for improved comparative studies," *Bioinformatics*, vol. 30, pp. 9–18, 2014.

[13] S. Pham and P. Pevzner, "DRIMM-Synteny: Decomposing genomes into evolutionary conserved segments," *Bioinformatics*, vol. 26, no. 20, pp. 2509–2516, Oct. 2010.

[14] P. SanMiguel, A. Tikhonov, Y. K. Jin, N. Motchoulskaia, D. Zakharov, A. Melake-Berhan, P. S. Springer, K. J. Edwards, M. Lee, Z. Avramova, and J. L. Bennetzen, "Nested retrotransposons in the intergenic regions of the maize genome," *Science*, vol. 274, no. 5288, pp. 765–768, 1996.

[15] C. W. Hill, "Large genomic sequence repetitions in bacteria: Lessons from rRNA operons and Rhs elements," *Res. Microbiol.*, vol. 150, no. 9-10, pp. 665–674, 1999.

[16] W. F. Doolittle and C. Sapienza, "Selfish genes, the phenotype paradigm and genome evolution," *Nature*, vol. 284, no. 5757, pp. 601–603, 1980.

[17] H. H. Kazazian, "Mobile elements: Drivers of genome evolution," *Science*, vol. 303, no. 5664, pp. 1626–1632, 2004.

[18] S. Saha, S. Bridges, Z. V. Magbanua, and D. G. Peterson, "Empirical comparison of ab initio repeat finding programs," *Nucleic Acids Res.*, vol. 36, no. 7, pp. 2284–2294, 2008.
[19] A. L. Price, N. C. Jones, and P. a. Pevzner, "De novo identification of repeat families in large genomes," *Bioinformatics*, vol. 21, no. Suppl. 1, pp. 351–358, 2005.
[20] J. Jurka, V. V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz, "Repbase Update, a database of eukaryotic repetitive elements," *Cytogenet. Genome Res.*, vol. 110, no. 1–4, pp. 462–467, 2005.
[21] Z. Bao and S. R. Eddy, "Automated de novo identification of repeat sequence families in sequenced genomes," *Genome Res.*, vol. 13, no. 1, pp. 1269–1276, 2003.
[22] R. C. Edgar, "PILER-CR: Fast and accurate identification of CRISPR repeats," *BMC Bioinformat.*, vol. 8, p. 18, 2007.
[23] O. Torreno and O. Trelles, "Breaking the computational barriers of pairwise genome comparison," *BMC Bioinformat.*, vol. 16, no. 1, p. 250, 2015.
[24] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, Mar. 1970.
[25] G. Tesler, "GRIMM: Genome rearrangements web server," *Bioinformatics*, vol. 18, no. 3, pp. 492–493, 2002.
[26] A. Muñoz-Mérida, E. Viguera, M. G. Claros, O. Trelles, and A. J. Pérez-Pulido, "Sma3s: A three-step modular annotator for large sequence datasets," *DNA Res.*, vol. 21, no. 4, pp. 341–353, 2014.

The Author(s) *BMC Genomics* 2016, **17**(Suppl 8):804
DOI 10.1186/s12864-016-3069-4

BMC Genomics

RESEARCH Open Access

Refining borders of genome-rearrangements including repetitions

JA Arjona-Medina[1*] and O Trelles[2]

From: 6th SolBio International Conference 2016 (SolBio-IC & W-2016)
Riviera Maya, Mexico. 22-26 April 2016

Abstract

Background: DNA rearrangement events have been widely studied in comparative genomic 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.

Methods and results: 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.

Conclusion: The accurate definition of the borders of SB and repeated genomic 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.

Keywords: Synteny block, Computational synteny block, Breakpoint, Refinement

Background

Large scale genomic rearrangements (LSGR) have been widely studied due to their implication in the evolution of the species. The study of rearrangements is strongly linked with Synteny Blocks (SB) defined as conserved regions between sequences [1]. The regions between SB are called breakpoints (BP), and their study might reveal clues towards evolutionary mechanisms [2, 3]. Both, SB and BP, have been used for phylogeny distance calculation [4], ancestral genome reconstruction [5], and others.

Although there are many methods to identify SBs, they usually do not refine their borders [3, 6, 7]. Those methods that refine SBs -and therefore BP- they usually focus on extending the borders of the SB, aiming to maximize a specific target function based on the alignment. Additionally, the lack of a well-accepted definition of SB [8] might be among the reasons that current tools yield widely different results. Furthermore, the presence of repeated regions or small blocks between the SBs increases the complexity of the detection, one of the main reasons why most methods do not take into account such repetitions. However, these repetitions -mostly associated with mobile elements- have been driving the evolution in many ways [9].

One of the main problems to identify BPs is the unclear definition. For example, Lemaitre et al. [10] reasoned that

*Correspondence: arjona@uma.es
[1]Advanced Computing Technologies Unit, RISC Software GmbH, 4232 Hagenberg, Upper Austria, Austria
Full list of author information is available at the end of the article

a BP is not a single "point" but a region between two SB; while others, for example Chu et al. [11] describe a method to determine the exact location of a BP at nucleotide level for inversions and block interchange events.

A second problem appears when trying to refine the SB by extending its borders. Current methods try to maximize the alignment in the region between two SBs, but boundaries are less conserved. Most of them [12–14], remove the small blocks or repetitions to simplify the SB detection. Clearly the resulting BPs might be contaminated by small subsequences which actually have a homologous region in the other sequence. Any analysis based on these contaminated sequences will be biased by these small subsequences.

In a recent work [15] we addressed the detection of blocks of large rearrangements, called Computational Synteny Blocks, taking into account repetitions. In this manuscript, we propose a method to refine these detected CSBs and detect also BPs taking into account small blocks and any kind of repetitions. Indeed, we use the repetitions alignment to improve the accuracy of the refinement process. In our model, we contemplate inversions, duplications and translocations.

Our results show a higher accuracy in terms of percentage of identity in refined SBs. Our results also indicate biological differences between refined SBs and detected BPs sequences. Sequences in the SBs borders are richer in DNA damage whereas sequences in the detected BPs are richer in DNA replication and stress response, strongly linked to evolution [16].

Methods

Our method starts with the collection of Computational Synteny Blocks (CSB) - similar to SB associated with coding regions, and CSB also covering non-coding regions. The CSBs are calculated using GECKO-CSB [15] (second step in Fig. 1). Applying linearity and collinearity functions (described in [15]) over the CSB provided by GECKO-CSB we identify LSGR (so far duplications, inversions and translocations). The next step — which is reported in this document- is the precise refinement of the borders of CSBs involved in every detected LSGR (third step in Fig. 1). This refinement is applied to the sequences involved in calculation (namely sequences X and Y) in two independent and separable processes. After that we combine the results to get the final refinement. Figure 1 describes the workflow step by step.

Once an LSGR is detected, we take the two CSBs involved. The repetitions in between them, if any, are also take into account. Then we define a region of interest (ROI) running from the tail of one CSB to the head of the other (step 4 in Fig. 1). This ROI includes an arbitrary *offset* to force the overlapping between CSBs and repetitions

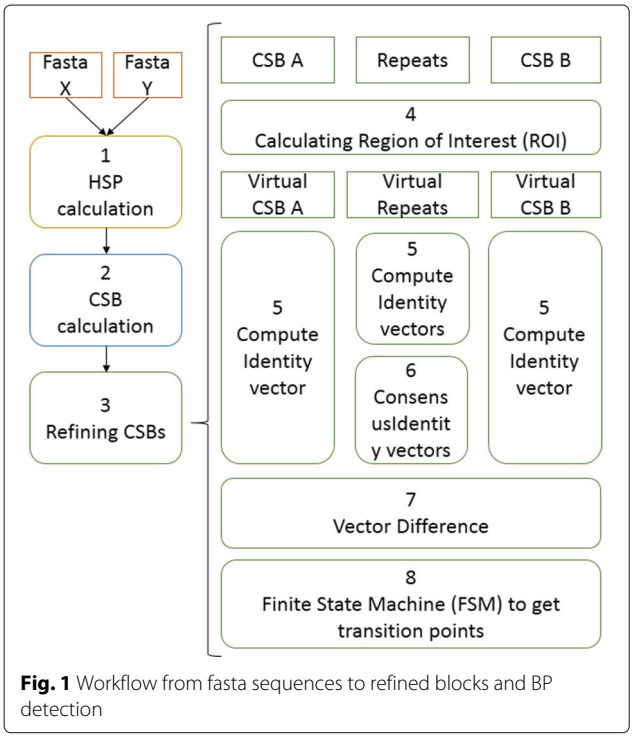


Fig. 1 Workflow from fasta sequences to refined blocks and BP detection

(see Figs. 2 and 12). A virtual CSB (CSB_V) and virtual repetitions are created by extending the borders in order to cover the ROI. Afterwards, these CSB_V and virtual repetitions are aligned using a fast customized implementation of the Needleman and Wunsch [17] global alignment method. The main idea of this process is to force overlapped regions to study the alignments within the ROI.

At this point an identity vector for every aligned CSB_V and all repetitions is computed (step 5 in Fig. 1. See Additional file 1 for more details). Then, a "difference vector" (V_{diff}) is calculated (step 7). If we are working with only two CSB_V, the V_{diff} contains the normalized absolute difference between the two identity vectors. If besides that we are working with repetitions, we compute the V_{diff} taking into account a consensus identity vector from the repetitions (step 6).

The rationale behind the method is the following: The V_{diff} vector contains high values when identity vectors are different. In those regions where values are similar in both identity vectors, the values contained in V_{diff} will be low. At some point we will observe a transition between high and low values along the V_{diff} vector. These transitions will define the BP. A finite-state machine (FSM) was designed to detect these transitions (step 8). At the end of the process, CSB borders are refined based on the BPs detected by the FSM. The method does not force the BP to be a region or a point. This will depend on the transition's features.

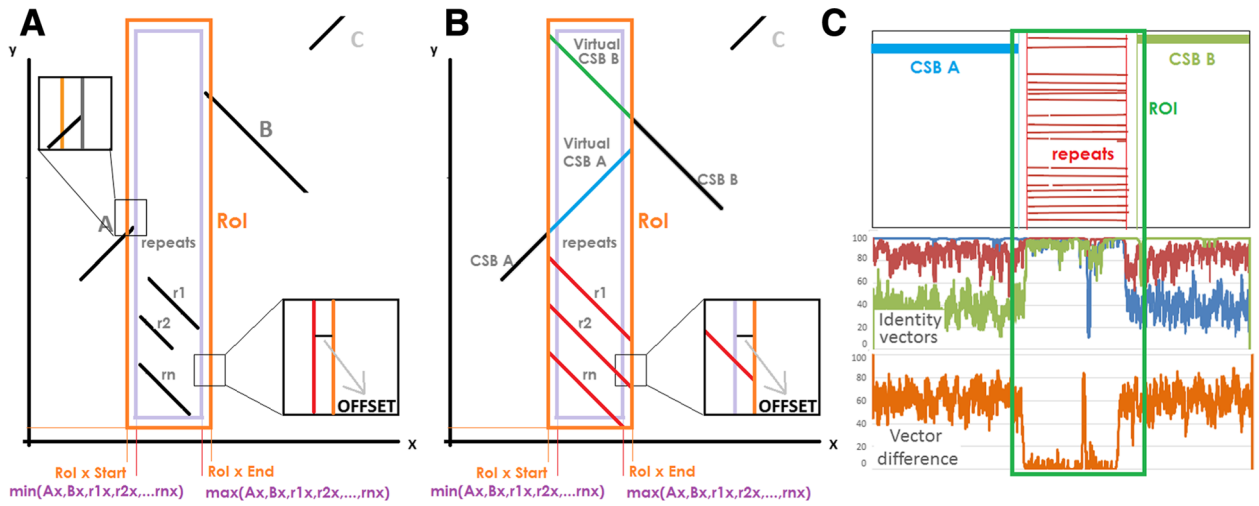


Fig. 2 Illustrative representation of the Region of Interest (ROI). **a** ROI region in an inversion event (CSB B). Coordinates are calculated following the Eqs. 1 (**b**) Virtual CSBs and repetitions. Virtual CSBs are calculated using the Eqs. 2 (**c**) Same representation but including identity vectors and vector difference graphs

Detection of CSBs repetitions and large-scale genomic rearrangements

CSBs and repetitions are detected using Gecko-CSB [15], an extension of Gecko [18] . This software has demonstrated its capacity to yield HSPs of high-quality beating reference software. In [15] we presented a set of formal definitions describing different levels of linearity and collinearity between CSBs. Using these definitions, a set of rules was defined to identify LSGR in single chromosome species, such a inversions, translocations, reverted translocations and duplications. Once a LSGR is detected, we perform our refining method over those CSBs involved in the LSGR.

After the detection of a LSGR two CSBs (namely A and B) are selected. Optionally, if collinearity between CSB A and CSB B is interrupted by a set of repeats, the repeats will be included in the selection as well. Repeats can be separated in two groups. Those repeats whose coordinates in the sequence X overlap with CSBs A and B are grouped in a collection named repeats-X. In the collection repeats-Y are the equivalents regarding sequence Y.

Refining CSBs

At this point the method splits in two branches. The refinement in the sequence X and Y are complementary and independent. In this document we will describe the refinement for the sequence X branch. The sequence Y branch is the same, but interchanging X by Y.

Calculating the region of interest

The CSBs and repeats define a ROI (see Eq. 5, Figs. 2 and 12). Since our method is focused on finding transitions between CSBs and repetitions, we introduce an *offset* parameter, which ensures overlapping between the end of CSBs and the beginning of virtual CSBs and the virtual repetitions, guaranteeing that transitions are present. In the worst case, the method will have *offset* number of nucleotides in both CSBs that share similarity and therefore, they can be aligned with a high value of identity. In other words, the *offset* parameter stabilises the beginning and the end of the signal (More details in "FSM thresholds selection" in the Additional file 1). The ROI is defined as follows:

$$\begin{aligned} ROI_{xStart} &= min(A_{xEnd}, B_{xStart}, Repeats_{xStart}) - offset \\ ROI_{xEnd} &= max(A_{xEnd}, B_{xStart}, Repeats_{xEnd}) + offset \\ ROI_{yStart} &= min(A_{yEnd}, B_{yStart}, Repeats._{yStart}) - offset \\ ROI_{yEnd} &= max(A_{yEnd}, B_{yEnd}, Repeats_{yEnd}) + offset \end{aligned} \quad (1)$$

After calculating the ROI, new CSBs named virtual CSBs (CSB_V) are created using the ROI X_{Start} and X_{End} coordinates. This means that all CSB_Vs will start and end at the same point. In this step we are extending or trimming the old CSBs concerning ROI start and end points. New CSB_Vs' Y coordinates will be calculated depending on how much we have trimmed or extended the coordinates in X regarding the old CSB. The equations that describe this process are the following:

$$\begin{aligned} CSB_{VxStart} &= ROI_{xStart} \\ CSB_{VxEnd} &= ROI_{xEnd} \\ \alpha_L &= CSB_{xStart} - CSB_{VxStart} \\ \alpha_R &= CSB_{VxEnd} - CSB_{VxEnd} \\ CSB_{VyStart} &= CSB_{yStart} - \alpha_L \\ CSB_{VyEnd} &= CSB_{yEnd} + \alpha_R \end{aligned} \quad (2)$$

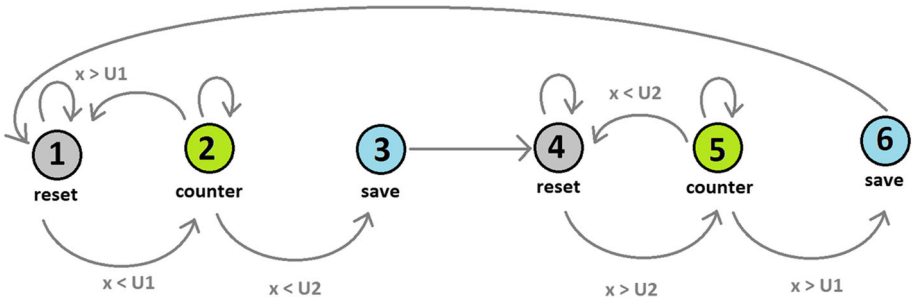


Fig. 3 Finite State Machine to detect transitions. This FSM has six states. The first three states (*1–3*) are to calculate the BP's start coordinate, and the last three states (*4–6*) to calculate the BP's end coordinate. Changes from one state to another will depend on vector difference values (x in the figure) and thresholds U1 and U2

Notice that α takes negative values when trimming and positive when extending. New CSB_Vs are aligned using a Needledman and Wunsch implementation.

Calculating identity vectors

After the alignment of CSB_Vs, identity vectors (I_V) are created for every CSB_V. All I_Vs have the same length and they represent the percentage of identity that a certain region of length W has in the alignment. We take a window of length W to calculate that percentage of identity.

First we create a binary vector (V_B) which represents matches in the alignment. V_B has the length of the alignment. Since V_B takes into account gaps, its length can be different from one CSB_V to another. By using a window of length W, we can compute the percentage of identity at any point in V_B. As long as we are going to compare I_V from different CSB_Vs, identity values from those points in the alignment that represent a gap in sequence X are not stored. This way, all identity vectors from different CSB_Vs will have the same length, ROI_{length}.

Low values in parameter W produce a noisy identity vector corresponding with high frequency changes of identity. On the contrary, high values in parameter W smooth the noise and produce a low frequency signal. The selection of a proper W value is not possible as it might change depending on the CSB_V involved. We could also

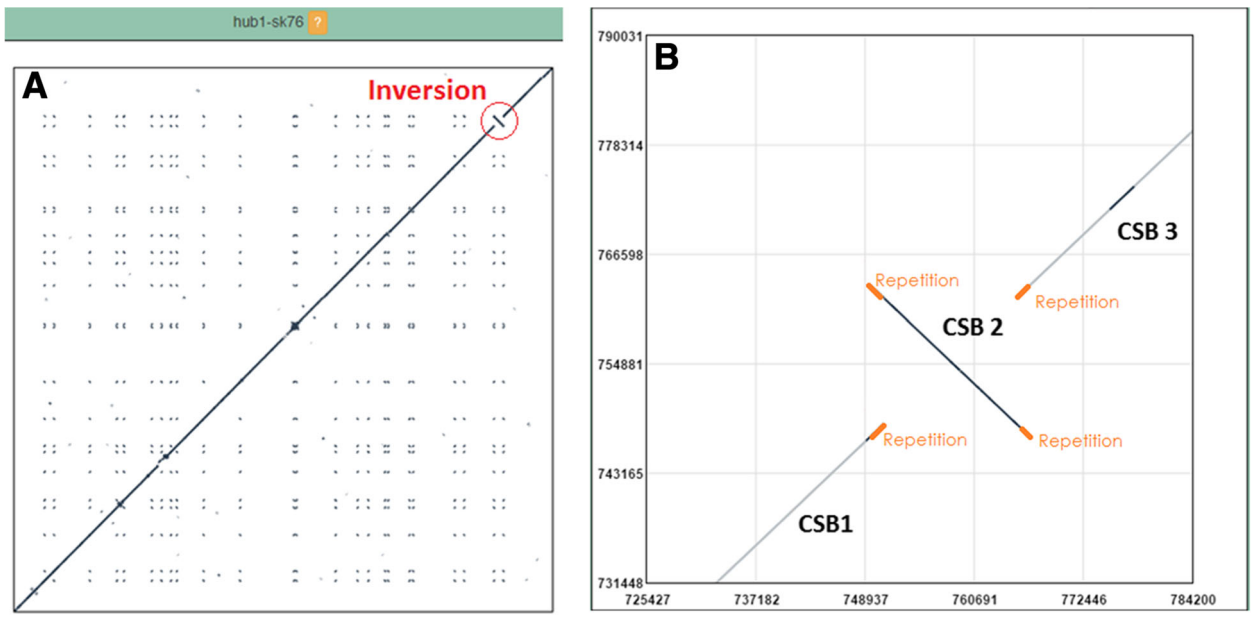


Fig. 4 a Full comparison of HUB-1 against SK76. Main diagonal represents that both subspecies are quite similar. Small points represent repetitions, with a notorious one (an inversion) present upper zone of the image (*circle*) (**b**) Zoomed display of the marked region in 4a. Three CSBs are going to be extended in this example. Repetitions are represented in a different colour

Table 1 CSB Coordinates before and after the refinement process

	X_{Start}	Y_{Start}	X_{End}	Y_{End}	Str	Length	% ΔX	% ΔY	% ident
CSB_1									
Before	711,591	710,528	747,965	746,902	f	36,379			99.69 %
After	711,591	710,528	748,001	746,940	f	36,413	0.1 %	0.1 %	99.72 %
CSB_2									
Before	749,573	761,860	762,895	748,534	r	13,348			99.37 %
After	749,564	761,853	762,933	748,505	r	13,349	0.35 %	0.17 %	99.59 %
CSB_3									
Before	764,581	763,521	780,474	779,414	f	15,895			99.70 %
After	764,494	763,439	780.474	779.414	f	15,976	0.55 %	0.52 %	99.69 %

be interested on changes that happen at different frequencies. Therefore, instead of choosing a fixed W value, which would mean changes at only one frequency, we build a vector containing all frequencies as follows:

$$I_V(x) = \sum_{i=0}^{N} A_i I_i(x) \tag{3}$$

where A_i is the weight of the identity vector at a certain frequency

$$\sum_{i=0}^{N} A_i = 1 \tag{4}$$

And the Identity vector at a certain frequency is calculated as follows:

$$I_i(x) = \frac{1}{2N+1} \sum_{j=x-N}^{x+N} V_B(j) \tag{5}$$

In this model, N defines the maximum window to compute the percentage of identity and also defines the start and end positions where the values of the vector can be used. From 0 to $2N + 1$ and from $2N + 1 - ROI_{length}$ to ROI_{length} the I_V is uncompleted. Therefore, N cannot be as long as we want. It should be at least lesser than *OFFSET*. In practice we have observed that a value of 50 is enough to get good results.

Finally, since identity vectors are going to be compared, they must to be normalized.

Calculating consensus identity vector

In the case that a group of repetitions are detected, we use the information of the consensus sequence to improve accuracy of the refinement method.

After repeats have been aligned and the V_Bs have been computed, a Sum Match Vector (V_{SM}) is calculated by adding all V_Bs vectors. This vector has a length of ROI_{length}, so only positions which are not representing a gap are taken into account -as we did in the previous section. Then, we calculate the percentage of repeats that cover one specific position in the V_{SM}. To calculate the Consensus Identity Vector (V_{CI}), only positions that comply with a given threshold are set to 1, and 0 otherwise. In this implementation the threshold was set to 25 %. This new vector is named Consensus Binary Vector. After this process, we calculate the V_{CI} by processing the Consensus Binary Vector as we already described in the previous section.

Vector difference

In order to detect transitions which delimitate the BP, we compute the absolute difference between the CSB_Vs identity vector. CSB_Vs are extracted from CSBs according to the *ROI*, using the *OFFSET* to ensure that similar regions are represented in CSB_Vs. As a result, the identity vectors for the CSB_V-A have a high value at the

Table 2 Repeated region coordinates

ID	Sequence	Start	End	Length	Description	Enzyme
1.x	*hyorhinis HUB-1*	748,012	749,513	1,501	tnp	Transposase
2.x	*hyorhinis HUB-1*	762,953	764,494	1,541	insK	Integrase core domain protein
1.y	*hyorhinis SK76*	746,988	748,493	1,505	tra	Transposase for insertion sequence element IS6290
2.y	*hyorhinis SK76*	761,936	763,425	1,489	insK	Mobile element protein

Table 3 Breakpoint coordinates

ID	Sequence	Ref seq	Start	End	Length
1.1a	M. *hyorhinis* HUB-1	NC-014448.1	748,001	748,012	11
1.2a	M. *hyorhinis* HUB-1	NC-014448.1	749,513	749,564	51
2.1a	M. *hyorhinis* SK76	NC-019552.1	746,940	746,988	48
2.2a	M. *hyorhinis* SK76	NC-019552.1	748,493	748,505	12
3.1a	M. *hyorhinis* HUB-1	NC-014448.1	762,933	762,953	20
3.2a	M. *hyorhinis* HUB-1	NC-014448.1	764,494	764,539	45
4.1a	M. *hyorhinis* SK76	NC-019552.1	761,853	761,936	83
4.2a	M. *hyorhinis* SK76	NC-019552.1	763,425	763,439	14

beginning and low value at the end. On the contrary, the identity vectors for the CSB_V-B have a low value at the beginning and high value at the end. This is the reason why the vector difference will start and end with high values. If repetitions are detected, then the difference vector will have high values in the middle as well.

Anyways, transitions will be found in between these high values (see Fig. 2).

Detecting transition points

To detect transitions a Finite-State Machine (*FSM*) was designed. Figure 3 shows the design. Basically, the *FSM* detects the coordinates where the vector difference value was the last time at a certain threshold ($U1$) before reaching the second threshold ($U2$). As a result, the selected region defined by the coordinates is the transition between high and low values along the vector difference.

We associate these transitions as a candidate for a BP. After this process, the refined *SB* can be trimmed or extended. The threshold selection is discussed in the next section.

Results

Simple case

We will use a simple case to illustrate the algorithm behaviour in the SB borders-refinement method using M. *hyorhinis* HUB-1 (Accession code NC-014448.1) and M. *hyorhinis* SK76 (Accession code NC-019552.1) genome sequences with a length of 839,615 bp and 836,897 bp, respectively.

Figure 4a shows the full comparison of HUB-1 against SK76. Figure 4b shows a particular area where a LSGR (an inversion) is presented, before the refinement.

Table 1 shows the coordinates of the CSBs involved in the inversion before and after the refinement process, where X represents M. *hyorhinis* HUB-1 and Y correspond to M. *hyorhinis* SK76. Str. column represents the strand of the Y sequence, forward or reverse. The percentage of extension in X and Y sequence is shown in ΔX and ΔY columns.

The percentage of identities has increased a bit due to the extension (the refined CSBs are a bit longer). Notice that in CSB_2 the refine process has extend the Y_{Start} coordinate making the CSBs 7 nucleotides shorter. On the other hand, in the opposite border (y_{End}) it has extended 29 nucleotides.

Four regions have been detected as repeated sequences. A database search (Uniprot bacteria at ftp://ftp.ebi.ac.uk) using SMA3s [19] was carried out. Results and sequence features are shown in Table 2.

And the BPs are shown in the Table 3.

In this case the method has found 8 BPs. Due to repetitions that the method detects between two CSBs, two BPs are detected in each sequence. For each BP found, we have performed a database search using Uniprot and NCBI non-redundant with no results. No annotation was found.

Comparing with CASSIS software

We have processed the CSBs detected by GECKO-CSB using CASSIS [12] in order to refine them. Since CASSIS cannot handle repetitions and following the recommendations from its article, we have masked all the repetitions in both sequences using RepeatMasker [20] (search Engine was *abblast*) and we did not include the repetitions in the input file. Data set and results can be found in the Additional file 1.

Results from CASSIS are widely different than those obtained by our method because, among other reasons, they do not take into account repetitions. Our method

Table 4 CASSIS software breakpoint coordinates

ID	Sequence	Start	End	length	Descript.	Enzyme
1b	M. *hyorhinis* HUB-1	747,965	749,573	1,608	tnp	Integrase core domain protein
2b		762,895	764,581	1,686	insK	Integrase core domain protein
3b	M. *hyorhinis* SK76	710,797	797,477	86,680	polC	DNA polymerase III PolC-type
4b		712,895	797,477	84,582	nanE	ManNAc-6-P epimerase

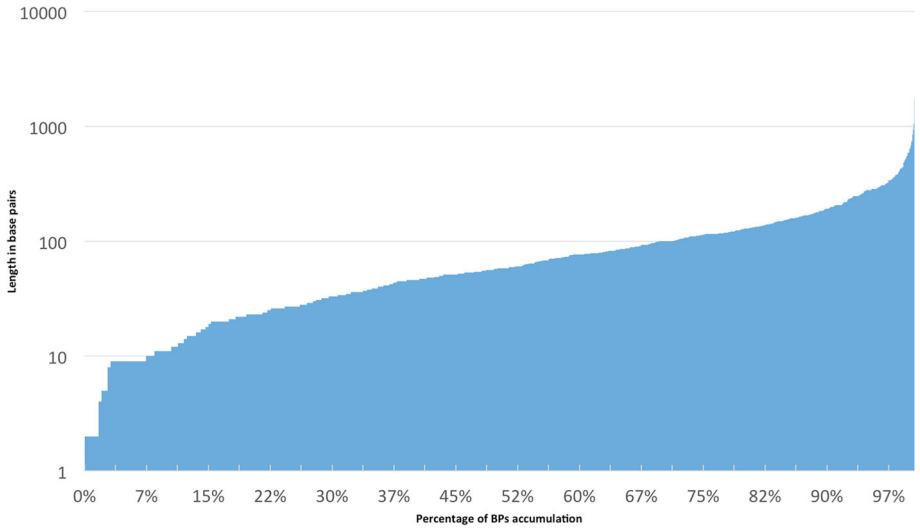


Fig. 5 Progressive distribution of Breakpoint length (bps)

detects 2 short BPs where CASSIS detects a big one. Indeed, BP 3b and 4b (SK76 sequence) cover the region contained by CSBs 1, 2 and 3. This result is incomprehensible because it implies that the SBs desapear, creating a huge BP of size around 85 Kbps, instead of these 3 SBs.

BP 1b has a length of 1,608 bps. We have performed a BLAST [21] search using the sequence of BP 1b with default parameters. The sequence has been found several times in different sub species of *hyorhinis* with high values of identity and coverage, which point-out that this sequence is a part of a conserved repetition (see BLAST Report-BreakPoint-1b in Additional file 1). An additionally BLAST search was carried out using sequences from BP 2a with similar results.

We have performed a database search using SMA3s over the BP detected by CASSIS. Results are shown in Table 4 (description and enzyme columns).

Testing the method with a 68 mycoplasmas dataset

For the next test, a collection of 68 Mycoplasmas was used. This test was performed with the aim to avoid bias in the analysis that a selection of two particular genomes could introduce. The genome collection and their gene bank annotations are available at http://bitlab-

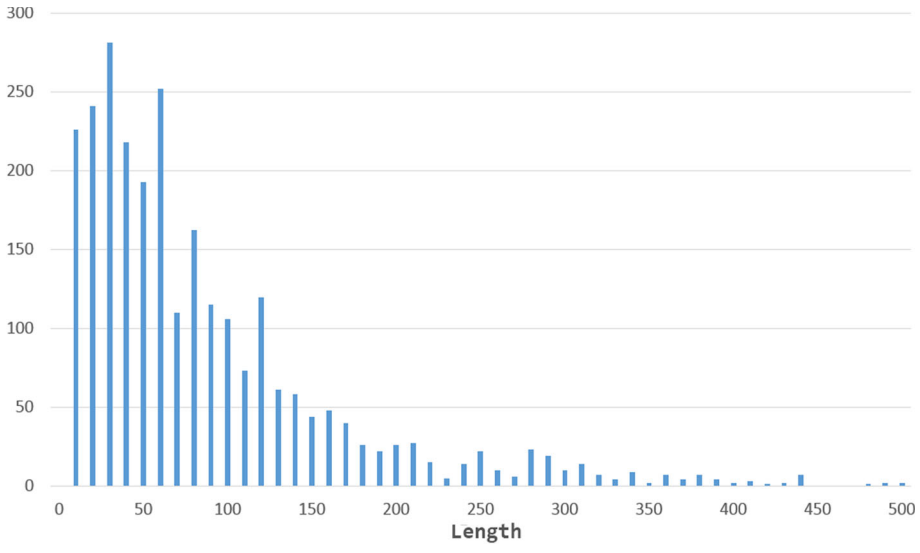


Fig. 6 Frequency distribution of Breakpoint length

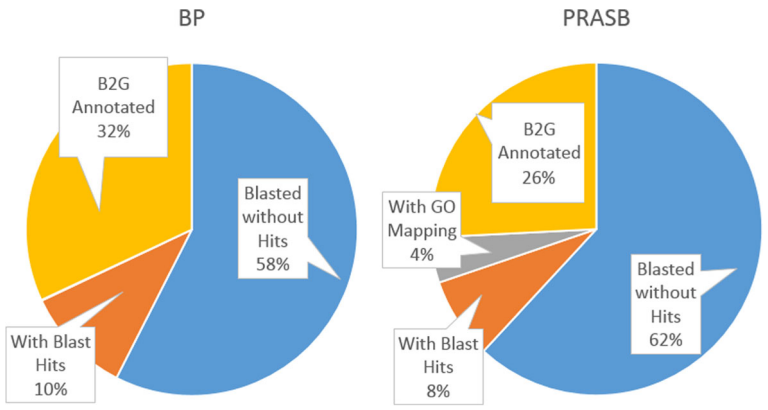


Fig. 7 Results of blast search, mapping and annotation process with blast2GO for BP and PRASB sequences

es.com/gecko/. For the biological analysis we have performed SMA3s [19] over the sequences to find annotations using the Uniprot bacteria database (ftp://ftp.ebi.ac.uk). Additionally blast2GO [22] was used to carry out a second annotation process using blastx and the non-redundant protein database filtered by Bacteria taxa.

We run first GECKO [18] over the resulting 2,278 comparisons following by GECKO-CSB [15]. After that, the refinement process was carried out giving the refined collection of CSBs as a result.

Our method refined 2,213 CSBs, 829 were trimmed after the refining process and 1,384 were extended. Around 70 % of the BPs detected are sized below 100 bps, 95 % below 300 bps (see Fig. 5). The BP detection was limited in the implementation at a size of 5000 bps to avoid spurious long BPs. As it can be observed in Fig. 6, the frequency of the length tends to zero at length of around 400 bps.

To analyse the results from a biological point of view, BPs sequences were extracted. The sequences of the proportional region of the adjacent Synteny Block (PRASB) of each BP were also extracted according with the BP length (the length of the PRASB sequence has the same length of the BP sequence, see Fig. 12). The purpose was to find biological differences by comparing results from annotations in BP and PRASB sequences. The sequences were compared against the NCBI non-redundant protein database, filtered by Bacteria taxa. After that, the sequences were mapped and annotated using blast2GO [22].

The number of sequences with annotation was higher in BPs (32 %) than in PRASBs (26 %). For more details, see Fig. 7. We also analysed the percentage of annotations by level of coverage that cover the CSBs in the comparison from which the BPs were detected. We found that at a lower level of coverage (meaning non related species), more sequences were annotated, especially in BPs sequences (27 % vs 17 %, see Fig. 8).

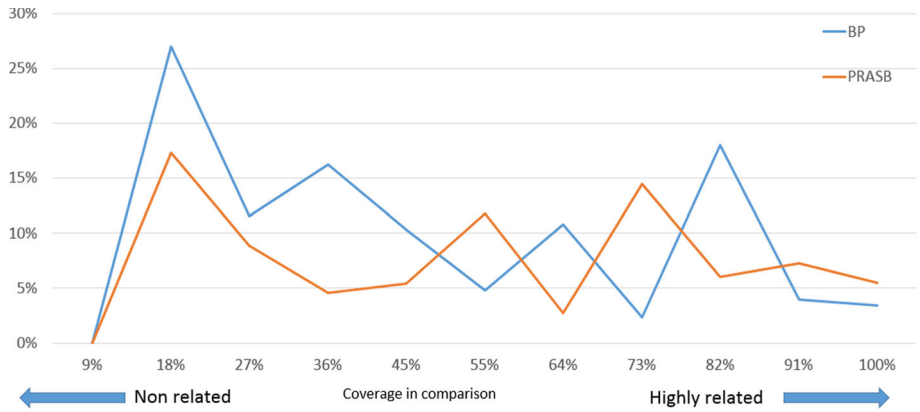


Fig. 8 Percentage of annotated sequences in BP and PRASB by percentage of coverage in the comparisons in which the sequences were extracted from

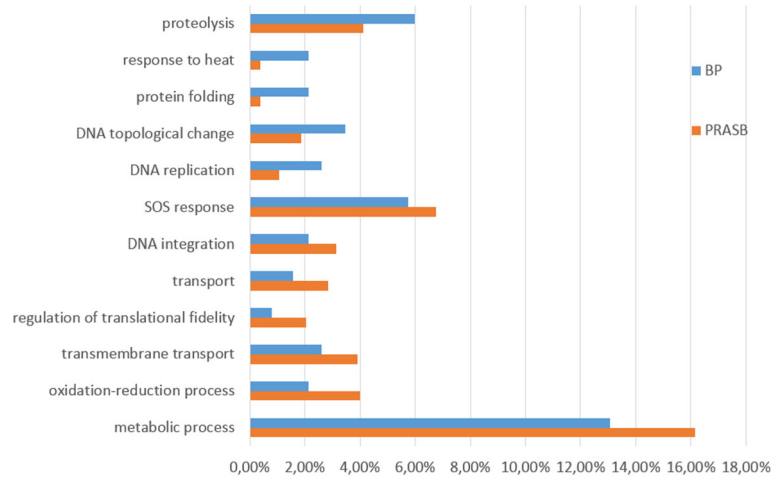


Fig. 9 Results from Blast2GO for biological process in annotations of BP and PRASB sequences. Percentages are over the total amount of annotations

Regarding the content of the annotation, we found several differences in the biological process and molecular function categories. Figure 9 shows a summary of the biological process category with the most significant differences between BPs sequences and PRASBs sequences. SOS response, DNA integration or metabolic process were more present in PRASB sequences. Proteolysis, response to heat, protein folding, DNA topological change and DNA replication were found in more proportion in BP sequences. Full reports are available as Additional file.

We also performed another database search, which was carried out using SMA3s [19] against the UNIPROT database. The results showed strong differences between annotations in BPs and PRASBs sequences. Figure 10 shows the UNIPROT keyword categories for Biological process. Stress response and DNA replication are more present in BP sequences. On the other hand, Glycolysis, Calvin cycle and DNA damage are significantly more present in the PRASB than in BP sequences.

Figure 11 shows the UNIPROT pathways. Carbohydrate degradation is by far more represented in PRASB sequences and Purine metabolism is more present in BP sequences. Full reports are available as Additional file.

The method we present in this manuscript detects two BPs when refining SBs, one at each border (tail or head) of the SB, instead of considering the whole region between these SBs as one large BP. Therefore, after the refining process we have two BPs and one region in between (gap), as it can be observed in the Fig. 12. The sequences corresponding with this region in between the BPs were extracted to be analysed.

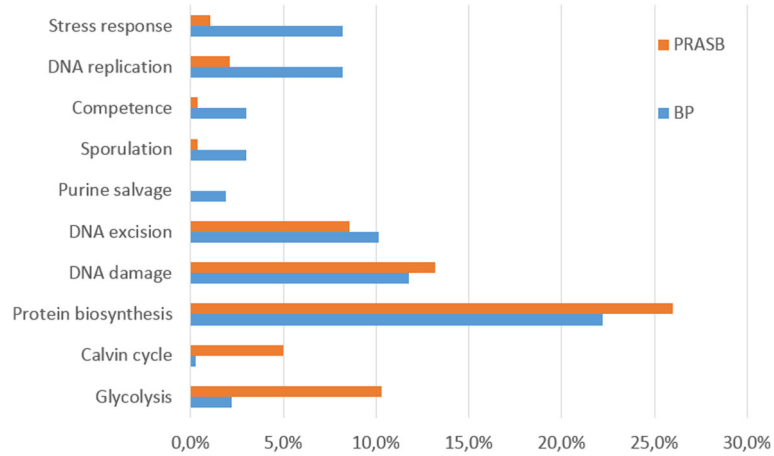


Fig. 10 Results of Uniprot keyword categories for biological process in annotations of BP and PRASB sequences. Percentages are over the total amount of annotations

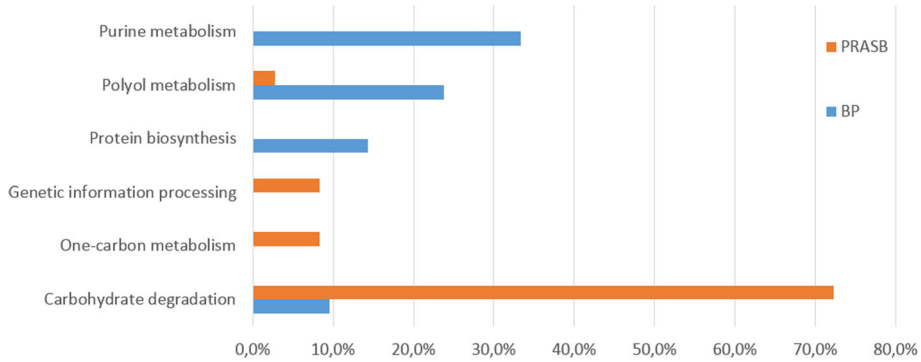


Fig. 11 Results of Uniprot pathways in annotations of BP and PRASB sequences. Percentages are over the total amount of annotations

Around 30 % of the gap regions in between two breakpoints are shorter than 100 bps of length, 88 % below 1,000 bps.

In order to analyse biological differences between BPs and the gap between two BPs once SBs borders have been refined, we have extracted the sequences corresponding with the gap regions between BPs.

A SMA3s search was carried out over BPs sequences and the gap sequences using the Uniprot database. The main difference according with these results is at the biological process (Fig. 13). DNA replication, Stress response and Purine salvage were found more often in the gap whereas transport, DNA damage and DNA excision were more present in the BP sequences.

Discussion

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. Many studies support that LSGR do not happen randomly but follow an unknown model. Some regions of the sequence seem to be more fragile or predispose to suffer a large-scale LSGR [2]. Indeed these BPs can be reused [3, 23] and the BP reuse rate is strongly linked with the resolution in which SB are detected [24].

Therefore, if a BP depends 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 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 means that the BP region will be a region without

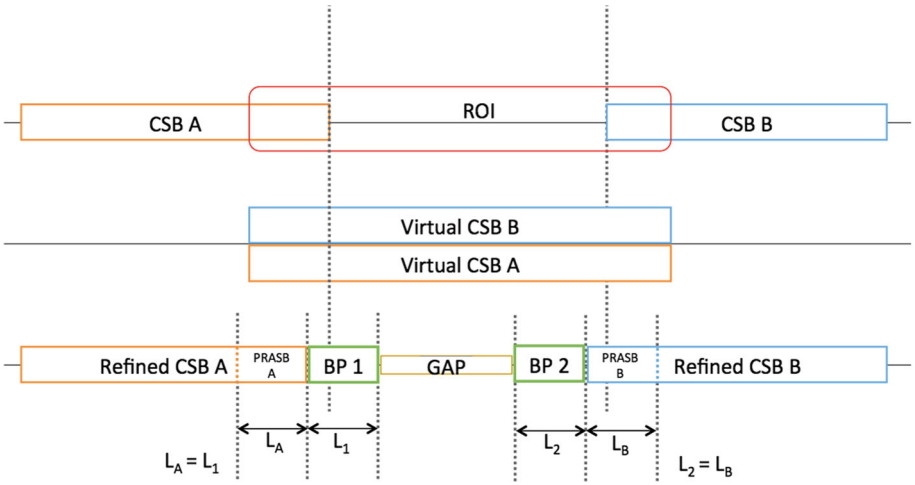


Fig. 12 CSBs before and after the refinement. At the end of the refinement process, we detect BPs. We also extract PRASB and GAP sequences to analyse accuracy of the method. PRASB and BP have the same length

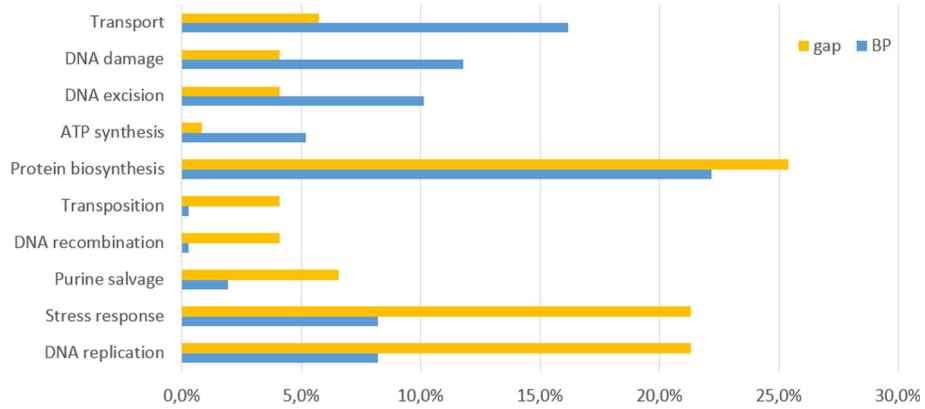


Fig. 13 Results of UniProt keyword categories for biological process in annotations of BP and gap sequences. Percentages are over the total amount of annotations

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.

Threshold selection in the finite state machine

Our method bases the BP detection on transition points in the differences of the percentage of identities. We have analysed the behaviour of the identity vector along SBs. We have found that coding regions and non-coding regions have different levels of identity, which can be explained because of different evolutionary level of pressure. But we also have found that in many cases there is a perceptible transition that could be detectable using a FSM (see Fig. 14). We think that something similar might happen between SBs and BPs, a detectable transition that could determine the BP region.

To identify these transitions we have designed a FSM which uses two thresholds. In the current version of the implementation of the method, which we have described in this document, thresholds are set to 80 and 20 respectively. The selection of the parameter values was made empirically. (see "FSM thresholds selection" in the Additional file 1 for more details).

We analysed the identity percentage of SBs and BPs at different length and have found a strong correlation between SB and BP levels of identity percentage (see Fig. 15). In general BPs have less identity percentage than SB.

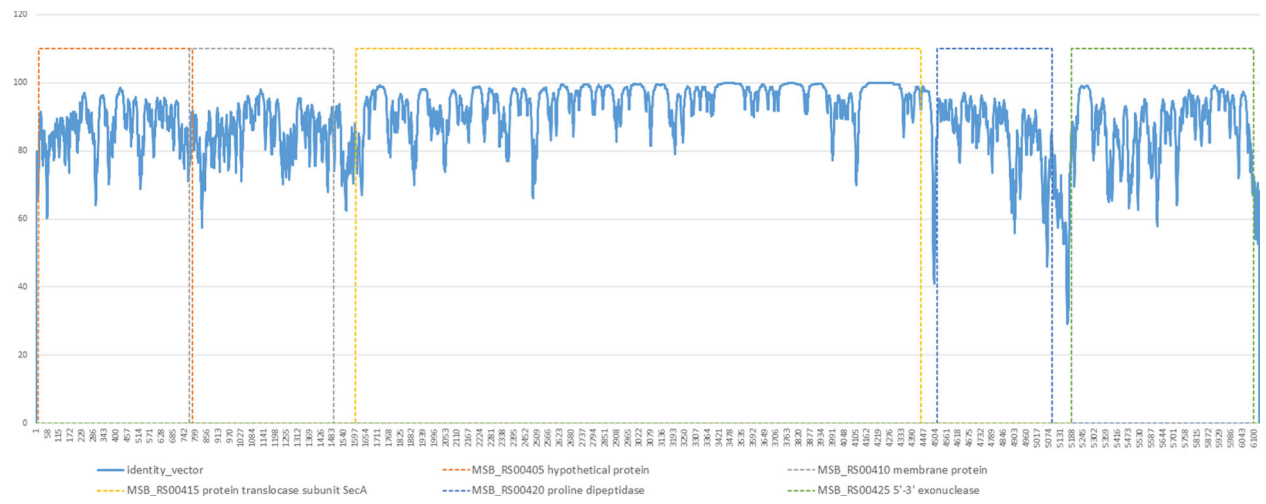


Fig. 14 Real case of SB identity vector. In dotted lines codding regions for sequence X. SB extracted from NC-014751.1 (sequence X) vs NC-015431.1 (Sequence Y) comparison. X_{Start}: 92,877, Y_{Start}: 115,660, X_{End}: 98,983, Y_{End}: 121,755

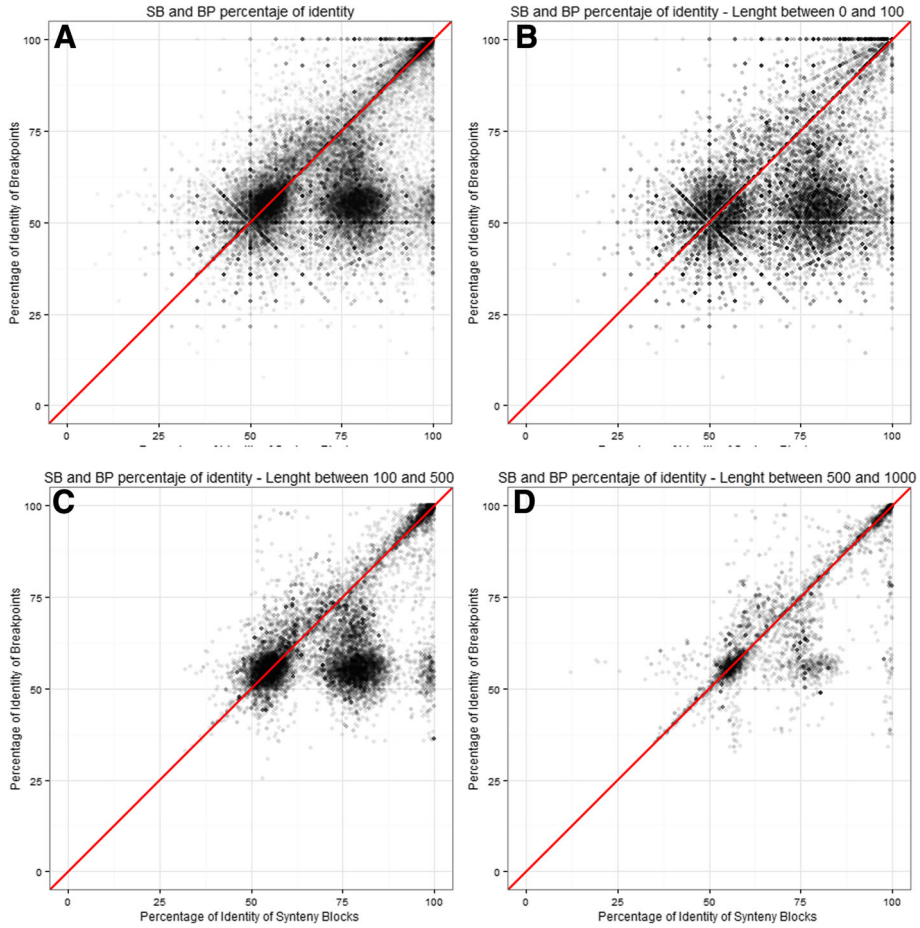


Fig. 15 Percentages of identities in SB (axis x) and BP (axis y) regions. **a** all pairs of SBs and BPs. **b** only BPs with length between 0 and 100. **c** length between 100 and 500. **d** length between 500 and 1,000

Conclusions

We have developed a method to refine the borders of CSBs taking into account repetitions and using them to improve the accuracy of the refinement. The method is not based on maximizing any target function, but studies the alignments to refine and uses a finite-state machine to find transition points in the alignment. These transition points set an accurate refinement of the involved blocks. Due to the method's features, BPs are detected as regions or as points, depending on the specific case. It also takes into account the repeated regions, so between two CSBs it can give 4 breakpoints, 2 for each sequence, demarcating start/end of one block and end/start of the region in between.

Several analysis were carried out in order to find biological differences between BPs, SBs borders and gap regions.

The results showed that there are biological differences between BPs sequences and the PRASB sequences. BPs sequences are biologically richer than PRASB. Both searches using Uniprot and NCBI databases gave more results in BPs sequences than the PRASB sequences. However, PRASB showed more diversity in annotations than those obtained for BPs.

Our experiments show that there may to 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.

We have also found that there are differences between what we consider as BPs and the region in between the BPs, whereas other methods just consider the whole region as BP.

Our method needs two thresholds to detect the transition points in the difference vector in which the BP is defined. Thresholds pick up the abrupt changes in the signal. These thresholds are fixed in this version of the method, however, we will work on a dynamic configuration of the threshold based on SB similarity that might produce more accurate results.

Additional file

Additional file 1: Supplementary material. (PDF 2796 kb)

Acknowledgements
Oscar Torreño Tirado, Michael Krieger and Paul Heinzlreiter for their valuable comments and review.

Declarations
This article has been published as part of *BMC Genomics* Volume 17 Supplement 8: Selected articles from the Sixth International Conference of the Iberoamerican Society for Bioinformatics on Bioinformatics and Computational Biology for Innovative Genomics. The full contents of the supplement are available online at https://bmcgenomics.biomedcentral.com/articles/supplements/volume-17-supplement-8.

Funding
This work has been partially supported by ISCIII, projects: PT13.001.012 and RD12.013.006) and the EU Mr.SBM project, code 324554. The latter EU project funds the publication fees of this work.

Availability of data and materials
The datasets, software, additional files and results can be found at http://www.bitlab-es.com/gecko at subsection GECKO-CSB. The EBI Uniprot Bacteria database used in this work is available at ftp://ftp.ebi.ac.uk.

Authors' contributions
JAM contributed in the conception of the work and software development. OT supervised the work and provided ideas. All authors contributed to the manuscript's preparation. Both authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Author details
[1]Advanced Computing Technologies Unit, RISC Software GmbH, 4232 Hagenberg, Upper Austria, Austria. [2]Department of Computer Architecture, University of Malaga, Campus de Teatinos, 29071 Malaga, Spain.

Published: 25 October 2016

References

1. Nadeau JH, Taylor Ba. Lengths of chromosomal segments conserved since divergence of man and mouse. Proc Natl Acad Sci U S A. 1984;81(February):814–8. doi:10.1073/pnas.81.3.814.
2. Bailey Ja, Baertsch R, Kent WJ, Haussler D, Eichler EE. Hotspots of mammalian chromosomal evolution. Genome Biol. 2004;5(4):23. doi:10.1186/gb-2004-5-4-r23.
3. Pevzner P, Tesler G. Genome Rearrangements in Mammalian Evolution: Lessons From Human and Mouse. Genome Res. 2003;13(1):37–45. doi:10.1101/gr.757503.
4. Blanchette M, Bourque G, Sankoff D. Breakpoint Phylogenies. Genome Inform Ser Workshop Genome Inform. 1997;8:25–34.
5. Alekseyev M, Pevzner P. Breakpoint graphs and ancestral genome reconstructions. Genome Res. 2009;2000:943–57. doi:10.1101/gr.082784.108.2.
6. Darling AE, Mau B, Perna NT. Progressivemauve: Multiple genome alignment with gene gain, loss and rearrangement. PLoS ONE. 2010;5(6). doi:10.1371/journal.pone.0011147.
7. Donthu R, Lewin HA, Larkin DM. SyntenyTracker: a tool for defining homologous synteny blocks using radiation hybrid maps and whole-genome sequence. BMC Res Notes. 2009;2(1):148. doi:10.1186/1756-0500-2-148.
8. Ghiurcuta CG, Moret BME. Evaluating synteny for improved comparative studies. Bioinformatics. 2014;30:9–18. doi:10.1093/bioinformatics/btu259.
9. Kazazian HH. Mobile elements: drivers of genome evolution. Science (New York). 2004;303(5664):1626–1632. doi:10.1126/science.1089670.
10. Lemaitre C, Tannier E, Gautier C, Sagot MF. Precise detection of rearrangement breakpoints in mammalian chromosomes. BMC Bioinforma. 2008;9:286. doi:10.1186/1471-2105-9-286.
11. Chu TC, Liu T, Lee DT, Lee GC, Shih ACC. GR-Aligner: An algorithm for aligning pairwise genomic sequences containing rearrangement events. Bioinformatics. 2009;25(17):2188–193. doi:10.1093/bioinformatics/btp372.
12. Baudet C, Lemaitre C, Dias Z, Gautier C, Tannier E, Sagot MF. Cassis: Detection of genomic rearrangement breakpoints. Bioinformatics. 2010;26(15):1897–1898. doi:10.1093/bioinformatics/btq301.
13. Brudno M, Malde S, Poliakov A, Do CB, Couronne O, Dubchak I, Batzoglou S. Glocal alignment: Finding rearrangements during alignment. Bioinformatics. 2003;19(1). doi:10.1093/bioinformatics/btg1005.
14. Darling AE, Mau B, Blattner FR, Perna NT. GRIL: Genome rearrangement and inversion locator. Bioinformatics. 2004;20(1):122–4. doi:10.1093/bioinformatics/btg378.
15. Arjona-Medina JA, Trelles O. Computational Synteny Block: A Framework to Identify Evolutionary Events. IEEE Trans NanoBioscience. 2016;15(4):1–11. doi:10.1109/TNB.2016.2554150.
16. Rocha EPC, Matic I, Taddei F. Over-representation of repeats in stress response genes: a strategy to increase versatility under stressful conditions? Nucleic Acids Res. 2002;30(9):1886–94.
17. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol. 1970;48(3):443–53.
18. Torreno O, Trelles O. Breaking the computational barriers of pairwise genome comparison. BMC Bioinforma. 2015;16(1):250. doi:10.1186/s12859-015-0679-9.
19. Muñoz-Mérida A, Viguera E, Claros MG, Trelles O, Pérez-Pulido AJ. 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); 2014. p. 1–13. doi:10.1093/dnares/dsu001.
20. Smith CD, Edgar RC, Yandell MD, Smith DR, Celniker SE, Myers EW, Karpen GH. Improved repeat identification and masking in Dipterans. Gene. 2007;389(1):1–9. doi:10.1016/j.gene.2006.09.011.
21. Bedell J, Korf I, Yandell M. Blast. Bioinformatics; 2003, p. 312.
22. Conesa A, Götz S, García-Gómez JM, Terol J, Talón M, Robles M. Blast2GO: A universal tool for annotation, visualization and analysis in functional genomics research. Bioinformatics. 2005;21(18):3674–676. doi:10.1093/bioinformatics/bti610.
23. Sankoff D, Trinh P. Chromosomal Breakpoint Reuse in Genome Sequence Rearrangement. J Comput Biol. 2005;12(6):812–21. doi:10.1089/cmb.2005.12.812.
24. Attie O, Darling AE, Yancopoulos S. The rise and fall of breakpoint reuse depending on genome resolution. BMC Bioinforma. 2011;12(Suppl 9):1. doi:10.1186/1471-2105-12-S9-S1.

The Author(s) *BMC Genomics* 2016, **17**(Suppl 8):802
DOI 10.1186/s12864-016-3063-x

BMC Genomics

RESEARCH Open Access

Computational workflow for the fine-grained analysis of metagenomic samples

Esteban Pérez-Wohlfeil[1], Jose A. Arjona-Medina[2], Oscar Torreno[1], Eugenia Ulzurrun[1] and Oswaldo Trelles[1*]

From 6th SolBio International Conference 2016 (SolBio-IC&W-2016)
Riviera Maya, Mexico. 22-26 April 2016

Abstract

Background: 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.

Results: 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.

Conclusions: 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.

Keywords: Metagenome analysis, Differential abundance, Annotational mapping, Mapping over specific regions, Open platform

Background

The purpose of metagenomics is to identify the species present in an environment. Different types of studies can be performed based on metagenomics. Some examples include the analysis of changes in the presence of species in a given environmental sample and the use of phylogenetic analysis to follow up the spread or determine the origin of a species. A large number of tools are emerging in the form of stand-alone programs (e.g. MEGAN [1]), interoperable Web services (e.g. MG-RAST [2]) or tools accessible through the Internet (e.g. EBI Metagenomics [3]).

MEGAN performs taxonomic analyses of a metagenome by mapping reads to different taxa based on BLAST [4] search results and the NCBI taxonomy. To perform this task, the program runs the lowest common ancestor (LCA) algorithm to classify input reads. Most metagenomic tools are constructed following a workflow scheme offering distinct stages of data processing. In this line, the open-source EBI Metagenomic workflow is split into two branches following the quality control step. The first branch performs taxonomic classification

*Correspondence: ots@ac.uma.es
[1]Department of Computer Architecture, University of Málaga, Boulevard Louis Pasteur 35, Málaga, Spain
Full list of author information is available at the end of the article

based on 16S rRNA, whereas the second branch performs functional analysis based on protein-coding sequences. Unannotated reads are kept out of the pipeline. However, these reads should be taken into account for whole metagenomic analysis in order to improve the accuracy of taxonomic classification and better understand the roles of species in environmental samples.

The number of comparative metagenomic tools is the key point of the metagenomic RAST (MG-RAST) platform. MG-RAST builds clusters of proteins at a given percentage of identity level using QIIME [5]. Once built, the longest sequence of each cluster is subject to similarity using sBLAT, an implementation of the BLAT [6] algorithm. MG-RAST also uses the NCBI taxonomy for taxonomic classification. Functional profiles are available through comparison against data sources that provide hierarchical information. Abundance profiles are the main output for displaying information on datasets. The MG-RAST annotation pipeline does not generally provide a single annotation for each submitted fragment of DNA. Steps in the pipeline map a read to multiple annotations and vice versa. Data privacy is one of the concerns of the scientists using this tool. Firstly, they are reluctant to upload their unpublished and/or confidential data to a public website. Secondly, the priority of analysis requests to the website is subject to the level of confidentiality of input data (with lower priority and therefore longer waiting times for private data).

Recently, a new DNA sequence analysis workflow called META-pipe [7] has been developed to find novel commercially exploitable enzymes from marine microbial communities. META-pipe uses tools such as MetaGeneAnnotator (MGA) [8] and Protein BLAST to identify sequences found in the UniProtKB database. MGA is a new version of MetaGene [9] where a prophage gene model is offered in addition to bacterial and archaeal models. MGA uses di-codon frequencies estimated by the GC content of an input sequence to map genes using regression models. In addition, MGA offers an approach for the analysis of ribosomal binding sites (RBSs) to detect specific patterns of ribosomal sequences in species. However, due to their tendency to undergo highly degenerative changes, RBSs are particularly difficult to identify [10]. In the line of pipelines used to facilitate the comparative analysis of high throughput sequencing, MOCAT [11] is a modular tool for processing raw sequence reads produced by the Illumina technology [12]. The main steps in MOCAT are 1) read trimming and filtering 2) read assembly, 3) gene prediction and 4) estimation of taxonomic abundance profiles.

The fine-grained metagenomic analysis workflow developed by our group can operate over a user-defined collection of genomes -thus accelerating the computational process- and with the advantages of being able to map reads over unannotated regions of genomes. Our software provides different mapping methods and different mapping alternatives apart from the best read-genome matching. In addition, it provides information about the quality of mapping and about differences between mapping options. Unlike the methods currently available, which deduce that a species is not present in a sample when its abundance is low (in number of reads), the proposed method can detect low-abundance species by finding reads mapping to particular specific regions of genomes. In addition, the developed workflow is an open platform composed of an expandable set of separate modules that use well-defined format datafiles. This enables the easy on-demand incorporation of new processing tools. Along with low-abundance species support, other tools have been included to verify the correctness of taxonomic assignation and extrapolate DNA sequencing data to gene expression levels.

System and methods

System and requirements

The designed workflow (see Fig. 1) includes all the needed steps for data processing. The quality control step can be performed using SeqTrimNext [13] for the case of 454-pyrosequencing [14] reads, whereas Trimmomatic [15] can be used for Illumina reads. These programs are available in our workflow implementation under our Galaxy [16] instance. With regards to the sequence comparison algorithm, we suggest to use the GECKO [17] package to accelerate the process. Since several metagenomic packages for 454-pyrosequencing reads are based on the matches provided by a BLAST run, the developed workflow offers a parser to translate BLAST's output, and therefore the same strategy (parsing) can be used when other sequence comparison software is employed. In addition, the sequence comparison tool SHRiMP [18] is included along with a parser that is also available and described in the Additional file 1.

In the line to offer a broad scope of the presented software, the proposed workflow can handle sequences of different length obtained with different sequencing technologies (e.g. SOLiD [19], Illumina, 454-Pyrosequencing). For instance, in the case of colour-space reads these can be compared using SHRiMP, which natively supports colour-space reads. The proposed method focuses in the comparison and mapping procedures, while the pre-processing steps can be carried out with common publicly available software.

The workflow operates over a user-defined collection of genomes. This database might as well be a custom selection of genomes which hold particular interest, a preselection of the most common species for the type of metagenome analysis, or even a complete database such as GenBank [20].

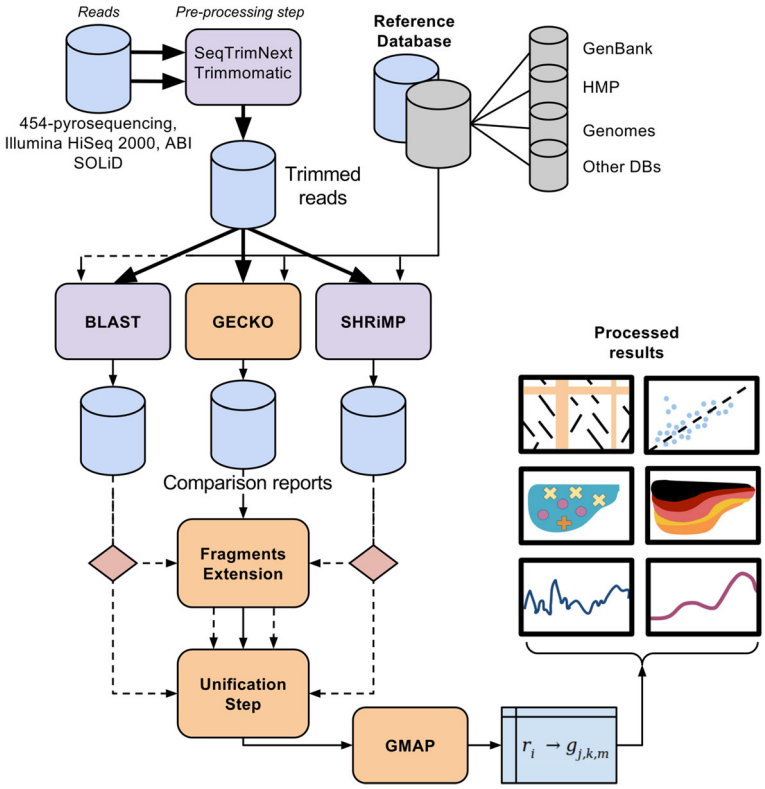


Fig. 1 The Workflow diagram. *Top*: Quality control layer and input files. *Center*: Comparison software layer. *Bottom*: Mapping kernel (GMAP), which provides open-source datafile definitions and enables many on-demand post-processing experiments (*Right*)

The workflow is specialized in matching (reads-species mapping) and post-processing procedures, which require the following input: (1) Sequence comparison files, (2) taxonomic description of the reference dataset, and (3) annotation files for the genomes (optional, only needed for post-processing).

(1) Sequence comparison files: the workflow has been designed so that it is compatible with any sequence comparison software (i.e. BLAST family, FASTA family, proprietary software, etc). The default comparison software used in this workflows is GECKO, however, the user can employ other packages. To include other comparison software a format conversion program would be needed. A parsing conversion system for BLAST is already included in the workflow. The parsing module converts sequence comparison files to a format composed of headers (read-genome tuples) followed by rows, where each row represents a fragment for the tuple. Fragments belonging to a read-genome match are defined by a 12-tuple:

$$t^{12}_{n,k} = (k, score, identities, length, similarity, igaps, egaps, strand, rStart, rEnd, gStart, gEnd)$$

Where k is the $k-th$ fragment reported by the read n (see Additional file 1 for further reading). The fields $rStart, rEnd, gStart, gEnd$ represent the anchoring positions in the read r and genome g. Reversed fragments are found by comparing the read with the reverse complement of genome g. Notice that *rEnd* and *gEnd* are redundant for ungapped fragments, but necessary for gapped fragments.

(2) A taxonomic description file allows the customization of hierarchical relationships between organisms in the reference database as assigning strain relationships between species or separating strains that belong to a common ancestor. Such file can be generated automatically using a module of the workflow and/or can be manually built to insert customized relationships between species. The format of the file generated is a text file including a 5-tuple per line, each tuple being a new genome:

$$t^{5}_{n,m} = (n, m, genome\ accession\ number, genome\ name, length)$$

Where n and m are the specie and subspecie id's. These can be used to set up custom boundaries. For further details, please see "Taxonomy files" in the Additional file 1).

(3) Annotation files are used to carry out all coding region-related computations in the post-processing phase.

Therefore, these files are optional and should be included if annotation post-processing modules are to be used. As in the case of comparison software, a parsing system is implemented; e.g. a parsing system for GenBank's annotation files has been included in our workflow.

Extension of ungapped HSPs

Comparative analysis of metagenomes is an expensive computational process that involves comparing a large set of DNA fragments against an enormous database of candidate sequences (genes, proteins or genomes). It should be noticed that, by definition, bacteria in the metagenome are uncultured species and the sequences in the databases that already exist are not –most likely– the corresponding to the species in the metagenome. Even some large mutations (inversions, deletions, etc) can happen regularly. Therefore a more flexible matching is proposed, which differs from an assembly by mapping in which there are quite close representatives of the sequenced bacteria.

Thereby we included the option of using a custom glocal [21] alignment, which yields longer fragments and larger evolutionary gaps. This method generally improves mapping results, as global alignment methods are less accurate when identifying species.

Once local alignments are calculated (using GECKO, BLAST or any other similar program), fragments are extended by joining those that are close enough according to a given maximum-gap parameter. This is done by calculating the Needleman-Wunsch matrix between the start and the end of the matching read within the genome region with a customized implementation.

Furthermore, glocal alignment can be performed by combining the local alignments produced by alignment tools such as BLAST or GECKO with the provided custom glocal alignment. All parameters can be user-defined, thus providing data processing flexibility. Table 1 shows an example of candidate fragments that are extended to conform a glocal alignment.

New score and expected value calculation

The extension of fragments requires the re-calculation of fragment scores to identify the best match out of a list of candidates during the mapping process. The properties of the extended fragment, namely length –of bases–, number of identities and inserted gaps stand for the raw score. The raw score has to be normalized in order to obtain the expected value of a reported fragment. This is performed using K and Lambda parameters using a similar approach to that of the BLAST family. K and Lambda parameter are calculated as described in [22].

To compute the raw score of the extended fragment produced by our custom glocal alignment we apply a traditional affine scoring model (with open and extension gap penalties), as shown in the following formula:

$$RS = I * M_r + (L - (G_i + G_e) - I) * M_p + G_i * P_i + G_e * P_e \quad (1)$$

Where RS stands for "Raw Score", I for the total number of identities in the fragment, M_r for the match score, L for the total length of the fragment in base pairs, G_i for the total number of open gaps in the fragment, G_e for the total number of extension gaps in the fragment, M_p for the mismatch penalty, P_i for the penalty of an open gap and P_e for the penalty of an extension gap.

Mapping

The mapping module (GMAP) process offers a three-level mapping option that not only discovers highly abundant species that hide others in terms of abundance due to high similarity or uncertainty in the alignment, but to also obtain quality distance measurements between the best 3 candidates for every match. The top three candidates are selected based on identity and coverage thresholds and expected values. Moreover, users can perform different mappings by restraining subsets of reads using different thresholds.

Table 1 Before-and-after extension example of a read with two candidate fragments to be joined

Before extension of local or ungapped alignments											
029701.102903 — NC_004663.1 — Bacteroides thetaiotaomicron 897405											
N	SCORE	IDEN	LEN	SIM	IGAPS	EGAPS	STRA	R1	R2	G1	G2
1	-	134	145	84	0	0	Plus\Plus	1	133	1631420	1631563
2	-	94	104	80	0	0	Plus\Plus	147	249	1631564	1631666
After extension ("glocal-like" alignment)											
029701.102903 — NC_004663.1 — Bacteroides thetaiotaomicron 897405											
1	-	224	248	90	1	2	Plus\Plus	2	249	1631420	1631665

In the top, the table before extension. Fragments (1) and (2) are separated by a relatively small gap of 14 base pairs (The ending read coordinate R2 of (1) is 14 base pairs away from R1 in (2)). These fragments represent an example of candidate fragments. The after subtable (bottom) displays the resulting extended fragment and shows a longer alignment with still high similarity and a low number of gaps (one opening gap and two extension gaps). The score is calculated afterwards (See "New score and expected value calculation").

In a scenario with a highly abundant organism, further analysis can be performed by only considering certain genomes using certain options e.g. to observe differences and extract statistical indicators of close candidates.

About the mapping decision

Every read yields a list of reported fragments to which the following algorithm is applied.

1. Filtering step: A filtering step allows the researcher to consider only a subset of reported fragments, enabling a levelling up mapping method. If a fragment does not reach pre-filtering thresholds, it will be discarded. Such filtering allows a two phases pre-filtering:
    (a) Coverage threshold phase: The length of the match divided by the length of the read.
    (b) Identity threshold phase: The number of identities in the match divided by the length of the match.
2. Repeat this step for 3-option mapping and, if fragments are still active: Select the fragment with the smallest expected value and if it is lower than the maximum allowed expected value. This fragment is included in the mapping file as first, second or third candidate depending on the number of options chosen and the genome is inactivated for the next option iteration.
3. If no more fragments are still active or none of them exceeds the thresholds, the read is decided with either no mapping option or up to 3 mapping options.

See "Mapping decision and fragments" in the Additional file 1 for more information.

Results and discussion

Rather than developing a monolithic application with graphical interfaces, we opted for a simple pipelining procedure in which new software modules can be used to exploit results. To facilitate user interaction, a complete web-based interface has been developed based on Galaxy workflow manager, which enables users to easily run their analyses in both a local instance or in dedicated servers. In addition, a User Guide [23] is available. Regarding software modules, all specifications about input and result file formats are shown, facilitating the use of third-party software, such as common graphical libraries and spreadsheets.

The results given by our workflow software are illustrated by an experiment where two collections of 6 metagenome samples each were extracted from faecal microbial communities of adult female monozygotic and dizygotic twin pairs concordant for leanness or obesity and their mothers [24]. Raw data (i.e. .sff files) were obtained by 454-pyrosequencing, and inherent artefacts or low-quality sequences were further filtered and removed using Replicates [25] software and SeqTrimNext (See "Filtering and trimming parameters" in the Additional file 1 for used parameters). The average size of the read collection ranged from 172 bp to 237 bp after quality control and sequence trimming. The total number of sequences was 2,724,867 for lean metagenomes and 2,972,697 for obese ones. For testing purposes, in this technically-oriented paper we opted to design a synthetic case-control study of two metagenomes by joining samples from lean and obese individuals.

Reads-abundance and taxonomy classification of reads

The analysis of the species present in metagenomic samples enables taxonomic classification based on abundance of mapped reads. Information about the species present in metagenomes and variations across a collection of species is yielded by GMAP in Comma Separated Value (csv) format files that can be edited using common spreadsheet software (see Fig. 2a).

Abundance data are primarily used to determine the species that are present in a metagenomic sample and can be exploited in comparative studies on the over- or under-abundance of species in different samples. However, abundance data does not provide information on the quality and certainty of mapping. This lack of reliability can be partially compensated by using the n-mapping method.

Three-options mapping analysis Our software has the ability to perform the mapping of reads through a multiple-level strategy. After the best read-genome mapping value is used, the used fragment is inactivated and the genomes belonging to different strains of the same species are optionally inactivated, and the process is repeated. This way, we get the second, third and subsequent best read-genome mapping values. A long separation between the mapping options provides stronger evidence supporting the validation of the mapping procedure.

When comparing 3-mapping options, the detection of peaks in second or third options means that a particular species is repeatedly the second or third candidate (see Fig. 2b). These peaks suggest that strong similarities exist between a specific pair of species and careful examination is required since the accuracy of mapping is not certain. For instance, it would be interesting to study if the alphabetical order of the BLAST output for sequences with the same expected value is affecting the mapping. These observations can be supported by the analysis of mapping precision (see Fig. 2c), which considers the closest reads given a distance parameter and shows the separation in mapping length, the number of identities, or any

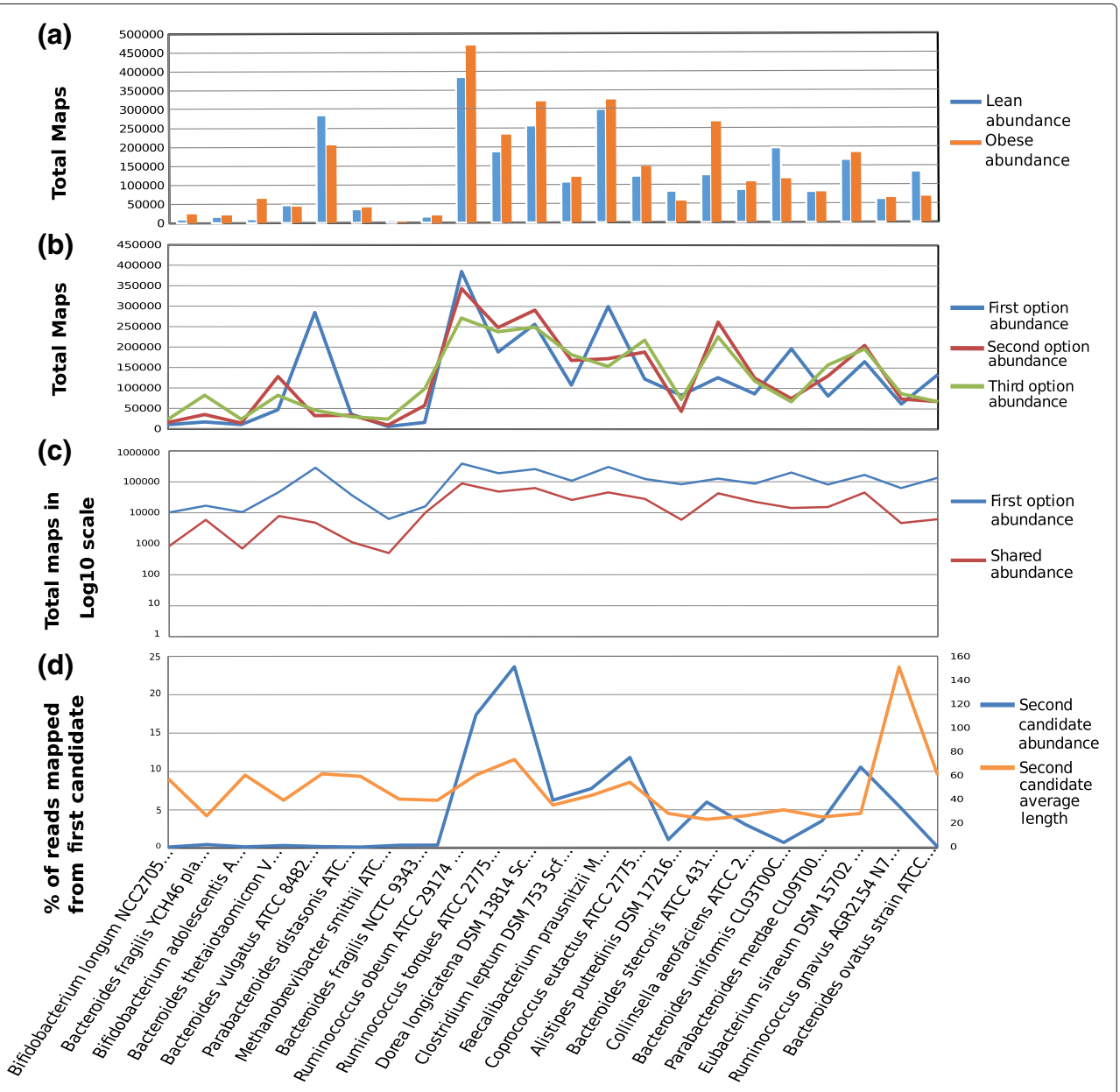


Fig. 2 Three-options mapping analysis. Some data from GMAP-based mapping analysis. **a** Abundance plot for the averaged Lean (*blue*) and Obese (*orange*) metagenomes of the most read-abundance genomes. The plot depicts total mapped reads per specie in the two averaged metagenomes. **b** Three-option abundance by organism. In blue, total first option abundance, (number of reads assigned). In red and green, the number of times an organism was the second and third best candidate for a read. Bacteria with red or green peaks reveal that another organism is probably hiding them (regarding abundance) and there is not a direct consensus. **c** Total reads assigned in log10 scale per species as best candidate (*first option, blue*) and from that total, the number of reads that had two very similar candidates (defined as a distance in terms of identity, length and coverage) from the second best candidate (*in red*). **d** An exhaustive-one-vs-all user-defined analysis where a bacterium is compared against all species in the database. The peak in the plot (near the middle) is the analyzed genome, Ruminococcus obeum ATCC 29,174. This particular scenario depicts a comparison of the target genome against all species by length and abundance. In blue, the percentage of reads that were mapped as second candidate when the best candidate was the target genome. In *orange*, the average length of such mapped reads

other chosen parameter between the assigned read and its second best candidate. Additionally, this separation shows the extent of differences between first and second candidates, and therefore is another indicator of the quality of mapping.

In addition, the 3-mapping approach allows to assess the mapping certainty at both reads and species level; at read level by comparing fragments quality indicators of particular genomes against the rest (see Fig. 2d), and at species level by comparing the abundance levels of the different

options for the particular genomes. For example, for all reads mapped over a given genome, information about the identity and coverage level of the second and third mapping option would provide information about the quality or certainty of the first option.

On the other hand, in a joint analysis of the Fig. 2b and c, no peaks in second or third options, along with a larger separation gap on mapping precision analysis suggest that the accuracy of mapping is high, which reduces the random assignment of reads to genomes and, therefore, the results obtained are more reliable.

Figure 2c displays the number of reads assigned to each species and, in relation to each assignation, the times the second option was almost as good as the first (namely, "shared" reads). The fact that the blue and red lines of two species are close to each other suggests that mapping is not accurate and careful examination is required.

Fine-grained tuning and closer examination In a scenario where a specific species has been the second option a higher number of times compared to the first option, as discussed in the previous Section, the mapping should be exhaustively analyzed and compared with other species. Such analysis would provide more certainty of the presence of a low-abundance genome by checking the properties of its matches, and would enable contrasting the variances in the matches between a high-abundance genome and its best second option. Moreover, it is possible to perform a one vs. one, one vs. some, or, some vs. some comparative analysis of the target species. This type of analysis can be performed based on any of the properties of the mapped reads, such as length, similarity, coverage, or any user-defined properties. This information is particularly useful when the first and second mapping options identify different species (in some cases, remotely-related species).

Figure 2d illustrates how a number of reads map to very similar sequence regions shared by different species (due to high similarity at genome level –i.e. conserved genes). For example, the mentioned Figure displays the second mapping option of the reads that were mapped as first solution to the Ruminococcus obeum ATCC 29,174. The blue peak in the middle of the plot stands for almost 25 % of the reads assigned to Ruminococcus as first option and to Dorea longicatena DSM 13,814 as second option, which evidences strong similarities in several areas of the two sequences. Additionally, the orange peak at the right side suggests a longer alignment in the second option –Ruminococcus gnavus AGR2154–, thus requiring in-depth analysis of such reads.

Statistical significance of variations between samples The presented software can provide statistical data on a number of aspects or characteristics, such as the Z-score test to detect significant variations in the abundance of species in different experimental conditions; or to contrast the significance of the variation at a species level between samples calculating the p-values. An interesting example is case-control studies in which differences in reads abundance along genomes can be identified. Z-scores provide accurate information on the significance of such differences (see "Statistical Significance" in the Additional file 1 for more information).

Genome-specific experiments and quality assessment

Reads mapping to specific regions of genomes Besides the proximity measures provided by three-option mapping, there is another important aspect concerning the provision of evidence about the presence of species with low-abundance of reads in the metagenome. The main idea is to find regions in a particular organism that do not exist or do not share similarity at all with other organisms present in the collection of genomes. To accomplish this, $N - 1$ comparisons between the reference genome and the N genomes contained in the collection are performed using GECKO. This process yields the detected regions and the assigned reads that have been mapped to these regions.

The extracted reads mapped to these regions provide strong evidence on the presence of low-abundance species in the metagenomic sample, since the mapped read does not fit over other genomes (see "Reads mapping to specific regions of genomes" in Additional file 1 for more information).

Differential abundance in annotated regions of genomes Another useful tool is the comparison plot of abundance of annotated regions (potential coding regions that could change abundance values in different samples). This assay is conducted on a particular genome by only considering the reads mapped to annotated regions of the genome and comparing abundance between different samples in the same way as RNA-seq transcriptome expression analysis is performed. Differences in the abundance of reads mapped to annotated regions –when sampling genomic DNA– might be related to environmental changes. This hypothesis is based on the experimental resemblance of the differential expression plot of annotated regions when two samples whose environmental conditions change are compared. Figure 3a suggests that some annotated regions are being over- or under-represented, thus suggesting that abundance in annotated regions may be related to variations in the samples.

Genome profiling of mapped reads A genomic profile of mapped reads is the accumulated number of reads mapping to a given position within the genome. Accumulated histograms of abundance of reads provide

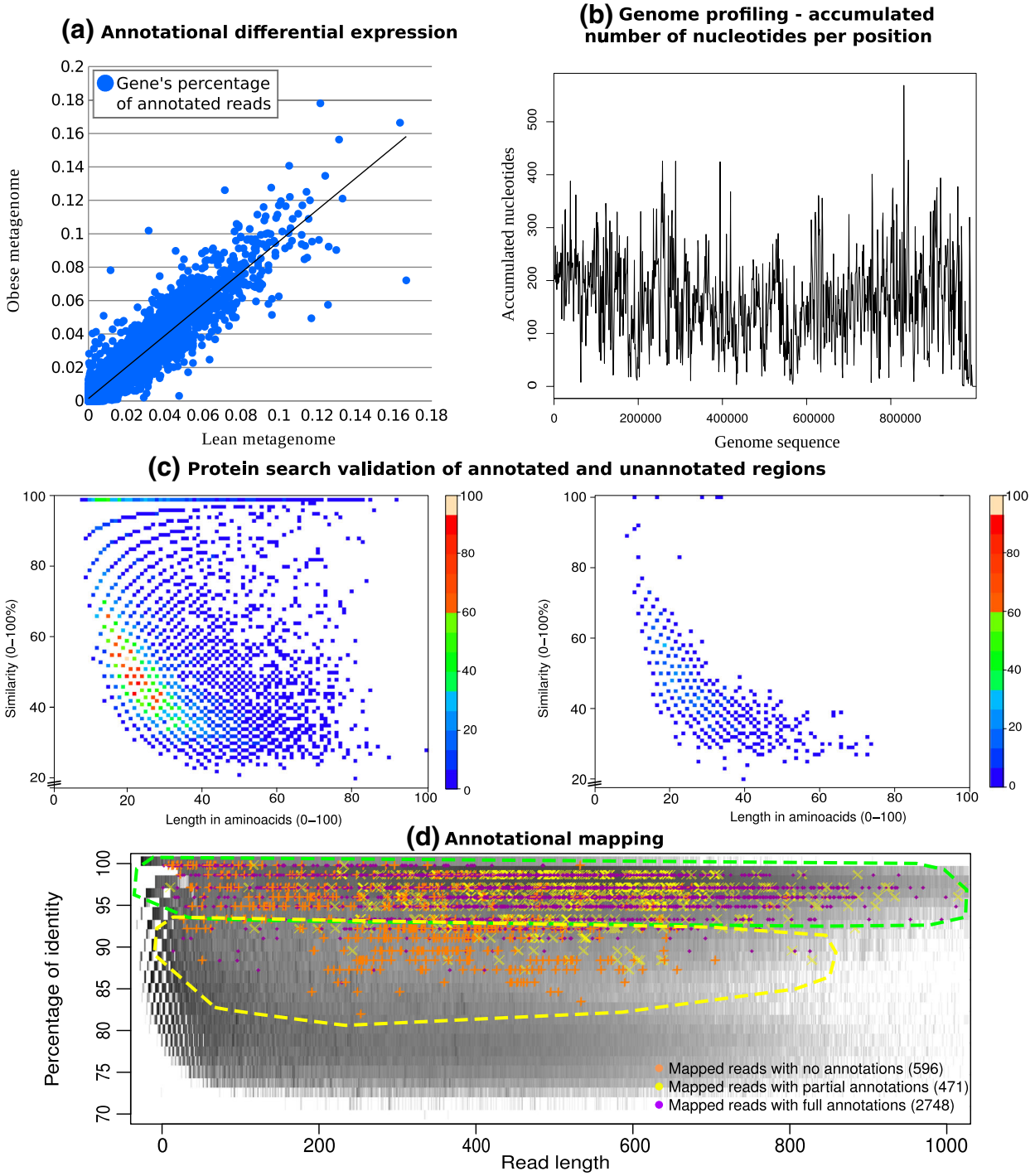


Fig. 3 Genome-specific experiments. Some of the results oriented at a genome-specific-level. **a** DNA-seq differential expression plot. Each point represents an annotated region for a particular genome. In the x-axis and y-axis, the percentage of reads that are mapped to each annotated region divided by the total mapped reads. **b** Accumulated reads mapped onto each position of the genome smoothed using a window of size 10000. In the x-axis, the genome bases from 1 to a portion of its length. In the y-axis, absolute accumulated number of reads mapped. **c** This plot shows how proteins found by searching with annotated (*Left*) and non annotated (*Right*) reads accumulate along similarity and length. The annotated search depicts higher length and similarity matches, resembling Sanders curve (reference in the main text), whereas non-annotated search shows mostly non significant matches. **d** Annotation mapping. This plot shows reads mapped to a particular genome distributed by annotation properties. The three groups are plotted in different colours and shapes, namely *a* orange crosses for unannotated reads, *b* yellow crosses for semi-annotated reads and *c* purple points for fully-annotated reads. The background grey area represents the accumulation of reads for the whole mapped metagenome in logarithmic scale; thus, darker areas represent higher accumulation

information about the number of reads at region level, and therefore about variations in such accumulation (in case-control experiments). In principle, when working with genome sequencing, a more or less flat profile would be expected, as opposed to transcriptomics sequence data. The genomic profile helps detect highly active regions or different number of copies in such regions. This visualization tool (see Fig. 3b) shows how reads are distributed in a particular species and whether the assigned reads are present along the whole genome or only in the most active areas. Another possibility offered by this tool is that it helps the user decide whether to perform or not a pre-assembly of the reads mapped to a specific genome to support the connections found between reads.

Extensive and further verification We propose that the distribution of fragments based on the comparison of reads versus genomes is now divided into two different distributions, as seen in Fig. 3c. Additional verification was performed by representing the matching values for reads falling into annotated and unannotated regions. This is obtained by blasting the set of annotated and unannotated reads mapped to a given genome against a database of proteins –such as swissprot [26]. As expected, different distributions are obtained, which evidences the suitability of using different thresholds. This affirmation is supported by the different levels of sequence conservation in annotated and unannotated regions.

Mapping over annotated regions of genomes Annotation mapping is another example of in-depth analysis of a specific genome and, in particular, of low abundance ones. Our workflow uses all the reads assigned to a genome and divides them into three groups: (1) No annotations, in the sense that the annotation files obtained did not contain any annotation at the position where the read was mapped; (2) Semi-annotations, when a part of the mapped read contains annotations, and (3) Full annotations, when the whole read contains an annotation.

These three groups are plotted onto the whole mapped metagenome distribution (see Fig. 3d). The background grey area represents the accumulation of reads for the whole mapped metagenome in logarithmic scale; darker areas represent higher accumulations. The identity-length distribution of reads for all fragments (with any filtering) is provided by GECKO and can be partially obtained from data evidencing significant alignments yielded by other programs (BLAST) (which can be tuned to also report random distribution). The rationale of this result comes from the experiments of Sanders et al. [27] and Rost [28] that significance is related to the tail of the distributions. Therefore, displaying mapping values on the grey area distribution provides first-glance information about the accuracy of mapping.

Comparison with other metagenomic tools

In order to prove that the results of the proposed workflow are consistent with those of other metagenomic analysis software suites (in terms of abundance in the taxonomic classification), the following test was performed using results from BLASTn based on metagenomic samples from faecal microbial communities. Both, our workflow (MG workflow) and MEGAN were executed using the same input from BLASTn and ran with default parameters (available in the Additional file 1 under "Comparison with MEGAN").

On comparison of the lean metagenome based on MEGAN, the abundance plot (see Fig. 4a) shows similar results to ours. Standard deviation from ratios (using abundance data provided by MEGAN and by our workflow) was 0.25, which is not significant enough to identify relevant variations (see Fig. 4b, c). However, whereas the analysis of a metagenome using MEGAN can last nearly an hour, our MG workflow took about six minutes to analyze the obese metagenome and five minutes for the lean one when the comparison had been done with BLAST. With GECKO, the duration of the process was further reduced, taking about only one minute for the lean sample and three minutes and a half for the obese metagenome. Runtime executions were measured using a regular Intel i5 machine with 4 GB of RAM.

Conclusions

Metagenomics is an effervescent field and there are still a number of questions that need to be addressed before a stable version of data analysis software becomes available. Currently, metagenomic analysis tools generally represent a closed environment and offer few configuration options and limited extension possibilities. Our aim was to develop a software framework to which other modules could be added. An additional motivation to develop this software was the need for software sensitive enough to detect the presence of low-abundance species. Finally, our intent was to provide data in standard and editable formats that facilitate further analysis with external software.

The proposed workflow software offers several notable advantages over the software currently available in the market. Firstly, the use of GECKO enables this software to compute similarity searches in the samples against a collection of genomes in a reasonable time. We found that better results are obtained if a collection of genomes –rather than genes or proteins– is used. At least this was the case when not all genes/proteins from the genomes were registered in reference databases. Moreover, if genomic samples are used (not only transcriptomics), a significant amount of reads would map to unannotated regions, and therefore they would not match to databases composed of genes or proteins.

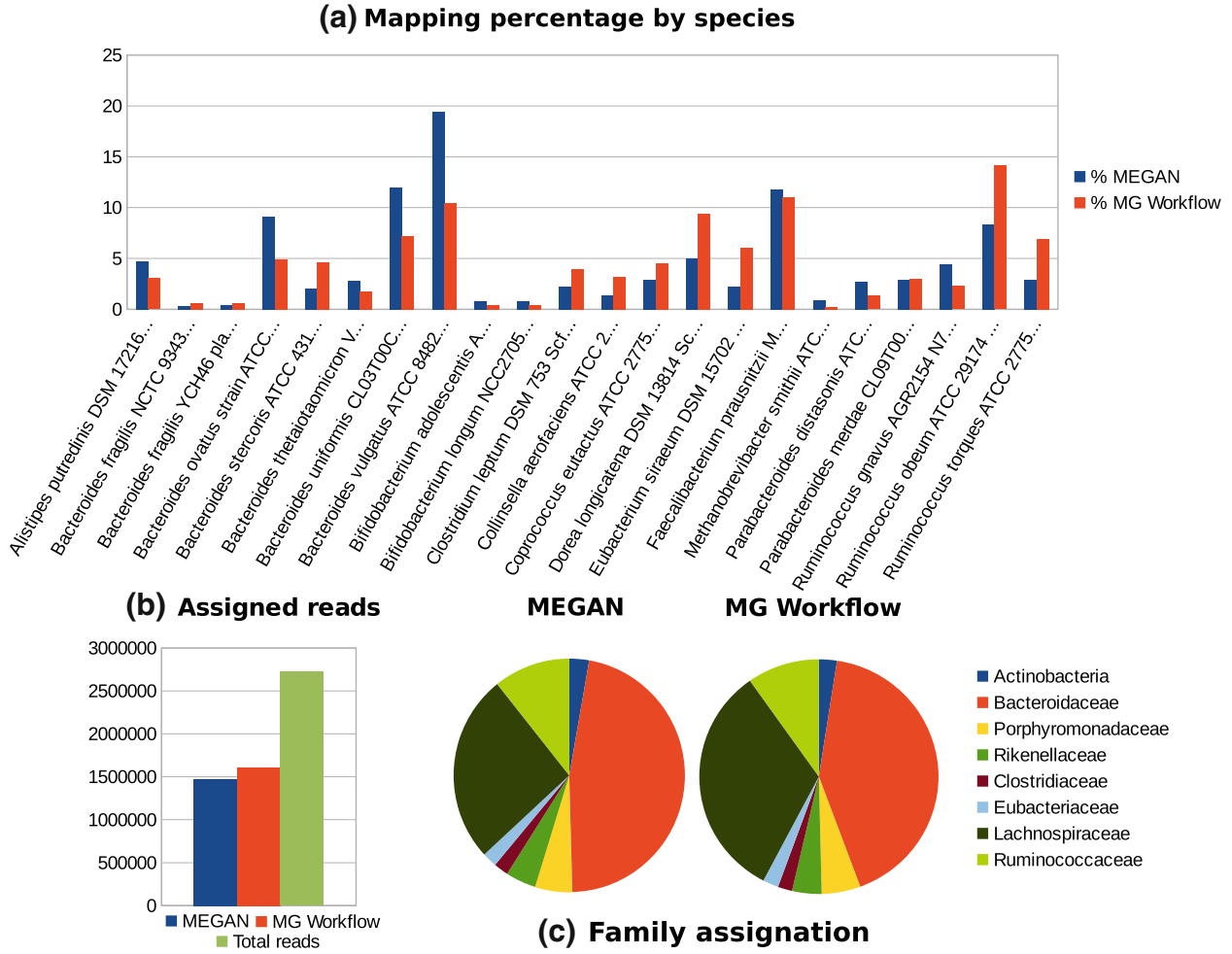


Fig. 4 MEGAN and MG Workflow comparison. Comparative analysis for the lean metagenome shows similar mapping abundances. **a** Abundance plot by species in percentages. **b** Total reads assigned by each method and total number of reads in the metagenome. **c** Abundance chart by family (except Actinobacteria, shown as Phylum)

Providing different mapping alternatives helps set up a sort of quality measures of the mapping process based on abundance differences across mapping alternatives. In addition, the study of the different alternatives could reveal hidden interactions or shared similarities between species that cooperate in some aspects.

The proposed software is designed to provide evidence of the presence of low-abundance species by finding particular specific regions of genomes with mapped reads. These mapped reads provide strong evidence of the species present in samples. The methods developed for assessing and evaluating the quality of mapping also improve accuracy and reliability in terms of the identification of the species present in a sample.

From our perspective, the most important contribution of this workflow software is that it offers the possibility of incorporating new software to extend the analysis workflow by showing datafile specifications enabling fine-grained metagenomic data analysis.

Additional file

Additional file 1: Supplementary material. (PDF 1269 kb)

Acknowledgements
The authors of the manuscript want to thank for all the support provided by the Franciele Maboni Siqueira, Arnaldo Zaha, Ana-Tereza Ribeiro de Vasconcelos, Gonzalo Claros and María del Carmen Morcillo Aixela.

Declarations
This article has been published as part of *BMC Genomics* Volume 17 Supplement 8: Selected articles from the Sixth International Conference of the Iberoamerican Society for Bioinformatics on Bioinformatics and Computational Biology for Innovative Genomics. The full contents of the supplement are available online at https://bmcgenomics.biomedcentral.com/articles/supplements/volume-17-supplement-8.

Funding
This work has been partially supported by the ISCIII (projects: PT13.001.012 and RD12.013.006) and the EU Comission through the Mr.SBM project, code 324554. The publication fees of this work have been funded by the ISCIII through the project RD12.013.006.

Availability of data and materials
The dataset(s) supporting the conclusions of this article is (are) available in the EBI Metagenomics repository https://www.ebi.ac.uk/metagenomics/. The software, additional files, user guides, training material and experiments can be found at http://www.bitlab-es.com/geckointhesubsectionMETA-GECKO.

Authors' contributions
EP-W and JA-M contributed in the software development. OT and OT supervised and coordinated the work and provided ideas for the development of new tools. EU contributed with the biological aspects of the paper. All of the authors contributed to the manuscript's preparation. All the authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Author details
[1] Department of Computer Architecture, University of Málaga, Boulevard Louis Pasteur 35, Málaga, Spain. [2] Advanced Computing Technologies Unit, RISC Software GmbH, Softwarepark 35, Hagenberg, Austria.

Published: 25 October 2016

References

1. Huson DH, Weber N. Microbial community analysis using MEGAN. Methods Enzymol. 2012;531:465–85.
2. Meyer F, et al. The metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes. BMC Bioinforma. 2008;9(1):386.
3. Hunter S, et al. EBI metagenomics—a new resource for the analysis and archiving of metagenomic data. Nucleic Acids Res. 2014;42(D1): D600–D6.
4. Altschul SF, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17): 3389–402.
5. Caporaso GJ, et al. QIIME allows analysis of high-throughput community sequencing data. Nat Methods. 2010;7(5):335–6.
6. Kent WJ. BLAT—the BLAST-like alignment tool. Genome Res. 2002;12(4): 656–64.
7. METApipe Metagenomics Analysis Pipeline. http://bdps.cs.uit.no/papers/nesus-metapipe.pdf.
8. Noguchi H, Taniguchi T, Itoh T. MetaGeneAnnotator: detecting species-specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes. DNA Res. 2008;15(6):387–96.
9. Noguchi H, Park J, Takagi T. MetaGene: prokaryotic gene finding from environmental genome shotgun sequences. Nucleic Acids Res. 2006;34(19):5623–30.
10. Oliveira MFdS, et al. Ribosome binding site recognition using neural networks. Genet Mol Biol. 2004;27(4):644–50.
11. Kultima JR, et al. MOCAT: a metagenomics assembly and gene prediction toolkit. PLoS ONE. 2012;7(10):e47656.
12. Illumina Sequencing Methods. http://www.illumina.com/techniques/sequencing.html.
13. Falgueras J, et al. SeqTrim: a high-throughput pipeline for pre-processing any type of sequence read. BMC Bioinforma. 2010;11(1):1.
14. 454 Life Sciences Technology. http://my454.com/products/technology.asp.
15. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014;30:2114–20.
16. Afgan E, Baker D, van den Beek M, Blankenberg D, Bouvier D, Čech M, Chilton J, Clements D, Coraor N, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res. 2016;44:W3–W10.
17. Torreno O, Trelles O. Breaking the computational barriers of pairwise genome comparison. BMC Bioinforma. 2015;16(1):1.
18. Rumble SM, et al. SHRiMP: accurate mapping of short color-space reads. PLoS Comput Biol. 2009;5(5):e1000386.
19. SOLiD Next-Generation Sequencing. http://www.thermofisher.com/es/es/home/life-science/sequencing/next-generation-sequencing/solid-next-generation-sequencing.html?cid=fl-WE111642.
20. Benson DA, et al. GenBank. Nucleic Acids Res. 2008;36(suppl 1):D25–D30.
21. Brudno M, et al. Glocal alignment: finding rearrangements during alignment. Bioinformatics. 2003;19(suppl 1):i54–i62.
22. Karlin S, Altschul SF. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc Natl Acad Sci. 1990;87(6):2264–8.
23. Galaxy Guided Exercise at BitLab's Research Site. www.bitlab-es.com/gecko/documents/GalaxyGuidedExercise.pdf.
24. Turnbaugh PJ, et al. A core gut microbiome in obese and lean twins. Nature. 2009;457(7228):480–4.
25. Gomez-Alvarez V, Teal TK, Schmidt TM. Systematic artifacts in metagenomes from complex microbial communities. ISME J. 2009;3(11): 1314–7.
26. Boeckmann B, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. 2003;31(1):365–70.
27. Sander Ch, Schneider R. Database of homology derived protein structures and the structural meaning of sequence alignment. Proteins. 1991;9(1):56–68.
28. Rost B. Twilight zone of protein sequence alignments. Protein Eng. 1999;12:85–94.

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.