Escape from parsimony of different models of genome evolution processes

Mona Meghdari Miardan

Thesis submitted in partial fulfillment of the requirements for the degree of
Doctorate in Philosophy Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The Ph.D. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

In the course of evolution, genomes diverge from their ancestors either via global mutations and by rearrangement of their chromosomal segments, or through local mutations within their genes. In this thesis (Chapters: 2, 3 and 4) we analyze the evolution of genomes based on different rearrangement operations including: in Chapter 2 both restricted and unrestricted double-cut-and-join (DCJ) operations, in Chapter 3 both internal and general reversal and translocation (IRT and HP, respectively) operations, and in Chapter 4 translocation, weighted reversal (WR) and maximum length reversal (MLR) operations. Based on the rearrangement operation chosen we can model the evolution of genomes as a discrete or continuous-time Markov chain process on the space of signed genomes.

For each model of evolution, we study the stochastic process by investigating the time up to which the difference between the number of operations along the evolutionary trajectory and the edit distance of the genome from its ancestor is negligible, as soon as these two values starts diverging drastically from one another we say the process escapes from parsimony. One of the major parameters in the known edit distance formulas between any two genomes (such as reversal, DCJ, IRT, HP and translocation) is the number of cycles in their breakpoint graph.

For DCJ, IRT and HP models by adopting the method elaborated by Berestycki and Durret [4], we estimate the number of cycles in the breakpoint graph of the genome at time t and its ancestor by the number of tree components of the random graph constructed from the model of evolution at time t, which is an Erdös-Rényi. We also proved that for each of the DCJ, IRT and HP models of evolution, the process on a genome of size n is bound to its parsimonious estimate up to $t \approx n/2$ steps.

Since the random graph constructed from the models of evolution for the translocation, WR and MLR processes are not Erdös-Rényi, the proofs of their parsimony-bound require more advanced mathematical tools, however our simulation shows for the translocation, two types of WR, and MLR (except for reversals with very short maximum length) models, the escape from parsimony do not occur before $n/2$ steps, where n is the number of genes in the genome.

A basic result in this field is due to Berestycki and Durrett [4], from 2006, who found that a random transposition (pairwise exchange of the elements in the corresponding permutation of the genome) evolves along its parsimonious path of evolution

up to $n/2$ steps, where n is the number of the genes. Although, this transposition model is applicable solely for evolution of a unichromosomal ancestor which remains unichromosomal at each step t of the process; however for the DCJ, IRT, HP and translocation models the genomes are multichromosomal which increases the difficulty of the problem at hand.

The models studied in Chapters 2 - 4 are all based on signed permutation representations of genomes, where each "gene" occurs exactly once, with either positive or negative polarity. The same genes occur in all the genomes being considered. There is no distinction between the same gene in two different genomes. In Chapter 5 we generalize our representation to genes that may have several copies of a gene, which differ only by a few point mutations. This leads to problems of identifying copies in two genomes that are primary orthologs, under the assumptions of differentials in point mutation rate. We provide algorithms, software and test examples.

Dedications

To my parents

Acknowledgement

First and foremost, I would like to express my greatest gratitude to my supervisor Prof. David Sankoff for all he has done for me during my graduate studies. I would like to sincerely thank him for his immense wisdom, vast knowledge, endless support and patience. During these years, your kindness and priceless guidance have been always a great ray of light and hope in my endeavors. Moreover, I would also like to express my sincere appreciation to Dr. Arash Jamshidpey for devoting his valuable time, effort and his generous supervision during this research. I am very grateful for your careful guidance, valuable inspiration, infinite patience and insightful comments which made this dissertation possible. In addition, I would like to thank Prof. Lafond for all of his support and inspiration, I am grateful for his kindness and guidance.

My gratitude extends to my committee Prof. Tom Salisbury, Prof. Daniel Panario, Prof. Mike Newman and Dr. Termeh Kousha. In addition, I would like to thank the Department of Mathematics and Statistics and its kind staff. Especially, I am so grateful to Prof. Benoit Dionne for being tremendously kind and supportive during these years and through ups and downs.

Furthermore, I am very fortunate to have worked alongside my amazing colleagues in the Sankoff lab. Especially, I am very thankful to Poly Hannah da Silva for the inspiration during my research. Thank you Nicole Yu, Alma Oladi, Chunfang Zheng, Fatemeh Pouryahya, Yue Zhang, Yuji Jeong, Qiaoji Xu, Daniella Santos Munoz, Eric Lam, Haitian Zhong and Sindeed Islam.

Lastly, I am beyond thankful to my parents, my sister and my brother in law for always supporting me, my beautiful niece for being a blessing in my life, Zahra and Sepideh for being the most amazing friends one can wish for.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The *genome* of an organism contains all of its genetic information and consists of one or more DNA molecules. Most of the information carried by the DNA is common to all members of the same species, but a small proportion may be characteristic of a few individuals.

A DNA molecule is made up of two long parallel strands, which take on the form of a double helix. Each strand is an oriented sequence of nitrogenous bases (or nucleobases) identified as adenine(A), thymine (T), guanine(G), and cytosine(C), attached to a backbone of sugar and phosphate molecules. The two strands of DNA bond together by pairing adenine(A) to thymine (T), and guanine(G) to cytosine(C), positioned facing each other in the two strands, which are parallel but of opposite orientation.

DNA consists of non-coding and coding segments, where the latter are called genes. In other words, genes are sequences of the bases A, T, G and C. Genes contain recipes for producing proteins. One can view DNA as an ordered sequence of genes. Genes can be on either of the two parallel strands of DNA, so each gene has the orientation of the DNA strand it is on.

The DNA in a genome is partitioned among one or more *chromosomes*. A linear chromosome is a sequence of genes with two ends called telomeres, whereas circular chromosomes have no telomere. A genome can have several chromosomes. For example, the human genome contains 23 pairs of linear chromosomes.

Similarities and divergences among genomes have long sparked scientists' curiosity about their role in the evolution of species. *Comparative genomics* is a field in biology, armed with mathematics and algorithms, aiming to discover and put together the pieces of the genomic evolution puzzle by comparing genomes of related organisms.

For decades, researchers had only access to segments of DNA sequences, such as individual genes, without knowing the complete genomic map of an organism, i.e. all of its DNA sequences. Analyzing those short sequences of DNA was significant in

understanding the evolution of genes via deletion, insertion, and substitution in their nucleotide sequence, called point mutations. More recently, access to the complete genomic mapping of some organisms reveals that genomes also evolve at greater scales via rearrangements of chromosomal segments, such as reversals of large segments. In this light, genomes evolve locally through point mutations of the genes, and globally via rearrangement of the their DNA sequences. The latter is known as *genome rearrangement*. It is worth noting that, genomic rearrangements do not occur as often as point mutations.

1.1 Genome rearrangement operations

One of the main goals of comparative genomics is to find the optimal rearrangement scenarios with the minimum number of allowed rearrangement operations to transform one genome into another. There are several rearrangement operations including *reversal, transposition, block interchange, translocation*, *fusion*, *fission*, and *double cut and join*.

A reversal operation on a segment of a chromosome reverses the order and orientation of the genes in that segment. That is to say, the mentioned changes in the order and orientation are due to the migration of the reversed segment from its initial location in one of the DNA strands to its parallel one. A transposition takes two consecutive segments of a chromosome and swaps them. A more general case of transposition is a block interchange which swaps any two segments of the chromosome. A translocation operation takes any two linear chromosomes within a genome, choose a segment containing a telomere in each chromosome and swaps them in one of the two possible ways. This translocation is called *reciprocal*. Let $\pi_1 =< A\ B >$ and $\pi_2 =< C\ D >$ be two chromosomes, such that A, B and C, D are segments of π_1 and π_2 containing telomeres, respectively. A translocation operation transform π_1 and π_2 to either $\pi_1' =< A\ D >$ and $\pi_2' =< C\ B >$ called prefix-prefix translocation, or $\pi_1'' =< A\ -C >$ and $\pi_2'' =< -B\ D >$ called prefix-suffix translocation, where the negative sign is addressing the inversion of the segment [17].

Fusion and fission are translocation operations, as well. Fusion takes two nonempty chromosomes like $\pi_1 =< A\,T >$ and $\pi_2 =< T\ B >$ as inputs (Ts are telomeres), and transforms them into a nonempty chromosome $\pi_1' =< A\ B >$ and an empty chromosome $\pi_2' =< T\ T >$, and fission does vice versa.

In 2005, Yancopoulos *et al.* [37] introduced *double cut and join* or in short *DCJ* operation. Their innovative genomic operation had a great importance in the field of comparative genomics since all the genomic operations mentioned so far can be obtained by one or two consecutive DCJ operations.

DCJ is a genomic operation that makes two cuts in the same chromosome or two different chromosomes of a genome, and rejoin the affected 4 parts in one of the

possible ways, i.e. DCJ cuts 2 adjacencies among gene pairs like (a, b) and (c, d), and rejoins them either as (a, c) and (b, d), or (a, d) and (b, c).

It is worth noting that DCJ on a unichromosomal genome can contribute to either a reversal operation or appearance of a new circular chromosome; moreover, applying another DCJ on the latter may result in a block interchange (a form of transposition). In addition to the previous scenarios, DCJ on a multichromosomal genome can result in translocation including fission and fusion.

A DCJ operation is *restricted* if creating a circular chromosome as the result of joining the extremities is prohibited, in this case one needs to immediately treat the intermediate circular chromosome with an appropriate DCJ and rejoin it back to an existing linear chromosome.

1.2 Genome representations

In order to model genome rearrangements mathematically, signed (unsigned) genomes are presented by signed (unsigned) permutations, where each integer corresponds to a distinct gene and the signs indicates the orientations of the gene in the chromosome i.e. its strandedness. This representation is possible by assuming no gene duplication in the genome.

Definition 1.2.1. *[13] A signed permutation on* $\{1, \ldots, n\}$ *is a permutation on*

$$\{-n, \ldots, -2, -1, 1, 2, \ldots, n\},$$

such that $\pi_{-i} = -\pi_i$*, and is presented by*

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

Here, we equivalently present it by $\pi = \pi_1 \pi_2 \cdots \pi_n$*.*

Representing genomes as permutations facilitated modeling reversals as permutations. For a signed genome $\pi = \pi_1 \cdots \pi_i \cdots \pi_j \cdots \pi_n$ the reversal operation on the segment $[\pi_i \cdots \pi_j]$ is represented by $\rho(i, j)$, where:

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

which transform π into the composition of two permutations π and $\rho(i, j)$ that is $\pi \circ \rho(i, j)$, such that :

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

The set of all signed permutations on $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$ with group operation $\circ$ (i.e. composition of permutations) is the signed symmetric group and is denoted by $S_n^{\pm}$. In this thesis (except Chapter 5) we assume the genes within each genome are not duplicated which allows us to represents every signed chromosome with n genes by a permutation in $S_n^{\pm}$.

It is worth noting that chromosome itself is not oriented, and by assigning a positive (respectively, negative) orientation to a strand of DNA, we assign positive (respectively, negative) sign to the genes within that strand. In addition to permutations, we can depict each genome via an individual graph. We will introduce the graph representation of the genome in more detail in the upcoming chapters.

1.3 Breakpoint graph

Besides representing single genomes, graphs are employed for comparing two genomes. Hannenhalli and Pevzner [18] following Bafna and Pevzner [1] introduced the breakpoint graph of two signed unichromosomal genomes, see Pevzner [28] for more details. Let $X = x_1 \cdots x_n$ and $X^{id} = 12 \cdots n$ (the identity genome) be two unsigned linear unichromosomal genomes on the same set of n genes such that $x_i \in \{1, 2, \ldots, n\}$, x_i's are distinct. We add two telomeres 0 and $n+1$ to both ends of X and X^{id} transforming them into $0\, x_1 \cdots x_n\, n+1$ and $0\, 1\, 2 \cdots n+1$, respectively. The *breakpoint* graph of X and X^{id}, denoted by $G_{BP}(X, X^{id})$, has the following properties:

- $G_{BP}(X, X^{id})$ has $n+2$ vertices representing the n genes and the two added telomeres.
- The edges in $G_{BP}(X, X^{id})$ represents the adjacencies among the genes in X and X^{id} and are coloured in black and grey, respectively.

Each black (respectively, grey) edge in $G_{BP}(X, X^{id})$ is called a reality (respectively, desired) edge. Figure 1.1 displays the breakpoint graph of two unsigned unichromosomal genomes.

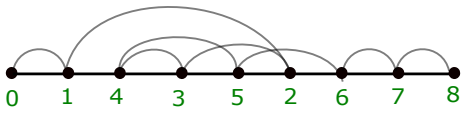

Figure 1.1: The breakpoint graph of two unsigned linear chromosomes X and X^{id} denoted by $G_{BP}(X, X^{id})$, where $X = 1\,4\,3\,5\,2\,6\,7$ and $X^{id} = 1\,2\,3\,4\,5\,6\,7$.

Let $\vec{X}$ and $\vec{X}^{id}$ be two signed linear unichromosomal genomes with n genes, represented by signed permutations on $\{-n, \ldots, -2, -1, 1, 2, \ldots, n\}$, such that we have $\vec{X}^{id} = 1 \cdots n$ and $\vec{X} = \vec{x}_1 \cdots \vec{x}_n$, where $\vec{X}^{id}$ (the identity permutation) represents the

ancestral genome. Hannenhalli and Pevzner [18] showed how to obtain $G_{BP}(\vec{X}, \vec{X}^{id})$ by first transforming $\vec{X}^{id}$ and $\vec{X}$ into two unsigned permutations on $\{1, \ldots, 2n\}$, where $X^{id} = 1 \cdots 2n$ and $X = x_1 \cdots x_{2n}$, respectively. For $i = 1, \ldots, n$, we transform $\vec{X}$ to X by replacing each element in $\vec{X}$ into

- $x_{2i-1} = 2a - 1$ and $x_{2i} = 2a$ for a positive element $\vec{x}_i = a$ (in graph representation, equivalently $x_{2i-1} = a^-$ and $x_{2i} = a^+$), and
- $x_{2i-1} = 2a$ and $x_{2i} = 2a - 1$ for a negative element $\vec{x}_i = -a$ (in graph representation, equivalently $x_{2i-1} = a^+$ and $x_{2i} = a^-$).

To put it in words, each positive gene a or negative gene $-a$ is represented by two extremities a^- and a^+, known as tail and head of the gene, respectively. In this transformation and while depicting the breakpoint graph linearly, we start from left with vertex 0, each positive gene a (negative gene $-a$) is being substituted from the left by a^- and a^+ (a^+ and a^-). Also, $x_0 = 0$ and $x_{2n+1} = 2n + 1$ are the added telomeres, (in graph representation, equivalently $x_0 = 0^+$ and $x_{2n+1} = (n+1)^-$). For $1 \leq i \leq n$, $G_{BP}(\vec{X}, \vec{X}^{id})$ is obtained from $G_{BP}(X, X^{id})$ by deleting all the black edges representing the $\{x_{2i-1}, x_{2i}\}$ adjacencies in X (equivalently $\{x_i^-, x_i^+\}$), and the grey edges representing the $\{2i - 1, 2i\}$ adjacencies in X^{id} (equivalently $\{i^-, i^+\}$). Each deleted black or grey edge is adjacent to the extremities of the same gene. Figure 1.2 displays the breakpoint graph of two signed unichromosomal genomes.

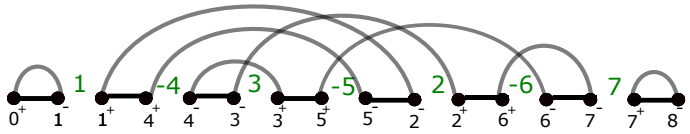

Figure 1.2: $BG(\vec{X}, \vec{X}^{id})$, where $\vec{X} = 1\ {-4}\ 3\ {-5}\ 2\ {-6}\ 7$ and $\vec{X}^{id} = 1\,2\,3\,4\,5\,6\,7$.

As shown in Figure 1.2, the breakpoint graph of two signed unichromosomal genomes decomposes into edge alternating even cycles. The breakpoint graphs for the signed multichromosomal genomes are slightly different. The reason is that [18] applied the breakpoint graph to formulate the reversal distance between two signed unichromosomal genomes while operations like DCJ can be performed on both unichromosomal and multichromosomal genomes. One way of representing the breakpoint graph of two signed multichromosomal genomes is by labeling the telomeres of linear chromosomes belonging to each genome via unique sets of symbols or numbers. Figure 1.3 displays the breakpoint graph of two signed multichromosomal genomes $\Pi = \{\vec{X}_1, \vec{X}_2\}$ and $\Pi^{id} = \{\vec{X}_1^{id}, \vec{X}_2^{id}\}$, where $\vec{X}_1 = 1\ \ {-2}\ 3\ \ {-5}$, $\vec{X}_2 = {-4}\ \ {-6}\ 7$ and $\vec{X}_1^{id} = 1\ 2\ 3\ 4$, $\vec{X}_2^{id} = 5\ 6\ 7$, and all are linear chromosomes. The unique set of telomeres for Π and Π^{id} are $\{T_1', T_2', T_3', T_4'\}$ and $\{T_1, T_2, T_3, T_4\}$, respectively.

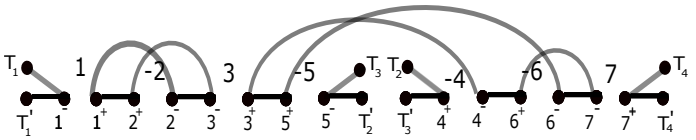

Figure 1.3: $G_{BP}(\Pi, \Pi^{id})$.

As shown in Figure 1.3, the breakpoint graph of two signed multichromosomal genomes decomposes into edge alternating paths and cycles.

1.4 Adjacency graph

Another graph representation for comparing genomes is the *adjacency graph*. Let Π and Π^{id} be two signed genomes, the adjacency graph of Π and Π^{id} introduced by Bergeron *et al.* [6] is denoted by $AG(\Pi, \Pi^{id})$ such that:

- $AG(\Pi, \Pi^{id})$ is a bipartite graph, whose vertices are partitioned into two sets of adjacency pairs belonging to Π and Π^{id}, respectively.
- Each edge in $AG(\Pi, \Pi^{id})$ connects two vertices that share at least one common non-telomeric extremity.

The degree of each vertex in the adjacency graph is either 1 or 2, such that every vertex with degree 1 represents an adjacency incident to one non-telomeric and one telomeric extremities. Whereas, vertices with degree two represent the adjacencies incident to two non-telomeric extremities. Therefore, the adjacency graph of two genomes are union of cycles and paths. A path in the adjacency graph is even (respectively, odd), i.e. has even (respectively, odd) number of edges, if it starts and ends at the vertices corresponding to the telomeres of the same genome (respectively, different genomes). Figure 1.4 displays the adjacency graph of two signed multichromosomal genomes.

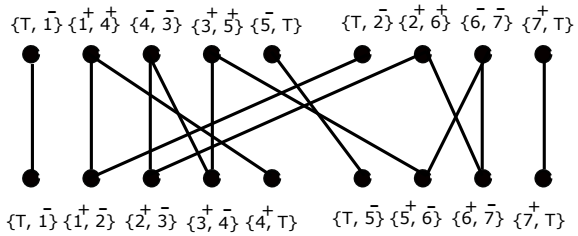

Figure 1.4: $AG(\Pi, \Pi^{id})$, where $\Pi = \{1 - 4\ \ 3 - 5,\ \ 2 - 6\ \ 7\}$ and $\Pi^{id} = \{1\ 2\ 3\ 4,\ \ 5\ 6\ 7\}$.

1.5 Genomic distances

Comparing the genomic sequences of different species initiated the mathematical study of genomic distance. Genomes diverged from the same ancestor can have either exactly the same set or only share a common subset of genes. The latter happens due to insertions or deletions of the genes during the course of evolution. In addition genes can be duplicated as well. In this thesis, except for Chapter 5, when comparing genomes we assume they have unique set of genes, and no gene is inserted, lost or duplicated. Consequently, the descendants inherit the same set of genes as their common ancestor, and their differences are due to the order of genes being scattered in their corresponding DNA sequences. The distance between two genomes measures how much they diverged from one another in the course of evolution. For two genomes Π, Π' with the same set of genes, the genomic distance d is metric if it satisfies:

1. $d(\Pi, \Pi') > 0$ if $\Pi \neq \Pi'$, and $d(\Pi, \Pi') = 0$ if $\Pi = \Pi'$,
2. $d(\Pi, \Pi') = d(\Pi', \Pi)$,
3. $d(\Pi, \Pi') \leq d(\Pi, \Pi'') + d(\Pi', \Pi'')$, for any genome Π'' with the same set of genes as Π and Π'.

1.5.1 Breakpoint distance

Let $\pi = \pi_1 \cdots \pi_n$ be a chromosome with n genes, where $\pi_i \in \{1, \ldots, n\}$ for $1 \leq i \leq n$. We can display the set of adjacencies in a genome as a set of unordered pairs of adjacent genes, such that an adjacency between π_i and π_{i+1} in π can be equivalently presented as either $\{\pi_i, \pi_{i+1}\}$ or $\{\pi_{i+1}, \pi_i\}$. Let $\gamma = \gamma_1 \gamma_2 \cdots \gamma_n$ be a linear chromosome with the same set of genes as π that is $\gamma_k \in \{1, \ldots, n\}$ and distinct for $1 \leq k \leq n$. A pair of adjacent genes $\{\pi_i, \pi_{i+1}\}$ in π that are not adjacent in γ is called a *breakpoint*. That is for $1 \leq k \leq n$, there is no adjacency pair $\{\gamma_k, \gamma_{k+1}\}$ in γ such that either $\gamma_k = \pi_i$ and $\gamma_{k+1} = \pi_{i+1}$, or $\gamma_k = \pi_{i+1}$ and $\gamma_{k+1} = \pi_i$. For two genomes with a common ancestor a set of contiguous genes appearing in the same order in both genomes is called a *conserved segment*. In other words, in each conserved segment the genes kept their adjacency by escaping from disruptions during the course of evolution. The breakpoint distance measures dissimilarities between two genomes by finding the maximum number of breakpoints between them. For two unsigned linear unichromosomal genomes π and γ with n genes, let $\alpha_{\pi,\gamma}$ be the number of common adjacencies among π and γ, then

$$bd(\pi, \gamma) = n - \alpha_{\pi,\gamma} - 1.$$

Note that the chromosome itself does not have an orientation, so that an unsigned chromosome π can be equivalently represented by both unsigned permutations $\pi_1 \cdots \pi_n$

and $\pi_n \cdots \pi_1$. Therefore, the breakpoint distance is a metric measure [15]. Although, calculating the breakpoint distance between two genomes does not provide any information about the number and the type of rearrangement operations required for transforming genomes into one another.

1.5.2 Genome edit distance

Edit distance between two genomes is the minimum number of rearrangement operations required to convert either of the genomes into another. The known algorithms and formulas for the edit distance are based on the chosen operations belonging to the repertoire of allowed genomic rearrangement operations.

1.5.3 Reversal distance

The *reversal distance* between two chromosomes is the minimum number of reversal operations that transform one of them into the other. This distance is one of the oldest and most studied edit distances. In 1982, Watterson *et al.* [36] studied the reversal distance between the configurations of two circular chromosomes, and suggested a heuristic algorithm for it. In 1993, Kececioglu and Sankoff [23] converted the problem of finding the reversal distance between two unsigned unichromosomal genomes into the combinatorial problem of finding the minimum number of reversals transforming the corresponding permutation of either one of the genomes into the other. Additionally, [23] conjectured that the reversal distance for unsigned genomes is NP-hard, this conjecture was proved by Caprara [11], afterward. In 1995, Hannenhalli and Pevzner [18] presented a polynomial time algorithm and an exact formula for calculating the reversal distance between two signed unichromosomal genomes. Theorem 1.5.1 presents the reversal distance formula between two signed permutations $\vec{\pi}$ and $\vec{\pi}^{id}$ belonging to $S_n^{\pm}$. The problem of finding reversal distance between $\vec{\pi}$ and the identity genome $\vec{\pi}^{id}$ is also known as *sorting* $\vec{\pi}$. Since $\vec{\pi}^{id}$ is an identity permutation we can equivalently denote $d^{rev}(\vec{\pi}, \vec{\pi}^{id})$ which is the reversal distance between $\vec{\pi}$ and $\vec{\pi}^{id}$ as $d^{rev}(\vec{\pi})$.

Theorem 1.5.1. *[18]* $d^{rev}(\vec{\pi}) = n + 1 - \mathcal{C}_{BP}(\vec{\pi}) + h(\vec{\pi}) + f(\vec{\pi})$.

The parameters $\mathcal{C}_{BP}(\vec{\pi})$ and $h(\vec{\pi})$ are the numbers of cycles and hurdles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, respectively, and $f(\vec{\pi}) = 1$ if $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ is a fortress, and $f(\vec{\pi}) = 0$ otherwise (see Chapter 3 for definition of hurdles and fortress). Clearly, $d^{rev}(\vec{\pi}^{id}, \vec{\pi}^{id}) = d^{rev}(\vec{\pi}^{id}) = 0$ and the breakpoint graph of $\vec{\pi}^{id}$ and $\vec{\pi}^{id}$ i.e. $BG(\vec{\pi}^{id}, \vec{\pi}^{id})$ has the maximum number of cycles such that $\mathcal{C}(\vec{\pi}^{id}) = n + 1$. While sorting $\vec{\pi}$ a reversal operation on two black edges of $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ belonging to two different cycles of $G(\vec{\pi}, \vec{\pi}^{id})$ merges their cycles into a larger one and reduces the number of cycles of the breakpoint graph by 1. Meanwhile, if the two black edges belong to the same cycle of $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ applying a

reversal operation may or may not split their cycle. Figure 1.5 displays an example of 3 different reversal operations on a same chromosome.

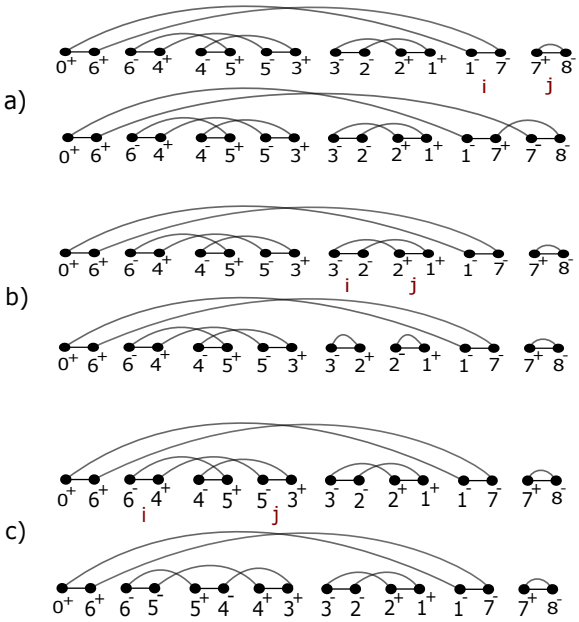

Figure 1.5: Reversal operations: a) i and j are in two different cycles of the breakpoint graph and the reversal decreases the number of cycles by 1. b) i and j are in the same cycle of the breakpoint graph and the reversal increases the number of cycles by 1. c) i and j are in the same cycle of the breakpoint graph and the reversal does not change the number of cycles.

Figure 1.5 shows an example of changes in the number of cycles of the breakpoint graph $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ by applying different reversal operations on a unichromosomal genome $\vec{\pi} = -6\ -4\ -5\ -3\ 2\ -1\ 7$, where $\vec{\pi}^{id} = 1\ 2\ 3\ 4\ 5\ 6\ 7$. As shown in Figure 1.5, applying a reversal operation increases the number of cycles in the breakpoint graph by at most 1. While sorting $\vec{\pi}$ to $\vec{\pi}^{id}$ via reversals, ideally, we would like to increase the number of cycles of the breakpoint graph by each reversal. Although, due to existence of some impediments, this is not always the case. Hannenhalli and Pevzner named such impediments hurdles and fortress, which we discuss in more detail in Chapter 3. Studying reversal distance between genomes enhanced the analysis of genomic evaluation drastically, although, sorting via reversals is most pertinent to unichromosomal genomes. Kececioglu and Ravi [22] pioneered the study of sorting multichromosomal genomes and via more than one type of rearrangement operation. Afterwards, Hannenhalli and Pevzner [17] presented an exact formula and algorithm for sorting multichromosomal genomes via internal reversal and internal translocation operations. An internal reversal (respectively, translocation) operation on a genome does not change the genes' extremities at either end of the chromosomes. Clearly,

fission and fusion are not internal translocations. Furthermore, [17] extended their results and presented a polynomial time algorithm and an exact formula (known as HP distance) for sorting multichromosomal genomes via reversal and translocation (including fission and fusion) operations. Their method includes:

- capping the chromosomes,
- obtaining a permutation for the multichromosomal genome by concatenating the permutations of the capped chromosomes, and
- imitating the reversal and translocation operations on the multichromosomal genome as reversal operations on its concatenation.

Later on, Ozery-Flato and Shamir [27] found a special cases in which HP distance is erroneous, and then corrected the formula, see Chapter 3 for more details.

1.5.4 DCJ distance

Yancopoulos *et al.* [37] presented an $O(n)$ time algorithm for sorting genomes via DCJ operations inspired by the capping procedure in [17]. The initial DCJ distance formula [37] for two genomes Π and Π^{id} with the same set of n genes is

$$d^{DCJ}(\Pi, \Pi^{id}) = b - c, \tag{1.5.1}$$

where b and c are the numbers of breakpoints and cycles (excluding the cycles of length 2) in $G_{BP}(\Pi, \Pi^{id})$, respectively. Shortly after, Bergeron *et al.* [6] formulated the DCJ distance by introducing the adjacency graph, such that

$$d^{DCJ}(\Pi) = d^{DCJ}(\Pi, \Pi_{id}) = n - \left[\mathcal{C}_{AG}(\Pi) + \frac{\mathcal{P}_o(\Pi)}{2} \right], \tag{1.5.2}$$

where $\mathcal{C}_{AG}(\Pi)$ and $\mathcal{P}_o(\Pi)$ are the numbers of cycles and odd paths in $AG(\Pi, \Pi_{id})$, respectively. After that, Bérard *et al.* [3] presented 1.5.2 based on the parameters of the breakpoint graph, such that

$$d^{DCJ}(\Pi) = d^{DCJ}(\Pi, \Pi_{id}) = n - \left[\mathcal{C}(\Pi) + \frac{\mathcal{P}_e(\Pi)}{2} \right], \tag{1.5.3}$$

where $\mathcal{C}(\Pi)$ and $\mathcal{P}_e(\Pi)$ are the numbers of cycles and even paths in $G_{BP}(\Pi, \Pi_{id})$, respectively. Note, finding an appropriate capping for the genomes is not required in 1.5.2 and 1.5.3.

1.6 Different models of genome evolution process and their escape from parsimony

We analyze the evolution of genomes under different types of rearrangement operations (i.e. reversals, DCJs, HPs, translocations, etc) by modeling genome evolution as stochastic processes on the space of genomes.

Definition 1.6.1. *[30] A stochastic process is a collection of random variables that are indexed by time. A stochastic process* $\{X(t), t \in \mathcal{T}\}$ *is a continuous-time (respectively, discrete-time) stochastic process if* $\mathcal{T}$ *is an interval on the real line (respectively, a countable set).*

Definition 1.6.2. *The state space of a stochastic process* $\{X(t), t \in \mathcal{T}\}$ *is all the possible values of* $X(t)$ *for* $t \in \mathcal{T}$.

Definition 1.6.3. *[30] Let* $\{X(n), n = 0, 1, 2, \cdots\}$ *be a discrete-time stochastic process with countable sample space* $\mathcal{S}$*, then* $\{X(n), n = 0, 1, 2, \ldots\}$ *is called a discrete-time Markov chain if*

$$\mathbf{P}(X_{m+1} = j | X_m = i, X_{m-1} = i_{m-1}, \cdots, X_0 = i_0) = \mathbf{P}(X_{m+1} = j | X_m = i),$$

where $j, i, i_{m-1}, \ldots, i_0 \in \mathcal{S}$*. Moreover,* $\mathbf{P}(X_{m+1} = j | X_m = i)$ *is called the transition probability which measures the chance of moving to the state* j *at time* $m + 1$ *from state* i *at time* m*. We can assume transition probabilities are independent of time, and denote*

$$\mathbf{P}_{ij} = \mathbf{P}(X_{m+1} = j | X_m = i).$$

Definition 1.6.4. *[30] A continuous-time finite state Markov chain* $X(t)$ *is defined by two components: a jump chain, and a set of holding time parameters* λ_i*. The jump chain consists of a set of states belonging to the countable state space* $\mathcal{S}$ *along with transition probabilities* $\mathbf{P}_{ij}$*. For* $i, j \in \mathcal{S}$ *and* $t \in \mathcal{T}$*, we assume*

1. *if* $X(t) = i$*, the time until the state changes has* ***Exponential***(λ_i) *distribution;*
2. *if* $X(t) = i$*, the next state will be* j *with probability* $\mathbf{P}_{ij}$.

The process satisfies the Markov property. That is, for all $0 \leq t_1 < t_2 < \cdots t_n < t_{n+1}$*, we have*

$$\mathbf{P}(X(t_{n+1}) = j | X(t_n) = i, X(t_{n-1}) = i_{n-1}, \cdots, X(t_1) = i_1) = \mathbf{P}(X(t_{n+1}) = j | X(t_n) = i),$$

where $j, i, i_{n-1}, \ldots, i_1 \in \mathcal{S}$.

Being parsimony-bound means that in transforming from genome Π^{id} into genome Π, Π^{id} has evolved along the shortest path among all the evolutionary trajectories to Π. Roughly speaking, any set of allowed genomic operations R on a space of genomes $\mathcal{G}$, under which the genome space is closed, induces an edit distance ρ on $\mathcal{G}$ that, for $G_1, G_2 \in \mathcal{G}$, is defined as the minimum number of elements of R needed to transform G_1 into G_2 or vice-versa. Of course, we assume that this is possible in finite steps, for any pair of genomes in $\mathcal{G}$. When a random walk $(\xi_t)_{t\geq 0}$ models the evolution of genomes under R-operations, evolution is parsimony-bound up to time t, if

$$\rho(\xi_{t'}, \xi_0) \approx t', \text{ for } 0 \leq t' \leq t.$$

Let $(X_t)_{t\geq 0}$ be a continuous-time Markov chain with a countable sample space $\mathcal{G}$ and transition probability $\mathbf{P}(G, G')$ for every $G, G' \in \mathcal{G}$, whose jump times are according to the Poisson process with rate 1. In subsequent chapters of this thesis, we investigate parsimony-binding for different models of genomic evolution obtained from considering different Markov chains, these models include DCJ, restricted DCJ, internal reversal and translocation (IRT) and general reversal and translocation (HP).

Edit distances (such as reversal, DCJ, HP) are maximum parsimony methods for comparing genomes. Analyzing genome evolution modeled as a stochastic processes under different genomic operations leads us to ask questions about the time up to which the evolutionary trajectory is parsimony-bound. A genome that has only moderately evolved under a model such as DCJ inevitably recovers the true number of steps that actually occurred, i.e. the evolution is parsimony-bound.

After some time, however, parsimony methods will eventually find a shorter path than the one actually traversed by the evolving genome. We say that evolution "escapes from parsimony" since parsimony underestimates the true length of the evolutionary trajectory.

The point at which the trajectory escapes depends on the stochastic process modeling evolution and the distance measure comparing points on the sample path to the starting point.

The fundamental result in this field is due to Berestycki and Durret [4], who proved that a random transposition process (exchange of two genes - not in the repertoire of biologically meaningful operations) on a unichromosomal genome of size n is bound to its parsimonious estimate up to $t \approx n/2$ steps. Later on Jamshidpey and Sankoff [20] proved a similar result for the reversal random process on the space of unichromosomal genomes.

A review by [7] finds little mathematical work relevant to this direct question for models of evolution under DCJ, HP, IRT, translocation, etc. Although there has been much research on probabilistic modeling of evolution using DCJ operations.
In this thesis, we study this problem in the context of genomic evolution models involving various Markov chains pertaining to DCJ, restricted DCJ, IRT, and HP, in Chapters 2 and 3. Using the Berestycki and Durret method [4] for these models, we

estimate the number of cycles in the breakpoint graph of a random genome at time t and its ancestral genome by the number of tree components of an Erdös-Rényi random graph constructed from the model of evolution. Then we prove for these models that the process on a genome of size n does not escape parsimony before $n/2$ steps, where n is the number of genes in the genome. Furthermore, we initiated the study of the problem for other models such as maximum length reversal, weighted reversal, and translocation, see Chapter 4 for more details. In order to adopt Berestycki and Durret's method, it is more difficult to prove whether the random graph constructed from each of these models of evolution is an Erdös-Rényi random graph. We find strong evidences that the majority of these random processes are parsimony-bound up to time $n/2$, evolving from an ancestral genome with n genes. However, providing mathematical proofs for parsimony binding requires more advanced mathematical tools, which we will address in subsequent research. Finally, in Appendix A we present the pseudocodes for the simulation algorithms.

1.7 Departing from the permutation model of genomes: problems arising from gene duplication

In reality genomes have more than one copy of some genes, which limits the applicability of permutation-based models. The "copies" are not identical, but differ by a small number of point mutations. In comparing two genomes, then, it is often the case where it is not obvious which copy of a certain gene in one genome corresponds to which copy of the same gene in the other genome. This is the "orthology detection" problem. In parallel to the permutation-based investigations described in the previous chapters, we undertook an analysis of the orthology problem under a particular model of contrasting mutation rates, as reported in Chapter 5.

Escape from parsimony of a double-cut-and-join genome evolution process

Mona Meghdari Miardan,[1] Arash Jamshidpey,[2] David Sankoff[1*]

[1]Department of Mathematics and Statistics, University of Ottawa,
Ottawa, Ontario, K1N 6N5, Canada

[2]Department of Mathematics, Columbia University,
New York, NY 10027, USA

*To whom correspondence should be addressed;
E-mail: sankoff@uottawa.ca

September 28, 2021

Keywords: breakpoint graph, double-cut-and-join (DCJ), random walk, parsimony binding, and Erdös-Rényi graphs

Abstract: We analyze models of genome evolution based on both restricted and unrestricted double-cut-and-join (DCJ) operations. We compare the number of operations along the evolutionary trajectory to the DCJ distance of the genome from its ancestor at each step, and determine at what point they diverge: the process escapes from parsimony. Adapting the method developed by Berestycki and Durrett (2006), we estimate the number of cycles in the breakpoint graph of a random genome at time t and its ancestral genome by the number of tree components of an Erdös-Rényi random graph

constructed from the model of evolution. In both models, the process on a genome of size n is bound to its parsimonious estimate up to $t \approx n/2$ steps.

1 Introduction

The introduction of the "double-cut-and-join" (DCJ) methods, by Yancopoulos et al. (2005), and in a slightly different formulation by Bergeron et al. (2006), greatly invigorated the field of genome distance algorithmics. DCJ incorporated the operation of interchanging blocks of genes between arbitrary regions of a chromosome (including "block transposition" of neighbouring regions) to the two previously studied operations of reversing of a block of genes in a chromosome ("inversion") and interchanging the prefixes or suffixes of two chromosomes ("reciprocal translocation", which formally includes chromosome fusion and fission). As a result, the exact DCJ distance is easily calculated in linear time, avoiding the cumbersome work necessitated by the ingenious solution (called "HP") of Hannenhalli and Pevzner (1995a) and Hannenhalli and Pevzner (1995b) a decade earlier, where only reversals and reciprocal translocations were considered.

Edit distances for comparing genomes, like DCJ and HP, minimize the number of steps, chosen from a small repertoire of operations, to transform one genome into another. A mistaken comment sometimes heard, whether approving or critical, is that these methods assume that "evolution takes the shortest path". The methods, however, make no assumption at all about the preferences of the evolutionary process; they are simply tools for inferring evolutionary history.

The minimization criterion basic to edit distances makes it a maximum parsimony method. As such it is model-free, unlike probability-based models. Nevertheless, under any reasonable model of evolution, calculations or simulations show that for a genome that has only moderately evolved, DCJ or similar methods inevitably recover the true number of steps that actually occurred. In this sense, evolution has indeed "followed" a most parsimonious path, but this is only an inferential result, not an evolutionary tendency. We say that the evolution is "parsimony-bound" during this initial period.

As a genome diverges more substantially from its initial state, however, parsimony methods

will eventually find a shorter path than the one actually traversed by the evolving genome. Parsimony underestimates the length of the evolutionary trajectory. When this occurs, we can say that evolution "escapes from parsimony".

The nature of this escape and the point at which it happens depends on the stochastic process modeling evolution and the distance measure comparing points on the sample path to the starting point. Little mathematical work has been done relevant to this direct question, although there has been much research on probabilistic modeling of evolution using DCJ operations, as reviewed in Biller et al. (2015). In the present paper, we model the evolution of genomes as a Markov jump process on the space of genomes, and use DCJ as a tool for tracking it.

A basic result in this field is due to Berestycki and Durrett (2006), who found that escape from parsimony for a random transposition (pairwise exchange) process does not occur before $n/2$ steps, where n is the number of genes in the genome.

2 Genomes and DCJ operations

The genes or markers in a genome can be represented by different positive integers. We denote by $\mathcal{G}_{n,k}$ the space of all genomes with n specific genes (or markers), denoted by $1, \ldots, n$, exactly k linear chromosomes, and a number of circular chromosomes (possibly 0). For the purpose of this paper, it is most convenient to represent a genome $G \in \mathcal{G}_{n,k}$ as an alternating graph $G = (V(G), E(G))$ on $2(n+k)$ vertices, $2n$ of which are of degree 2 and the rest are of degree 1, as in Figure 1. More precisely, we denote by $+i$ and $-i$ the extremities (head and tail) of gene (or marker) $i \in [n] := \{1, \ldots, n\}$, and denote by τ_j, $j \in [2k]$, the telomeres of k linear chromosomes, i.e.

$$V(G) = \{\pm 1, \ldots, \pm n\} \cup \{\tau_1, \ldots, \tau_{2k}\}.$$

On the other hand, let $E_1(G)$ be invariant for all genomes G, and define it to be the set of edges $\{-i, +i\}$, for $i \in [n]$, and let $E_2(G)$ vary for different genomes G and be defined as a perfect matching on $V(G)$. Then, $E(G)$ consists of the edges in $E_1(G)$ and $E_2(G)$, where multiple edges are allowed.

We call $E_1(G)$ and $E_2(G)$ the set of genes and set of adjacencies, respectively, of genome G. We notice that $|E_2(G)|= n+k$, for any $G \in \mathcal{G}_{n,k}$. We suppose the edges in $E_1(G)$ and $E_2(G)$ are dotted and solid, respectively. The solid edges which are incident to the telomeric vertices $\{\tau_1, \ldots, \tau_{2k}\}$ are called telomeric edges, and those vertices in $\{\pm 1, \ldots, \pm n\}$ which are adjacent to $\{\tau_1, \ldots, \tau_{2k}\}$ are called telomeric gene extremities. It is then clear that the graph G is a union of disjoint alternating paths (linear chromosomes) and cycles (circular chromosomes). Indeed, $\deg(+i) = \deg(-i) = 2$ and $\deg(\tau_j) = 1$, for any $i \in [n]$ and $j \in [2k]$. Note that, under this definition, a linear chromosome may be null (i.e. with no genes). In this case, the null chromosome only contains two telomeres τ_i, τ_j and their connecting edge $\{\tau_i, \tau_j\}$.

A natural orientation can be assigned to the connected components (chromosomes) of each genome in $\mathcal{G}_{n,k}$. To establish this, recall that an orientation of a graph G is a digraph whose underlying graph is G. In other words, we can assign a direction to each edge $\{u, v\} \in E(G)$ by assigning exactly one of the two possible arcs (u, v) or (v, u) to it. An orientation of G is then a digraph obtained from G by assigning a direction to each of its edges. Clearly, there are $2^{2(n+k)}$ orientations for $G \in \mathcal{G}_{n,k}$. Let C be a cycle component, and let P be a path component of G. A direction of C, namely $\overrightarrow{C}$, is an orientation for which $\overrightarrow{C}$ is a directed cycle, that is $deg_+(v) = deg_-(v) = 1$, for $v \in V(C)$, where $deg_+(v)$ and $deg_-(v)$ denote the outdegree and indegree of v. Similarly, one can give a direction to a linear chromosome P, by choosing one of its telomeres as a starting point. To be more precise, an orientation $\overrightarrow{P}$ is called a direction for P, if it is a directed path. Hence, there are exactly two directions for each component of G. Denoting by $\kappa(G)$ the number of connected components of G, this implies that the number of directions of G is $2^{\kappa(G)}$. Now, let x be the gene with the smallest label in a circular chromosome C (or a linear chromosome P, respectively). The standard direction is the unique direction of C (P, respectively) for which the direction of $\{-x, +x\}$ is given by arc $(-x, +x)$. Moreover, the standard direction of G is the unique orientation of that for which each component is assigned to its standard direction. We henceforth assume that each genome $G \in \mathcal{G}_{n,k}$ is furnished with its standard direction.

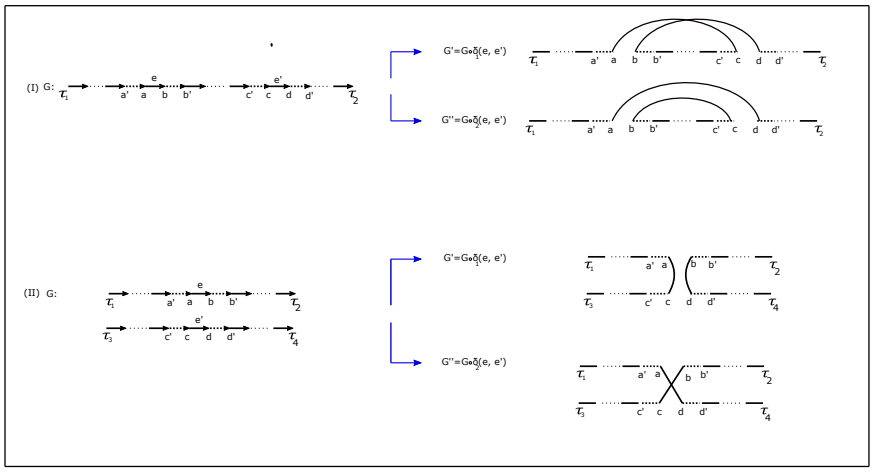

Figure 1: DCJ operations.

A double-cut-and-join (DCJ) operation is a genomic operation that cuts two given adjacencies (edges) of a genome and rejoins the free extremities of the genes involved in one of the two possible ways (Figure 1). Two types of the DCJ operations are defined as follows. For $G \in \mathcal{G}_{n,k}$, and $e = (a, b), e' = (c, d) \in E_2(G)$, a first-type DCJ (DCJ-1 or δ_1) operation, denoted by $\delta_1(e, e')$ is defined by $G' = G \circ \delta_1(e, e')$, where $V(G') = V(G)$, $E_1(G') = E_1(G)$, and

$$E_2(G') = E_2(G) \cup \{(a, c), (b, d)\} \setminus \{e, e'\}.$$

Similarly, a second-type DCJ (DCJ-2 or δ_2) operation on G is defined by $G'' = G \circ \delta_2(e, e')$, where $V(G'') = V(G)$, $E_1(G'') = E_1(G)$, and

$$E_2(G'') = E_2(G) \cup \{(a, d), (b, c)\} \setminus \{e, e'\}.$$

Note that for e, e' taken from the same chromosome, $\delta_1(e, e')$ is an inversion (reversal) operation, as shown in Figure 1-*I*. From the definition, $\mathcal{G}_{n,k}$ is closed under the DCJ operations, and the DCJ distance between two genomes $G_1, G_2 \in \mathcal{G}_{n,k}$, $d(G_1, G_2)$, is the minimum number of DCJ operations

required to transform G_1 into G_2 or vice-versa. Equivalently, this distance can be interpreted as the length of the shortest path on the graph $\mathscr{G}_{n,k}$ with the set of vertices $\mathcal{G}_{n,k}$ where each pair of vertices are connected with an edge if their DCJ distance is 1. The formula for the DCJ distance of two genomes can be given in terms of the numbers of cycles and even paths in their breakpoint graph. In mathematical terms, letting $T := \{\tau_1, \ldots, \tau_{2k}\}$ and $T' := \{\tau'_1, \ldots, \tau'_{2k}\}$, the breakpoint graph of two genomes $G_1, G_2 \in \mathcal{G}_{n,k}$, denoted by $BP(G_1, G_2)$, is a graph whose vertices are given by $\{\pm 1, \ldots, \pm n\} \cup T \cup T'$ and whose edges are the adjacencies (edges) of G_1 and G_2^* (multiple edges are allowed), where G_2^* is obtained from G_2 by changing the labels of its telomeric vertices τ_j to τ'_j, for $j = 1, \ldots, 2k$. The adjacencies of G_1 and G_2^* are usually presented in two different colours, say gray and black. We notice that every vertex of $BP(G_1, G_2)$ which is obtained from a telomeric vertex of G_1 or G_2^* has degree 1. The rest of the vertices of $BP(G_1, G_2)$ are all of degree 2, that is each such vertex is incident to exactly one gray edge and one black edge. Hence, $BP(G_1, G_2)$ is the disjoint union of some alternating cycles and $2k$ alternating paths (or lines). The DCJ distance between two genomes G_1 and G_2 is then given by

$$d(G_1, G_2) = n - \mathbf{C}(G_1, G_2) - \frac{\mathbf{P}_e(G_1, G_2)}{2}, \tag{1}$$

where $\mathbf{C}(G_1, G_2)$ and $\mathbf{P}_e(G_1, G_2)$ determine the number of cycles and the number of even paths in $BP(G_1, G_2)$, respectively. A line is called a TT-line or with TT ends if both telomeric vertices of that line belong to T. Lines with TT' or $T'T'$ ends are defined similarly. It is straightforward to see that a line is even (of even size) if and only if it has TT' ends. In the same spirit of the standard direction assigned to linear and circular chromosomes of a genome in $\mathcal{G}_{n,k}$, one can give a natural direction to the cycles and lines of a breakpoint graph. This can be done by determining a start and an end point for a line. More precisely, the direction of a TT' line is given by traversing it, starting at its T telomeric vertex and ending at its T' telomeric vertex. For TT or $T'T'$ lines, the starting point is the telomeric vertex with a smaller index. On the other hand, a cycle can be traversed, starting at its gene extremity with the minimum index, through the unique black edge incident to it. It is clear that the described direction of a cycle or a line of $BP(G_1, G_2)$ does not

necessarily coincide with its orientation induced from the direction of edges in genomes G_1 and G_2.

The effect of a DCJ operation on the distance between two genomes can be studied as follows. Let $G_1, G_2 \in \mathcal{G}_{n,k}$ and $e = [a, b], e' = [c, d] \in E_2(G_2)$, where by $[a, b] = [a, b]_{G_1,G_2}$, we mean that the direction of e is from a to b, in the natural direction of $BP(G_1, G_2)$, described above. Note once again that this direction may be different from the direction of $\{a, b\}$ in the genome G_2. The DCJ operation $\Delta_1(e, e') = \Delta_1^{G_1,G_2}(e, e')$ is defined on G_2 by cutting e and e' and joining a to c, and b to d. Similarly $\Delta_2(e, e') = \Delta_2^{G_1,G_2}(e, e')$ cuts e and e' and joins a to d, and b to c. Note that δ_i and Δ_i operations are different due to the different orders they use to join vertices.

Let $\alpha_i := d(G_1, G_2 \circ \Delta_i(e, e')) - d(G_1, G_2)$. In fact, α_i measures the displacement of the genome G_2, with respect to a reference genome, after a Δ_i-DCJ operation is performed on it. If e, e' are in two different cycles of $BP(G_1, G_2)$, then $\alpha_1 = \alpha_2 = 1$, since two cycles merge following the DCJ operation on e and e', as in Figure 2-III. If both are in one cycle (Figure 2-IV), then Δ_1 does not change the number of cycles or paths, but Δ_2 fragments it into two smaller cycles. Hence $\alpha_1 = 0$ and $\alpha_2 = -1$. If they belong to one line, then, similarly to the cycle case, $\alpha_1 = 0$. In this case, Δ_2 splits the line into a cycle and a line whose length has the same parity as that of the original line (e.g. if the original line is of an even size, then so is the size of the new line). In this case, since a new cycle is added, we get $\alpha_2 = -1$. When one edge belongs to an odd line (TT or $T'T'$ line) and the other belongs to an even line (TT'-line), as in Figure 2-I, clearly, after splitting and rejoining two lines together, both Δ_1 and Δ_2 operations result in an odd and an even lines, which means $\alpha_1 = \alpha_2 = 0$. If both lines are odd, but all of their 4 telomeric vertices come from one genome, say with TT ends, then both types of DCJ operator generate two odd TT-lines, i.e. $\alpha_1 = \alpha_2 = 0$. If both are odd, one with TT ends and the other with $T'T'$-ends, then Δ_1 and Δ_2 operations generate two TT'-lines which are even, hence $\alpha_1 = \alpha_2 = -1$. For two even TT'-lines, the Δ_1 operation gives two odd lines, one with TT ends and the other with $T'T'$ ends, while Δ_2 operator gives two even TT'-lines. Therefore, $\alpha_1 = 1, \alpha_2 = 0$. Finally, as shown in Figure 2-II, if e belongs to a line and e' belongs to a cycle, the cycle and line merge after implementing Δ_1 or Δ_2 operations. This

gives rise to a line whose length has the same parity as that of the original one. Hence in this case $\alpha_1 = \alpha_2 = 1$.

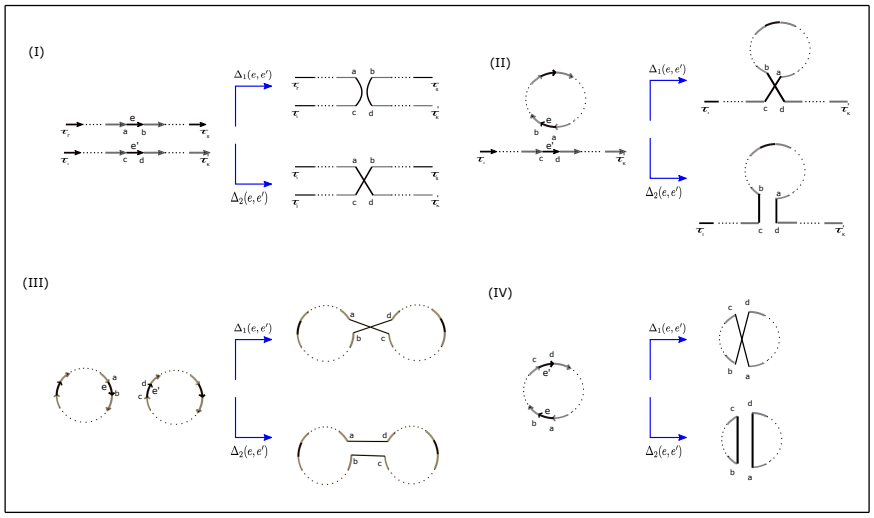

Figure 2: Effects of DCJ operations on the components of the breakpoint graph of two genomes.

3 DCJ evolution and DCJ parsimonious paths

As mentioned in the previous section, the reversals are, in fact, DCJ-1 operations. DCJ can also generate translocations, fissions and fusions. In addition, two consecutive DCJ's may generate a genomic transposition or a block interchange operation (Yancopoulos et al., 2005; Bergeron et al., 2006; Fertin et al., 2009). Consequently, DCJ's provide a rich family of mathematical operations which model the genomic evolution of some species and their edit distances. We can model the DCJ evolution of genomes as a Markov jump process on the space of genomes $\mathcal{G}_{n,k}$. Given an initial weight $p \in [0,1]$, let $\iota_1, \iota_2, \ldots$ be an *i.i.d.* sequence of random variables with $\mathbb{P}(\iota_k = 1) = p = 1 - \mathbb{P}(\iota_k = 2)$, for any $k \in \mathbb{N}$. The DCJ process $X^{(p)} = (X_t^{(p)})_{t \geq 0}$ starts at genome $X_0^{(p)} = G_0 \in \mathcal{G}_{n,k}$, and jumps at random times $0 \leq \tau_1 < \tau_2 < \ldots$, where $\{\tau_1, \tau_2, \ldots\}$ are points of a Poisson process on $\mathbb{R}_+$, with rate 1. At a jump time τ_i, having $X_{\tau_i}^{(p)} = G$, we choose two different solid edges $(a,b), (c,d) \in E_2(G)$,

uniformly at random, without replacement, and let the random walk jump to $G\circ\delta_{t_i}((a,b),(c,d))$. In other words, at each time τ_i, $X^{(p)}$ jumps to $G\circ\delta_1(e,e')$ with probability $p_{n,k} := p/\{(n+k)(n+k-1)\}$, and to $G\circ\delta_2(e,e')$ with probability $q_{n,k} := (1-p)/\{(n+k)(n+k-1)\}$, for any pair $e \neq e' \in E_2(G)$. Equivalently, $X^{(p)}$ can be considered as a continuous-time biased nearest-neighbour random walk on $\mathscr{G}_{n,k}$ starting at G_0, for which at a Poisson time τ_i, it jumps to one of its neighbours with a probability proportional to its weight where the weights of each δ_1-neighbour and each δ_2-neighbour of G are given by p and $1-p$, respectively. We can see that, for $p = 1/2$, $X^{(p)}$ is a continuous-time (symmetric) simple random walk on $\mathscr{G}_{n,k}$. Hereafter, we suppress the superscript p when p is fixed and there is no risk of ambiguity.

We can generalize the above DCJ process as follows. Let $\mathbf{p} := (p_t)_{t\geq 0}$, $p_t \in [0,1]$ for $t \in \mathbb{R}_+$, be a given function from $\mathbb{R}_+$ to $[0,1]$. In order to obtain $X^{(\mathbf{p})} = (X_t^{(\mathbf{p})})_{t\geq 0}$ from $X^{(p)}$, we modify the above definition by giving the weight p_{τ_k} and $1-p_{\tau_k}$ to δ_1 and δ_2 operations which can be performed on the current genome at each jump time τ_k. Then, the probability of choosing each δ_1 operation and each δ_2 operation, at time τ_k, are given by $p_{\tau_k}/\{(n+k)(n+k-1)\}$ and $(1-p_{\tau_k})/\{(n+k)(n+k-1)\}$. In fact, $X^{(p)}$ is a special case of $X^{(\mathbf{p})}$, for $p_t = p$; $t \in \mathbb{R}_+$.

Being parsimony-bound means that in transforming into another genome B, a genome A has evolved along a shortest path to B. This is, in fact, the fastest way that A can be transformed into B through the paths of evolution. Roughly speaking, any set of allowed genomic operations R on a space of genomes S, under which the genome space is closed, induces an edit distance ρ on S that, for $G_1, G_2 \in S$, is defined as the minimum number of elements of R needed to transform G_1 into G_2 or vice-versa. Of course, we assume that this is possible in finite steps, for any pair of genomes in S. When a random walk $(\xi_t)_{t\geq 0}$ models the evolution of genomes under R-operations, evolution is parsimony-bound up to time t, if

$$\rho(\xi_0, \xi_t) \approx t.$$

Setting $S := \mathcal{G}_{n,k}$, and letting R include all DCJ operations, the simulation results in Figure 3 show

that the parsimony binds the trajectory up to time cn, for $c \approx 0.5$.

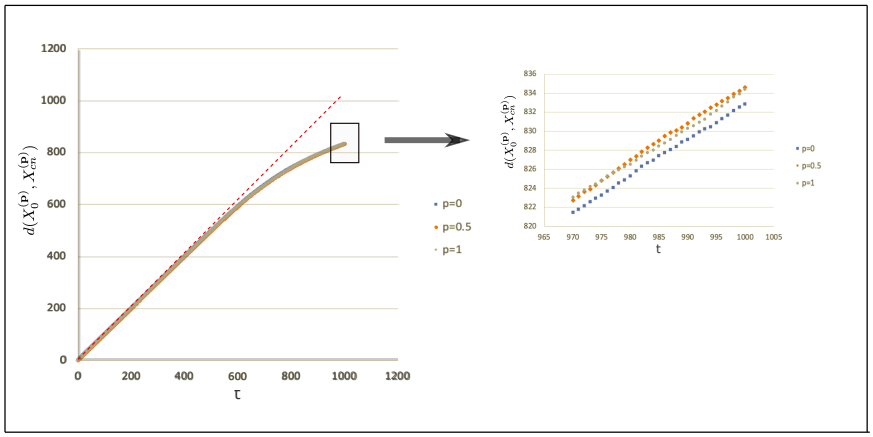

Figure 3: Simulation for unrestricted DCJ model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 100 runs, for $p = 0, 0.5, 1$. The ancestral genome contains 1000 genes in 4 linear chromosomes of size $100, 200, 300$ and 400.

Berestycki and Durrett (2006) proved that the transposition random walk is bound to parsimony up to time $n/2$. Counting the number of hurdles of a reversal random walk, and making use of their results, one can see this also holds for the reversal walk up to the same time $n/2$ (Jamshidpey and Sankoff, 2013). Here, we make use of the Berestycki-Durrett method to show that the same result is true for the DCJ stochastic process. To this end, we need a relabeling mechanism that is updated at each jump of the DCJ process. This technique is important for constructing a random graph process whose number of components estimates the number of cycles of the $BP(X_0, X_t)$, and therefore the DCJ distance between the genomes X_0 and X_t. A labeling of a genome $G \in \mathcal{G}_{n,k}$ is a bijection from $E_2(G)$ onto $[n+k] := \{1, 2, \ldots, n+k\}$. The labeling process $L = (L_t)_{t \geq 0}$ can be obtained from the DCJ process X as follows. We start from an arbitrary label $L_0 = \hat{L}_0$ for X_0. From exchangeability, the results in the next theorems do not really depend on any particular choice of $\hat{L}_0$. Let t be a jump time such that X_t jumps to $X_{t+} := X_t \circ \delta_i(e, e')$ for $e = (a, b), e' = (c, d) \in X_t$. Let $x := \min\{a, b, c, d\}$, and let $\tilde{u}_x$ be the black edge of X_t, either e or e', which is incident to

x, while $\tilde{v}_x$ be the other edge, i.e. $\tilde{v}_x = \{e, e'\} \setminus \tilde{u}_x$. Similarly, let u_x be the edge of X_{t+} which is incident to x after performing $\delta_i(e, e')$ on X_t, and let v_x be the other new edge in X_{t+} that is created as a result of joining the two other free gene extremities. We update the labeling process L by

$$L_{t+}(w) = \begin{cases} L_t(\tilde{u}_x) & \text{if } \ w = u_x \\ L_t(\tilde{v}_x) & \text{if } \ w = v_x \\ L_t(w) & \text{if } \ w \notin \{u_x, v_x\}. \end{cases} \tag{2}$$

Using L and X, we can now define a random graph process $Z = (Z_t)_{t \geq 0}$ whose number of tree components estimates the number of cycles of the breakpoint graph, after a convenient rescaling. Let Z_0 be the trivial graph with vertices $1, 2, \ldots, n+k$ (representing the labels) and with no edges. Let $(\lambda_1(i), \lambda_2(i))$ be the labels of two uniformly random edges (without replacement) in X_{τ_i}, which is selected for a DCJ operation at time τ_i. Connect $\lambda_1(i)$ and $\lambda_2(i)$ by an edge in Z_{τ_i+}, if they are not already adjacent in Z_{τ_i}. As a matter of fact, the number of times at which the random sequence $((\lambda_1(i), \lambda_2(i)))_{i \in \mathbb{N}:\ \tau_i < cn}$ takes an ordered pair of labels $(\ell, \ell') \in [n+k]^2$, is a Poisson random variable with mean $cn/\{(n+k)(n+k-1)\}$. Hence, the probability that each edge $\{\ell, \ell'\}$ appears in the random graph is given by

$$1 - e^{-\frac{2cn}{(n+k)(n+k-1)}} \approx \frac{2cn}{(n+k)(n+k-1)} + O(n^{-2}).$$

Thus, letting Z^*_{cn} be the Erdös-Rényi random graph on the vertex set $[n+k]$ in which each edge is independently present with probability $2c/n$, the expected difference of number of edges in Z_{cn} and Z^*_{cn} is of order $O(1)$. Therefore, from Bollobás (2001, Theorem 5.12) (see page 109),

$$\mathbb{E}[\mathcal{T}_{cn}] = (1 - \gamma(c))n + O(1),$$

where $\mathcal{T}_{cn}$ denotes the number of tree components of Z_{cn}, and, for $c > 0$,

$$\gamma(c) := 1 - \frac{1}{2c} \sum_{j=1}^{\infty} \frac{j^{j-2}}{j!} (2ce^{-2c})^j.$$

In fact, $\gamma(c) = c$, for $0 < c \leq \frac{1}{2}$, and $\gamma(c) < c$, for $c > 1$. Similarly, once again from Bollobás (2001, Theorem 5.12), we get

$$var(\mathcal{T}_{cn}) = O(n).$$

We can then easily see that the Erdös-Rényi random graph Z^*_{cn} approximates Z_{cn}. In mathematical terms, from the Chebyshev's inequality

$$\frac{\mathcal{T}_{cn} - (1 - \gamma(c))n}{c_n\sqrt{n}} \xrightarrow{p} 0, \quad n \to \infty, \tag{3}$$

for an arbitrary sequence $(c_n)_{n\in\mathbb{N}}$, with $c_n \to \infty$, as $n \to \infty$.

Theorem 1. *Fix* $\mathbf{p} = (p_t)_{t\geq 0}$ *and* $c > 0$. *Let* $(c_n)_{n\geq 1}$ *be an arbitrary sequence for which* $c_n \to \infty$, *as* $n \to \infty$. *Then, as* $n \to \infty$,

$$\frac{d(X_0^{(\mathbf{p})}, X_{cn}^{(\mathbf{p})}) - \gamma(c)n}{c_n\sqrt{n}} \xrightarrow{p} 0. \tag{4}$$

Proof. We follow the proof of Berestycki and Durrett (2006, Theorem 3) and adapt it for the DCJ stochastic process. First, note that the number of paths of even length in $BP(X_0, X_{cn})$ is bounded by $2k$, and hence,

$$\mathbf{P}_e(X_0, X_{cn})/\{2c_n\sqrt{n}\} \to 0,$$

pointwisely, as $n \to \infty$. Using (1), it then suffices to analyze the asymptotic behavior of $\mathbf{C}(X_0, X_{cn})$. The number of cycles of size greater than $2\sqrt{n+k}$ is less than $\sqrt{n+k}$. Therefore, we only need to handle the cycles whose sizes are less than or equal to $2\sqrt{n+k}$. To this end, we say a cycle in $BP(X_0, X_{cn})$ is non-fragmented, if it is the result of a finite number of merging events, without any fragmentation in its history. Otherwise, it is called a fragmented cycle. Note that a fragmented cycle always has at least one fragmentation event (either a fragmentation on a cycle or on a line) in its history. For any $t \geq 0$, denote by $\mathcal{C}_t^{(f)}$ and $\mathcal{C}_t^{(nf)}$ the number of fragmented and non-fragmented cycles of $BP(X_0, X_t)$, respectively, with size less than or equal to $2\sqrt{n+k}$. Clearly, $\mathcal{C}_{cn}^{(f)}$ is bounded by 2 times the number of fragmentation events, up to time cn, which results in a fragmented cycle of size at most $2\sqrt{n+k}$. In order that a DCJ operation on two black edges e, e', located in one

connected component of $BP(X_0, X_t)$, splits that component and produces a cycle of size less than $2\sqrt{n+k}$, e and e' should be in distance at most $2\sqrt{n+k}$, in $BP(X_0, X_{cn})$, from each other. At each jump time τ_i, regardless of the value of p_{τ_i}, the probability that $(\lambda_1(i), \lambda_2(i))$ takes a pair of edges with the mentioned property is bounded by $2\sqrt{n+k}/\{n+k-1\}$. Therefore, the number of such fragmentation events, up to time cn, is bounded by a Poisson random variable with parameter $cn \cdot 2\sqrt{n+k}/\{n+k-1\} \approx 2c\sqrt{n}$. This implies that, as $n \to \infty$,

$$\frac{\mathcal{C}_{cn}^{(f)}}{c_n\sqrt{n}} \xrightarrow{p} 0.$$

To estimate $\mathcal{C}_{cn}^{(nf)}$, for any pair of labels $\{\ell, \ell'\}$, let $I_{cn}(\ell, \ell') = I_{cn}(\ell', \ell) = 1$, if the random sequence $(\{\lambda_1(i), \lambda_2(i)\})_{i:\ \tau_i < cn}$ takes $\{\ell, \ell'\}$ at least twice, and let it be equal to 0, otherwise. We have

$$\mathbb{E}[I_{cn}(\ell, \ell')] = \mathbb{P}(I_{cn}(\ell, \ell') = 1) = e^{-\frac{2cn}{(n+k)(n+k-1)}} \sum_{j=2}^{\infty} \left(\frac{2cn}{(n+k)(n+k-1)}\right)^j / j! = O(n^{-2}),$$

and

$$var(I_{cn}(\ell, \ell')) = \mathbb{P}(I_{cn}(\ell, \ell') = 1) - \mathbb{P}(I_{cn}(\ell, \ell') = 1)^2 = O(n^{-2})$$

Letting $I_{cn} = \sum_{\ell,\ell'} I_{cn}(\ell, \ell')$, we have $\mathbb{E}[I_{cn}] = O(1)$ and $var(I_{cn}) = O(1)$, that means the number of edges in Z_{cn} for which the corresponding labels are taken more than once is negligible, and in fact, as $n \to \infty$,

$$\frac{I_{cn}}{c_n\sqrt{n}} \xrightarrow{p} 0.$$

We notice that the tree components of Z_{cn} represent either the non-fragmented cycles and paths of X_{cn} or those fragmented cycles and paths of X_{cn} for which a same pair of labels for a DCJ operation is selected more than once (the number of these cycles or paths is bounded by I_{cn}). As the number of even paths is bounded by $2k$, and the number of cycles or tree components of size greater than $2\sqrt{n+k}$ are both bounded by $\sqrt{n+k}$, the tree components of Z_{cn} can approximate $\mathcal{C}_{cn}^{(nf)}$, and

$$\frac{\mathcal{T}_{cn} - \mathcal{C}_{cn}^{(nf)}}{c_n\sqrt{n}} \xrightarrow{p} 0,$$

as $n \to \infty$. Therefore (3) yields

$$\frac{\mathcal{C}_{cn}^{(nf)} - (1-\gamma(c))n}{c_n\sqrt{n}} \xrightarrow{p} 0,$$

as $n \to \infty$, which finishes the proof. □

Remark 1. *For $c \leq 0.5$, the convergence in (4) implies that DCJ stochastic evolution X is bound by parsimony, since $\gamma(c) = c$.*

4 Restricted DCJ evolution and parsimonious paths

In the previous section we studied the parsimony binding for the DCJ model when there is no constraint for the number of circular genomes. In other words, an arbitrary number of δ_2-operations, say r, could be consecutively performed on one or more linear chromosomes in order to produce r new circular chromosomes in the genome. In this section we study the parsimony problem for the DCJ operations restricted to the unichromosomal linear genomes with the same genes, that is we assume that consecutive δ_1-operations (reversals) are allowed without any restriction, but on the contrary, no two consecutive δ_2-operations are allowed, which means that once an intermediate circular chromosome appears as a result of a δ_2-operation, two adjacencies (black edges), one from each chromosome, are chosen and a DCJ on them generates a unichromosomal linear genome. Another way of looking into this, is by considering a particular subgraph of $\mathcal{G}_{n,1}$, denoted by $\mathcal{G}_n^*$, induced by those vertices (genomes) in $\mathcal{G}_{n,1}$ which have at most one circular chromosome. The vertices of $\mathcal{G}_n^*$ can be partitioned into U, the unichromosomal linear genomes, and $\tilde{U}$, the genomes with exactly one linear and one circular chromosomes. From the definition, it is clear that no two vertices in $\tilde{U}$ are adjacent in $\mathcal{G}_n^*$. The edges between two vertices in U represent reversals, while the ones between a vertex from U and a vertex from $\tilde{U}$ represent the non-reversal DCJ operations needed to transform a unichromosomal linear genome into a genome with one linear and one circular chromosome, and vice versa. The restricted DCJ distance d^* between two genomes $G_1, G_2 \in \mathcal{G}_n^*$ is then the shortest path between G_1 and G_2 in $\mathcal{G}_n^*$. Yancopoulos et al. (2005) showed that, in fact,

for any pair of genomes $G_1, G_2 \in \mathcal{G}_n^*$,

$$d^*(G_1, G_2) = d(G_1, G_2).$$

Hence, once again, one can use the number of cycles and even paths in $BP(G_1, G_2)$, to find $d^*(G_1, G_2)$.

For $\mathbf{p} = (p_t)_{t\in\mathbb{R}_+}$, where $0 \leq p_t \leq 1$, $t \in \mathbb{R}_+$, we define a continuous-time biased nearest-neighbour random walk $Y^{(\mathbf{p})} = (Y_t^{(\mathbf{p})})_{t\geq 0}$ on $\mathcal{G}_n^*$, where $Y^{(\mathbf{p})}$ jumps at Poisson times with rate 1. At each jump time t, if $Y_t^{(\mathbf{p})} \in \tilde{U}$, then it chooses one of its neighbours, uniformly at random, and jumps to it, and if $Y_t^{(\mathbf{p})} \in U$, then it jumps to each of its δ_1-neighbours (obtained as a result of a reversal DCJ) with probability $2p_t/\{n(n+1)\}$ and jumps to each of its δ_2-neighbours (obtained as a result of a non-reversal DCJ) with probability $2(1-p_t)/\{n(n+1)\}$. The case $p_t = 1$, for any $t \geq 0$, gives the reversal (inversion) random walk on the symmetric group of size n. It is known that the reversal random walk diverges from its origin at its maximum speed up to time $n/2$, for large n, where n is the number of genes in a unichromosomal linear genome (Berestycki and Durrett, 2006; Jamshidpey and Sankoff, 2013).

We define the labeling $\tilde{L} = (\tilde{L}_t)_{t\geq 0}$ and the random graph process $\tilde{Z} = (\tilde{Z}_t)_{t\geq 0}$, from $Y^{(\mathbf{P})}$, in the same manner as we defined L and Z from $X^{(\mathbf{P})}$. As before, we drop the superscript $\mathbf{p}$ when there is no risk of confusion. The following theorem estimates $d(Y_0, Y_{cn})$ using the number of tree components of $\tilde{Z}_{cn}$.

Theorem 2. *Let $\mathbf{p} = (p_t)_{t\geq 0}$, such that $0 \leq p_t \leq 1$, $c > 0$, and let $(c_n)_{n\geq 1}$ be an arbitrary sequence of real numbers such that $c_n \to \infty$, as $n \to \infty$. Then, as $n \to \infty$,*

$$\frac{d(Y_0^{(\mathbf{p})}, Y_{cn}^{(\mathbf{p})}) - n + \tilde{\mathcal{T}}_{cn}}{c_n\sqrt{n}} \xrightarrow{p} 0,$$

where $\tilde{\mathcal{T}}_t$ denotes the number of tree components of $\tilde{Z}_t$.

Proof. The proof is similar to the proof of Theorem 1. We define the number of fragmented and non-fragmented cycles of $\tilde{Z}_t$, as we defined the similar notions for Z_t, and denote by $\tilde{\mathcal{C}}_t^{(f)}$ and $\tilde{\mathcal{C}}_t^{(nf)}$, the number of fragmented and non-fragmented cycles of $\tilde{Z}_t$ whose sizes are not greater than $2\sqrt{n+1}$. As before, there are at most 2 paths and at most $\sqrt{n}$ cycles of size greater than $2\sqrt{n+1}$ in $BP(Y_0, Y_{cn})$. Therefore, we only need to handle the asymptotic behaviors of $\tilde{\mathcal{C}}_t^{(f)}$ and $\tilde{\mathcal{C}}_t^{(nf)}$. Once again, the former is bounded by 2ζ, where $\zeta \sim Po(2c\sqrt{n+1})$, and hence, as $n \to 0$,

$$\frac{\tilde{\mathcal{C}}_{cn}^{(f)}}{c_n\sqrt{n}} \xrightarrow{p} 0.$$

Now , it suffices to show, as $n \to \infty$,

$$\frac{\tilde{\mathcal{T}}_{cn} - \tilde{\mathcal{C}}_{cn}^{(nf)}}{c_n\sqrt{n}} \xrightarrow{p} 0.$$

To establish this, for any pair of labels $\{\ell, \ell'\}$ in $(BP(\tilde{Z}_0, \tilde{Z}_t))_{0\le t\le cn}$, we define the indicator functions $\tilde{I}_{cn}(\ell, \ell') = \tilde{I}_{cn}(\ell', \ell)$ and their total sum $\tilde{I}_{cn}$, similarly to their counterparts introduced in the proof of Theorem 1. To find an upper bound for $\mathbb{E}[\tilde{I}_{cn}]$, first observe that the probability that any pair of labels $\{\ell, \ell'\}$ is picked for a DCJ at a jump time t is given by $2/\{n(n+1)\}$, if $Y_t \in U$. If $Y_t \in \tilde{U}$, given that the size of the circular chromosome is $2j$ (which means there are exactly j black edges in the circular chromosome), this probability is given by $1/\{j(n+1-j)\}$. Letting q_j be the probability that a non-reversal DCJ results in a circular chromosome of size $2j$, this means that the probability that any pair of labels $\{\ell, \ell'\}$ is picked for a DCJ, at time t, for $Y_t \in \tilde{U}$ is bounded by

$$\begin{aligned}\sum_{j=1}^{n} \frac{q_j}{j(n+1-j)} \le& \frac{2}{n}\sum_{j=1}^{n} \frac{1}{j(n+1-j)} \\ =&\frac{2}{n(n+1)}\sum_{j=1}^{n}\left(\frac{1}{j} + \frac{1}{n+1-j}\right) \\ =&\frac{4h_n}{n(n+1)} \approx \frac{4\ln n}{n(n+1)},\end{aligned} \tag{5}$$

where $h_n := \sum_{j=1}^{n} 1/j$ is the harmonic number. Therefore, integrating this on $[0, cn]$ and letting

$\tilde{\zeta} \sim Po(q^*_{cn})$ for

$$q^*_{cn} := cn\left(\frac{2}{n(n+1)} \vee \frac{4h_n}{n(n+1)}\right) \approx \frac{4c\ln n}{n},$$

we can couple the random variables $I_{cn}(\ell, \ell')$ with indicator random variables $J_{cn}(\ell, \ell')$ defined independently for any pair of labels $\{\ell, \ell'\}$ by

$$\mathbb{P}(J_{cn}(\ell, \ell') = 1) = 1 - \mathbb{P}(J_{cn}(\ell, \ell') = 0) = \mathbb{P}(\tilde{\zeta} \geq 2),$$

such that $J_{cn}(\ell, \ell') = 1$ if $I_{cn}(\ell, \ell') = 1$ (the converse is not assumed), i.e. $I_{cn}(\ell, \ell') \leq J_{cn}(\ell, \ell')$. We have

$$\begin{aligned}\mathbb{P}(J_{cn}(\ell, \ell') = 1) =& e^{-q^*_{cn}} \sum_{j=2}^{\infty} (q^*_{cn})^j / j! \\ =& O\left(\left(\frac{\ln n}{n}\right)^2\right).\end{aligned}$$

Defining $J_{cn} = \sum_{\ell,\ell'} J_{cn}(\ell, \ell')$, we get $\mathbb{E}[J_{cn}] = O(\ln^2 n)$ and $var(J_{cn}) = O(\ln^2 n)$. Thus, by Chebyshev's inequality, $J_{cn}/\{c_n\sqrt{n}\} \to 0$, in probability, which implies

$$\frac{\tilde{I}_{cn}}{c_n\sqrt{n}} \xrightarrow{p} 0, \quad n \to \infty,$$

since $0 \leq \tilde{I}_{cn} \leq J_{cn}$. The same lines of arguments as those in the proof of Theorem 1 finish the proof. □

Remark 2. *Note that $\tilde{Z}_{cn}$ is not an Erdös-Rényi random graph. Therefore, an exact analysis of the number of trees components of $\tilde{Z}_{cn}$ is not easy and needs further investigation.*

We have seen that the number of cycles in $BP(Y_0, Y_{cn})$ can be estimated by the number of tree components of the random graph $\tilde{Z}_{cn}$. This provides a method to see to what extent evolution by the restricted DCJ process is bound by parsimony. Of course, this could also be checked by simulating Y and deriving the DCJ distance between Y_{cn} and Y_0, using the existing linear-time algorithms for computing the DCJ distance (cf. Yancopoulos et al. (2005); Bergeron et al. (2006)). But to make

use of the method described for the Erdös-Rényi random graph, we investigated the maximum time for which evolution is bound to parsimony by counting the number of tree components in $\tilde{Z}_{cn}$. The simulation study summarized in Figure 4) shows that the binding holds up to time $n/2$, for large n.

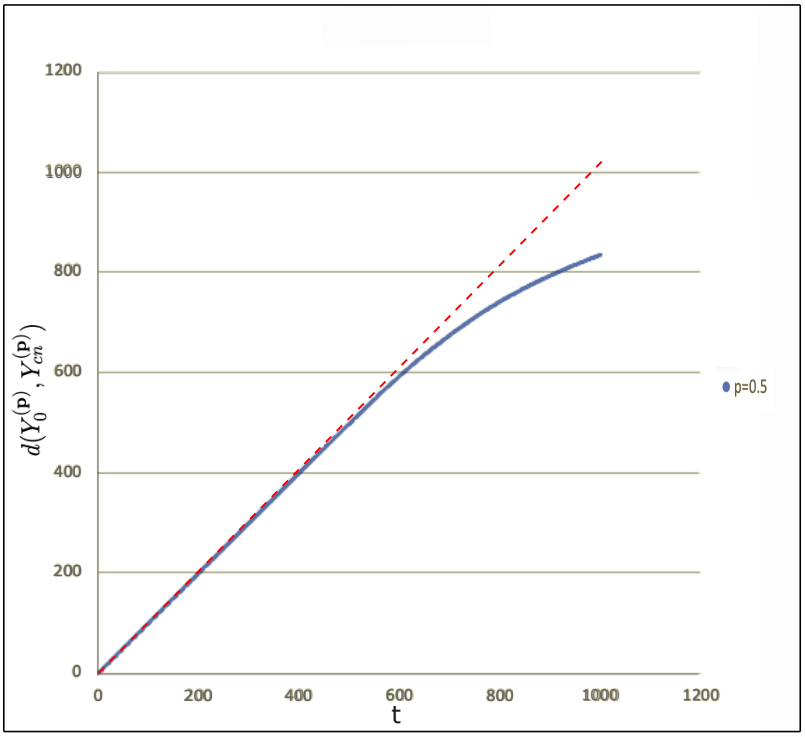

Figure 4: Simulation for the restricted DCJ model illustrating parsimony binding for restricted DCJ up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 100 runs, for 1000 genes; $p = 0.5$.

5 Discussion

We have adapted the analysis of Berestycki and Durrett (2006), originally developed for mathematical transposition random walk on permutations, to the biologically relevant DCJ operations. We have shown that as n gets large, the parsimonious DCJ inference algorithms accurately reflect the number of evolutionary steps traced by evolutionary processes as long as this remains less than $n/2$. Parsimony binding and the escape from parsimony are illustrated by simulation experiments.

Funding

Research supported in part by a Discovery grant to DS from the Natural Sciences and Engineering Research Council of Canada. DS holds the Canada Research Chair in Mathematical Genomics.

References

Berestycki, N. and Durrett, R. A phase transition in the random transposition random walk. *Probability theory and related fields*, 136(2):203–233, 2006.

Bergeron, A., Mixtacki, J., and Stoye, J. A unifying view of genome rearrangements. *Workshop on Algorithms in Bioinformatics*, 163–173. Springer, 2006.

Biller, P., Guéguen, L., and Tannier, E. Moments of genome evolution by double cut-and-join. *BMC Bioinformatics*, 16(14):1–10, 2015.

Bollobás, B. *Random graphs*. Cambridge University Press, 2001.

Fertin, G., Labarre, A., Rusu, I., et al. *Combinatorics of Genome Rearrangements*. MIT Press, 2009.

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

Hannenhalli, S. and Pevzner, P.A. Towards a computational theory of genome rearrangements. *Computer Science Today*, 184–202. Springer, 1995.

Jamshidpey, A. and Sankoff, D. Phase change for the accuracy of the median value in estimating divergence time. *BMC Bioinformatics*, 14 (15), 1–6, 2013.

Yancopoulos, S., Attie, O., and Friedberg, R. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

Chapter 3

Escape from parsimony of a reversal and translocation genome evolution process

Following Chapter 2, in this chapter we analyze models of genome evolution based on two types of reversal and translocation operations and study the time up to which they evolve along their parsimonious paths of evolution. We model the internal reversal and translocation (respectively, general reversal and translocation) process as a Markov chain process on the space of genomes, and measure the dissimilarities between the identity (ancestor) genome and the genome obtained at time t via the internal reversal and translocation (respectively general reversal and translocation) distance. Both distance formulas have been introduced by Hannenhalli and Pevzner [17], although Ozery-Flato and Shamir [27] revised the general reversal and translocation distance formula (known as HP distance) which is the one we apply here. We got inspired by the simulation results of these models in discrete time, see Chapter 4 for more details. We then proved (as expected from the simulations) both of these models do not escape from their parsimonious paths of evolution before $t \approx n/2$, where n is the number of genes in the genome.

3.1 Genomes and reversal and translocation operations

We assume there is no duplication in genomes which means every gene or marker can be represented by a unique positive integer. A *genome* is a set of linear and circular chromosomes, where each chromosome is a sequence of genes or markers. Let $\vec{X}$ be a signed linear chromosome with r different genes $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_r$, such that $\vec{x}_i \in \{-r, \ldots, -2, -1, 1, 2, \cdots, r\}$, where $|\vec{x}_i| \in \{1, 2, \ldots, r\}$ is unique for $1 \leq i \leq r$,

we denote the signed unichromosomal genome $\vec{X}$ by a signed permutation $\vec{X} := \vec{x}_1\vec{x}_2\cdots\vec{x}_r$. That is for $1 \leq i \leq r$, every two genes $\vec{x}_i$ and $\vec{x}_{i+1}$ are *adjacent* in $\vec{X}$. Each chromosome has no orientation itself. By flipping a chromosome $\vec{X}$ or reading it from the other side, the order and orientation of its genes get reversed such that $-\vec{X} = -\vec{x}_r \cdots - \vec{x}_2 - \vec{x}_1$, which is equivalent representation of the chromosome $\vec{X}$ [17][27]. Each gene in the chromosome represented by either a positive integer (marker) a or a negative integer (marker) $-a$ has two extremities denoted by a^+ and a^-, which are known as the head and the tail of the gene, respectively. The sign of each gene indicates its strandedness or orientation within the chromosome. For a gene in X represented by a positive (respectively, negative) marker $+a$ (respectively, $-a$), we denote an ordered pair (a^-, a^+) (respectively, (a^+, a^-)). The set of tails $\mathcal{T}(\vec{X})$ for a linear chromosome $\vec{X} = \vec{x}_1\vec{x}_2\cdots\vec{x}_r$, is defined as the left most extremity of $\vec{x}_1$, and the right most extremity of $\vec{x}_r$ in their corresponding ordered pairs of extremities, therefore flipping a chromosome does not change its tails. A genome with more than one chromosome is called a *multichromosomal* genome. We define $\Pi = \{\vec{X}_1, \ldots, \vec{X}_M\}$ to be a signed multichromosomal genome with n genes and M chromosomes. Furthermore, for $1 \leq i \leq M$, each chromosome $\vec{X}_i$ of Π can be represented as $-\vec{X}_i$ as well. Thus there are 2^M distinct sets equivalently representing Π. We denote $\mathcal{T}(\Pi)$ to be the set of tails for Π such that

$$\mathcal{T}(\Pi) = \bigcup_{i=1}^{M} \mathcal{T}(\vec{X}_i).$$

For the purpose of this chapter, we present genomes with graphs as well. More precisely, for the graph representation of a unichromosomal genome $\vec{X} = \vec{x}_1 \cdots \vec{x}_r$, we define three sets V, E^R and E^B, where

$$V := \{|\vec{x}_1|^-, |\vec{x}_1|^+, \ldots, |\vec{x}_r|^-, |\vec{x}_r|^+\},\ E^R := \Big\{\{|\vec{x}_i|^-, |\vec{x}_i|^+\} | 1 \leq i \leq r\Big\}, \text{ and}$$
$$E^B := \{e_i | 1 \leq i \leq r-1\}.$$

Here, $|\vec{x}_i|$ is the absolute value of the integer representing the genome $\vec{x}_i$. Each edge in E^R represents the adjacency among the two extremities of a gene. Furthermore, for $1 \leq i \leq r-1$, each $e_i \in E^B$ represents the adjacency among two consecutive genes $\vec{x}_i$ and $\vec{x}_{i+1}$ in $\vec{X}$, and is one of the following:

- $e_i = \{\vec{x}_i^+, \vec{x}_{i+1}^-\}$, if $\vec{x}_i, \vec{x}_{i+1} > 0$,
- $e_i = \{|\vec{x}_i|^-, |\vec{x}_{i+1}|^+\}$, if $\vec{x}_i, \vec{x}_{i+1} < 0$,
- $e_i = \{|\vec{x}_i|^-, \vec{x}_{i+1}^-\}$, if $\vec{x}_i < 0$ and $\vec{x}_{i+1} > 0$,
- $e_i = \{\vec{x}_i^+, |\vec{x}_{i+1}|^+\}$, if $\vec{x}_i > 0$ and $x_{i+1} < 0$.

We represent $\vec{X}$ with an edge coloured graph $G_{\vec{X}} = (V_{\vec{X}}, E^R_{\vec{X}} \cup E^B_{\vec{X}})$, where edges in $E^R_{\vec{X}}$ and $E^B_{\vec{X}}$ are coloured in red and black, respectively. $E^R_{\vec{X}}$ is the set of edges representing the adjacencies between the two extremities of any gene in $\vec{X}$, and $E^B_{\vec{X}}$ is the set of proper adjacencies between the extremities of every two consecutive genes in $\vec{X}$. Note that $E^B_{\vec{X}}$ is slightly different for circular and linear chromosomes. To be more precise, when $\vec{X} = (\vec{x}_1 \cdots \vec{x}_r)$ is a circular chromosome, then $V_{\vec{X}} = V$, $E^R_{\vec{X}} = E^R$, and $E^B_{\vec{X}} = E^B \cup \{e_r\}$, where

- $e_r = \{\vec{x}_r^+, \vec{x}_1^-\}$, if $\vec{x}_r, \vec{x}_1 > 0$,
- $e_r = \{|\vec{x}_r|^-, |\vec{x}_1|^+\}$, if $\vec{x}_r, \vec{x}_1 < 0$,
- $e_r = \{|\vec{x}_r|^-, \vec{x}_1^-\}$, if $\vec{x}_r < 0$ and $\vec{x}_1 > 0$,
- $e_r = \{\vec{x}_r^+, |\vec{x}_1|^+\}$, if $\vec{x}_r > 0$and $\vec{x}_1 < 0$.

In this case, cardinality of the sets $V_{\vec{X}}$, $E^R_{\vec{X}}$ and $E^B_{\vec{X}}$ are $|V_{\vec{X}}| = 2r$, $|E^R_{\vec{X}}| = |E^B_{\vec{X}}| = r$, and the edge-coloured graph $G_{\vec{X}}$ has only one edge alternating cycle as its component. When $\vec{X} = \vec{x}_1 \cdots \vec{x}_r$ is a linear chromosome, $E^B_{\vec{X}}$ has two additional adjacencies created by adding two telomeres to the both ends of $\vec{X}$. In this case $V_{\vec{X}} = V \cup \{T_1, T_2\}$, where T_1 and T_2 are the telomeres, $E^R_{\vec{X}} = E^R$, and $E^B_{\vec{X}} = E^B \cup \{e_r, e_{r+1}\}$, where

- $e_r = \{T_1, \vec{x}_1^-\}$ if $\vec{x}_1 > 0$,
- $e_r = \{T_1, |\vec{x}_1|^+\}$ if $\vec{x}_1 < 0$,
- $e_{r+1} = \{\vec{x}_r^+, T_2\}$ if $\vec{x}_r > 0$,
- $e_{r+1} = \{|\vec{x}_r^-|, T_2\}$ if $\vec{x}_r < 0$.

It is worth emphasizing that, e_r and e_{r+1} are the two black edges in $G_{\vec{X}}$ representing the telomeric adjacencies of $\vec{X}$. The non-telomeric extremities incident to e_r and e_{r+1} are actually the tails of $\vec{X}$. So, when X is linear $|V_{\vec{X}}| = 2r + 2$, $|E^R_{\vec{X}}| = r$, and $|E^B_{\vec{X}}| = r + 1$, and the edge-coloured graph $G_{\vec{X}}$ has only one component, which is an alternating path.

In addition, for each unichromosomal genome $\vec{X}$ we assign a standard orientation to $G_{\vec{X}}$. To do so, let $\overrightarrow{G_{\vec{X}}}$ be a digraph obtained by replacing each edge

$$\{a, b\} \in (E^R_{\vec{X}} \cup E^B_{\vec{X}})$$

by two arcs (a, b) and (b, a) in $\overrightarrow{G_{\vec{X}}}$. If $\vec{X}$ is a linear (respectively, circular) chromosome, there are exactly two directed subgraphs which are directed Hamiltonian paths

(respectively, cycles). For a chromosome $\vec{X}$, we assign the standard orientation of $G_{\vec{X}}$ to be the same as the directed Hamiltonian path or cycle of $\overrightarrow{G_{\vec{X}}}$, containing the arc $(|\vec{x}_i|^-, |\vec{x}_i|^+)$, where $\vec{x}_i$ is the gene whose marker has the largest absolute value among the other gene markers in the same chromosome. In words, for such $\vec{x}_i$ we choose the Hamiltonian path or cycle in which the direction of the arc between the two extremities of $\vec{x}_i$ is from $|\vec{x}_i|^-$ to $|\vec{x}_i|^+$. Hereafter, for any edge $e \in E^R_{\vec{X}} \cup E^B_{\vec{X}}$ incident to $\{a, b\}$, we let $d_{\vec{X}}(\{a, b\}) = (a, b)$, if the arc incident to $\{a, b\}$ starting at a and ending at b in the standard orientation of $G_{\vec{X}}$, and $d_{\vec{X}}(\{a, b\}) = (b, a)$, vice versa.

Furthermore, for each linear chromosome $\vec{X}$, with graph representation $G_{\vec{X}}$ and telomeres T_1 and T_2, we define $\widetilde{G}_{\vec{X}} = G_{\vec{X}} \setminus \{T_1, T_2\}$, to be the graph obtained by removing both telomeres of $\vec{X}$ and the two black edges incident to them. In other words, $\widetilde{G}_{\vec{X}} = (\widetilde{V}_{\vec{X}}, \widetilde{E}^R_{\vec{X}} \cup \widetilde{E}^B_{\vec{X}})$ such that $\widetilde{V}_{\vec{X}} = V_{\vec{X}} \setminus \{T_1, T_2\}$, $\widetilde{E}^R_{\vec{X}} = E^R_{\vec{X}}$ and $\widetilde{E}^B_{\vec{X}} \subseteq E^B_{\vec{X}}$, where $|\widetilde{E}^B_{\vec{X}}| = |E^B_{\vec{X}}| - 2 = r - 1$. Moreover, we can find the tails of $\vec{X}$ as the two extremities with degree 1 in $\widetilde{G}_{\vec{X}}$, that is

$$\mathcal{T}(\vec{X}) = \{v \in \widetilde{V}_{\vec{X}} | deg(v) = 1\}.$$

The graph representation of the multichromosomal genome $\Pi = \{\vec{X}_1, \ldots, \vec{X}_M\}$ is an edge-coloured graph

$$G_\Pi = G_{\vec{X}_1} + \cdots + G_{\vec{X}_M},$$

which is the disjoint union of $G_{\vec{X}_i}$'s for $1 \leq i \leq M$. Assume Π has l linear chromosomes, where $l \leq M$, and $\{T_1, T_2, \ldots, T_{2l}\}$ is the set of telomeres, then G_Π decomposes into l alternating paths, and $M - l$ alternating cycles. The standard orientation of G_Π is the one considering the standard orientation of $G_{\vec{X}_i}$'s, for $1 \leq i \leq M$. We also define

$$\widetilde{G}_\Pi = \widetilde{G}_{\vec{X}_1} + \cdots + \widetilde{G}_{\vec{X}_M}.$$

Note that for a circular chromosome $\vec{X}$, $G_{\vec{X}} = \widetilde{G}_{\vec{X}}$.

A *reversal* is a rearrangement operation which takes a segment of a chromosome and reverses the order and orientation of genes within that segment. For $1 \leq i \leq j \leq r$, a reversal on a segment $[\vec{x}_i \cdots \vec{x}_j]$ of $\vec{X} = \vec{x}_1 \cdots \vec{x}_{i-1}\vec{x}_i \cdots \vec{x}_j\vec{x}_{j+1} \cdots \vec{x}_r$ denoted by $\rho(\vec{X}, i, j)$, transforms $\vec{X}$ into a new chromosome

$$\vec{X} \circ \rho(\vec{X}, i, j) = \vec{x}_1 \cdots \vec{x}_{i-1} - \vec{x}_j \cdots - \vec{x}_i\vec{x}_{j+1} \cdots \vec{x}_r.$$

Let $\vec{X}$ be a linear chromosome, we call the segment $[\vec{x}_i \cdots \vec{x}_j]$ of $\vec{X}$ *internal*, if $2 \leq i \leq j \leq r - 1$. A reversal which inverses an internal segment of a chromosome is called an *internal reversal*. For every internal reversal $\rho(\vec{X}, i, j)$ on $\vec{X}$, the set of tails does not change that is $\mathcal{T}(\vec{X}) = \mathcal{T}(\vec{X} \circ \rho(\vec{X}, i, j))$.

We can define the reversal operation as a genomic operation that cuts two adjacencies among genes belonging to the same chromosome of a genome (black edges of the graph representation of the genome) and rejoins the free extremities, in a way that no circular chromosome is created. To be more precise, let $e_1, e_2 \in E^B_{\vec{X}}$ such that $d_{\vec{X}}(e_1) = (a, b)$ (i.e. e_1 is incident to $\{a, b\}$ in $G_{\vec{X}}$ and the direction of its arc connecting them in the standard orientation of $G_{\vec{X}}$ is from a to b), and $d_{\vec{X}}(e_2) = (c, d)$. A reversal operation reversing the segment of a chromosome trapped between two edges e and e' transforms $\vec{X}$ into $\vec{X}'$ whose graph representation is $G_{\vec{X}'} = G_{\vec{X}} \circ \rho_{e_1,e_2}$, such that

$$G_{\vec{X}'} = (V_{\vec{X}}, E^R_{\vec{X}} \cup E^B_{\vec{X}'}),$$

where

$$E^B_{\vec{X}'} = \Big(E^B_{\vec{X}} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}.$$

In this light, a reversal is internal if $e_1, e_2 \in \widetilde{E}^B_{\vec{X}}$. It is worth remembering that, reversal operations are genomic operations that act on a single chromosome of the genome.

A *translocation* is another well known genomic operation which acts on two linear chromosomes. Let $\vec{X} =< A, B >$ and $\vec{Y} =< C, D >$ be two signed linear chromosomes each partitioned into two non-internal (containing one telomere) segments A, B and C, D, respectively. A translocation on $\vec{X}$ and $\vec{Y}$ is a rearrangement that swaps their non-internal segments and transforms them into either $< A, D >$ and $< C, B >$ (prefix-prefix translocation) or $< A, -C >$ and $< -B, D >$ (prefix-suffix translocation). Fission and fusion are non-internal translocations. Fusion takes two chromosomes $\vec{X} =< A, T >$ and $\vec{Y} =< T, B >$ (where T represents telomers), and transforms them into a chromosome $< A, B >$ and an empty chromosome $< T, T >$, and fission does vice versa. Here, we assume an empty chromosome is linear and has only two telomeres and no gene. Let $\vec{X} = \vec{x}_1 \cdots \vec{x}_r$ and $\vec{Y} = \vec{y}_1 \cdots \vec{y}_s$, be two signed linear chromosomes. For $1 \leq i \leq r + 1$ and $1 \leq j \leq s + 1$, we denote $\delta_1(\vec{X}, \vec{Y}, i, j)$ to be a prefix-prefix translocation on $\vec{X}$ and $\vec{Y}$ transforming $\vec{X}$ and $\vec{Y}$ into $\vec{x}_1 \cdots \vec{x}_{i-1}\vec{y}_j \cdots \vec{y}_s$ and $\vec{y}_1 \cdots \vec{y}_{j-1}\vec{x}_i \cdots \vec{x}_r$. Likewise, we denote $\delta_2(\vec{X}, \vec{Y}, i, j)$ to be a prefix-suffix translocation transforming $\vec{X}$ and $\vec{Y}$ into $\vec{x}_1 \cdots \vec{x}_{i-1} - \vec{y}_{j-1} \cdots - \vec{y}_1$ and $-\vec{x}_r \cdots - x_i y_j \cdots y_s$ [16]. A translocation either $\delta_1(\vec{X}, \vec{Y}, i, j)$ or $\delta_2(\vec{X}, \vec{Y}, i, j)$ is internal, if $\vec{X}$ and $\vec{Y}$ are nonempty and neither $i \in \{1, r + 1\}$ nor $j \in \{1, s + 1\}$. It is worth noting that reciprocal translocations on two chromosomes of Π does not alter the set of tails for Π i.e. $\mathcal{T}(\Pi)$.

We can also define a translocation operation as a genomic operation that cuts two adjacencies among genes belonging to two different linear chromosomes of a genome (black edges belonging to different components of the graph representation of the genome) and rejoins the free extremities in one of the two possible ways.

More precisely, let genome $\Pi = \{\vec{X}, \vec{Y}\}$, where $\vec{X}$ and $\vec{Y}$ are linear chromosomes and $e_1, e_2 \in E^B_\Pi$ such that $e_1 \in E^B_{\vec{X}}$ where $d_{\vec{X}}(e_1) = (a, b)$, and $e_2 \in E^B_{\vec{Y}}$ where $d_{\vec{Y}}(e_2) = (c, d)$. We define the translocation $\delta^1_{e_1,e_2}$ transforming genome Π to Π'_1, such that G_Π transforms to

$$G_{\Pi'_1} = G_\Pi \circ \delta^1_{e_1,e_2} = (V_\Pi, E^R_\Pi \cup E^B_{\Pi'_1}),$$

where

$$E^B_{\Pi'_1} = \Big(E^B_\Pi \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}.$$

On the other hand, we define the translocation $\delta^2_{e_1,e_2}$ transforming Π to Π'_2, such that G_Π converted into

$$G_{\Pi'_2} = G_\Pi \circ \delta^2_{e_1,e_2} = (V_\Pi, E^R_\Pi \cup E^B_{\Pi'_2}),$$

where

$$E^B_{\Pi'_2} = \Big(E^B_\Pi \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, d\}, \{b, c\}\Big\}.$$

A translocation $\delta^1_{e_1,e_2}$ or $\delta^2_{e_1,e_2}$ is *internal* if $e_1 \in \widetilde{E}^B_{\vec{X}}$ and $e_2 \in \widetilde{E}^B_{\vec{Y}}$, or equivalently $e_1, e_2 \in \widetilde{E}^B_\Pi$.

Let $\vec{X}$ and $\vec{X}^{id}$ be two signed linear unichromosomal genomes represented by signed permutations on $\{-n, \ldots, -1, 1, \ldots, n\}$, where $\vec{X} = \vec{x}_1 \cdots \vec{x}_n$ and $\vec{X}^{id} = 1 \cdots n$. We obtain the breakpoint graph of $\vec{X}$ and $\vec{X}^{id}$, $G_{BP}(\vec{X}, \vec{X}^{id})$, with some adjustments to the union of $G_{\vec{X}}$ and $G_{\vec{X}^{id}}$. To do so, we assume the vertices representing the two telomeres of both genomes are $T_1 = 0^+$ and $T_2 = (n+1)^-$. We then define

$$G'_{\vec{X}} := (V_{\vec{X}}, E^B_{\vec{X}}) = G_{\vec{X}} \setminus E^R_{\vec{X}}, \text{ and}$$
$$G'_{\vec{X}^{id}} := (V^{id}_{\vec{X}}, E^B_{\vec{X}^{id}}) = G_{\vec{X}^{id}} \setminus E^R_{\vec{X}^{id}}.$$

Clearly, $V_{\vec{X}} = V_{\vec{X}^{id}}$. We attain $G_{BP}(\vec{X}, \vec{X}^{id})$ as the union of $G'_{\vec{X}} = (V_{\vec{X}}, E^B_{\vec{X}})$ and $G'_{\vec{X}^{id}} = (V_{\vec{X}^{id}}, E^B_{\vec{X}^{id}})$, where multiple edges are feasible and the edges belonging to $E^B_{\vec{X}}$ and $E^B_{\vec{X}^{id}}$ are coloured in black and grey respectively. For the purpose of this chapter, while sketching the breakpoint graph, for a linear chromosome $\vec{X}$ we first depict $G_{\vec{X}}$ linearly starting from the vertex 0^+ and ending with vertex $(n+1)^-$, where 0^+ is the left most vertex. In this ordering the vertices representing the two extremities of each gene in $\vec{X}$ appear from left as a^- and a^+ for $+a$, and a^+ and a^- for $-a$. We then remove the red edges of $G_{\vec{X}}$ obtaining $G'_{\vec{X}} = (V_{\vec{X}}, E^B_{\vec{X}})$, and finally add the black edges of $G'_{\vec{X}^{id}} = (V_{\vec{X}^{id}}, E^B_{\vec{X}^{id}})$ as grey edges.

Every vertex in $G_{BP}(\vec{X}, \vec{X}^{id})$ has degree two and is incident to exactly one black and one grey edge, and $G_{BP}(\vec{X}, \vec{X}^{id})$ decomposes into edge alternating even cycles. We represent the number of cycles in $G_{BP}(\vec{X}, \vec{X}^{id})$ by $\mathcal{C}_{BP}(\vec{X}, \vec{X}^{id})$. Each cycle of size 2 in $G_{BP}(\vec{X}, \vec{X}^{id})$ represents a common adjacency between $\vec{X}$ and $\vec{X}^{id}$.

As explained in Chapter 1, in order to obtain $G_{BP}(\vec{X}, \vec{X}^{id})$ for two signed permutations $\vec{X} = \vec{x}_1 \cdots \vec{x}_n$ and $\vec{X}^{id} = 1 \cdots n$, where $\vec{x}_i \in \{-n, \ldots, -1, 1, \ldots, n\}$, in [18] Hannenhalli and Pevzner defined a function transforming $\vec{X}$ and $\vec{X}^{id}$ into two unsigned permutations X and X^{id} of length $2n$, where $X = x_1 \cdots x_{2n}$ and $X^{id} = 1 \cdots 2n$, respectively. More precisely, for $1 \leq i \leq 2n$, the function defined in [18] transforms every ith element in the signed permutation $\vec{X}$, into

- $x_{2i-1} = 2a - 1$ and $x_{2i} = 2a$ for each positive element $\vec{x}_i = a$ in $\vec{X}$ (in graph representation, equivalently presented by $x_{2i-1} = a^-$ and $x_{2i} = a^+$), or
- $x_{2i} = 2a$ and $x_{2i-1} = 2a - 1$ for each negative element $\vec{x}_i = -a$ in $\vec{X}$ (in graph representation, equivalently presented by $x_{2i} = a^+$ and $x_{2i-1} = a^-$).

By assigning $x_0 = 0$ and $x_{2n+1} = 2n + 1$ (in graph representation, equivalently presented by $x_0 = 0^+$ and $x_{2n+1} = (n+1)^-$) and independent from the direction of the edges in $G_{\vec{X}}$ or $G_{\vec{X}^{id}}$, we can present every black and grey edges in $G_{BP}(\vec{X}, \vec{X}^{id})$ as an ordered pair of extremities. To do so for $0 \leq i < j \leq 2n + 1$, an edge $\{x_i, x_j\}$ in $G_{BP}(\vec{X}, \vec{X}^{id})$ is presented by an ordered pair $(x_i, x_j)_{BP}$. Note that this ordering is beneficial for explaining hurdles (see page 43) which is involved in the reversal distance formula.

3.2 Concatenation, IRT and HP distance

Let $\Pi^{id} = \{\vec{\Pi}_1^{id}, \ldots, \vec{\Pi}_M^{id}\}$ be a signed multichromosomal genome with n genes and M linear chromosomes, represented by

$$\begin{aligned} \vec{\Pi}_1^{id} &= 1 \cdots n_1, \\ \vec{\Pi}_2^{id} &= n_1 + 1 \cdots n_2, \\ &\vdots \\ \vec{\Pi}_M^{id} &= n_{M-1} + 1 \cdots n_M, \end{aligned}$$

where $n_M = n$. For $1 \leq i \leq M$, the set of tails for Π^{id} is

$$\mathcal{T}(\Pi^{id}) = \{1^-, {n_1}^+, (n_1 + 1)^-, \ldots, n_i^+, (n_i + 1)^-, \ldots, n^+\}.$$

Let $\Pi = \{\vec{\Pi}_1, \ldots, \vec{\Pi}_M\}$ be another signed multichromosomal genome with the same set of n genes as Π^{id}, and M linear chromosomes. The two genomes Π^{id} and Π are called *co-tailed* if $\mathcal{T}(\Pi^{id}) = \mathcal{T}(\Pi)$.

Definition 3.2.1. *A* concatenation *for a signed multichromosomal genome* Π *is a signed permutation obtained in the following two steps:*

1. *we choose one of the equivalent* 2^M *representations of* Π*,*
2. *we then sequence the signed permutations of its* M *chromosomes with respect to the representation in one of the* $M!$ *possible order.*

Roughly speaking, each concatenation of Π represents a corresponding unichromosomal genome obtained by lining up all the chromosomes of Π in a unique order and orientation.

Let $\vec{\pi}$ be a concatenation of Π whose graph representations are $G_{\vec{\pi}}$ and G_Π, respectively. Then $V_{\vec{\pi}} = \widetilde{V}_\Pi \cup \{T_1, T_2\}$, where T_1 and T_2 are telomeres of $\vec{\pi}$, and since $\vec{\pi}$ and Π have the same set of genes $E^R_{\vec{\pi}} = E^R_\Pi$. Let $|E^B_{\vec{\pi}}|$ be the number of black edges in $G_{\vec{\pi}}$, then $|E^B_{\vec{\pi}}| = n + 1$ since $\vec{\pi}$ is a linear unichromosomal genome with n genes. It is worth noting that, $E^B_{\vec{\pi}}$ representing the adjacencies among genes in $\vec{\pi}$, where $E^B_{\vec{\pi}} \cap E^B_\Pi$ are the set of common adjacencies between $\vec{\pi}$ and Π. Therefore, $E^B_{\vec{\pi}} \bigcap E^B_\Pi = \widetilde{E}^B_\Pi$ which is the set of non-telomeric adjacencies in Π and is the same for any concatenation of Π, where $|\widetilde{E}^B_\Pi| = n - M$. Furthermore, $E^B_{\vec{\pi}} \setminus E^B_\Pi$ are the set of edges created in $\vec{\pi}$ as the result of concatenating the M chromosomes of Π, so $|E^B_{\vec{\pi}} \setminus E^B_\Pi| = M + 1$, and each edge in $E^B_{\vec{\pi}} \setminus E^B_\Pi$ is incident to two unique vertices in $\mathcal{T}(\Pi) \cup \{T_1, T_2\}$, and the set of such edges is unique for any concatenation of Π.

The problem of sorting co-tailed multichromosomal signed genomes via finite sequence of internal reversals and internal translocations, which we call IRT operations hereafter, have been initiated by Kececioglu and Ravi [22]. Hannenhalli and Pevzner [17] proved their duality theorem and presented a polynomial time algorithm and a distance formula for sorting co-tailed multichromosomal genomes via IRT operations. They have done so by mimicking IRT operations on multichromosomal genomes as reversals on the concatenation of their chromosomes. By extending their results for IRT, Hannenhalli and Pevzner [17] found a polynomial time algorithm and a distance formula for sorting any multichromosomal genome, via a finite sequence of reversals and translocations. In the latter, the operations are not restricted to be internal, and we call them HP operations, from now on. This method was inspired by IRT and by mimicking HP operations on a multichromosomal genome as reversals on any concatenation of the capping of its chromosomes. Later on, Ozery-Flato and Shamir [27] corrected the HP distance formula and the algorithm have been revised by Tesler [35], Ozery-Flato and Shamir [27] and Jean and Nikolski [21].

Let $\vec{\pi}$ and $\vec{\pi}^{id}$ be any concatenations of two co-tailed signed multichromosomal genomes Π and Π^{id} with n genes and M linear chromosomes, respectively. For simplicity, we assume $\vec{\pi}^{id} = 1\,2\,\cdots\,n$. Let $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ be the breakpoint graph for the unichromosomal genomes $\vec{\pi}$ and $\vec{\pi}^{id}$, since we imitate the IRT operations on Π and as reversals on $\vec{\pi}$, the reversal distance between $\vec{\pi}$ and $\vec{\pi}^{id}$ depends on the order and orientation chosen for the chromosomes in the concatenations of Π and Π^{id}. Note that $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ has $2n+2$ vertices and $n+1$ black (respectively, grey) edges representing the adjacencies among the extremities of the unichromosomal genome $\vec{\pi}$ (respectively, $\vec{\pi}^{id}$).

An edge $(\pi_i, \pi_j)_{BP}$ in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ is called *intrachromosomal* if both π_i and π_j belong to the same chromosome of Π, and *interchromosomal*, otherwise. For any concatenation $\vec{\pi}$ and $\vec{\pi}^{id}$, $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ has exactly (respectively, at least) $M+1$ black (respectively, grey) interchromosomal edges, $M-1$ of which are obtained by concatenating the M chromosomes of Π $(respectively, \Pi^{id})$, and the remaining two edges by adding the telomeres 0 and $2n+1$, equivalently 0^+ and $(n+1)^-$, to both ends of $\vec{\pi}$ $(respectively, \vec{\pi}^{id})$. Thus, by choosing different concatenations of Π and/or Π^{id}, the number of interchromosomal black edges remains the same.

Denoted by $\mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id})$, the number of cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, let $e'_1 \in c'$ and $e'_2 \in c''$ be two black edges i.e. adjacencies of $\vec{\pi}$ in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ such that c' and c'' are two cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$. Let $G_{\vec{\pi}'} = G_{\vec{\pi}} \circ \rho_{e'_1, e'_2}$ be the graph representation of $\vec{\pi}'$ obtained by reversing the segment trapped between e'_1 and e'_2 in $\vec{\pi}$, one can see that [2]:

- if $c' \neq c''$, then $\mathcal{C}_{BP}(\vec{\pi}', \vec{\pi}^{id}) - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) = -1$, and
- if $c' = c''$, then $\mathcal{C}_{BP}(\vec{\pi}', \vec{\pi}^{id}) - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) = 0$ or 1.

That is if the two edges e'_1 and e'_2 are in two different cycles c' and c'', a reversal operation on the segment of chromosome trapped between e'_1 and e'_2 merges their cycles into a larger one, while if e'_1 and e'_2 are in the same cycle the same reversal operation may or may not split their cycle into two smaller ones. Let e be a grey edge incident to two black edges e'_1 and e'_2 in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, i.e. $\{e, e'_1, e'_2\} \in c'$, where c' is a cycle in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, and $\vec{\pi}'$ is obtained by reversing the segment trapped between e'_1 and e'_2 in $\vec{\pi}$, then e is *oriented* if $\mathcal{C}_{BP}(\vec{\pi}', \vec{\pi}^{id}) - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) = 1$, and *unoriented* otherwise. In words, every grey edge (e) is called an oriented edge if applying a reversal on the two black edges incident to it (e'_1 and e'_2), breaks its cycle (c') into two smaller cycles, and unoriented otherwise. A cycle of the breakpoint graph that has only unoriented edges, is called an *unoriented cycle*, otherwise it is an oriented cycle. Now, assume $e'_1, e'_2 \in c'$ are any two black edges in the the same cycle c' in the breakpoint graph $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ such that they are not necessarily adjacent to the same grey edge, and let $\vec{\pi}'$ be obtained by reversing the segment trapped between e'_1 and e'_2 in $\vec{\pi}$ i.e. $G_{\vec{\pi}'} = G_{\vec{\pi}} \circ \rho_{e'_1, e'_2}$. We say e'_1 and e'_2 are *convergent* if

$\mathcal{C}_{BP}(\vec{\pi}', \vec{\pi}^{id}) - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) = 1$, and *divergent* otherwise. Based on this definition, every two black edges of every unoriented cycle of the breakpoint graph are divergent towards one another. One can check the orientation of a cycle in the breakpoint graph of $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ [32] by starting from any vertex of the cycle and traversing it in a clockwise direction. For each black edge being traversed, we assign the label zero if it agrees with its direction in the standard orientation of $G_{\vec{\pi}}$, and label one otherwise. This way, every two black edges of the traversed cycle with the same label are divergent. Thus, the cycle is unoriented, only if the labels of the black edges assigned this way are either all zeros or all ones.

Two grey edges $(\pi_i, \pi_j)_{BP}$ and $(\pi_k, \pi_l)_{BP}$ in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ are *interleaving* if $0 \leq i < k < j < l \leq 2n+1$. The upcoming notations closely follow the notations used in [18], [27] and [21]. Two cycles c', c'' in the breakpoint graph $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ are *interleaving* if there exist at least two grey edges $e' \in c'$ and $e'' \in c''$ such that e' and e'' are interleaving in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$. An *interleaving* graph is denoted by $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$, whose vertices are the cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ with size greater than two, such that every two vertex c', c'' in $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ are connected if c' and c'' are interleaving in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$. $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ decomposes into connected components, and the cycles corresponding to the vertices of each connected components of $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ are *interleaving*. A connected component $\mathcal{C}$ in $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ is *unoriented*, if none of its interleaving cycles have an oriented grey edge, and *oriented*; otherwise. Similarly, each connected component $\mathcal{C}$ in $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ is *intrachromosomal*, if none of its interleaving cycles have an *interchromosomal* grey edge, and *interchromosomal*, otherwise.

Let $\mathcal{U}(\vec{\pi})$ be the set of unoriented components in $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$. We define two sets of integers $rg(\mathcal{C})$ and $lg(\mathcal{C})$, for $0 \leq i < j \leq 2n+1$, such that

$$rg(\mathcal{C}) = \{i : (\pi_i, \pi_j)_{BP} \text{ is a grey edge in } \mathcal{C}\},$$

$$lg(\mathcal{C}) = \{j : (\pi_i, \pi_j)_{BP} \text{ is a grey edge in } \mathcal{C}\}.$$

The interval corresponding to each component $\mathcal{C}$ of $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$ is

$$INT(\mathcal{C}) = [min(rg(\mathcal{C})), max(lg(\mathcal{C}))].$$

Let $\mathcal{C}, \mathcal{C}', \mathcal{C}''$ be three components of $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$, then $\mathcal{C}''$ *separates* $\mathcal{C}$ and $\mathcal{C}'$, if there exist a grey edge $(\pi_i, \pi_j)_{BP} \in \mathcal{C}''$ such that $INT(\mathcal{C}) \subset [i, j]$ and $INT(\mathcal{C}') \not\subset [i, j]$, or $INT(\mathcal{C}') \subset [i, j]$ and $INT(\mathcal{C}) \not\subset [i, j]$ [21]. A component $\mathcal{H} \in \mathcal{U}(\vec{\pi})$ is a *hurdle* if there are no $\mathcal{C}, \mathcal{C}' \in \mathcal{U}(\vec{\pi})$ such that $\mathcal{H}$ separates $\mathcal{C}$ and $\mathcal{C}'$. A hurdle $\mathcal{H}$ is the *greatest* if its interval i.e. $INT(\mathcal{H})$ contains the interval of all the hurdles, and it is *minimal* otherwise. Let $\mathcal{H}, \mathcal{H}' \in \mathcal{U}(\vec{\pi})$, where $\mathcal{H}$ is a hurdle and $\mathcal{H}'$ is a non-hurdle, then $\mathcal{H}$ *protects* $\mathcal{H}'$, if $\mathcal{H}'$ becomes a hurdle in $\mathcal{U}(\vec{\pi}) \setminus \mathcal{H}$. A hurdle is *simple* if it does not protect any non-hurdle, and *super* otherwise. The signed permutation representing the unichromosomal genome $\vec{\pi}$ is called a *fortress*, if $\mathcal{U}(\vec{\pi})$ has odd number of super

hurdles, and no simple hurdle. Let $h(\vec{\pi}, \vec{\pi}^{id})$, represent the number of hurdles in $\mathcal{U}(\vec{\pi})$, and $f(\vec{\pi}, \vec{\pi}^{id}) = 1$ if $\vec{\pi}$ is a fortress, and $f(\vec{\pi}, \vec{\pi}^{id}) = 0$, otherwise.

Theorem 3.2.2 presents the reversal distance formula for two signed permutation $\vec{\pi}$ and $\vec{\pi}^{id}$, representing two signed unichromosomal genomes, this distance represents the minimum number of reversal operations that is required to transform $\vec{\pi}$ to $\vec{\pi}^{id}$. Figure 3.1 shows an example for the reversal distance between two genomes.

Theorem 3.2.2. *[17] Let $\vec{\pi}$ and $\vec{\pi}^{id}$ be two signed unichromosomal genomes with the same set of genes. Then $d^{rev}(\vec{\pi}, \vec{\pi}^{id}) = n + 1 - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) + h(\vec{\pi}, \vec{\pi}^{id}) + f(\vec{\pi}, \vec{\pi}^{id})$.*

Note that we assumed $\vec{\pi}$ and $\vec{\pi}^{id}$ are the concatenations of signed multichromosomal genomes Π and Π^{id}, which are applied to mimic sorting Π to Π^{id} via a sequence of IRT operations as sorting $\vec{\pi}$ to $\vec{\pi}^{id}$ via a sequence of reversals. Furthermore, the duality theorem ensures that there exist optimal concatenations $\tilde{\pi}$ and $\tilde{\pi}^{id}$ for Π and Π^{id}, such that sorting $\tilde{\pi}$ to $\tilde{\pi}^{id}$ via reversals is parsimonious and mimics the parsimonious transformation of Π to Π^{id} via IRT operations. Let $d^{IRT}(\Pi, \Pi^{id})$ be the IRT distance between Π and Π^{id}, i.e. the minimum number of IRT operations needed to transform Π into Π^{id}. All the parameters in the IRT distance formula for Π and Π^{id} presented by [17] are independent from the choice of the selected concatenations. One can obtain the mentioned parameters by constructing the breakpoint graph of any concatenations of Π and Π^{id}.

We denote $\mathcal{C}(\Pi)$ to be the number of cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ with no interchromosomal black edges. Following closely the notations used in [17], [27] and [21] in this subsection, let $\mathcal{IU}(\Pi)$ be the set of unoriented intrachromosomal components in $\mathcal{IG}(\vec{\pi}, \vec{\pi}^{id})$, clearly $\mathcal{IU}(\Pi) \subseteq \mathcal{U}(\vec{\pi})$. *Knots, super knots* and *fortress of knots* are hurdles, super hurdles and fortress of hurdles for the set $\mathcal{IU}(\Pi)$, respectively. We represent the number of knots in $\mathcal{IU}(\Pi)$ by $k(\Pi)$.

It is worth noting that for any concatenations $\vec{\pi}$ and $\vec{\pi}^{id}$ of Π and Π^{id}, $k(\Pi)$, $\mathcal{C}(\Pi)$, $f(\Pi)$, and the number of interchromosomal black edges in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ remain the same. The independency of the mentioned parameters to the choice of concatenation is because each concatenation keeps the content and orientation of the intrachromosomal components intact, while the orientation of the interchromosomal components may differ.

Theorem 3.2.3. *[17] Let Π and Π^{id} be two co-tailed multichromosomal genomes with n genes scattered into M linear chromosomes, then*

$$d^{IRT}(\Pi, \Pi^{id}) = n - M - \mathcal{C}(\Pi) + k(\Pi) + f(\Pi).$$

In order to introduce the HP distance formula, let Π and Π^{id} be two multichromosomal genomes with n genes and some linear chromosomes (and no circular chromosomes), such that they are not necessary co-tailed. Following [17], we need to find capping for the chromosomes of Π and Π^{id}, and then concatenate them. This

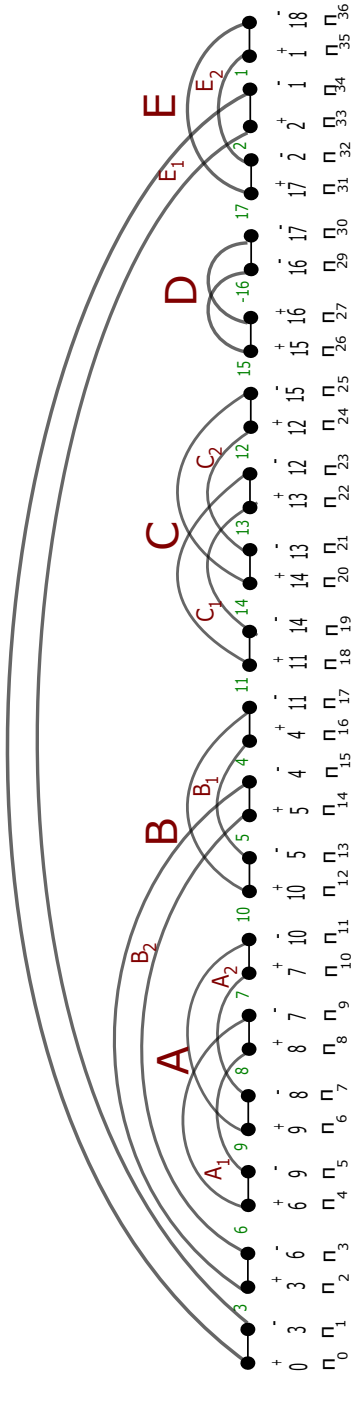

Figure 3.1: $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, the break point graph of two unichromosomal signed genomes $\vec{\pi}$ and $\vec{\pi}^{id}$. The component D is oriented, and the components $\{A, B, C, E\}$ are unoriented such that A, C and E are hurdles, and B is a non-hurdle. Furthermore, A and C are super-hurdles, and E is the greatest hurdle. Although, we have odd number of hurdles, since E is not a super hurdle then $\vec{\pi}$ is not a fortress. So the reversal distance between $\vec{\pi}$ ans $\vec{\pi}^{id}$ is $d^{rev}(\vec{\pi}, \vec{\pi}^{id}) = n + 1 - \mathcal{C}_{BP}(\vec{\pi}, \vec{\pi}^{id}) + h(\vec{\pi}, \vec{\pi}^{id}) + f(\vec{\pi}, \vec{\pi}^{id})) = 17 + 1 - 9 + 3 + 0 = 12$

way sorting multichromosomal genomes Π to Π^{id} via HP operations is like sorting a capping of Π to a capping of Π^{id} via IRT operations. Consequently, Hannenhalli and Pevzner [17] displayed that sorting Π to Π^{id} via a sequence of HP operations, can be mimicked as a sequence of reversals on their capped concatenations. Furthermore, the duality theorem proved that there exists an optimal capping and concatenation $\hat{\pi}$ and $\hat{\pi}^{id}$ for Π and Π^{id} respectively, such that the parsimonious sorting $\hat{\pi}$ to $\hat{\pi}^{id}$ via a sequence of reversals mimics a parsimonious sorting of Π to Π^{id}, via a sequence of HP operations.

Let $\Pi = \{\vec{\Pi}_1, \ldots, \vec{\Pi}_N\}$ and $\Pi^{id} = \{\vec{\Pi}_1^{id}, \ldots, \vec{\Pi}_M^{id}\}$, be two signed multichromosomal genomes, with n genes, and respectively N and M linear chromosomes such that $N \leq M$. Clearly, Π and Π^{id} can have different sets of tails. Without loss of generality, we assume the M chromosomes of Π^{id} are

$$\vec{\Pi}_1^{id} = 1 \cdots n_1,\ \vec{\Pi}_2^{id} = n_1 + 1 \cdots n_2, \ldots,\ \vec{\Pi}_M^{id} = n_{M-1} + 1 \cdots n_M,$$

for $1 \leq i \leq M$, where $n_M = n$. By capping the chromosomes of Π and Π^{id}, we artificially make them co-tailed. If $M - N > 0$, we add $M - N$ empty chromosomes to Π. A reminder that we assumed empty chromosomes are linear with two telomeres and no gene. Let $Cap = \{c_0, \ldots, c_{2M-1}\}$ be the set of caps whose element are $2M$ positive integers which are distinct from the n integers representing the genes. Out of the two extremities corresponding to each cap one is going to be adjacent to a tail of the original chromosome and the other is becoming a new tail for the capped chromosome. Let $\widehat{\Pi}^{id} = \{\widehat{\Pi}_1^{id}, \ldots, \widehat{\Pi}_M^{id}\}$, and $\widehat{\Pi} = \{\widehat{\Pi}_1, \ldots, \widehat{\Pi}_M\}$ be one of the $2M!$ capping of Π. For $1 \leq i \leq M$, each chromosome $\vec{\Pi}_i = \pi_{i,1} \cdots \pi_{i,n_i}$ of Π is capped into

$$\widehat{\Pi}_i = \mathbf{c}_{2i-2}\ \pi_{i,1} \cdots \pi_{i,n_i}\ \mathbf{c}_{2i-1}.$$

Let $\widehat{\Pi}^{id} = \{\widehat{\Pi}_1^{id}, \ldots, \widehat{\Pi}_M^{id}\}$ be a capping for Π^{id} such that

$$\widehat{\Pi}_i^{id} = (\mathbf{-1})^{\alpha}\mathbf{c}_{\alpha}\ n_{i-1} + 1 \cdots n_i\ (\mathbf{-1})^{\beta+1}\mathbf{c}_{\beta},$$

where $1 \leq i \leq M$ and $0 \leq \alpha \neq \beta \leq 2M-1$. Note that the coefficient of each cap is for keeping the consistency between the capping of Π^{id} and Π such that $\mathcal{T}(\widehat{\Pi}) = \mathcal{T}(\widehat{\Pi}^{id})$.

Let $\hat{\pi}$ and $\hat{\pi}^{id}$ be the capped concatenations of Π and Π^{id}, respectively, and $G_{BP}(\Pi, \Pi^{id})$ be the breakpoint graph of multichromosomal genomes Π and Π^{id}. We obtain $G_{BP}(\Pi, \Pi^{id})$ from $G_{BP}(\hat{\pi}, \hat{\pi}^{id})$ by removing the following edges:

- all the $M + 1$ black (respectively, grey) edges in $G_{BP}(\hat{\pi}, \hat{\pi}^{id})$ that exist due to concatenation of $\hat{\Pi}$ (respectively, $\hat{\Pi}^{id}$) and the two added telomeres to both ends of $\hat{\pi}$ (respectively, $\hat{\pi}^{id}$);
- the $2M$ grey edges connecting Π^{id}-tails to Π-caps, i.e. the edges corresponding to the capping assigned to Π^{id}.

$G_{BP}(\Pi, \Pi^{id})$ has $2n + 4M + 2$ vertices, where $2M + 2$ of those have degree 0 and are called tails, $4M$ vertices have degree 1 such that $2M$ of which are Π-caps and the other $2M$ are Π^{id}-tails, and the remaining $2n - 2M$ vertices have degree 2. Thus, $G_{BP}(\Pi, \Pi^{id})$ decomposes into edge-alternating cycles and paths, and isolated vertices. We denote $c(\Pi, \Pi^{id})$ to be the total number of cycles and paths in $G_{BP}(\Pi, \Pi^{id})$. Each cycle of length two represents a common adjacency in Π and Π^{id}, and each non-empty path starts and ends with a black edge and it is alternating if it has more than one edge. Each path in $G_{BP}(\Pi, \Pi^{id})$ is named based on the type of vertices it starts and ends with, and is called as one of: $\Pi\Pi$-path, $\Pi^{id}\Pi^{id}$-path, $\Pi\Pi^{id}$-path, or $\Pi^{id}\Pi$-path. Among paths in $G_{BP}(\Pi, \Pi^{id})$ with length more than one, a path is *oriented* if it has at least one oriented grey edge, and *unoriented* otherwise. A Π-cap (respectively, Π^{id}-tail) vertex in $\Pi\Pi$-path (respectively, $\Pi^{id}\Pi^{id}$-path) is called a $\Pi\Pi$-vertex (respectively, $\Pi^{id}\Pi^{id}$-vertex). We denote $p(\Pi, \Pi^{id})$, to represents the number of $\Pi\Pi-$paths (or equivalently, $\Pi^{id}\Pi^{id}-$paths) in $G_{BP}(\Pi, \Pi^{id})$.

Let $\mathcal{IG}(\Pi, \Pi^{id})$ be the interleaving graph corresponding to $G_{BP}(\Pi, \Pi^{id})$, and let $\mathcal{IU}(\Pi)$ be the set of intrachromosomal unoriented components in $\mathcal{IG}(\Pi, \Pi^{id})$. Since some components of $\mathcal{IG}(\Pi, \Pi^{id})$ have unoriented intrachromosomal paths, we update the interval of components in $\mathcal{IG}(\Pi, \Pi^{id})$ such that for a component $\mathcal{C} \in \mathcal{IG}(\Pi, \Pi^{id})$, the interval of $\mathcal{C}$ is $INT(\mathcal{C}) = [min(rg(\mathcal{C})), max(lg(\mathcal{C}))]$, where

- $rg(\mathcal{C}) = \{i : (\pi_i, \pi_j)_{BP}$ is a grey edge or $(\pi_i, \pi_{i+1})_{BP}$ is a black edge in $\mathcal{C}\}$, and
- $lg(\mathcal{C}) = \{j : (\pi_i, \pi_j)_{BP}$ is a grey edge or $(\pi_{j-1}, \pi_j)_{BP}$ is a black edge in $\mathcal{C}\}$.

A component $R \in \mathcal{IU}(\Pi)$ is called a *real component* if there are no Π^{id}-tails and/or Π-caps in $INT(R)$. Let $\mathcal{RU}(\Pi)$ be the set of real components in $\mathcal{IU}(\Pi)$. Hurdles, super hurdles, and fortress of hurdles for $\mathcal{RU}(\Pi)$, are called real-knots, super-real -knots, and fortress-of-real-knots. Following [17], we denote $r(\Pi, \Pi^{id})$ to represent the number of real-knots in $G_{BP}(\Pi, \Pi^{id})$.

A component SK in $\mathcal{IU}(\Pi) \setminus \mathcal{RU}(\Pi)$ is called a *semi-knot* if it is a knot in $\mathcal{IU}(\Pi)$, and there are no $\Pi\Pi$-vertex or $\Pi^{id}\Pi^{id}$-vertex in $INT(SK)$. Although, there exists at least one $\Pi\Pi^{id}$-path or $\Pi^{id}\Pi$-path in $INT(SK)$. A component RK in $\mathcal{IU}(\Pi) \setminus \mathcal{RU}(\Pi)$ is called a *semi-real-knot* if it is a semi-knot and by closing all the $\Pi\Pi^{id}$-paths and/or $\Pi^{id}\Pi$-paths in the interval of SK, it turns to a simple real-knot and not a super-real-knot [27]. Following [27], we denote $s'(\Pi, \Pi^{id})$ to represent the number of semi-real-knot in $G_{BP}(\Pi, \Pi^{id})$.

A component $\mathcal{C}$ in $G_{BP}(\Pi, \Pi^{id})$ is called a *simple* component if:

1. there is a $\Pi\Pi^{id}$-path in $\mathcal{C}$, and
2. $\mathcal{C}$ is not a semi-real-knot.

A graph $\bar{G}_{BP}(\Pi, \Pi^{id})$ is obtained by closing all $\Pi\Pi^{id}$-paths in every simple component of $G_{BP}(\Pi, \Pi^{id})$. We close a $\Pi\Pi^{id}$-path by adding a grey edge connecting the two ends of that path. A graph $G_{BP}(\Pi, \Pi^{id})$ is a *weak-fortress-of-real-knots* if:

1. it has an odd number of real-knots,
2. it has a greatest-real-knot,
3. all the real-knots except the greatest-real-knot are super-real-knots,
4. it has at least one semi-real-knot i.e. $s' > 0$.

It is worth noting that if we close all the $\Pi\Pi^{id}$-paths in one of the semi-knots belonging to a *weak-fortress-of-real-knots* $G_{BP}(\Pi, \Pi^{id})$, it converts to a *fortress-of-real-knots*. Furthermore, $fr'(\Pi, \Pi^{id})$ can be either 0 or 1, such that $fr'(\Pi, \Pi^{id}) = 1$ if either

- $\bar{G}_{BP}(\Pi, \Pi^{id})$ is a fortress-of-real-knots and there is no semi-greatest-real-knot in $\bar{G}_{BP}(\Pi, \Pi^{id})$, or
- $\bar{G}_{BP}(\Pi, \Pi^{id})$ is a weak-fortress-of-real-knots,

otherwise $fr'(\Pi, \Pi^{id}) = 0$. And similarly, $gr'(\Pi, \Pi^{id})$ can be either 0 or 1, such that $gr'(\Pi, \Pi^{id}) = 1$ if there is a greatest-real-knot in $G_{BP}(\Pi, \Pi^{id})$ and $s' > 0$, otherwise $gr'(\Pi, \Pi^{id}) = 0$. The following theorem is the HP distance revised by Ozery-Flato and Shamir.

Theorem 3.2.4. *[27] Let Π and Π^{id} be two signed multichromosomal genomes with n genes and at most M linear chromosomes (and no circular chromosomes), then*

$$\begin{aligned} d^{HP}(\Pi, \Pi^{id}) =& n + M - c(\Pi, \Pi^{id}) + p(\Pi, \Pi^{id}) + r(\Pi, \Pi^{id}) \\ &+ \left\lceil \frac{s'(\Pi, \Pi^{id}) - gr'(\Pi, \Pi^{id}) + fr'(\Pi, \Pi^{id})}{2} \right\rceil. \end{aligned}$$

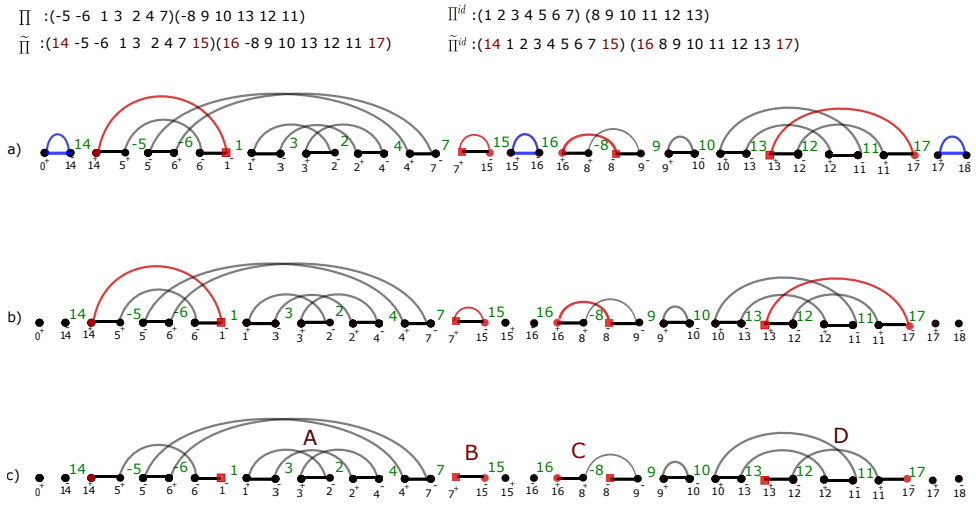

Figure 3.2: a) $G_{BP} = (\tilde{\pi}, \tilde{\pi}^{id})$. b) $G_{PB} = (\widetilde{\Pi}, \widetilde{\Pi}^{id})$. c)$G_{BP} = (\Pi, \Pi^{id})$.

In Figure 3.2, part (a) shows the breakpoint graph of $\tilde{\pi} = (14\ -5\ -6\ 1\ 3\ 2\ 4\ 7\ 15\ 16\ -8\ 9\ 10\ 13\ 12\ 11\ 17)$ and $\tilde{\pi}^{id} = (14\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 15\ 16\ 8\ 9\ 10\ 11\ 12\ 13\ 17)$, where $\tilde{\pi}$ and $\tilde{\pi}^{id}$ are the concatenations of capping for Π and Π^{id}, respectively. In Figure 3.2, part (b) shows $G_{BP}(\tilde{\pi}, \tilde{\pi}^{id})$ after removing the adjacencies appeared due to concatenation of the capped chromosomes. In Figure 3.2 part (c) shows the breakpoint graph of Π and Π^{id}, $G_{BP} = (\Pi, \Pi^{id})$ which is obtained by removing the grey edges (to distinguish them we coloured them red) connecting Π-caps (red filled circles) to Π^{id}-tales (red filled squares). One can see component A is a real-knot, components B and C are simple, and component D is a semi-real-knot. Furthermore, $fr'(\Pi, \Pi^{id}) = 1$ since $\bar{G}_{BP}(\Pi, \Pi^{id})$ is a fortress of real-knot, and $gr'(\Pi, \Pi^{id}) = 0$ since there is no greatest real-knot in $\bar{G}_{BP}(\Pi, \Pi^{id})$. By applying HP distance formula represented in Theorem 3.2.4 we have:

$$d^{HP}(\Pi, \Pi^{id}) = 13 + 2 - 8 + 0 + 1 + \left\lceil \frac{1 - 0 + 1}{2} \right\rceil = 9.$$

3.3 Internal reversal and translocation (IRT)

Prior to defining an IRT model of genomic evolution on the space of signed multichromosomal genomes, we explain the procedure of mimicking IRT operations on a signed multichromosomal genome Π as reversals on any desired concatenation of Π. The parameters in IRT distance formula between any two multichromosomal genomes are dependent to the components of the breakpoint graph of any desired concatenation of them. Let $\Pi = \{\vec{\Pi}_1, \ldots, \vec{\Pi}_M\}$ be a multichromosomal genome with n genes scattering into M linear chromosomes, and $\Pi^{id} = \{\vec{\Pi}_1^{id}, \ldots, \vec{\Pi}_M^{id}\}$ be a reference multichromosomal genome that has the same set of n genes and is co-tailed with $\vec{\Pi}$. Without loss of generality, we assume for $1 \leq i \leq M$ we have

$$\vec{\Pi}_1^{id} = 1 \cdots n_1,\ \vec{\Pi}_2^{id} = n_1 + 1 \cdots n_2, \ldots,\ \vec{\Pi}_M^{id} = n_{M-1} + 1 \cdots n_M,$$

such that $n_M = n$ and $\vec{\pi}^{id} = 1 \ldots n$ is a concatenation of Π^{id}. Let $\vec{\pi}$ be any desired concatenation of Π. We define $\omega_{\vec{\pi}}^{BP}(e)$ to be a function on the set of black edges in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ (i.e. for every e in $E_{\vec{\pi}}^{B}$), whose range is $\{-0.5, 0, 0.5\}$. Moreover, $\omega_{\vec{\pi}}^{BP}(e)$ is defined based on the standard orientation of $G_{\vec{\pi}}$ and G_{Π}. For each black edge $e = \{a, b\}$ in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$:

- $\omega_{\vec{\pi}}^{BP}(e) = 0.5$, if $e \in E_{\vec{\pi}}^{B} \cap E_{\Pi}^{B}$, and $d_{\vec{\pi}}(\{a, b\}) = d_{\Pi}(\{a, b\})$;
- $\omega_{\vec{\pi}}^{BP}(e) = -0.5$, if $e \in E_{\vec{\pi}}^{B} \cap E_{\Pi}^{B}$ and $d_{\vec{\pi}}(\{a, b\}) \neq d_{\Pi}(\{a, b\})$;
- $\omega_{\vec{\pi}}^{BP}(e) = 0$, if $e \in E_{\vec{\pi}}^{B} \setminus E_{\Pi}^{B}$.

In words, for a black e edge in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ that exists in both $G_{\vec{\pi}}$ and G_Π, we define $\omega^{BP}_{\vec{\pi}}(e) = 0.5$ if e has the same direction in the standard orientation of $G_{\vec{\pi}}$ and G_Π, and $\omega^{BP}_{\vec{\pi}}(e) = -0.5$, otherwise. And for every black edge e in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ that exists only in $G_{\vec{\pi}}$ and not in G_Π, $\omega^{BP}_{\vec{\pi}}(e) = 0$. The edges with assigned zero value are the result of concatenations. It is worth noting that this function is distinct from the labeling we used for distinguishing the oriented cycles from the unoriented ones. Furthermore, since Π and Π^{id} are co-tailed, thus every grey or black edge in the breakpoint graph for any concatenations $\vec{\pi}$ and $\vec{\pi}^{id}$ of Π and Π^{id} is either incident to vertices both in $V_\Pi \setminus \mathcal{T}(\Pi)$ or $\mathcal{T}(\Pi) \cup \{T_1, T_2\}$. As the result, the value of $\omega^{BP}_{\vec{\pi}}$ of the black edges belonging to each cycle of $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$, are either all non-zero in the former or all zero, in the latter case. We also assign a labeling function to the black edges of G_Π. To do so, we first label the non-telomeric black edges in G_Π, i.e. elements in $\widetilde{E}^B_\Pi$ or equivalently the black edges in $\widetilde{G}(\Pi)$. As mentioned previously, $|\widetilde{E}^B_\Pi| = n - M$, thus we define the labelling $\tilde{l}_\Pi$ to be a bijection from $\widetilde{E}^B_\Pi$ to $\{1, \ldots, n-M\}$. We then label the black edges in G_Π such that l_Π is a function from E^B_Π to $\{0, 1, \ldots, n-M\}$ where

$$l_\Pi(e) = \begin{cases} \tilde{l}_\Pi(e) & \text{if } e \in E^B_\Pi \cap \widetilde{E}^B_\Pi, \\ 0 & \text{if } e \in E^B_\Pi \setminus \widetilde{E}^B_\Pi. \end{cases}$$

This labeling assigns distinct labels to the black edges of G_Π corresponding to non-telomeric adjacencies in Π, while giving zero labels to all the edges corresponding to telomeric ones. Finally, we define a labeling function $\hat{l}_{\vec{\pi}}(e)$ for the black edges of $G_{\vec{\pi}}$ such that

$$\hat{l}_{\vec{\pi}}(e) = \begin{cases} \tilde{l}_\Pi(e) & \text{if } e \in E^B_{\vec{\pi}} \cap E^B_\Pi, \\ \hat{l}_{\vec{\pi},1}(e) & \text{if } e \in E^B_{\vec{\pi}} \setminus E^B_\Pi. \end{cases}$$

Where $\hat{l}_{\vec{\pi},1}$ is a bijection from $E^B_{\vec{\pi}} \setminus E^B_\Pi$ to $\{n - M + 1, \ldots, n + 1\}$. It is worth reminding that $|E^B_{\vec{\pi}}| = n + 1$, and $|E^B_{\vec{\pi}} \cap E^B_\Pi| = n - M$.

We choose two distinct black edges labelled by $\tilde{l}_\Pi(e_1)$ and $\tilde{l}_\Pi(e_2)$ in $\widetilde{G}(\Pi)$ (corresponding to two non-telomeric adjacencies in Π uniformly and at random. The labels $\tilde{l}_\Pi(e_1)$ and $\tilde{l}_\Pi(e_2)$ are in $\{1, \ldots, n - M\}$ and let $d_\Pi(e_1) = (a, b)$ and $d_\Pi(e_2) = (c, d)$. For $i \neq j \in \{1, \ldots, M\}$, one of the two possible cases happens:

- **Case 1:** $e_1, e_2 \in E^B_{\vec{\Pi}_i}$, i.e. edges e_1, e_2 are in the same chromosome of Π, which we denote by $e_1 \underset{\Pi}{\sim} e_2$. In this case the possible IRT operation is a reversal on the segment of the chromosome $\vec{\Pi}_i$ trapped between e_1 and e_2, or equivalently $G_{\vec{\Pi}_i} \circ \rho_{e_1,e_2}$.
- **Case 2:** $e_1 \in E^B_{\vec{\Pi}_i}$ and $e_2 \in E^B_{\vec{\Pi}_j}$, i.e. the edges e_1, e_2 are in two distinct chromosomes of Π, which we denote by $e_1 \underset{\Pi}{\nsim} e_2$. In this case the possible IRT operation is a translocation and is either $\delta^1_{e_1,e_2}$ or $\delta^2_{e_1,e_2}$ on G_Π.

We imitate IRTs on Π as reversals on $\vec{\pi}$, and we then update $\vec{\pi}$ and $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ accordingly:

Case 1: since $e_1, e_2 \in E^B_{\vec{\Pi}_i}$ they have the same direction both in $\vec{\pi}$ and Π, and $\omega^{BP}_{\vec{\pi}}(e_1) = \omega^{BP}_{\vec{\pi}}(e_1) \neq 0$. The appropriate IRT operation turns Π to Π', with graph representation

$$G_{\Pi'} = G_{\Pi} \circ \rho_{e_1,e_2},$$

such that

$$G_{\Pi'} = (V_{\Pi}, E^R_{\Pi} \cup E^B_{\Pi'}),$$

where

$$E^B_{\Pi'} = \Big(E^B_{\Pi} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}.$$

We can mimic this as a reversal on $\vec{\pi}$ transforming it into $\vec{\pi}'$ whose graph representation is $G_{\vec{\pi}'} = G_{\vec{\pi}} \circ \rho_{e_1,e_2}$ such that $G_{\vec{\pi}'} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}'})$, where

$$E^B_{\pi'} = \Big(E^B_{\pi} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}.$$

Note that, $\vec{\pi}'$ is a concatenation of Π'. We also obtain $G_{BP}(\vec{\pi}', \vec{\pi}^{id})$ from $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ by removing the black edges e_1 and e_2, and creating the new black edges via joining a to c, and b to d.

Case 2: since $e_1 \in E^B_{\vec{\Pi}_i}$ and $e_2 \in E^B_{\vec{\Pi}_j}$, where $i \neq j$, if e_1 and e_2 have the same direction in $\vec{\pi}$ and Π then $\omega^{BP}_{\vec{\pi}}(e_1) = \omega^{BP}_{\vec{\pi}}(e_2) \neq 0$, otherwise $\omega^{BP}_{\vec{\pi}}(e_1) = -\omega^{BP}_{\vec{\pi}}(e_2) \neq 0$.

Case 2, $(\delta^1_{e_1,e_2})$: in this case Π transforms to Π' whose graph representation is

$$G_{\Pi'} = G_{\Pi} \circ \delta^1_{e_1,e_2},$$

such that

$$G_{\Pi'} = (V_{\Pi}, E^R_{\Pi} \cup E^B_{\Pi'}),$$

where

$$E^B_{\Pi'} = \Big(E^B_{\Pi} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}.$$

Based on the values of $\omega^{BP}_{\vec{\pi}}(e_1)$ and $\omega^{BP}_{\vec{\pi}}(e_2)$, we can mimic the IRT operation on Π by one or at most two consecutive reversals on $\vec{\pi}$.

- If $\omega^{BP}_{\vec{\pi}}(e_1) = \omega^{BP}_{\vec{\pi}}(e_2) \neq 0$, $\delta^1_{e_1,e_2}$ on Π can be mimicked as ρ_{e_1,e_2} on $\vec{\pi}$, such that $G_{\Pi'} = G_{\Pi} \circ \delta^1_{e_1,e_2}$ and $G_{\vec{\pi}'} = G_{\vec{\pi}} \circ \rho_{e_1,e_2}$, and $\vec{\pi}'$ is a concatenation of Π'. Furthermore, one can obtain $G_{\vec{\pi}'} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}'})$, and $G_{BP}(\vec{\pi}', \vec{\pi}^{id})$ as explained in case 1.

- If $\omega^{BP}_{\vec{\pi}}(e_1) = -\omega^{BP}_{\vec{\pi}}(e_2) \neq 0$, $\delta^1_{e_1,e_2}$ on Π can not be mimicked as a single reversal ρ_{e_1,e_2} on $\vec{\pi}$. This is due to creation of a new circular chromosome in $\vec{\pi}$ by removing e_1 and e_2 and joining a to c and b to d. To solve this problem, we first need to apply an appropriate reversal on $\vec{\pi}$, and turning it to $\vec{\pi}_1$ which is another concatenation of Π such that e_1 and e_2 have the same direction in Π and $\vec{\pi}_1$. We can obtain such $\vec{\pi}_1$ by flipping either the chromosome $\vec{\Pi}_i$ or $\vec{\Pi}_j$ in $\vec{\pi}$, here we choose to flip $\vec{\Pi}_j$. Since $\vec{\Pi}_j$ is a linear chromosome in Π, so it has two tails, let $\mathcal{T}(\vec{\Pi}_j) = \{v_1, v_2\}$, each are incident to two edges $e_3,\ e_4 \in E^B_{\vec{\pi}}$, where $\omega^{BP}_{\vec{\pi}}(e_3) = \omega^{BP}_{\vec{\pi}}(e_4) = 0$. Let $x_1, x_2 \in V_{\vec{\pi}}$ such that $e_3 = \{x_1, v_1\}$ and $e_4 = \{x_2, v_2\}$.

  Since one of the v_1 and v_2 is the rightmost and the other the left most genes' extremities of $\vec{\Pi}_j$ in Π, thus either $d_{\vec{\pi}}(e_3) = (x_1, v_1)$ and $d_{\vec{\pi}}(e_4) = (v_2, x_2)$, or $d_{\vec{\pi}}(e_3) = (v_1, x_1)$ and $d_{\vec{\pi}}(e_4) = (x_2, v_2)$. In any case $G_{\vec{\pi}_1} = G_{\vec{\pi}} \circ \rho_{e_3,e_4}$ such that

$$G_{\vec{\pi}_1} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}_1}),$$

  where

$$E^B_{\vec{\pi}_1} = \Big(E^B_{\vec{\pi}} \setminus \{e_3, e_4\}\Big) \cup \Big\{\{x_1, v_2\}, \{v_1, x_2\}\Big\}.$$

  It is worth to point out that $\vec{\pi}_1$ is another concatenation of Π, where all the chromosomes are in the same order and orientation as in $\vec{\pi}$, except $\vec{\Pi}_j$ which is flipped. This results in $\omega^{BP}_{\vec{\pi}_1}(e_1) = \omega^{BP}_{\vec{\pi}_1}(e_2) \neq 0$. Now, $\delta^1_{e_1,e_2}$ on Π can be mimicked as a single a reversal ρ_{e_1,e_2} on $\vec{\pi}_1$. So $G_{\vec{\pi}'_1} = G_{\vec{\pi}_1} \circ \rho_{e_1,e_2}$, such that $G_{\vec{\pi}'_1} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}'_1})$, where $E^B_{\vec{\pi}'_1} = \Big(E^B_{\vec{\pi}_1} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, c\}, \{b, d\}\Big\}$, and we update $G_{BP}(\vec{\pi}'_1, \vec{\pi}^{id})$ by removing e_1 and e_2 in $G_{BP}(\vec{\pi}_1, \vec{\pi}^{id})$ and connecting a to c and b to d. Here, $\vec{\pi}'_1$ is a concatenation of Π'.

  Case 2, ($\delta^2_{e_1,e_2}$): in this case Π transforms to Π'' whose graph representation is

$$G_{\Pi''} = G_{\Pi} \circ \delta^2_{e_1,e_2},$$

  such that

$$G_{\Pi''} = (V_{\Pi}, E^R_{\Pi} \cup E^B_{\Pi''}),$$

  where

$$E^B_{\Pi''} = \Big(E^B_{\Pi} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, d\}, \{b, c\}\Big\}.$$

Based on the values of $\omega^{BP}_{\vec{\pi}}(e_1)$ and $\omega^{BP}_{\vec{\pi}}(e_2)$, we can mimic IRT operation on Π as one or at most two consecutive reversals on $\vec{\pi}$. Although, this time:

- If $\omega^{BP}_{\vec{\pi}}(e_1) = -\omega^{BP}_{\vec{\pi}}(e_2) \neq 0$, $\delta^2_{e_1,e_2}$ which transforms Π to Π'', can be mimicked as ρ_{e_1,e_2} on $\vec{\pi}$. To be more precise $G_{\vec{\pi}''} = G_{\vec{\pi}} \circ \rho_{e_1,e_2}$, where $G_{\vec{\pi}''} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}''})$,

and $E^B_{\vec{\pi}''} = \Big(E^B_{\vec{\pi}} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, d\}, \{b, c\}\Big\}$. Here, $\vec{\pi}''$ is a concatenation of Π''. We can update $G_{BP}(\vec{\pi}'', \vec{\pi}^{id})$ by removing black edges e_1 and e_2 in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ and connecting a to d and b to c.

- If $\omega^{BP}_{\vec{\pi}}(e_1) = \omega^{BP}_{\vec{\pi}}(e_2) \neq 0$, this time $\delta^2_{e_1,e_2}$ on Π can not be mimicked as a simple reversal on $\vec{\pi}$ due to creation of a circular chromosome by connecting a to d and b to c. Consequently, we need to flip one of the chromosomes either $\vec{\Pi}_i$ or $\vec{\Pi}_j$ by an appropriate reversal on $\vec{\pi}$ transforming it to $\vec{\pi}_1$. This results in $\omega^{BP}_{\vec{\pi}_1}(e_1) = -\omega^{BP}_{\vec{\pi}_1}(e_2) \neq 0$, where $\vec{\pi}_1$ is another concatenation of Π. Finally, $\delta^2_{e_1,e_2}$ on Π can be mimicked as a single a reversal ρ_{e_1,e_2} on $\vec{\pi}_1$ such that $G_{\vec{\pi}''_1} = G_{\vec{\pi}_1} \circ \rho_{e_1,e_2}$. Furthermore, $G_{\vec{\pi}''_1} = (V_{\vec{\pi}}, E^R_{\vec{\pi}} \cup E^B_{\vec{\pi}''_1})$, where $E^B_{\vec{\pi}''_1} = \Big(E^B_{\vec{\pi}_1} \setminus \{e_1, e_2\}\Big) \cup \Big\{\{a, d\}, \{b, c\}\Big\}$, and $\vec{\pi}''_1$ is a concatenation of Π''. We can obtain $G_{BP}(\vec{\pi}''_1, \vec{\pi}^{id})$ by removing e_1 and e_2 in $G_{BP}(\vec{\pi}_1, \vec{\pi}^{id})$ and connecting a to d and b to c.

The purpose of this explanation is to state while mimicking IRT on any concatenation of Π, chromosomal flips do not change $\mathcal{C}(\Pi)$ or $k(\Pi)$, where $\mathcal{C}(\Pi)$ is the number cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ with only intrachromosomal black and grey edges. It is worth reminding that for every concatenation $\vec{\pi}$ and $\vec{\pi}^{id}$ of Π and Π^{id}, the value of ω^{BP} for all the black edges in a cycle in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ are all nonzero (0.5 or -0.5) or all zero. We call the latter *zero cycles*, hereafter. Thus, every chromosomal flip is performed on two black edges in zero cycle(s). If those edges are in two different zero cycles, the reversal (chromosomal flip) merges their cycles into one zero cycle. And if those black edges belong to the same zero cycle, the cycle may split into two zero cycles, or may not split at all. There are not going to be any mimicking reversals on two black edges one having a zero label, and the other having a nonzero label. So, chromosomal flip does not change the number of $\mathcal{C}(\Pi)$. Furthermore, since the knots are made of intrachromosomal cycles of the breakpoint graph they do not interleave with any zero cycles. So the chromosomal flips do not affect the knots, as well.

3.3.1 IRT evolution and parsimonious paths

Let $\mathcal{G}_{n,M}$ be the space of all signed multichromosomal genomes with n genes, M linear chromosomes and no circular chromosomes. Let $G \in \mathcal{G}_{n,M}$ be the graph representation of a given genome with n genes scattered into M linear chromosomes. As stated earlier, G has $n - M$ black edges representing the non-telomeric adjacencies of the genome i.e. $|E^B_G \cap \widetilde{E}^B_G| = |\widetilde{E}^B_G| = n - M$. For every two distinct edges $i, j \in \widetilde{E}^B_G$, we denote $i \underset{G}{\sim} j$ if i and j are in the same chromosome of G, and $i \underset{G}{\nsim} j$, otherwise. Equivalently, there is an $ij-$path in G if and only if $i \underset{G}{\sim} j$. When $i \underset{G}{\sim} j$ then the only feasible IRT operation is a reversal $\rho_{i,j}$. On the other hand when $i \underset{G}{\nsim} j$, there

are two feasible IRT operations, either a translocation $\delta^1_{i,j}$ or a translocation $\delta^2_{i,j}$. Let $\mathcal{A} = \{(i,j)|\{i,j\} \in \widetilde{E}^B_G \text{ and } i \neq j\}$. Then the set of all operators on G can be described as

$$\Delta(G) = \{\rho_{i,j}|(i,j) \in \mathcal{A}, i \underset{G}{\sim} j\} \cup \{\delta^1_{i,j}|(i,j) \in \mathcal{A}, i \underset{G}{\nsim} j\} \cup \{\delta^2_{i,j}|(i,j) \in \mathcal{A}, i \underset{G}{\nsim} j\}.$$

Suppose that we have a probability distribution μ on $\mathcal{A}$ with weight $\mu_{i,j}$ on (i,j) such that

$$\sum_{(i,j)\in\mathcal{A}} \mu_{i,j} = 1,\ \mu_{i,j} \geq 0.$$

Let $\mathcal{G}_{n,M,G_0}$ be the space of graph representations of all the genomes with n genes, and M linear chromosomes co-tailed with G_0. Thus, $\mathcal{G}_{n,M,G_0}$ is inside $\mathcal{G}_{n,M}$. Every genome $G \in \mathcal{G}_{n,M,G_0}$ can be obtained by a finite sequence of operators on G_0. More precisely, for every $G \in \mathcal{G}_{n,M,G_0}$, there exists a finite sequence of operations $f_1, f_2, \ldots, f_m$ such that

$$G = G_0 \circ f_1 \circ \cdots \circ f_m,$$

where $f_1 \in \Delta(G_0)$, $f_2 \in \Delta(G_0 \circ f_1)$, ..., $f_m \in \Delta(G_0 \circ f_1 \circ \cdots \circ f_{m-1})$. We will model the genome evolution via IRT operations as a Markov chain with the following transition probabilities:

$$p(G,G') = \begin{cases} \mu_{ij} & G' = G \circ \rho_{ij} \quad \text{and } i \underset{G}{\sim} j, \\ \theta\mu_{ij} & G' = G \circ \delta^1_{ij} \quad \text{and } i \underset{G}{\nsim} j, \\ (1-\theta)\mu_{ij} & G' = G \circ \delta^2_{ij} \quad \text{and } i \underset{G}{\nsim} j, \\ 0 & \text{otherwise.} \end{cases}$$

We establish our IRT evolution model as a random process $(X_t)_{t\geq 0}$ on the space of graph representations of multichromosomal genomes with n genes and M linear chromosomes $\mathcal{G}_{n,M}$. More precisely, let $(X_t)_{t\geq 0}$ be a continuous-time Markov chain on $\mathcal{G}_{n,M}$, with transition probability $p(G,G')$ whose jump times are according to the Poisson process with rate 1. We are going to focus on a special case when $\mu_{i,j}$ is uniformly distributed on $\mathcal{A}$ and $\theta = 1/2$. That is the pair of distinct edges $\{i,j\}$ (corresponding to non-telomeric adjacencies of the genome) picked uniformly at random (without replacement) and if i and j are in different chromosomes the transposition $\delta^1_{i,j}$ and $\delta^2_{i,j}$ are chosen with the same probability. We start at $X_0 = G_0 \in \mathcal{G}_{n,M}$, and jump occurs with the Poisson rate 1. Upon a jump time t, an ordered pair of distinct non-telomeric adjacencies $(i,j) \in \widetilde{E}^B_{G_t}$ are chosen uniformly at random with probability $\frac{1}{|\mathcal{A}|}$, where $|\mathcal{A}| = (n-M)(n-M-1)$. The Markov chain jumps from G to $G \circ \rho_{i,j}$ if $i \underset{G}{\sim} j$, that is i and j belong to the same chromosome of G.

Whereas, if $i \underset{G}{\nsim} j$ then the Markov chain jumps from G to either $G \circ \delta^1_{i,j}$ or $G \circ \delta^2_{i,j}$ with probability $1/2$.

The IRT evolution model implies that starting with the initial genome $X_0 = G_0$ then $X_t \in \mathcal{G}_{n,M,G_0}$ for $t \geq 0$, this means the genomes at each step of the IRT evolution are co-tailed with G_0. Motivated by the method initially applied by Berestycki and Durret [4] we prove the parsimony bound for IRT evolution that is

$$d^{IRT}(X_t, X_0) \approx t,$$

holds up to time $n/2$, for large n. This means the IRT random process escapes from G_0 by getting one step away from G_0 at each jump up to $n/2$. One of the parameters in IRT distance formula $d^{IRT}(X_t, X_0)$ (Theorem 3.2.3) is the number of intrachromosomal cycles, i.e. $\mathcal{C}(X_t)$. These cycles exist in the breakpoint graph of any concatenations of X_t and X_0, and are made of only intrachromosomal edges corresponding to non-telomeric adjacencies of X_t and X_0. Each cycle in $\mathcal{C}(X_t)$ has the same content in any concatenation of X_t and X_0, although its orientation might differ. Another parameter of $d^{IRT}(X_t, X_0)$ is the number of knots denoted by $k(X_t)$. It is worth reminding that any intrachromosomal cycle of $\mathcal{C}(X_t)$ has the same orientation in the breakpoint graphs of any concatenations of X_t and X_0. Consequently, the knots in any concatenations of X_t and X_0 are the same.

Let (i, j) be two distinct non-telomeric adjacencies. Similar to the reversal, if i and j belong to two different cycles of the breakpoint graph, then applying an IRT on them merges their cycle into one. And if i and j lie in the same cycle of the breakpoint graph, then an IRT may split their cycle into two smaller cycles or may not. Thus, the evolution of the cycles that matter in the IRT distance formula depends only on the edges representing the non-telomeric adjacencies. Inspired by Berestycki and Durret [4], we leverage the properties of Erdös-Rényi random graph to study the evolution of such cycles in the breakpoint graph in order to develop similar results. We need to construct a random graph process, $(\widetilde{R}_t)_{t\geq 0}$.

First, we define a labelling process for the edges corresponding to non-telomeric adjacencies in X_t, denoted by $(\widetilde{L}_t)_{t\geq 0}$. For each $t \geq 0$, let $\widetilde{L}_t$ be a bijection from $\widetilde{E}^B_{X_t}$ to $\{1, \cdots, n - M\}$. Without loss of generality, let $\widetilde{L}_0 = \tilde{l}_{X_0}$. From exchangeability, the results in the next theorems do not really depend on any particular choice of $\tilde{l}_{X_0}$. Upon each jump t, we select $(i, j) \in \widetilde{E}^B_{X_t}$ uniformly at random such that $i = (a, b)$ and $j = (c, d)$. We remove i and j in X_t and define m_t to be one of the extremities $\{a, b, c, d\}$, whose corresponding gene has the smallest absolute value. For example, let $i = (5^-, 2^+)$ and $j = (3^+, 9^-)$ be the two selected edges at time t, then $m_t = 2^+$. We define $\widetilde{L}_{t^+}(e)$ for $e = (f, g) \in \widetilde{E}^B_{X_{t^+}}$, such that

$$\widetilde{L}_{t^+}(e) = \begin{cases} \widetilde{L}_t(e) & \text{if } \ e \in \widetilde{E}^B_{X_{t^+}} \cap \widetilde{E}^B_{X_t}, \\ \widetilde{L}_t(i) & \text{if } \ e \in \widetilde{E}^B_{X_{t^+}} \setminus \widetilde{E}^B_{X_t} \quad \textit{and} \quad m_t \notin \{f, g\}, \\ \widetilde{L}_t(j) & \text{if } \ e \in \widetilde{E}^B_{X_{t^+}} \setminus \widetilde{E}^B_{X_t} \quad \textit{and} \quad m_t \in \{f, g\}. \end{cases}$$

Based on the IRT random stochastic process and the labeling process, we construct a random graph process $(\widetilde{\mathcal{R}}_t)_{t\geq 0}$. We start with $\widetilde{\mathcal{R}}_0$, a graph with $n-M$ vertices and no edge. The vertices of $\widetilde{\mathcal{R}}_0$ are labelled by $1, 2, \ldots, n-M$ corresponding to the labels of the non-telomeric edges in the breakpoint graph of $G_{BP}(X_t, X_0)$. Let $\{i, j\} \in \widetilde{E}^B_{X_t}$ be the two distinct randomly picked non-telomeric edges in X_t, whose labels are $\{\widetilde{L}_t(i), \widetilde{L}_t(j)\}$. We connect $\widetilde{L}_t(i)$ to $\widetilde{L}_t(j)$ in $\widetilde{\mathcal{R}}_{t^+}$, if they are not already adjacent.

For $l_1, l_2 \in \{1, 2, \ldots, n-M\}$, the number of times an ordered distinct pair of edge labels (l_1, l_2) are chosen up to time t, is a Poisson random variable with the expected value $\frac{t}{|\mathcal{A}|}$, where $|\mathcal{A}| = (n-M)(n-M-1)$. Therefore, the probability of there existing an $\{l_1, l_2\}$ edge in $\widetilde{\mathcal{R}}_{cn/2}$ is

$$1 - e^{-\frac{cn}{(n-M)(n-M-1)}} \approx \frac{cn}{(n-M)(n-M-1)} + O(\tfrac{1}{n^2}).$$

We assume $\mathcal{R}^*$ to be an Erdös-Rényi random graph with $n-M$ vertices, and edge probability c/n. This means the probability of existing an edge between any two vertices of $\mathcal{R}^*$ is independent from the other edges and it equals to c/n. Thus, there is an $O(1)$ difference between the expected number of the edges in $\widetilde{\mathcal{R}}_{cn/2}$ and $\mathcal{R}^*$. Hence the number of the tree components of $\widetilde{\mathcal{R}}_{cn/2}$ can be estimated by the number of the tree components of $\mathcal{R}^*$.

Theorem 3.3.1. *(Theorem 5.12 in [8]) Let $c > 0$, $c \neq 1$, $\mathcal{R}^*$ be an Erdös-Rényi random graph with n vertices, and with edge probability c/n. We denote $\mathcal{T}_{cn/2}$ to be the number of tree components of $\widetilde{\mathcal{R}}_{cn/2}$. Then*

$$\mathbb{E}[\mathcal{T}_{cn/2}] = \alpha(c)(n) + O(1)$$

where

$$\alpha(c) = \tfrac{1}{c} \sum_{k=1}^{\infty} \tfrac{k^{k-2}}{k!}(ce^{-c})^k$$

and $var(\mathcal{T}_{cn/2}) = O(n)$. For an arbitrary sequence $(b_n)_{n\in\mathbb{N}}$ such that $b_n \to \infty$ as $n \to \infty$, we have

$$|\mathcal{T}_{cn/2} - \alpha(c)n| \leq b_n\sqrt{n}.$$

We can then easily see that the Erdös-Rényi random graph $\mathcal{R}^*$ approximates $\widetilde{\mathcal{R}}_{cn/2}$. By Chebyshev's inequality as $n \to \infty$, we have

$$\tfrac{\mathcal{T}_{cn/2} - \alpha(c)n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Furthermore, $1-\alpha(c)=\frac{c}{2}$ for $0<c<1$.

Let $G_{BP}(\vec{\pi},\vec{\pi}^{id})$ be the breakpoint graph of $\vec{\pi}$ and $\vec{\pi}^{id}$ which are concatenations of X_t and X_0, respectively. It is worth remembering that $k(X_t)$ represents the number of knots which are made of intrachromosomal unoriented interleaving cycles of $G_{BP}(\vec{\pi},\vec{\pi}^{id})$ for any concatenations $\vec{\pi}$ and $\vec{\pi}^{id}$. The interchromosomal cycles of G_{BP} have the same content for any concatenation of X_t and X_0, but their orientation depend to the chromosomal flip and the order in which the chromosomes are placed into the concatenations. This is not the case for the knots. For each knot, not only the contents but also the orientations of its cycles are the same and independent from the concatenations. In other words, cycles of the knots are sheltered against chromosomal flips or different order of the chromosomes.

It is worth emphasizing that if we choose any concatenation $\vec{\pi}$ for X_t, we can obtain the concatenation for X_{t^+} by applying the appropriate reversals on $\vec{\pi}$ (one or at most two) simulating the IRT, although if we have to apply two reversals one of them is just for flipping the chromosome which effects the number of zero cycles, and any IRT whose equivalent reversal increases or decreases or does not change the number of non-zero cycles of $G_{BP}(\vec{\pi},\vec{\pi}^{id})$ behaves similarly for any concatenation $\vec{\pi}$ of X_t. A reminder that Berestycki and Durret (2006) [4] proved that the transposition random walk is bound to parsimony up to time $n/2$. Jamshidpey and Sankoff (2013) [20] by following [4], and counting the number of hurdles of a reversal random walk, proved that the parsimony-binding holds for the reversal walk up to $n/2$, as well. In the upcoming theorem we follow the proof of [4] and also counting hurdles in [20] and extend it for counting knots.

Theorem 3.3.2. *Let c be a fixed positive number, and $(X_t)_{t>0}$ be a continuous-time IRT Markov chain on the space of signed multichromosomal genomes with n genes, M linear chromosomes and no circular chromosome as defined above. Let $k(X_t)$ be the number of knots in $G_{BP}(\vec{\pi},\vec{\pi}^{id})$, where $\vec{\pi}$ and $\vec{\pi}^{id}$ are any desired concatenations of X_t and X_0, respectively. Then*

$$\frac{k(X_{cn/2})}{b_n\sqrt{n}} \xrightarrow{p} 0,$$

where $(b_n)_{n\in\mathbb{N}}$ is an arbitrary sequence such that $b_n\to\infty$ as $n\to\infty$.

Proof: Each knot in the breakpoint graph is made of one or several unoriented intrachromosomal cycles. One or more of the cycles in each knot should have been involved with at least an IRT operation on two edges of the same cycle of the breakpoint graph, in their history. Otherwise, the component is oriented. We call such operations *non-merging* IRT operations, hereafter. A non-merging IRT operation either fragments the cycle into two smaller ones, or changes its orientation, or neither of those. To be more precise, the unoriented intrachromosomal cycles resulting from fragmentation are obtained via translocations (respectively, reversals) on two edges

of the larger interchromosomal (respectively, interchromosomal or intrachromosomal) cycles. Otherwise, the unoriented intrachromosomal cycle is created via reversals on two divergent edges belonging to an intrachromosomal cycle. It is worth reminding that a reversal on two divergent edges of the same cycle does not always create an unoriented cycle.

We follow Theorem 3 in [4] and Theorem 3 in [20]. For $t \geq 0$, let $\mathcal{C}_t$ be the set of cycles in $G_{BP}(\vec{\pi}, \vec{\pi}^{id})$ with non-telomeric edges, where $\vec{\pi}$ and $\vec{\pi}^{id}$ are any desired concatenations of X_t and X_0. It is worth remembering that, each cycle in $\mathcal{C}_t$ has an even number of edges with an equal number of black and grey edges. Hence, we have exactly $n - M$ black edges so the number of non-zero cycles in $\mathcal{C}_t$ with size greater than $2\sqrt{n-M}$ is less than $\sqrt{n-M}$. We define $\mathcal{C}_t^{nf}$ to be the set of cycles in $\mathcal{C}_t$ with size at most $2\sqrt{n-M}$ that have never been involved in a non-merging IRT operation. Besides, we define $\mathcal{C}_t^{f}$ to be the set of cycles in $\mathcal{C}_t$ with size at most $2\sqrt{n-M}$ that have experienced a non-merging IRT operation at least once. Note that the sets $\mathcal{C}_t$, $\mathcal{C}_t^{nf}$ and $\mathcal{C}_t^{f}$ have the same content for any concatenation of X_0 and X_t. It is worth remembering that the non-merging IRT are performed on two black edges i and j belonging to the same cycle of the breakpoint graph. The cycles of $\mathcal{C}_t^{nf}$ with size greater than 2 have been solely engaged in coagulation process, up to time t. The cycles of $\mathcal{C}_t^{nf}$ with length 2 never experienced any IRT, and the cycles of $\mathcal{C}_t^{f}$ might have been fragmented, or not. Consequently, the number of cycles in $\mathcal{C}_t^{f}$, i.e. $|\mathcal{C}_t^{f}|$, has an upper bound of two times the number of non-merging IRT operations up to time t.

Each knot has one to several cycles (which are intrachromosomal) that each either belong to $\mathcal{C}_t^{f}$ or the intrachromosomal unoriented cycles with length more that $2\sqrt{n-M}$. Thus, for $t \geq 0$, $k(X_t)$ is less than or equal to $|\mathcal{C}_t^{f}| + \sqrt{n-M}$. We need to find an upper bound for $|\mathcal{C}_t^{f}|$. Each cycle experienced a non-merging IRT operation with maximum size $2\sqrt{n-M}$ is either the result of fragmentation of a larger cycle or change of the orientation of a cycle with the same size (non-merging IRT). The former must have at most $\sqrt{n-M}$ black edges separating i from j in the cycle, while the latter requires the cycle itself to have at most $\sqrt{n-M}$ black edges. The probability that an ordered pair of edge labels belonging to the randomly selected edges (without replacement) at time t has the mentioned property is at most $2\sqrt{n-M}/(n-M-1)$. Therefore, the number of non-merging IRT operations resulting to the cycles of length at most $2\sqrt{n-M}$ up to time $cn/2$, is bounded by a Poisson random variable with parameter $cn\sqrt{n-M}/n - M - 1 \approx c\sqrt{n}$. As $n \to \infty$, $\frac{|\mathcal{C}^f_{cn/2}|}{b_n\sqrt{n}} \xrightarrow{p} 0$, and since $k(X_{cn/2}) \leq |\mathcal{C}^f_{cn/2}| + \sqrt{n-M}$, then $\frac{k(X_{cn/2})}{b_n\sqrt{n}} \xrightarrow{p} 0$. ∎

Theorem 3.3.3. *Let c be a fixed positive number, $(X_t)_{t\geq 0}$ be a continuous-time IRT Markov Chain on the space of signed multichromosomal genomes with n genes and M linear chromosomes and no circular chromosomes as defined above, and let d^{IRT}*

be the IRT-distance, $(b_n)_{n\in\mathbb{N}}$ *be an arbitrary sequence diverging to* ∞*, and* $n \to \infty$ *then*

$$\frac{d^{IRT}(X_{cn/2},X_0)-(1-\alpha(c))n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Proof: We follow Theorem 3 in [4] and apply it for the IRT random stochastic process. We use Theorem 3.2.3 for the IRT distance formula. In Theorem 3.3.2, we proved $\frac{k(X_{cn/2})}{b_n\sqrt{n}} \xrightarrow{p} 0$. Clearly, $\frac{f(X_{cn/2})}{b_n\sqrt{n}} \to 0$, since $f(X_{cn/2})$ is either 0 or 1. As we have discussed in the proof of Theorem 3.3.2, at time $t = cn/2$ the number of cycles of length greater than $2\sqrt{n-M}$ is bounded by $\sqrt{n-M}$ and the number of cycles of length at most $2\sqrt{n-M}$ is $|\mathcal{C}^f_{cn/2}| + |\mathcal{C}^{nf}_{cn/2}|$ where $\frac{|\mathcal{C}^f_{cn/2}|}{b_n\sqrt{n}} \xrightarrow{p} 0$. The number of the tree components of $\widetilde{\mathcal{R}}_{cn/2}$ corresponds to the number of the cycles of the breakpoint graph such that either they have never experienced a non-merging IRT operation up to time $cn/2$, or they have experienced a non-merging IRT operation on a pair of edges whose labels have been selected before. Occurrence of the latter is negligible (proof is similar to unrestricted DCJ models). So the number of tree components of $\widetilde{\mathcal{R}}_{cn/2}$ estimates $|\mathcal{C}^{nf}_{cn/2}|$, so we have

$$\frac{\mathcal{T}_{cn/2}-|\mathcal{C}^{nf}_{cn/2}|}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Theorem 3.3.1 yields

$$\frac{|\mathcal{C}^{nf}_{cn/2}|-\alpha(c)n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Therefore, we have

$$\frac{d^{IRT}(X_{cn/2},X_0)-(1-\alpha(c))n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

∎

Since for $0 < c < 1$, $1-\alpha(c) = \frac{c}{2}$, consequently

$$\frac{d^{IRT}(X_{cn/2},X_0)-\frac{cn}{2}}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

This means an IRT evolution process is parsimony-bound up to $t \approx n/2$, where n represents the number of genes in the genome.

3.4 General reversal and translocation (HP)

In this section we study the HP model of genomic evolution. We will prove that similar to the reversal, DCJ and IRT random stochastic processes, the HP stochastic process does not escape from its parsimonious path before a certain time. More precisely, this means for a large n, a multichromosomal genome with n genes under the HP operation evolves around its parsimonious path of evolution up to time $n/2$. A quick reminder that an HP operation on a multichromosomal genome Π, either reverses a segment of a chromosome in Π, or involves two chromosomes of Π in a translocation operation. No circular chromosome is created as the result of the HP operations, although in the case of fusion two chromosomes get merged and create a longer linear chromosome, and an empty chromosome. In addition, unlike the IRT operations neither the reversal nor the translocation operations in HP is required to be internal. Let Π and Π^{id} be two signed multichromosomal genomes, with the same set of n genes scattered into linear chromosomes. One can obtain Π form Π^{id} by a finite sequence of HP operations, where Π and Π^{id} are not necessarily co-tailed. The minimum number of such operations that transform Π to Π^{id} is the HP distance between Π and Π^{id}, i.e. $d^{HP}(\Pi, \Pi^{id})$. The parameters of $d^{HP}(\Pi, \Pi^{id})$ are attained from the breakpoint graph $G_{BP}(\Pi, \Pi^{id})$ which is obtained by removing certain edges from the breakpoint graph of any capped concatenations of Π and Π^{id}, as explain previously.

By capping the chromosomes of a genome, we artificially add fake genes to both ends of each chromosome. These added fake genes enable us to mimic the HP operations on the original genome as IRT operations on any capping of the genome. To be more precise, let Π be a signed multichromosomal genome with n genes and M linear chromosome, and no circular chromosomes. Let $\widetilde{\Pi}$ be any desired capping for Π, and $\tilde{\pi}$ be a random concatenation of $\widetilde{\Pi}$. Then the HP operations on Π can be mimicked as the IRT operations on $\widetilde{\Pi}$, which can be imitated as the reversals on $\tilde{\pi}$. In other words, by applying the IRT operations on $\widetilde{\Pi}$, which can be simulated as the HP operation on Π and transforms Π into a genome not necessary co-tiled with its ancestor.

We can also obtain the graph representation of the original genome from the graph representation of any of its capping, and vice versa. The following explains this in more details.

HP-IRT Frankenstein

We call the following explanation HP-IRT Frankenstein since we explain how to cap and uncap genomes, and also how to mimic an HP operation on a genome as an IRT operation on its corresponding capping and vice versa. Let $G_{\widetilde{\Pi}}$ be the graph representation of $\widetilde{\Pi}$ a capping of the genome Π. For $1 \leq i \leq 2M$, let c_i be a cap in $\widetilde{\Pi}$,

such that it has two extremities c_i^+ and c_i^- in $G_{\widetilde{\Pi}}$. To obtain the graph representation of Π i.e. G_Π from $G_{\widetilde{\Pi}}$, we apply the following steps for each i, where $1 \leq i \leq 2M$:

Step 1: Let e_i be the red edge connecting c_i^- and c_i^+ in $G_{\widetilde{\Pi}}$. We obtain $G_{\widetilde{\Pi}}/e_i$ by contracting the edge e_i in $G_{\widetilde{\Pi}}$. A reminder that contracting an edge in a graph consists of deleting that edge and identifying its ends. Here by contracting e_i we identify c_i^- and c_i^+ into c_i. The vertex c_i in $G_{\widetilde{\Pi}}/e_i$ is adjacent with two black edges, and one of its neighbours is a telomere T_i, and the other one is an original tail of Π.

Step 2: Let e_i' be the black edge connecting c_i to the telomere T_i, we contract e_i' and name their identified vertex T_i.

Furthermore, based on the capping $\widetilde{\Pi}$ of Π, we can obtain $G_{\widetilde{\Pi}}$ from G_Π, by the following three steps. A reminder that since all the chromosomes of Π are linear each are being represented by an alternating paths in G_Π. Each of those paths starts and ends with two distinct telomeres. The following steps explain how to get $G_{\widetilde{\Pi}}$ which is a graph representation of a desired capping of Π from G_Π.

Step1: We delete all the black edges connecting the tails of Π to the telomeres, that means all the edges in $E_\Pi^B \setminus \widetilde{E}_\Pi^B$. There are $2M$ such edges.

Step 2: For $1 \leq i \leq 2M$, each removed black edge $e_i = \{T_i, a_i\}$ was incident to telomere T_i, and $\Pi-$tail a_i. With respect to the capping we insert two vertices c_i^+ and c_i^-, representing the two extremities for each cap c_i instead of e_i. Based on the sign of c_i, we connect one of the c_i^+ or c_i^- to T_i and the other one to a_i. These new edges are coloured in black.

Step 3 Finally, for $1 \leq i \leq 2M$, we connect each c_i^+ to c_i^- via a red edge.

Assigning a capping to genome Π enables us to simulate HP operations on Π as IRT operations on $\widetilde{\Pi}$. Assigning different capping is just giving different labels to the added fake genes.

The graph representation of genome Π (G_Π) has $n+M$ black edges representing both telomeric and non-telomeric adjacencies in Π. The graph representation of genome $\widetilde{\Pi}$ ($G_{\widetilde{\Pi}}$) has $n+3M$ black edges, $n+M$ of which are non-telomeric. Moreover, each edge belonging to $2M$ telomeric edges in $G_{\widetilde{\Pi}}$, is incident to exactly one extremity of a distinct cap. Thus, G_Π and $G_{\widetilde{\Pi}}$ have $n-M$ common edges which are the non-telomeric adjacencies of Π. For $1 \leq j \leq 2M$, each edge e_j in $2M$ telomeric adjacencies of G_Π is incident to a tail of Π and a telomere T_j. For each such e_j there exists a corresponding non-telomeric adjacencies of $G_{\widetilde{\Pi}}$ which is adjacent to the same tail of Π and an extremity of a cap, while the other extremity of that cap is adjacent to T_j. Thus, when we choose any two black edges e_1, e_2 in G_Π and apply an HP operation on them, we can mimic this operation as an IRT operation on $G_{\widetilde{\Pi}}$ such that:

- If e_1 and e_2 are both non-telomeric in G_Π i.e. they both belong to $\widetilde{E}_\Pi^B \cap \widetilde{E}_{\widetilde{\Pi}}^B$, we simulate an HP operation on e_1 and e_2 in G_Π as an IRT operation on e_1 and e_2 in $G_{\widetilde{\Pi}}$.

- If e_1 and e_2 are both telomeric in G_Π i.e. both of them belong to $E_\Pi^B \setminus \widetilde{E}_\Pi^B$. This

means $e_1 = \{a_1, b_1\}$ and $e_2 = \{a_2, b_2\}$ where a_1 and a_2 belong to the set of tails of Π and b_1 and b_2 are telomeres. We simulate an HP operation on e_1 and e_2 in G_Π as an IRT operation on the corresponding non-telomeric edges e'_1 and e'_2 in $G_{\widetilde{\Pi}}$, such that e'_1 is incident to a_1 (a tail of Π) and a cap extremity (incident to telomere b_1) in $G_{\widetilde{\Pi}}$, and e'_2 is incident to a_2 and another cap extremity (incident to telomere b_2) in $G_{\widetilde{\Pi}}$.

- If e_1 is a non-telomeric, and e_2 is a telomeric edge in G_Π i.e. $e_1 \in \widetilde{E}^B_\Pi \cap \widetilde{E}^B_{\widetilde{\Pi}}$ and $e_2 \in E^B_\Pi \setminus \widetilde{E}^B_\Pi$, such that $e_2 = \{a_2, b_2\}$ where a_2 is a tail of Π and b_2 is a telomere in Π. In this case we can mimic an HP operation on e_1 and e_2 in G_Π as an IRT operation on e_1 and e'_2 in $G_{\widetilde{\Pi}}$, where e'_2 is incident to a_2 and a cap extremity (incident to telomer b_2) in $G_{\widetilde{\Pi}}$.

Figure 3.3 shows an example of IRT-HP Frankenstein for a given genome. Part a, shows how to cap genome

$$\Pi = 1\ -7\ 4\ -8,\ -6\ 3\ 2\ -5,$$

into

$$\widetilde{\Pi} = c_1\ 1\ -7\ 4\ -8\ c_2,\ c_3\ -6\ 3\ 2\ -5\ c_4,$$

and uncap $\widetilde{\Pi}$ into Π, where c_1, c_2, c_3 and c_4 are the caps and can be any positive integers except $\{1, 2, \ldots, 8\}$. Part b, is an HP operation on two edges $\{1^+, 7^+\}$ and $\{4^+, 8^+\}$ in Π and its corresponding IRT operation on $\widetilde{\Pi}$. Part c, is an HP operation on two edges $\{7^-, 4^-\}$ and $\{3^+, 2^-\}$ in Π and its corresponding IRT operation on $\widetilde{\Pi}$. Part d, is an HP operation on two edges $\{T_1, 1^-\}$ and $\{6^-, 3^-\}$ in Π and its corresponding IRT operation on $\widetilde{\Pi}$. Part e, is an HP operation on two edges $\{8^-, T_2\}$ and $\{5^-, T_4\}$ in Π and its corresponding IRT operation on $\widetilde{\Pi}$.

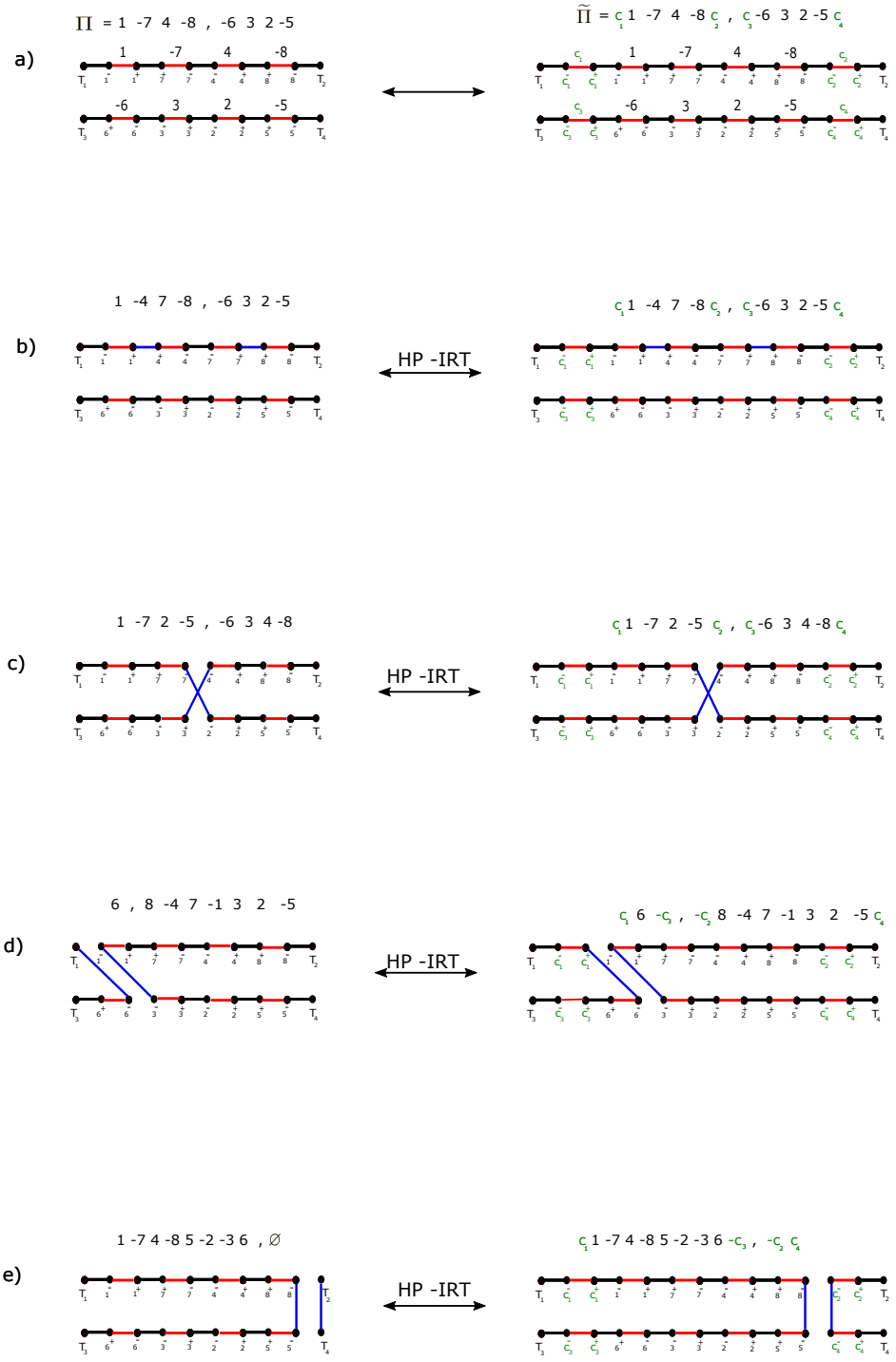

Figure 3.3: HP-IRT Frankenstein

3.4.1 HP evolution and parsimonious paths

In Mathematical language we have an operator Φ to embed the space of multichromosomal genomes with n genes and M linear chromosomes into the space of multichromosomal genomes with $n+2M$ genes and M linear chromosomes, that is $\Phi : \mathcal{G}_{n,M} \to \mathcal{G}_{n+2M,M}$ such that $\Phi(G_\Pi) = G_{\widetilde{\Pi}}$. We provided an operator Ψ from the space of all HP operators on the multichromosomal genomes with n genes and M linear chromosomes into the space of all IRT operators on the multichromosomal genomes with $n+2M$ genes and M linear chromosomes (for more details see HP-IRT Frankenstein).

The advantage of this is every transition probabilities on HP can be extended to transition probabilities on IRT via Φ embeddings, which implies that every stochastic process on HP operators on $\mathcal{G}_{n,M}$ will carry on to an stochastic process on IRT operators on $\mathcal{G}_{n+2M,M}$. We establish our HP model of evolution to be a continuous-time Markov chain $(Y_t)_{t\geq 0}$ on $\mathcal{G}_{n,M}$ transforming into its corresponding IRT model (a continuous-time Markov chain on $\mathcal{G}_{n+2M,M}$, as defined in in Subsection 3.3.1). Therefore, transforming stochastic HP models to IRT models enables us to provide an analogue of Theorem 3.3.2 and 3.3.3 in the context of HP models. In the upcoming theorem we follow the theorem of counting hurdles (Theorem 3) in [20] and Theorem 3.3.2 and extend it for counting real-knots and semi-real-knots.

Theorem 3.4.1. *Let $c > 0$ be a fixed number, and $(Y_t)_{t\geq 0}$ be a continuous-time HP Markov chain on the space of signed multichromosomal genomes with n genes, M linear chromosomes and no circular chromosome, as defined above. Let $r(Y_t, Y_0)$ and $s'(Y_t, Y_0)$ be the numbers of real knots and semi-real-knots in $G_{BP}(Y_t, Y_0)$, respectively. Then*

$$\frac{r(Y_{cn/2},Y_0)+s'(Y_{cn/2},Y_0)}{b_n\sqrt{n}} \xrightarrow{p} 0,$$

where $(b_n)_{n\in\mathbb{N}}$ is an arbitrary sequence of real numbers such that $b_n \to \infty$ as $n \to \infty$.

Proof: Following [4], [20] and Theorem 3.3.2, let $\widetilde{Y}_0$ be any desired capping of Y_0, and $\widetilde{Y}_t$ be a capping of Y_t obtained after t corresponding IRT operation on $\widetilde{Y}_0$. Thus, $\widetilde{Y}_0$ and $\widetilde{Y}_t$, are two co-tailed multichromosomal genomes with $n+2M$ genes scattered into M linear chromosomes. We can obtain the parameters $r(Y_t, Y_0)$ and $s'(Y_t, Y_0)$ from the breakpoint graph of $G_{BP}(Y_t, Y_0)$ which we get by removing $2M$ grey edges connecting Y_0-tails to Y_t-caps in $G_{BP}(\widetilde{Y}_t, \widetilde{Y}_0)$ and the $M+1$ black edges belonging to the zero cycles ($M-1$ of which the result of concatenation, plus the two telomeric edges) and the gray edges incident to them. The breakpoint graph of $G_{BP}(Y_t, Y_0)$ has M paths and $n+M$ black edges (corresponding to non-telomeric edges in in $G_{BP}(\widetilde{Y}_t, \widetilde{Y}_0)$). Let π and π^{id} be any concatenations of $\widetilde{Y}_t$ and $\widetilde{Y}_0$, respectively. The number of non-zero cycles of size greater than $2\sqrt{n+M}$ in $G_{BP}(\pi, \pi^{id})$ has an upper bound of $\sqrt{n+M}$. Let $\widetilde{\mathcal{C}}_t^f$ be the number of cycles of size at most $2\sqrt{n+M}$ in

$G_{BP}(\pi, \pi^{id})$ that experienced at least one non-merging IRT operation up to time t. Applying Theorem 3.3.2 yields to $\frac{\widetilde{\mathcal{C}}^f_{cn/2}}{b_n\sqrt{n}} \xrightarrow{p} 0$, as $n \to \infty$.

Furthermore, the intrachromosomal cycles in $G_{BP}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0)$ (which are the same for any concatenation of $\widetilde{Y}_{cn/2}$ and $\widetilde{Y}_0$) which have no grey edge connecting Y_0-tails to $Y_{cn/2}-$caps keep their orientations, while removing such grey edges from the cycles containing $\{Y_0 - \text{tails}, Y_{cn/2} - \text{caps}\}$ edges could create paths with different orientations. More precisely, any oriented or unoriented cycle containing exactly one $\{Y_0 - \text{tails}, Y_{cn/2} - \text{caps}\}$ grey edge becomes an oriented or unoriented path after removing such edge. So, if an oriented or unoriented cycle (intrachromosomal or interchromosomal) has α grey $\{Y_0-\text{tails}, Y_{cn/2}-\text{caps}\}$ edges, where $\alpha \in \{2, 3, \cdots, 2M\}$, it can create up to α unoriented intrachromosomal paths. Let $\mathcal{C}^*_t$ and $\mathcal{P}^*_t$ be the number of intrachromosomal unoriented cycles and paths in $G_{BP}(Y_t, Y_0)$, hence

$$\mathcal{C}^*_{cn/2} + \mathcal{P}^*_{cn/2} \leq \widetilde{\mathcal{C}}^f_{cn/2} + 4M + \sqrt{n+M}.$$

Real knots and semi-real knots are made of unoriented intrachromosomal cycles and paths in $G_{BP}(Y_t, Y_0)$, respectively. Although, each real knot does not have any Y_0Y_t-path in its interval, every semi-real knot has at least one. Consequently,

$$r(Y_{cn/2}, Y_0) + s'(Y_{cn/2}, Y_0) \leq \mathcal{C}^*_{cn/2} + \mathcal{P}^*_{cn/2} + \sqrt{n+M} \leq \widetilde{\mathcal{C}}_{cn/2} + 4M + \sqrt{n+M},$$

thus, for $n \to \infty$

$$\frac{r(Y_{cn/2}, Y_0) + s'(Y_{cn/2}, Y_0)}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

∎

Theorem 3.4.2. *Let c be a fixed positive number, $(Y_t)_{t\geq 0}$ be a continuous-time HP Markov chain on the space of signed multichromosomal genomes with n genes and M linear chromosomes as defined above. Let d^{HP} be the HP-distance, and $(b_n)_{n\in\mathbb{N}}$ be an arbitrary sequence diverging to ∞, then*

$$\frac{d^{HP}(Y_{cn/2}, Y_0) - (1-\alpha(c))n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Proof: We follow Theorem 3 in [4] and adjust it for HP model. Based on the Theorem 3.2.4

$$\begin{aligned} d^{HP}(Y_{cn/2}, Y_0) = \quad & n + M - c(Y_{cn/2}, Y_0) + p(Y_{cn/2}, Y_0) + r(Y_{cn/2}, Y_0) \\ & + \left\lceil \frac{s'(Y_{cn/2}, Y_0) - gr'(Y_{cn/2}, Y_0) + fr'(Y_{cn/2}, Y_0)}{2} \right\rceil. \end{aligned}$$

Since there are at most $2M$ paths, so clearly $p(Y_{cn/2}, Y_0) \leq 2M$. Furthermore, $gr'(Y_{cn/2}, Y_0)$ and $fr'(Y_{cn/2}, Y_0)$ can be either 0 or 1, thus by Theorem 3.4.1 as $n \to \infty$

$$\frac{p(Y_{cn/2},Y_0)+r(Y_{cn/2},Y_0)+\left\lceil \frac{s'(Y_{cn/2},Y_0)-gr'(Y_{cn/2},Y_0)+fr'(Y_{cn/2},Y_0)}{2} \right\rceil}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

We obtain the breakpoint graph of $G_{BP}(Y_{cn/2}, Y_0)$ by removing grey $\{Y_0-$tails$, Y_{cn/2}-$caps$\}$ edges in $G_{BP}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0)$. Moreover, it is worth reminding that $\mathcal{C}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0)$ represents the number of non-zero cycles in $G_{BP}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0)$ and $c(Y_{cn/2}, Y_0)$ represents the number of cycles and paths in $G_{BP}(Y_{cn/2}, Y_0)$, so

$$\mathcal{C}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0) \leq c(Y_{cn/2}, Y_0) \leq \mathcal{C}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0) + 4M.$$

The lower bound happens when all the grey edges connecting $\widetilde{Y}_0-$tails to $\widetilde{Y}_{cn/2}-$caps in $G_{BP}(\widetilde{Y}_{cn/2}, \widetilde{Y}_0)$ are in distinct cycles. It is worth remembering that $\widetilde{Y}_0, \widetilde{Y}_{cn/2} \in \mathcal{G}_{n+2M,M}$, where $\widetilde{Y}_0$ and $\widetilde{Y}_{cn/2}$ are cappings of Y_0 and $Y_{cn/2}$ such that each $\widetilde{Y}_t$ is obtained by t equivalent IRT operations on $\widetilde{Y}_0$. By Theorem 3.3.3 as $n \to \infty$

$$\frac{\mathcal{C}(\widetilde{Y}_{cn/2},\widetilde{Y}_0)-\alpha(c)n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

Therefore, by squeeze theorem we have

$$\frac{c(Y_{cn/2},Y_0)-\alpha(c)n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

So, as $n \to \infty$ we have

$$\frac{d^{HP}(Y_{cn/2},Y_0)-(1-\alpha(c))n}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

∎

Furthermore, since for $0 < c < 1$, $1 - \alpha(c) = \frac{c}{2}$, consequently

$$\frac{d^{HP}(Y_{cn/2},Y_0)-\frac{cn}{2}}{b_n\sqrt{n}} \xrightarrow{p} 0.$$

This means the HP model does not escape from its parsimonious path of evolution before $t \approx n/2$, for genomes with n genes.

3.5 Translocation evolution process

The translocation evolution process is in fact a more restricted version of the DCJ (Chapter 2) or the HP processes, in which upon each jump no two pair of black edges belonging to the same chromosome of the genome are allowed to be chosen.

A reminder that the translocation distance between two signed multichromosomal genomes is the minimum number of translocation operations converting either of the genomes to the other one. Theorem 3.5.1 represents the translocation distance formula between two multichromosomal genomes formulated by Hannenhalli [16].

Theorem 3.5.1. *[16][5][13] Let* Π *and* Π^{id} *be two signed multichromosomal genomes with* n *genes and* N *linear chromosomes, where* Π^{id} *represents the identity (ancestral) genome. The translocation distance between* Π *and* Π^{id} *denoted by* $d^{td}(\Pi, \Pi^{id}) = d^{td}(\Pi)$ *is*

$$d^{td}(\Pi) = n - N - \mathcal{C}(\Pi) + |\mathcal{S}| + o + 2b,$$

where $\mathcal{C}(\Pi)$ *represents the number of cycles in* $G_{BP}(\Pi, \Pi^{id})$*,* $\mathcal{S}$ *represents the set of all minimal subpermutations of* Π *, and* $|\mathcal{S}|$ *is the number of such subpermutations. If* $|\mathcal{S}|$ *is even then* $o = 0$*, otherwise* $o = 1$*. Moreover,* $b = 1$ *if there are even number of minimal subpermutations (i.e.* $o = 0$*), all of which are inside a single subpermutations, while residing in a single chromosome of* Π*,and* $b = 0$ *otherwise.*

By modeling the evolution as a random translocation process and measuring the distance between the starting point and any other points on a sample path of this model via translocation distance, we can investigate the time when the evolution escapes from parsimony. Although, adopting the previous methods for finding the maximum period in which this model is parsimony-bound is more challenging than before. In fact, the restriction on the choice of edges at each jump time, makes the constructed random graph corresponding to the translocation process not to be an Erdös-Rényi. Furthermore, the parameter $|\mathcal{S}|$, in the distance formula increases the complexity of the problem.

For the reasons mentioned above, we simulated the translocation model, see Chapter 4 for more details. The simulation results shows that the random translocation process evolves along its parsimonious path of evolution at least up to $t \approx n/2$, where n represents the number of genes in the genome.

3.6 Discussion

In this chapter, we have adapted the analysis of Berestycki and Durrett [4], originally developed for mathematical transposition random walk on permutations, to the biologically relevant IRT and HP operations (more restricted than DCJ operations). We

have shown that as n gets large, the parsimonious IRT and HP inference algorithms accurately reflect the number of evolutionary steps traced by evolutionary processes as long as this remains less than $n/2$, where n represents the number of genes.

Chapter 4

Simulation

In this chapter we present the simulation studies of different models of genomic evolution and investigate the time up to which they evolve along their parsimonious paths of evolution. Let $\mathcal{G}$ be the space of genomes and $\mathcal{A}$ be the set of genomic operations allowed in the model of evolution. For every two genomes $G_1, G_2 \in \mathcal{G}$ there exists a finite sequence of operations $\sigma_1, \sigma_2, \ldots, \sigma_s \in \mathcal{A}$ where

$$G_2 = G_1 \circ \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_s.$$

The edit distance ρ between any pair of genomes in $\mathcal{G}$ with respect to $\mathcal{A}$ is the minimum number of operations in $\mathcal{A}$ that transforms either of the two genomes into another.

We study the evolution of genomes under different types of rearrangement operations (such as: reversals, DCJs, IRTs, HPs, WRs, MLRs, translocations) by modeling genome evolution as continuous-time or discrete-time stochastic processes on the space of genomes $\mathcal{G}$. Here, we analyzed different models of genome evolution processes via simulation and by modeling them as discrete-time Markov chains $\{X_t : t = 0, 1, 2, \ldots\}$ on the countable sample space $\mathcal{G}$ with transition probability (based on the model) $\mathbf{P}(G, G')$ for every $G, G' \in \mathcal{G}$. For each model, we investigate the time up to which its evolutionary trajectory is parsimony-bound, which depends on not only the model but also the distance measure comparing points on the sample path to the starting point.

It is worth noting that, the discrete-time and continuous-time processes we used for modeling the genomic evolution here behave similarly. Let N_t be the number of jumps up to time t, for $t = cn$, where $c > 0$ and $n \to \infty$. By the Central limit theorem (CLT) we have

$$\frac{N_{cn} - cn}{\sqrt{cn}} \to \mathcal{N}(0, 1).$$

Then, for an arbitrary sequence $(b_n)_{n\geq 1}$ where $b_n \to \infty$ as $n \to \infty$

$$\frac{N_{cn} - cn}{b_n\sqrt{n}} \xrightarrow{p} 0,$$

which means after an appropriate rescaling the number of jumps up to time cn is approximately cn.

We simulated different types of stochastic processes on the space of signed genomes sharing the same set of genes with no duplication. Inspired by the simulation study of Bourque and Pevzner [9], Berestycki and Durret [4] proved that the random transposition walk on a permutation of length n (representing a unichromosomal genome with n genes) does not escape from its parsimonious path of evolution before $n/2$ steps. In fact, Berestycki and Durret [4] found a relation between the speed of escape of the random transposition walk and the number of tree components of an Erdös-Rényi random graph. Later on, inspired by Berestycki and Durret [4], Jamshidpey and Sankoff [20] proved that the reversal random walk does not escape from parsimony up to the same time. Analogous to the transposition random walk, the corresponding random graph for the reversal random walk is Erdös-Rényi, as well. In the previous chapters, we proved similar results for DCJ, IRT and HP stochastic processes. Although, the questions about other stochastic processes with some sorts of restrictions are still open since their corresponding random graphs are more complicated than Erdös-Rényi, consequently, the numbers of tree components in their random graphs remain unknown. These models include weighted reversal, maximum length reversal, and translocation. Here we employed simulation to study various stochastic processes modeling the evolution and for each model we compare the number of genomic operations along its evolutionary trajectory to the distance measure (based on the model reversal, DCJ, HP, etc) of the genome from its ancestor.

4.1 Different models of reversal genome evolution process

First, we simulate different reversal stochastic processes with additional restrictions on the space of signed unichromosomal genomes and compare them with the reversal random walk on the same space. In addition, for each process and at each time t we find the average number of tree components of its corresponding random graph and the average number of cycles in breakpoint graph of the genome at time t and the ancestor at time $t = 0$.

For each model, we simulate $\{X_t : t = 0, 1, 2, \ldots\}$ which is a reversal (with special properties) stochastic process on the space of signed unichromosomal genomes with n genes and no circular chromosomes. Without loss of generality, we always assume the stochastic processes starts at the identity genome represented by the identity

permutation $X_0 = 1 \ \cdots \ n$, where X_0 is a linear chromosome. We then add 0 and $n+1$, to both ends of its permutation corresponding to the two telomeres of the linear chromosome. As mentioned previously, we can present each chromosome with a graph representation as well. For example the graph representation of a linear chromosome $X_0 = 1 \cdots n$ has $2n+2$ vertices. We represent the extremities of each gene $i \in \{1, \ldots, n\}$ by two vertices $2i-1$ and $2i$, which got connected via a red edge. The two vertices labelled by 0 and $2n+1$ represent the telomers in X_0. Furthermore, for $i \in \{0, 1, \ldots, n\}$, we connect the pair of vertices $\{2i, 2i+1\}$ via a black edge representing the adjacencies among the genes in X_0. The two black edges $\{0, 1\}$ and $\{2n, 2n+1\}$ are telomeric adjacencies i.e. they are adjacent to vertices representing the telomeres. Note that the graph representation of X_0 is an edge alternating path.

Moreover, in order to study the number of tree components in the random graph $\mathcal{G}_t$ where $t = 0, 1, 2, \ldots$ we define a labelling process for the black edges of X_t. The labelling process starts by assigning the labels $\{1, \ldots, n+1\}$ to all the black edges of X_0, such that for each $i \in \{0, 1, \ldots, n\}$ the black edge $\{2i, 2i+1\}$ is labelled by $i+1$, although after each jump the edge labels get updated as well, which is similar to the labelling process explained in Chapters 2 and 3. Following [4], we define the random graph $\mathcal{G}_t$ such that $\mathcal{G}_0$ has $n+1$ vertices labelled by $\{1, 2, \ldots, n+1\}$ and no edge. We simulate the stochastic process such that at each jump time $t \geq 1$, we pick two black edges, and apply a reversal operation between them, we then connect their corresponding vertices in $\mathcal{G}_t$.

For example for a jump time t, let $e = \{a, b\}$ and $e' = \{c, d\}$ be two black edges where a, b, c, d are vertices representing the extremities in X_{t-1}, applying a reversal between e and e' on X_{t-1} is equivalent to:

- removing the black edges $\{a, b\}$ and $\{c, d\}$, and
- connecting the two extremity pairs either $\{a, c\}$ and $\{b, d\}$, or $\{a, d\}$ and $\{b, c\}$ such that no circular chromosome is created.

Creating a circular chromosome results in the appearance of an edge alternating cycle in the graph representation of the genome which increases the number of connected components of the graph into two. Hence, while applying the reversal operations, we are required to make sure no circular chromosome is created. We then update the labels of the black edges in X_t, as explained in Chapters 2 and 3. More informally, we label the edges of X_t that already exist in X_{t-1} the same as X_{t-1}, and for the remaining two unlabeled edges in X_t incident with $\{a, b, c, d\}$, we label the one edge incident to $x = min\{a, b, c, d\}$ by e, and the other one by e'.

The difference among our models of experiments are due to the restrictions we place while choosing the two edges e and e'. The codes in this thesis are developed in python and via the package NetworkX [14], since genomic operations including reversals are applied on the graph representations of the genomes. In addition, our

codes translate the graph representation of genome X_t, back into its equivalent permutation. To do so, we start reading the graph (which is an edge alternating path) starting from any of its telomeres either 0 or $2n+1$. For $1 \leq i \leq n$, we obtain the corresponding permutation such that while traversing the path we write:

- i, if we visit $2i-1$ and then $2i$, and
- $-i$, if we visit $2i$ and then $2i-1$.

Note that we have to follow the order in which the vertices appear in the traversed path. Furthermore, since the chromosome does not have any orientation itself (i.e. $X = x_1 \cdots x_n$ are equivalent to $-X = -x_n \cdots - x_1$). So reading the vertices on the path by starting from either of the telomers will result in the equivalent representation of the chromosome. The purpose of translating the genomes back into their equivalent permutation is to calculate $d^{rev}(X_t, X_0)$, which is the reversal distance of X_t from the initial genome X_0. To calculate this distance we applied the software UniMoG [19][10] and obtained the reversal distance for each step, where the inputs are the two genomes X_0 and X_t. UniMoG calculates different edit distances (such as reversal, DCJ, HP, etc) for unichromosomal and multichromosomal genomes with linear and circular chromosomes. Although, while inserting the input we need to differentiate the linear and circular chromosomes, such that for a chromosome Y (with 4 consecutive genes y_1, y_2, y_3, and y_4) we enter "$y_1\, y_2\, y_3\, y_4|$" if Y is linear, and "$y_1\, y_2\, y_3\, y_4)$ if Y is circular. Our codes include different models of reversal random processes, each experiment had been repeated a fixed number of times and the outputs present the average distance to the identity genomes at each time. Here, we also compared our results with the output of the simulation related to the reversal random walk where upon each jump the two edges are selected uniformly at random.

For each model our codes take 2 positive integers n and R as inputs, where

- n is the number of genes,
- R is the number of reversal operations (steps of the experiment).

We start from the identity genome $X_0 = 1 \cdots n$, for each $t \in \{1, \ldots, R\}$, we choose two black edges e and e' belonging to the graph representation of X_{t-1}, and apply the reversal operation on them. For the reversal model we pick e and e' uniformly at random. This means we can apply the reversal operation on any segment of the chromosome. The followings are the models with extra restrictions on the choice of e and e':

1. **Maximum Length Reversal (MLR):** this model is inspired by Chen and Skiena [12], and in addition to n and R, it takes an extra integer M indicating the maximum number of red edges separating e from e' in X_t. It is worth

remembering that X_t is an edge alternating path, and for e and e' to be separated with m red edges, where $m \leq M$, there should be $2m+1$ edges on the shortest path on the graph representation of the genome X_t containing the two edges e and e', where m of which are red, and the remaining $m+1$ are black. This means, we are only allowed to reverse the segments of the chromosome that do not have more than M genes. For every two black edges e and e' in X_t, we will call the number of red edges on the shortest path containing e and e' in X_t, the *distance* between e and e' in X_t and denote it by $\Delta_{e,e'}(t)$, hereafter. For this model, first we pick a black edge e uniformly at random from X_t, and we calculate $\Delta_{e,e''}(t)$ for every black edge e'' in X_t ($e'' \neq e$), and finally we choose e' from the set of edges with maximum distance M from e, uniformly at random. We present the pseudocode for this model in Algorithm 2.

2. **Weighted Reversal (WR):** this model is inspired by Pinter and Skiena [29] emphasizing that from the biological point of view the probability of a reversal on a segment of a genome depends on the length of that segment. Our codes include two models such that: first, each model chooses a black edge e in X_t uniformly at random, and then calculates the distance between e and any black edge e'' in X_t, i.e. $\Delta_{e,e''}(t)$, where $e \neq e''$. We then define a weight function assigning different probabilities to every black edge e'' based on $\Delta_{e,e''}(t)$. Finally, we choose the edge e' at random and with respect to the weight function, and then apply a reversal on e and e'. Here, we have defined two different weight functions:

   - **Weighted Reversal, Type I (Step Model):** in this model, we define the weight function such that the closer the black edge is to e, the more probable it is for it to be chosen. In other words, after picking e uniformly at random from X_t, the chance of picking e' decreases as $\Delta_{e,e'}(t)$ increases. Thus, the chance is maximum for e' with $\Delta_{e,e'}(t) = 1$. After choosing the distance, if there are two edges with the same distance from e, we choose either of them with probability $p = 1/2$. We present the pseudocode for this model in Algorithm 3.
   - **Weighted Reversal, Type II (Uniform Model):** biologically reversals on very short and very long segments of a chromosome are probably very scarce [29]. Therefore, we model this reversal process such that very short and very long reversals have the smallest probability to occur. We denote the maximum distance of black edges from e in X_t by $m_{e,t}$. In this model, we define the weight function such that the probability of choosing a black edge whose distance to e is 1 is the smallest, and the probability linearly increases as the distance increases up to half of $m_{e,t}$, and then the further the distance of an edge gets from the half of $m_{e,t}$ the smaller its

probability becomes. This way the probability of the farthest edge from e (with distance $m_{e,t}$) is the lowest and the same as an edge with distance 1. Although, this model slightly changes based on the parity of the maximum distance. If $m_{e,t}$ is an even number then we find $p_{k,t}$ such that the probability of choosing a black edge with distance k from e in X_t is

$$p_{k,t} = \begin{cases} pk & \text{if } k \in \{1, \ldots, \frac{m_{e,t}}{2}\}, \\ p(m_{e,t} + 1 - k) & \text{if } k \in \{\frac{m_{e,t}}{2} + 1, \ldots, m_{e,t}\}, \end{cases}$$

and $\sum_{k=1}^{m_{e,t}} p_{k,t} = 1$. A numerical example for this case is for a set of edge distances from e like $\{1, 1, 2, 2, 3, 4\}$, we have $p_1 = p_4 = p$ and $p_2 = p_3 = 2p$, where $p = \frac{1}{6}$. If $m_{e,t}$ is an odd number, we find $p_{k,t}$ such that

$$p_{k,t} = \begin{cases} pk & \text{if } k \in \{1, \ldots, \lceil \frac{m_{e,t}}{2} \rceil\}, \\ p(m_{e,t} + 1 - k) & \text{if } k \in \{\lceil \frac{m_{e,t}}{2} \rceil + 1, \ldots, m_{e,t}\}, \end{cases}$$

and $\sum_{k=1}^{m_{e,t}} p_{k,t} = 1$. A numerical example for this model is for a set of edge distances from e like $\{1, 1, 2, 2, 3, 3, 4, 5\}$, $p_1 = p_5 = p$, $p_2 = p_4 = 2p$, and $p_3 = 3p$, where $p = \frac{1}{9}$. In both cases after choosing the edge distance, if there exist two edges with the same distance in X_t, we can pick either of them with probability $p = 1/2$. We present the pseudocode for this model in Algorithm 4.

Our simulation results show the average reversal distances from the identity genome (ancestor) of the unichromosomal genome under reversals, weighted reversals (step model and uniform model), and maximum length reversals for different lengths. Starting from an identity genome with $n = 500$ genes, and over $R = 500$ random reversals, each simulation has been repeated 50 times. As [9] and [20] proved, the reversal random walk does not escape from its parsimony-bound before $t \approx n/2$. It is theoretically more complex to present the same results for the other types of reversal operations since while having restriction on selecting edge pairs to apply reversal, the corresponding random graph of the stochastic process is not Erdös-Rényi any more. Although, our simulation results show that for both models of weighted reversals we can expect the same kind of behaviour from the stochastic processes, that is they evolve along their parsimonious paths up to step $n/2$ which is $t = 250$, here. The maximum length reversal stochastic process behaves similarly unless the maximum length is too small in comparison to the size of the genome. Figures 4.1 and 4.2, and Table 4.1 display the reversal distance from the identity genome at each step of our models. Unlike the other reversal models, the maximum length reversal stochastic process with $k = 10$ (the yellow curve) escapes from its parsimonious path at an earlier step.

For the maximum length reversal (MLR) model, we have MLR process but we use d^{rev} to measure the distance of the genome at time t from the identity genome, since d^{MLR} is not available at UniMoG. This is not parsimony but we can easily fix this issue since

$$d^{rev}(X_t, X_0) \leq d^{MLR}(X_t, X_0) \leq t,$$

so

$$0 \leq \frac{cn - d^{MLR}(X_{cn}, X_0)}{a_n\sqrt{n}} \leq \frac{cn - d^{rev}(X_{cn}, X_0)}{a_n\sqrt{n}}, \tag{4.1.1}$$

for $t = cn$, where $c > 0$, $n \to \infty$ and $(a_n)_{n\geq 1}$ is an arbitrary sequence of real numbers such that $a_n \to \infty$ as $n \to \infty$. So our simulation shows the right hand side of 4.1.1 is very small in probability, so the same is true for the middle term. We can justify using the $d^{rev}(X_t, X_0)$, for both models of weighted reversal processes similarly.

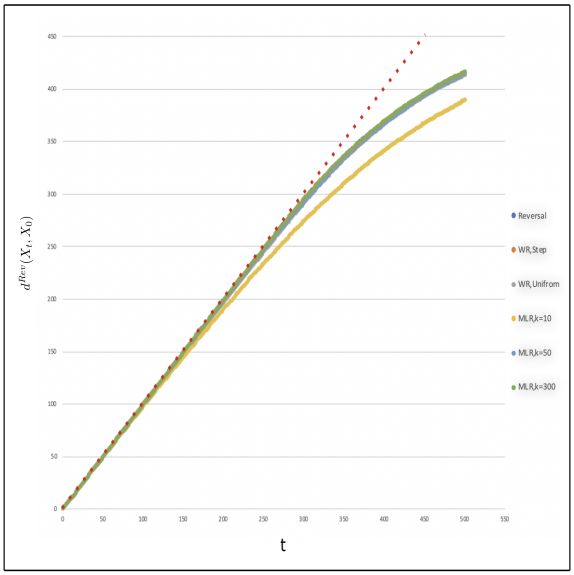

Figure 4.1: Simulation for different types of reversal models. The average reversal distance estimates are based on 50 runs, for 500 genes. The dashed line represents $d^{Rev}(X_t, X_0) = t$.

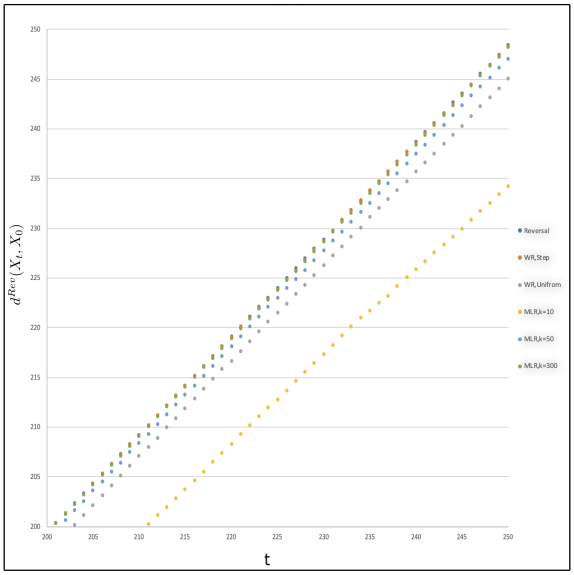

Figure 4.2: A closer look to Figure 4.1, where $200 \leq t \leq 250$.

Table 4.1: The average reversal distance after t reversal operations on the identity genome $X_0 = 1 \dots 500$, applied on different reversal stochastic processes, based on 50 runs.

The Average Reversal Distance $d^{Rev}(X_t, X_0)$	t=50	t=100	t=150	t=200	t=250	t=300	t=350	t=400	t=450	t=500
Reversal	49.96	99.92	149.70	199.36	248.42	295.42	336.22	368.90	395.32	415.60
Weighted Reversal, Step	50.00	99.92	149.70	199.36	248.30	295.28	335.96	369.08	395.02	415.10
Weighted Reversal, Uniform	49.88	99.24	148.60	197.26	245.04	291.80	333.44	367.04	393.26	413.56
Maximum Length Reversal, $k = 10$	49.60	97.76	144.80	190.64	234.18	274.54	310.34	341.54	367.60	389.56
Maximum Length Reversal, $k = 50$	49.86	99.62	149.30	198.62	247.00	292.46	334.12	366.90	393.74	414.80
Maximum Length Reversal, $k = 100$	49.98	99.84	149.66	198.98	247.82	294.44	335.24	368.18	394.04	415.10
Maximum Length Reversal, $k = 200$	50.00	99.94	149.84	199.46	248.52	294.94	335.58	368.82	394.46	416.04
Maximum Length Reversal, $k = 300$	49.96	99.74	149.62	199.36	248.20	295.22	336.38	369.48	395.82	416.02

Furthermore, Table 4.2 presents the average number of cycles in the breakpoint graph of the genome X_t and the identity genome X_0, $G_{BP}(X_t, X_0)$. And Table 4.3 shows the average number of trees in the random graph $\mathcal{G}_t$ corresponding to the reversal stochastic processes at step t. In both tables, t represents the number of reversals and $0 \leq t \leq 500$. It is worth reminding that the the reversal distance between any two genomes with the same set of genes depends on the number of genes, number of cycles in their breakpoint graph, number of hurdles and existence of fortress. Jamshidpey and Sankoff [20] proved that in the reversal random walk as n approaches to infinity, the number of hurdles are negligible after a convenient rescaling. Furthermore, when a jump occurs at time t, the number of tree components in the corresponding random graph (which is a realization of Erdös-Rényi) estimates the number of cycles in the breakpoint graph of the genome at time t and the ancestor. Note that the random graphs corresponding to the reversal stochastic processes studied here (i.e. maximum length reversal, weighted reversal) are not Erdös-Rényi. Although, by comparing the values of Tables 4.2 and 4.3 one can observe at each step the number of cycles in the breakpoint graph and the number of tree components in the random graph are approximately close in all the models except the maximum length reversal random process with $k = 10$. For example, for the the weighted reversal (step model) and at $t = 500$, the number of cycles in $G_{BP}(X_{500}, X_0)$ is 85.76 and the number of tree components of its random graph $\mathcal{G}_{500}$ is 81.86, while for the maximum length reversal with $k = 10$ these numbers are 111.44 and 86.85, respectively.

Table 4.2: The average number of cycles in the breakpoint graph $G_{BP}(X_t, X_0)$ for different reversal models evolving from the identity genome $X_0 = 1 \dots 500$, based on 50 runs.

The Average Number of Cycles in $G_{BP}(X_t, X_0)$	t=50	t=100	t=150	t=200	t=250	t=300	t=350	t=400	t=450	t=500
Reversal	451.04	401.08	351.30	301.64	252.56	205.52	164.68	131.90	105.48	85.18
Weighted Reversal, Step	451.00	401.08	351.30	301.64	252.68	205.72	165.00	131.82	105.92	85.76
Weighted Reversal, Uniform	451.12	401.76	352.4	303.74	255.96	209.20	167.52	133.84	107.68	87.22
Maximum Length Reversal, $k = 10$	451.40	403.28	356.24	310.4	266.90	226.50	190.68	159.46	133.40	111.44
Maximum Length Reversal, $k = 50$	451.14	401.38	351.70	302.38	254.00	208.54	166.84	134.08	107.16	86.20
Maximum Length Reversal, $k = 100$	451.02	401.16	351.34	302.02	253.18	206.56	165.7	132.74	106.92	85.82
Maximum Length Reversal, $k = 200$	451.00	401.06	351.16	301.54	252.48	206.06	165.38	132.16	106.38	84.88
Maximum Length Reversal, $k = 300$	451.04	401.26	351.38	301.64	252.78	205.74	164.56	131.42	105.16	84.80

Table 4.3: The average number of the tree components in the random graph $\mathcal{G}_t$ for different reversal models evolving from the identity genome $X_0 = 1 \dots 500$, based on 50 runs.

The Average Number of Tree Components of $\mathcal{G}_t$	t=50	t=100	t=150	t=200	t=250	t=300	t=350	t=400	t=450	t=500
Reversal	451.02	401.06	351.12	301.20	251.46	203.44	161.78	128.60	102.14	81.26
Weighted Reversal, Step	451.00	401.04	351.10	301.18	251.44	203.26	162.02	128.68	102.48	81.86
Weighted Reversal, Uniform	451.06	401.38	351.64	302.18	252.92	204.62	161.92	127.66	101.26	80.74
Maximum Length Reversal, $k = 10$	451.20	402.08	353.26	304.90	257.48	212.24	171.46	136.64	108.88	86.56
Maximum Length Reversal, $k = 50$	451.08	401.20	351.32	301.46	252.06	204.84	162.52	129.16	102.54	81.48
Maximum Length Reversal, $k = 100$	451.02	401.10	351.16	301.40	251.72	203.56	162.14	128.94	103.00	82.16
Maximum Length Reversal, $k = 200$	451.00	401.02	351.08	301.16	251.38	203.50	162.06	128.52	102.74	81.38
Maximum Length Reversal, $k = 300$	451.00	401.08	351.10	301.12	251.34	203.02	161.04	127.96	101.56	80.88

4.2 Different models of DCJ genome evolution process

As mentioned in the previous chapters double-cut-and-join or in short DCJ operation introduced by [37] is a genomic operation that cuts two adjacencies of the genome like $\{a, b\}$ and $\{c, d\}$ and reconnect the four extremities in one of the two possible ways, either $\{a, c\}$ and $\{b, d\}$ or $\{a, d\}$ and $\{b, c\}$. The DCJ operation includes other operations such as reversal, translocation (including fission and fusion), and block interchange, and a DCJ operation on a segment of a single chromosome either reverses that segment or extract it into a circular chromosome. If a DCJ operation is *restricted*, then a DCJ resulting in creation of a circular chromosome should be followed only by another DCJ reinserting that intermediate cycle into a linear chromosome. The outcome of these two consecutive DCJ operations is a block interchange.

In Chapter 2 we proved both DCJ and restricted DCJ evolutionary models evolve along their parsimonious paths up to $n/2$, where n is the number of genes in the genome. Here we present the simulation methods and results for evolutionary models evolving from unichromosomal genomes under DCJ and restricted DCJ operations, with probability p of creating circular chromosomes (different than the DCJ models in chapter 2). We then extend the DCJ evolutionary model for multichromosomal genomes.

•**DCJ Evolution Starting from UniChromosomal Genomes**: let $\{Y_t : t = 0, 1, 2, \ldots\}$ be a DCJ process on the space of signed genomes with n genes. We start the process at the identity genome Y_0 and construct the graph representation of $Y_0 = 1 \cdots n$. We label the vertices and the edges of Y_0. Similarly, we define the labelling process and the random graph $\mathcal{G}_t$, for $t \in \{1, 2, \ldots, D\}$, where D is the number of desired DCJ operation. We simulate a DCJ process in which at each jump time t we pick two black edges labelled by e and e' in Y_{t-1} such that $e, e' \in \{1, 2, \ldots, n + 1\}$, and apply a DCJ operation on them. Let $0 \leq p \leq 1$ be a given number, we pick two black edges $e = \{a, b\}$ and $e' = \{c, d\}$ in Y_{t-1} uniformly at random. Applying a DCJ operation effecting e and e' on the genome Y_{t-1} is equivalent to:

- removing the black edges $\{a, b\}$ and $\{c, d\}$,
- if e and e' were not on the same chromosome connect either $\{a, c\}$ and $\{b, d\}$, or $\{a, d\}$ and $\{b, c\}$, with probability $1/2$,
- if e and e' were on the same chromosome connect the extremities such that the probability of non-reversal DCJ (creation of a cycle) and reversal DCJ be p and $1 - p$, respectively.

We update the labelling in Y_t (as explained in Chapter 2) and connect the two vertices labelled by e and e' in $\mathcal{G}_t$ if they are not already adjacent. We then calculate

the DCJ distance between Y_0 and Y_t, $d^{DCJ}(Y_t, Y_0)$, via UniMoG [10]. We present the pseudocode for this model in Algorithm 5.

•**Restricted DCJ Evolution Starting from UniChromosomal Genomes**: here again, we simulated restricted DCJ stochastic process $\{Y'_t : t = 0, 1, 2, \ldots\}$ on the space of signed uni-chromosomal genomes with n genes. We start our random process at $Y'_0 = 1 \cdots n$. Let D' be the number of desired restricted DCJ operations. The graph representation of Y'_0, the labelling process and $\mathcal{G}_t$ are defined similar as in the previous models. Let $0 \leq p \leq 1$ be a given number,we apply a restricted DCJ operation based on the existence of a circular chromosome in Y'_{t-1}, such that:

1. if Y'_{t-1} has no circular chromosome
   - pick two black edges $e = \{a, b\}$ and $e' = \{c, d\}$ in Y'_{t-1} uniformly at random,
   - remove the black edges $\{a, b\}$ and $\{c, d\}$,
   - connect the extremities such that the probability of a non-reversal DCJ (creation of a cycle) and a reversal DCJ are p and $1 - p$, respectively.
2. if Y'_{t-1} has a circular chromosome
   - pick $e = \{a, b\}$ from the linear and $e' = \{c, d\}$ from the circular chromosome uniformly at random,
   - remove the black edges $\{a, b\}$ and $\{c, d\}$,
   - connect either $\{a, c\}$ and $\{b, d\}$, or $\{a, d\}$ and $\{b, c\}$, with probability 1/2.

Applying a DCJ operation on two edges of the genome, one belonging to a circular and the other to a linear chromosome results in a longer linear chromosome. In our models the number of linear chromosomes remains the same, and equals the number of linear chromosomes in the identity genome. An empty linear chromosome is made of two telomers that are connected with a black edge. For unichromosomal models $\{0, 2n+1\}$ is the empty linear chromosome. We present the pseudocode for this model in Algorithm 6. Figure 4.3 and Figure 4.4 show the average DCJ distance from the identity genome of the random genome sampled at time t for DCJ and Restricted DCJ processes, respectively. The DCJ distance is obtained via the tool introduced by [10] in UniMoG. Both models starting from the identity genome with $n = 1000$ genes, and over 1000 operations. Each simulation has been repeated 50 times and with probabilities 0.1, 0.5 and 0.7 of creating circular chromosomes. Both models do not escape from their parsimonious paths of evolution before $t \approx 500$, and behave very similarly for different probabilities.

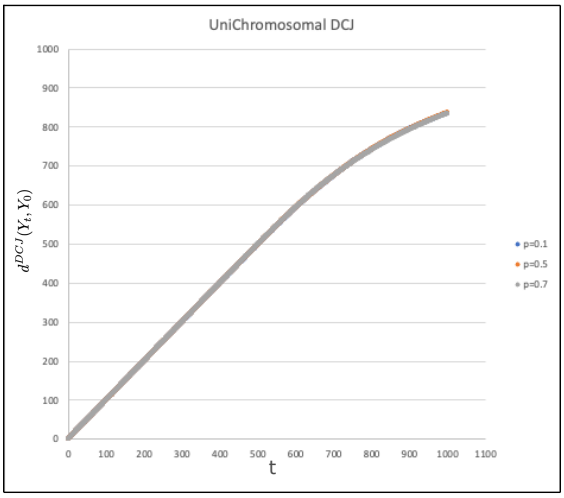

Figure 4.3: Simulation for DCJ model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 50 runs, for $p = 0.1, 0.5, 0.7$ of creating circular chromosomes. The ancestral genome is unichromosomal and contains 1000 genes.

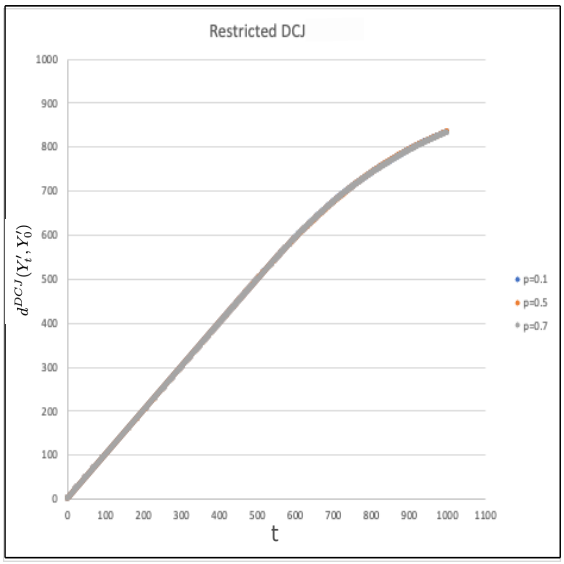

Figure 4.4: Simulation for restricted DCJ model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 50 runs, for $p = 0.1, 0.5, 0.7$ of creating circular chromosomes. The ancestral genome is unichromosomal and contains 1000 genes.

As presented in Tables 4.4 and 4.5, the average distance from the identity genome, the average number of the cycles in the breakpoint graph and the average number of tree components in the random graph at each step for different probabilities are quite

similar in both models. Moreover, at each time t, the number of tree components in the random graph estimates the number of cycles in the breakpoint graph with a small error, in comparison to the size of the genome.

Table 4.4: The average data obtained from simulating DCJ model of evolution with probability p of creating circular chromosomes, starting from the identity unichromosomal genome with 1000 genes, based on 50 runs.

DCJ Random Process (Unichromosomal Identity Genome)	p	t=100	t=200	t=300	t=400	t=500	t=600	t=700	t=800	t=900	t=1000
$d^{DCJ}(Y_t, Y_0)$	0.1	99.96	199.94	299.68	399.34	497.9	592.64	674.46	741.82	794.78	835
	0.5	100	199.98	299.84	399.6	498.14	593.48	674.96	742.02	794.74	836.22
	0.7	100	199.96	299.8	399.54	498.42	593.28	674.74	740.78	793.16	834.06
The number of cycles in $G_{BP}(Y_t, Y_0)$	0.1	901.04	801.06	701.32	601.66	503.1	408.36	326.5	259.1	206.06	165.86
	0.5	901	801.02	701.16	601.4	502.86	407.52	326	258.88	206.1	164.64
	0.7	901	801.04	701.2	601.46	502.58	407.7	326.24	260	207.66	166.8
The number of trees in $\mathcal{G}_t$	0.1	901.02	801.02	701.08	601.16	501.4	405.42	322.98	255.28	202.38	161.86
	0.5	901	801	701.04	601.1	501.28	405	322.34	255.18	202.4	160.68
	0.7	901	801.02	701.06	601.1	501.52	404.96	322.56	256.18	203.94	162.7
The number of circular chromosomes	0.1	0.56	0.56	0.46	0.58	0.72	0.76	0.58	0.58	0.66	0.58
	0.5	2.68	2.92	3.48	3.72	3.08	3.14	3.06	3.34	3.18	3.5
	0.7	2.98	3.6	3.82	4.1	4.38	4.44	4.36	4.48	4.18	4.16
The maximum length of circular chromosomes	0.1	88.1	86.54	52.16	100.7	86.42	113.76	93.8	44.5	76.18	40.74
	0.5	311.16	242.66	262.58	265.84	327.42	242.94	220.54	243.94	227.16	218.24
	0.7	340.92	324.2	378.44	290.8	287.34	249.04	364.38	277.16	298.26	269.8
The length of linear chromosome	0.1	905.1	908.48	944.34	893.2	907	876.92	901.12	947.72	920.84	954.28
	0.5	609.8	660	628.58	633.96	579.78	689.4	710.9	657.66	691.28	695.92
	0.7	567.48	546.76	505	597.34	592.98	630.24	515.4	606.7	588.16	616.8

Table 4.5: The average data obtained from simulating restricted DCJ model of evolution with probability p of creating circular chromosomes, starting from the identity unichromosomal genome with 1000 genes, based on 50 runs.

Restricted DCJ Random Process (Unichromosomal Identity Genome)	p	t=100	t=200	t=300	t=400	t=500	t=600	t=700	t=800	t=900	t=1000
$d^{DCJ}(Y'_t, Y'_0)$	0.1	100	199.94	299.92	399.6	498.48	593.04	674.62	740.6	793.52	834.46
	0.5	99.96	199.9	299.86	399.48	498.3	592.68	673.84	741.04	793.66	834.38
	0.7	100	199.96	299.64	399.34	498.12	593.12	674.44	740.44	792.44	833.4
The number of cycles in $G_{BP}(Y'_t, Y'_0)$	0.1	901	801.06	701.08	601.4	502.52	407.94	326.28	260.32	207.26	166.18
	0.5	901.04	801.1	701.14	601.52	502.7	408.28	327.06	259.88	207.18	166.36
	0.7	901	801.04	701.36	601.66	502.88	407.82	326.44	260.42	208.38	167.4
The number of trees in $\mathcal{G}_t$	0.1	901	801.04	701.06	601.14	501.34	405.46	322.44	256.84	203.42	162.2
	0.5	901.02	801.06	701.06	601.14	501.4	405.6	323.34	256.18	203.38	162.66
	0.7	901	801.02	701.12	601.26	501.5	405.06	323.18	256.64	204.16	163.64

•**DCJ Evolution of MultiChromosomal Genomes**: we present the pseudocode for this model in Algorithm 7. This model is very similar to the first DCJ model, except the identity genomes have N linear chromosomes. Therefore, in addition to the number of DCJ operations D, we have $N+1$ integers as input, where N is the number of linear chromosomes and $n_1, n_2, \ldots, n_N$ are the numbers of the genes in each chromosome. In this case, since the identity genome has more than one chromosome and we applied a different method to construct the graph representation of it, see Algorithm 8 for more details. Also, the codes for translating the multichromosomal genomes to their permutations are more complicated.

Figure 4.5 shows the simulation of DCJ evolution model of multichromosomal genomes (with different probabilities of creating circular chromosomes), and the DCJ model does not escape from its parsimonious path before $t \approx n/2$. Here, we simulated the DCJ evolution starting from a multichromosomal genome with 4 linear chromosomes containing 1000 genes with probability 0.1, 0.5 and 0.7 of creating circular chromosomes.

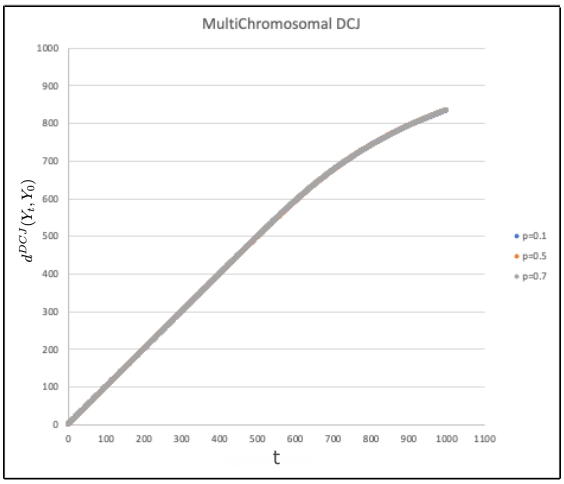

Figure 4.5: Simulation for unrestricted DCJ model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 50 runs, for $p = 0.1, 0.5, 0.7$ of creating circular chromosomes. The ancestral genome contains 1000 genes in 4 linear chromosomes of size $100, 200, 300$ and 400.

Table 4.6 presents more detailed data regarding this simulation. Similar to Table 4.4, by increasing the probability of circular chromosomes, the average number of the circular chromosomes and their maximum length increase as well.

Table 4.6: The average data obtained from simulating DCJ model of evolution with probability p of creating circular chromosomes, starting from the identity multichromosomal genome with 4 linear chromosomes of size $100, 200, 300$ and 400, based on 50 runs.

MultiChromosomal DCJ Random Process	p	t=100	t=200	t=300	t=400	t=500	t=600	t=700	t=800	t=900	t=1000
$d^{DCJ}(X_t, X_0)$	0.1	100	199.98	299.7	399.1	497.98	592.22	674.98	741.12	793.24	833.88
	0.5	100	199.92	299.84	399.3	498.16	592.58	674.9	740.56	793.3	834.74
	0.7	100	199.94	299.86	399.44	498.44	593.44	675.6	740.4	792.72	835.14
The number of circular chromosomes	0.1	0.36	0.48	0.32	0.38	0.52	0.42	0.52	0.68	0.3	0.52
	0.5	1.9	2.48	2.22	2.72	2.58	2.68	2.44	2.7	2.64	2.52
	0.7	2.42	3.3	3.3	3.76	4.02	3.74	4.24	4.42	4.34	3.9
The maximum length of circular chromosomes	0.1	11.78	33.3	12.6	10.8	44.26	26.32	42.06	38.58	7.54	23.4
	0.5	86.26	82.8	83.88	90.62	89.78	108.88	74.38	114.4	73.7	98.84
	0.7	96.42	118.02	120.86	95.62	121.22	124.62	130.52	114.04	109.78	117.48
The maximum length of linear chromosomes	0.1	537.28	494.64	507.28	545.44	491.64	517.62	474.86	449.48	505.64	515
	0.5	476.56	461.3	447.64	459.42	491.78	413.72	465.02	441.42	475.9	441.64
	0.7	427.3	424.36	415.32	432.5	426.96	450.94	406.36	426.08	456.8	467.08

4.3 HP, IRT, and translocation models of genome evolution

In Chapter 3, we proved that a random HP process and a random IRT process do not escape from their parsimonious paths of evolution before $t \approx n/2$, where n is the number of genes in the genome. Here we present the simulation results of these two models under HP distance, which inspired their mathematical proof. We also present the simulation result for random translocation process under DCJ distance. The distances are chosen based on their availability in UniMoG software. In all of the following models let X_t be the genome obtained after applying random t operations on X_0.

•**General reversal and translocation (HP)**: this model is inspired by [17], who obtained a linear time distance formula and algorithm for the minimum number of reversal and translocation operations required to transform one genome into another (HP distance). This model is very similar to the random DCJ process, except for each randomly picked pair of black edges e and e', if e and e' are in the same chromosome we must apply a reversal operation. We present the pseudocode for this model in Algorithm 9.

•**Internal reversal and translocation (IRT)**: this model is the restricted version of the HP model and is inspired by the distance formula obtained in [17]. At each step, we pick two non-telomeric edges of the genome uniformly at random (without replacement). This means we are not allowed to pick any black edge incident to the telomers. Similar to the HP model if the two edges belong to the same chromosome we are only allowed to apply a reversal operation on them. We present the pseudocode for this model in Algorithm 10. In our simulations we defined the IRT process but we applied $d^{HP}(X_t, X_0)$ to measure the distance between the genome at step t and the identity genome (UniMoG availability). Similar to MLR and WR in order to check the parsimony-bound for the IRT evolution process we have

$$d^{HP}(X_t, X_0) \leq d^{IRT}(X_t, X_0) \leq t,$$

so

$$0 \leq \frac{cn - d^{IRT}(X_{cn}, X_0)}{a_n\sqrt{n}} \leq \frac{cn - d^{HP}(X_{cn}, X_0)}{a_n\sqrt{n}}, \tag{4.3.1}$$

for $t = cn$, where $c > 0$, $n \to \infty$ and $(a_n)_{n\geq 1}$ is an arbitrary sequence of real numbers such that $a_n \to \infty$ as $n \to \infty$. So our simulation shows the right hand side of 4.3.1 is very small in probability, so the same is true for the middle term.

•**Translocation**: this model is inspired by [16], which formulated the minimum number of translocation operations required to transform one genome into another. At each step of this model, we are allowed to pick two black edges $\{a, b\}$ and $\{c, d\}$ that do not belong to the same chromosome of the genome, and after deleting the two

selected adjacencies, reconnect the extremities in one of the two possible ways: either $\{a, c\}$ and $\{b, d\}$, or $\{a, d\}$ and $\{b, c\}$. We present the pseudocode for this model in Algorithm 11. The translocation distance ($d^{td}(X_t, X_0)$) available in UniMoG is only applicable for genomes with the same tails; therefore, to demonstrate parsimony we applied DCJ distance instead and since

$$d^{DCJ}(X_t, X_0) \leq d^{td}(X_t, X_0) \leq t,$$

so

$$0 \leq \frac{cn - d^{td}(X_{cn}, X_0)}{a_n\sqrt{n}} \leq \frac{cn - d^{DCJ}(X_{cn}, X_0)}{a_n\sqrt{n}}, \tag{4.3.2}$$

for $t = cn$, where $c > 0$, $n \to \infty$ and $(a_n)_{n\geq 1}$ is an arbitrary sequence of real numbers such that $a_n \to \infty$ as $n \to \infty$. So our simulation shows the right hand side of 4.3.2 is very small in probability, so the same is true for the middle term.

Table 4.7, and Figures 4.6, 4.7 and 4.8 represent the simulation results for HP, IRT and translocation model. All of these simulations started from the identity genome with 3 linear chromosomes and $\{250, 350, 400\}$ genes, and repeated 50 times. As proved the random HP and the random IRT processes do not escape their parsimonious paths of evolution before $t \approx n/2$, which is $t = 500$ here. The simulation results are similar for the translocation random process.

	t=100	t=200	t=300	t=400	t=500
$d^{HP}(X_t, X_0)$ HP Random Process	100	199.86	299.86	399.45	498.23
$d^{HP}(X_t, X_0)$ IRT Random Process	100	199.94	299.76	399.42	498.44
$d^{DCJ}(X_t, X_0)$ Translocation Random Process	100	200	299.86	399.43	498.9
	t=600	t=700	t=800	t=900	t=1000
$d^{HP}(X_t, X_0)$ HP Random Process	592.36	673.45	739.72	793.84	831.59
$d^{HP}(X_t, X_0)$ IRT Random Process	592.4	672.88	737.68	790.12	830.56
$d^{DCJ}(X_t, X_0)$ Translocation Random Process	592.28	674.13	741.05	794.21	836.92

Table 4.7: The average data obtained from simulating HP, IRT and translocation Models of Evolution starting from the identity multichromosomal genome with 3 linear chromosomes of size $250, 350$ and 400, based on 50 runs.

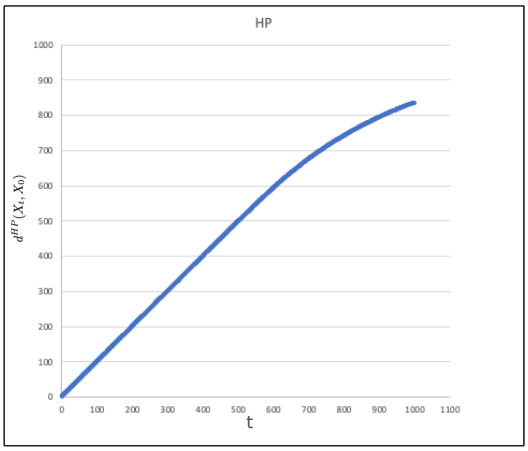

Figure 4.6: Simulation for HP model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average HP distance estimates are based on 50 runs. The ancestral genome contains 1000 genes in 3 linear chromosomes of size $250, 350$ and 400.

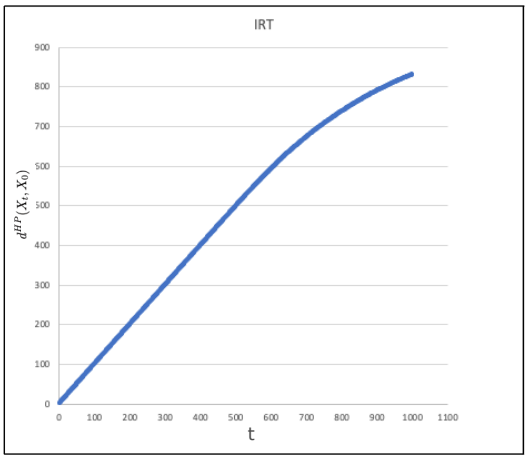

Figure 4.7: Simulation for IRT model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average HP distance estimates are based on 50 runs. The ancestral genome contains 1000 genes in 3 linear chromosomes of size $250, 350$ and 400.

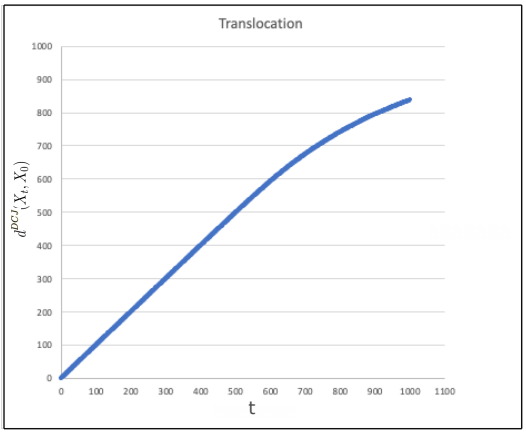

Figure 4.8: Simulation for translocation model illustrating parsimony binding up to time cn, for $c \approx 0.5$. The average DCJ distance estimates are based on 50 runs. The ancestral genome contains 1000 genes in 3 linear chromosomes of size $250, 350$ and 400.

Bioinformatics, 34, 2018, i366–i375
doi: 10.1093/bioinformatics/bty242
ISMB 2018

Accurate prediction of orthologs in the presence of divergence after duplication

Manuel Lafond[1,2,*], Mona Meghdari Miardan[1] and David Sankoff[1]

[1]Department of Mathematics and Statistics, University of Ottawa, Ottawa, ON K1N 6N5, Canada and [2]Department of Computer Science, Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada

*To whom correspondence should be addressed

Abstract

Motivation: When gene duplication occurs, one of the copies may become free of selective pressure and evolve at an accelerated pace. This has important consequences on the prediction of orthology relationships, since two orthologous genes separated by divergence after duplication may differ in both sequence and function. In this work, we make the distinction between the *primary* orthologs, which have not been affected by accelerated mutation rates on their evolutionary path, and the *secondary* orthologs, which have. Similarity-based prediction methods will tend to miss secondary orthologs, whereas phylogeny-based methods cannot separate primary and secondary orthologs. However, both types of orthology have applications in important areas such as gene function prediction and phylogenetic reconstruction, motivating the need for methods that can distinguish the two types.

Results: We formalize the notion of divergence after duplication and provide a theoretical basis for the inference of primary and secondary orthologs. We then put these ideas to practice with the Hybrid Prediction of Paralogs and Orthologs (HyPPO) framework, which combines ideas from both similarity and phylogeny approaches. We apply our method to simulated and empirical datasets and show that we achieve superior accuracy in predicting primary orthologs, secondary orthologs and paralogs.

Availability and implementation: HyPPO is a modular framework with a core developed in Python and is provided with a variety of C++ modules. The source code is available at https://github.com/manuellafond/HyPPO.

Contact: mlafond2@uOttawa.ca

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

During the course of evolution, speciation and duplication create pairs of homologous genes, which can be classified into two categories. Two genes are orthologs if they descend from an ancestral gene that has undergone speciation and paralogs if they result from duplication. Distinguishing orthologs from paralogs is of considerable importance in biology, owing to their functional and evolutionary implications (Gabaldón and Koonin, 2013; Koonin, 2005). Notably, the well-known 'orthologs conjecture' states that orthologs tend to perform similar functions, in contrast with paralogs, which tend to diverge in their functional role. The reasoning underlying this conjecture is that when a gene creates a copy in the same genome, then this genome has two genes that are able to perform the same function. Hence, one of these copies may become free of selective pressure, resulting in a higher rate of mutations and facilitating a possible change in functionality. We refer the reader to (Altenhoff *et al.*, 2012; Chen and Zhang, 2012; Nehrt *et al.*, 2011; Studer and Robinson-Rechavi, 2009; Thomas *et al.*, 2012) for some recent developments on this conjecture.

The two traditional methods to predict orthology relations are *similarity*- and *phylogeny-based* (see Altenhoff and Dessimoz, 2012; Kristensen *et al.*, 2011 for a survey). Similarity-based methods aim to partition the set of genes into 'groups' of orthologs under the assumption that genes that share similar sequences are orthologous. The precise notion of a group depends on the particular method used, since in some cases, these groups include multiple genes from the same species (e.g. recent paralogs). For instance in OrthoMCL (Li *et al.*, 2003), Markov clustering is used to identify groups of orthologs and close paralogs. A similar idea is implemented in Proteinortho using spectral clustering techniques (Lechner *et al.*, 2011). The OMA software is somewhat more stringent and aims to find cliques of pairwise orthologous genes (Roth *et al.*, 2008). This was extended in later versions with the addition of the GETHOGS algorithm (Altenhoff *et al.*, 2013; Train *et al.*, 2017). The program is able to potentially find all

orthologous gene pairs by grouping genes into *hierarchical orthologous groups* (HOGs), which are the orthologs that belong to a set of species within a given taxonomic range. Some other similarity-based grouping methods and databases include COG (Tatusov *et al.*, 2003), EggNOG (Powell *et al.*, 2012), InParanoid (O'Brien *et al.*, 2004) and OrthoFinder (Emms and Kelly, 2015). The OMG approach (Zheng *et al.*, 2011) identifies sets of pairwise *orthogonal* genes from a homology graph inferred from synteny.

On the other hand, phylogeny-based methods do not attempt in the first instance to group genes together but rather aim to identify orthologs by inferring the lowest common ancestor (lca) relationships between genes directly. This can be done by inferring a gene tree and identifying its speciation and duplication events using *reconciliation* with a species tree (Stolzer *et al.*, 2012; Ullah *et al.*, 2015). This approach is however sensitive to errors in the gene tree and the species tree. Methods of this type include notably LOFT (Van der Heijden *et al.*, 2007) and COCO-CL (Jothi *et al.*, 2006). Recently, a number of methods exploit a fundamental property of orthology graphs, in which vertices are genes and edges depict orthology: a graph G is a valid orthology graph if and only if it is P_4-free (Dondi *et al.*, 2017a; Dondi *et al.*, 2017b; Hernandez-Rosales *et al.*, 2012; Hellmuth *et al.*, 2013; Hellmuth *et al.*, 2015; Jones *et al.*, 2016; Lafond and El-Mabrouk, 2014; Lafond *et al.*, 2016), i.e. G has no path on four vertices without a shortcut. The essential reason underlying this characterization is that P_4-free graphs have a special tree representation called a *co-tree*, which turns out to be interpretable as a gene tree that depicts all the lca relationships prescribed by G. These methods usually construct an approximate putative orthology graph and perform a minimum number of modifications on the graph so that it becomes P_4-free.

The output from similarity- and phylogeny-based methods can be different, as some orthologies may be irrelevant from the perspective of grouping genes with similar functions, but will be inferred by phylogenetic methods. Consider the example of Figure 1. The node x represents an ancestral duplication event in which *neofunctionalization* hypothetically occurred, where one copy (here the left child of x) retains its original functionality, whereas the other copy (the right child of x) diverges and starts accumulating mutations at a higher rate. As a result of this event, the a and b genes remain similar, as no major divergence event separates the two. However, a and c will have differentiated significantly, even though the two genes are orthologous (since their lca is a speciation). Consequently, similarity-based methods are unlikely to mark a and c as orthologs, as the two genes do not share sequence similarity. From the point of view of functional annotation, this is not problematic, since the a, c orthology pair essentially behave as paralogs and may differ in function.

This raises the question as to which type of inference is the most relevant: should we focus on the groups of similar orthologs, or do we need *all* the orthologs? Ideally, both types of orthologies should be predicted, since similarity groups are useful for function-related applications (Doyle *et al.*, 2010), while the complete set of pairwise relations is useful for phylogenetic reconstructions (Hellmuth *et al.*, 2015).

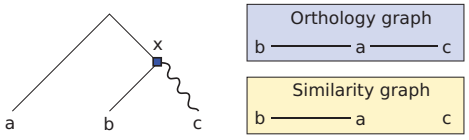

Fig. 1. An example of gene tree for the gene family {*a, b, c*}, along with the underlying orthology and similarity graphs. The root of the tree is a speciation, the square is a duplication, and the wiggly edge represents an event of divergence after duplication. The gene pairs *ab* and *ac* are orthologs. However, *a* and *c* will not appear as 'similar', as there was a significant divergence event on their evolutionary path

Of course, post-duplication phenomena other than neofunctionalization as described above can arise. For instance, *bifunctionalization* occurs when both copies preserve the parental function without diverging, and *subfunctionalization* occurs when the functions of the common ancestor are partitioned among the descending paralogs (Zhang, 2003), in which case it is possible for both copies to undergo a new rate of mutations. Nevertheless, the phenomenon of a single copy undergoing functional change has been reported to occur frequently (Cardoso-Moreira *et al.*, 2016; Jordan *et al.*, 2004; Lynch and Conery, 2000; Woods *et al.*, 2013; Soria *et al.*, 2014), as duplication models other than neofunctionalization, for instance, *adaptive radiation* or *modified duplication*, also predict this type of divergence after duplication. We redirect the reader to (Innan and Kondrashov, 2010; Table 1) for more details.

In this work, we are interested in how the task of predicting orthology is affected when divergence after duplication does occur. We propose an orthology prediction framework that makes the distinction between the *primary orthologs*, the orthologous gene pairs that have not been separated by an event of duplication followed by an increased rate of mutation, and the *secondary orthologs*, which consist of the pairs of orthologs that have had at least one such event along their evolutionary path. For example in Figure 1, {a, b} are primary orthologs, whereas {a, c} are secondary orthologs. Note that the relevance of making this distinction was also discussed by Swenson and El-Mabrouk (2012), where the primary orthologs are called *isoorthologs*. Also observe that this categorization is different from the co-orthology relationship, which is usually defined with respect to a given node of a gene tree. We formalize this notion of primary orthology and show that they must form cliques in the orthology graph, thereby providing a formal justification of the clustering steps often performed by orthology prediction methods.

Our algorithmic framework is a hybrid approach that makes use of both similarity clustering and phylogenetic reconstruction. More specifically, we use exact graph partitioning techniques to find the primary orthologs, construct a species tree from the orthology clusters found and use this species tree to obtain the secondary orthologs. The method also puts to practical use the theory of P_4-free orthology graphs developed recently. The framework, which we call Hybrid Prediction of Paralogs and Orthologs (HyPPO), is implemented as a fully modular pipeline in which each task can have its own independent implementation.

We evaluate HyPPO on both simulated and real empirical datasets. We compare our method with OrthoMCL and OMA-GETHOGS—to the best of our knowledge, the latter is the only other program that can distinguish primary and secondary orthologs. We show that HyPPO achieves superior accuracy in the set of predicted pairwise relations and is better at finding the primary orthology clusters. Table 1 presents a summary of our results.

Table 1. Average accuracy, precision, recall and cluster scores, taken over the set of all simulated gene trees (see the Results section for the definitions)

	Accuracy	Precision	Recall	Cluster score
HyPPO	0.940	0.911	0.875	0.915
HyPPO + Species tree	0.949	0.924	0.905	0.915
OMA-GETHOGS	0.877	0.940	0.699	0.831
OrthoMCL	0.812	0.845	0.496	0.690

2 Materials and methods

In this section, we first introduce the preliminary notions necessary for the understanding of this paper. We then formalize the notion of divergence after duplication in gene trees and its consequences on distinguishing primary and secondary orthologs. We then present the main algorithmic components of the HyPPO framework. Note that due to space constraints, most of the proofs are relegated to the Supplementary Material.

2.1 Preliminary notions

In this paper, every tree T is assumed to be rooted and binary, i.e. each internal node has two children, and each node v has a unique parent $p(v)$, with the exception of the root, denoted $r(T)$, which has no parent. The set of vertices of T is denoted by $V(T)$ and its set of leaves by $L(T)$. A node u of T is an *ancestor* of a node v if u is on the path from v to $r(T)$. Then v is a *descendant* of u, in which case we write $v \leq u$. If $v \neq u$, we write $v < u$. If none of $v \leq u$ nor $u \leq v$ holds, then u and v are *incomparable*, and we write $u \oplus v$. The *least common ancestor* of a set of leaves $X \subseteq L(T)$, denoted $lca_T(X)$, is the node u of T that is the ancestor of every node in X, and that is the most distant from the root. For convenience, we may write $lca_T(a, b)$ instead of $lca_T(\{a, b\})$. For a graph G, a *clique* is a set of vertices of G in which each pair shares an edge. A P_k is a path on k vertices with no chord.

A *gene family* Γ is a set of genes related by homology. A *gene tree* for Γ is a tree T in which $L(T) = \Gamma$. Likewise, if Σ is a set of species, a *species tree* for Σ is a tree S satisfying $L(S) = \Sigma$. Each gene $g \in \Gamma$ belongs to a species denoted $\sigma(g)$. If $X \subseteq \Gamma$ is a set of genes, then we write $\sigma(X) = \{\sigma(g) : g \in X\}$.

A *DS*-tree (T, ℓ) is a pair in which T is a gene tree, and $\ell : V(T) \setminus L(T) \to \{\mathbb{D}, \mathbb{S}\}$ is a function labeling internal nodes by either $\mathbb{D}$ or $\mathbb{S}$, respectively standing for duplication and speciation. We will often omit mentioning ℓ explicitly and say that T is a *DS*-tree with the understanding that its internal nodes are labeled by ℓ. Given a species tree S, we map each node of T to a node of S with the *lca-mapping* $s : V(T) \to V(S)$ defined as follows: if $g \in L(T)$, then $s(g) = \sigma(g)$, and otherwise $s(g) = lca_S(\{\sigma(l) : l \in L(T) \text{ and } l \leq g\})$. By a slight abuse of notation, if $C \subseteq \Gamma$, we denote $s(C) = lca_S(\sigma(C))$. Note that s depends on the species tree S. In case of ambiguity, we may write $s(g, S)$ or $s(C, S)$ to denote the lca-mapping with respect to S. An example of lca-mapping is presented in Figure 3.

We use the definition of Fitch (2000) for orthology and paralogy.

Definition 1. With respect to a DS-tree T, two genes a and b are orthologs if $\ell(lca_T(a, b)) = \mathbb{S}$, and paralogs if $\ell(lca_T(a, b)) = \mathbb{D}$.

An *orthology graph* $G = (V, E)$ is a graph in which V is a set of genes and $uv \in E$ if and only if u and v are orthologs. An *orthology cluster* C is a set of genes that are pairwise orthologous. In particular, C may contain at most one gene from any given species.

2.2 Orthology and divergence after duplication

As mentioned in the Section 1, duplication may introduce divergence when one of the child copies is redundant. In this section, our goal is to investigate the implications of this phenomenon on the structure of orthology relations. That is, if divergence after duplication occurs, what are the orthologs that are expected to be recovered using similarity-based methods? Or using phylogeny-based methods? We present the tools that we will use to devise methods tailored for this type of evolution. As we will demonstrate in the Section 3, the methodology developed here can also be used in case that divergence after duplication does not always occur, or even when if it does not happen at all.

We formalize divergence after duplication as follows. Let T be a *DS*-tree. Suppose that for each $\mathbb{D}$ node v of T, v has one child v_1 that evolved at a normal rate and another child v_2 that underwent an accelerated pace of evolutionary changes. We will say that the vv_2 edge is a *divergent duplication edge*. Now, we will assume that if, for two leaves g_1 and g_2 of $L(T)$, there is a divergent duplication edge on the path between g_1 and g_2, then g_1 and g_2 will be considered 'not similar'. Conversely, we will assume that if there is no such edge on the path between g_1 and g_2, then no unexpected differentiation between g_1 and g_2 will be observed, and thus g_1 and g_2 will be considered 'similar'.

Note that here, we have not defined 'similarity' in a quantitative manner. For our purposes, we prefer to leave this notion abstract, and we will assume that divergence after duplication leaves behind some traces that can be used to detect similarity versus non-similarity (for instance, one could compare the sequence of a gene to all its homologs in a given species). This is because our goal is to evaluate the consequences of divergence in the orthology inference process, and not necessarily to detect such divergence. We thus prefer to keep the notion of similarity as general as possible. We do note however that in the future, further investigation will be needed to establish concrete methods to recognize similarity.

We now introduce divergence after duplication (DAD)-trees.

Definition 2. A DAD-tree T for Γ is a DS-tree for Γ in which each $\mathbb{D}$ node v has exactly one child v' such that the edge vv' is marked as divergent, and the other edges of T are marked as non-divergent.

Two genes $g_1, g_2 \in L(T)$ are called *similar* if there is no divergent edge on the path between g_1 and g_2 in T. Otherwise, g_1 and g_2 are called *non-similar*.

It then becomes quite easy to characterize similarity graphs, using the above notion of 'similarity'.

Proposition 3. Let G be the graph in which $V(G) = \Gamma$, and two genes share an edge if and only if they are similar with respect to a DAD-tree T. Then each connected component of G is a clique.

Therefore, unlike orthology, this notion of similarity is a transitive relation. Proposition 3 then tells us precisely what we should expect from similarity graphs when divergence after duplication always occurs: we should obtain cliques of genes. Moreover, the genes in these cliques are pairwise orthologous, as two paralogous genes cannot appear as 'similar' under the above definition. Hence, the primary orthologs resulting from a DAD-tree form the cliques in the similarity graph. For this reason, we may sometimes call *orthology clusters* the cliques of primary orthologs. This motivates the first essential step of our orthology prediction framework, which is to transform the similarity graph into a cluster graph. However, this will only identify a subset of all orthologies, as there may be inter-cluster orthologies.

One idea might be to infer orthology between the 'closest' inter-cluster genes. However, this may introduce invalid relations or inconsistencies. Here, as by Hellmuth *et al.*, (2015) and Lafond *et al.* (2016), we take the viewpoint that for a set of relations to be correct, there must be an evolutionary scenario that could have given rise to the observed relations. In other words, there should exist a *DS*-tree that displays the inferred orthologies, and the speciation nodes on this *DS*-tree should agree with the species history.

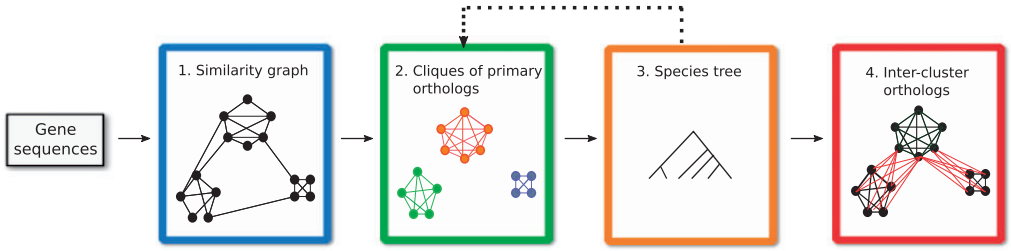

Fig. 2. The four essential steps of the HyPPO pipeline. First, a similarity graph is constructed from the gene sequences. This graph is refined into orthology clusters, which are cliques of primary orthologs. These clusters are used to infer a species tree, which is then used to find the secondary orthologs (which we also call inter-cluster orthologs)

This notion of agreement with a species tree is defined as in (Jones *et al.*, 2016; Lafond and El-Mabrouk, 2014).

Definition 4. Let T be a DS-tree and S be a species tree. Moreover, let $v \in V(T) \backslash L(T)$ be an $\mathbb{S}$ node of T, and let v_1 and v_2 be its children. Then we say that v is S-consistent if $s(v) \neq s(v_1)$ and $s(v) \neq s(v_2)$.

Furthermore, we say that T is S-consistent if each of its $\mathbb{S}$ nodes is S-consistent.

In the context of our work, the tree T is usually unknown, and only the orthology graph G is known. We use the existence/non-existence of a tree T corresponding to the relations in G to validate the orthology graph.

Definition 5. An orthology graph G is S-consistent if there exists a DS-tree T such that the two following conditions hold:

1. $ab \in E(G)$ if and only if a and b are orthologs in T;
2. T is S-consistent.

The property of S-consistency has been investigated in depth in the recent years and admits a convenient graph-theoretical characterization. First, we need to define a *triplet*. Let T be a tree, and let $x, y, z \in L(T)$ be three distinct leaves. We say that T contains the triplet $xy|z$ if $lca_T(x, y) < lca_T(x, z) = lca_T(y, z)$.

THEOREM 6. An orthology graph G is S-consistent if and only if the two following conditions hold (Lafond and El-Mabrouk (2014):

1. G is P_4-free, i.e. has no path on four vertices without a chord;
2. for every $x, y, z \in V(G)$ such that $\sigma(x), \sigma(y)$ and $\sigma(z)$ are distinct and such that $xy, yz \in E(G)$ but $xz \notin E(G)$, we have that $\sigma(x)\sigma(z)|\sigma(y)$ is a triplet of S.

Therefore, S-consistent graphs can be characterized by the absence of P_4's and a set of forbidden P_3's given by the structure of the species tree. Given the set of primary orthology clusters, our goal is to infer the inter-cluster orthologs in a way that the resulting relations could be generated from a DAD-tree in a S-consistent manner.

2.3 The HyPPO orthology framework

We may now describe the main ingredients of our orthology inference framework. The flow of the pipeline is illustrated in Figure 2. Starting only from the DNA or amino acid sequences of a given set of genes (or proteins), our ultimate goal is to identify the cliques of primary orthologs, and the orthology relations between these cliques. Recall that this framework is modular, and that an implementation can be provided for each step individually. We first provide an outline of the tasks that each step must perform and then proceed with the algorithmic details of how steps 2–4 were realized in the default implementation of our framework.

Step 1. We start by building a tentative *similarity graph*, in which two genes share an edge if they are 'similar'. The pairwise similarity scores between the genes are typically used for this step. For instance, the graph may contain an edge xy iff the BLAST score between x and y is above a certain threshold. The edges can also be weighted by this score.

Step 2. As stated in Proposition 3, we expect the above similarity graph to contain only cliques. In practice, this might of course not always be the case and so in this step, we partition the graph into cliques of pairwise orthologous genes, which are assumed to be primary orthologs. Note that virtually every similarity-based inference method performs this task, although it should be observed that our requirement of pairwise orthology is strict, and no in-paralogs are allowed in the inferred groups.

Step 3. In order to obtain the inter-cluster relations in a S-consistent manner, a species tree S is needed. If known, this step can be omitted, but otherwise, a species tree can be inferred from the orthology clusters. Indeed, as we will show, some phylogenetic signal can be extracted from these clusters.

Note that in Figure 2, step 3 points back to step 2. This is because the framework allows reinferring the orthology clusters, but this time with the additional information of the species tree. The new clusters can then be used to reinfer the species tree, and this process can be looped until convergence. Although our framework supports such a loop, we did not use it in our experimentations and leave this exploration for future work.

Step 4. In this final step, we assume that the orthology clusters and the species tree are known, but the orthologous gene pairs containing genes from distinct clusters are missing. Our goal is to recover these inter-cluster relations in a way to ensure S-consistency while preserving the properties of a DAD-tree. That is, we assume that in their history, two genes from two distinct clusters are separated by a divergent duplication edge, and that the predicted orthology relations can be explained by this type of history.

In the rest of this section, we describe how steps 2, 3 and 4 were implemented in HyPPO.

2.3.1 Prediction of orthology clusters

As partitioning a graph into cliques is a difficult algorithmic problem, most of the orthology prediction methods that perform clustering use a heuristic in order to deal with thousands, or even millions, of genes. In our case, we assume that genes are partitioned into families, which do not usually exceed a few hundred genes. In some

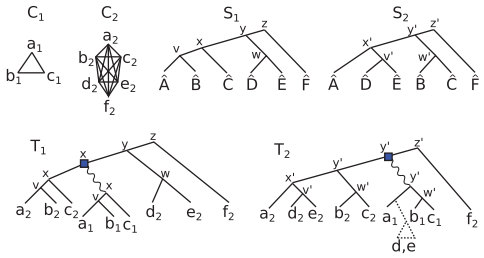

Fig. 3. Two orthology clusters C_1 and C_2, and two putative species trees S_1 and S_2. The genes are named according to their species, e.g. $\sigma(a_1) = \widehat{A}$. Each internal node u of T_1 (respectively, T_2) is labeled by its lca-mapping $s(u, S_1)$ (resp. $s(u, S_2)$). The tree T_1 is the DAD-tree that could have given rise to C_1 and C_2 if S_1 was the true species tree, assuming that C_1 was born during the history of the C_2 genes. T_2 is the DAD-tree for C_1 and C_2 under the assumption that S_2 is the true species tree. In the case of T_2, a loss in the clade $\{\widehat{D}, \widehat{E}\}$ is required to explain the birth of the C_1 cluster at a time prior to $lca_{S_2}(\sigma(C_1)) = y'$

cases, this makes it possible to use exact algorithms instead of heuristics. In fact, we provide two implementations for gene clustering: an exact method and a greedy heuristic. The exact method can be used on gene families with a reasonable number of genes (up to 200) and the heuristic for the other families.

The exact algorithm solves the cluster editing problem, which asks for the minimum number of edges to add or delete in a given graph so that every connected component of the resulting graph is a clique. We used the BBH graph, in which vertices are genes and edges are bidirectional best hits (BBH), where two genes x, y form a BBH if y is the gene of $\sigma(y)$ of maximum score with x, and vice-versa. The cluster editing problem is well-studied, and there are multiple methods that can find an optimal solution in an acceptable amount of time, including integer linear programming (ILP) formulations (Böcker *et al.*, 2011; Hartung and Hoos, 2015) and fixed-parameter algorithms (Böcker *et al.*, 2009). We used the ILP method (see Böcker *et al.* 2011 for details), ensuring that genes a and b from the same species would never belong to the same cluster by making the cost of adding the ab edge infinite.

The greedy heuristic proceeds in an agglomerative fashion using the pairwise scores. This can be seen as an adaptation of UPGMA with the restriction that clusters with a common species can never be joined. More precisely, at the start, we treat each individual gene as a cluster. Then, we merge the two clusters C_1 and C_2 if $\sigma(C_1) \cap \sigma(C_2) = \varnothing$ and if they maximize $\max_{a \in C_1, b \in C_2} \text{score}(a, b)$ (i.e. C_1 and C_2 have the two closest genes among all possible pairs of clusters). The algorithm stops when no more clusters can be merged. Note that we also tested other join criteria, for instance, joining the two clusters that minimize the average distance as in UPGMA, but the above algorithm yielded the best results.

2.3.2 Inference of a species tree from orthology clusters

We now show how orthology clusters may guide the construction of a species tree. In order to avoid confusion between species and clusters, we shall denote species with a *hat* symbol, e.g. $\widehat{C}$ is a species. Consider the example provided in Figure 3. Here, the set of species $\sigma(C_1)$ appearing in cluster C_1 forms a subset of the species $\sigma(C_2)$ from cluster C_2. This suggests that C_1 was given birth somewhere during the evolution of the C_2 cluster. Moreover, the duplication that gave rise to C_1 should have occurred in a species that existed at least as far back in time as the lca of $\sigma(C_1)$ in the true species tree S. This means that in the true gene tree for $C_1 \cup C_2$, the duplication node d must satisfy $s(d) \geq s(C_1)$ (where here s is with respect to the true species tree S). We argue that this suggests that the true species tree S contains the clade $\{\widehat{A}, \widehat{B}, \widehat{C}\}$, meaning that some node u of S has exactly $\widehat{A}, \widehat{B}$ and $\widehat{C}$ as its descendant leaves.

To justify this claim, observe that some species trees may require *losses* to explain the clusters, in a way that is analogous to gene tree and species tree *reconciliation* (see Doyon *et al.* 2011 for more details on this topic). Consider the two species tree S_1 and S_2 from Figure 3. If S_1 is the true species tree and d is the duplication node that gave birth to the C_1 cluster, then $s(d, S_1) \geq s(C_1, S_1) = x$, as argued in the previous paragraph. T_1 is an example of an S_1-consistent DAD-tree that satisfies this condition. If S_2 is the true species tree, then T_1 does not meet this requirement anymore. The tree T_2 does satisfy $s(d, S_2) \geq s(C_1, S_2) = y'$. However, $s(C_1, S_2)$ has $\widehat{A}, \widehat{B}, \widehat{C}, \widehat{D}$ and $\widehat{E}$ as its descendants, but $\widehat{D}$ and $\widehat{E}$ do not appear in C_1. This can be explained by a gene loss prior to the ancestor of $\widehat{D}$ and $\widehat{E}$, resulting in a loss in the $\{\widehat{D}, \widehat{E}\}$ clade. The tree T_1 does not require a loss to be explained, and therefore its corresponding species tree is preferred. Observe that any species tree that contains the $\{\widehat{A}, \widehat{B}, \widehat{C}\}$ clade leads to a loss-less history, suggesting that this clade should be in the species tree.

Of course, the situation gets more difficult when $\sigma(C_1)$ is not a subset of $\sigma(C_2)$, or when there are more than two clusters. Below, we formalize the problem of reconstructing a species tree from for an arbitrary set of clusters while minimizing losses.

Let C be an orthology cluster. For convenience, we will make no distinction between C and its set of species $\sigma(C)$, as this has no bearing on the inference procedure. For a species tree S and $x \in V(S)$, the *clade* of x is defined as $\text{clade}(x) = \{l \in L(S) : l \text{ is a descendant of } x\}$. The set of clades of S is $\text{clades}(S) = \{\text{clade}(x) : x \in V(S)\}$. For a given cluster C, let S_C be the subtree of S rooted at $lca_S(C)$. The number of losses of C with respect to S, denoted $l_S(C)$, is defined as the minimum number of clades to remove from $\text{clades}(S_C)$ so that the union of the remaining clades is exactly C. Another way to view $l_S(C)$ is as follows. Call a non-root node v of S_C *maximal C-free* if $\text{clade}(v) \cap C = \varnothing$ but $\text{clade}(p(v)) \cap C \neq \varnothing$. Then $l_S(C)$ is the number of maximal C-free nodes in S_C. As an example, the reader can verify in Figure 3 that $l_{S_2}(C_1) = 1$ and $l_{S_2}(C_2) = 0$. We may now formally define our species tree reconstruction problem.

> The species tree using clusters (STUC) problem:
> **Given**: a set of orthology clusters $\mathscr{C} = \{C_1, \ldots, C_k\}$.
> **Find**: a species tree S that minimizes $\sum_{i=1}^{k} l_S(C_i)$.

Denote by $OPT(\mathscr{C})$ the minimum number of losses of a species tree for the STUC instance $\mathscr{C}$. Note that $\mathscr{C}$ might not be sufficient to determine the complete species tree exactly, as in Figure 3 where any species tree containing the $\{\widehat{A}, \widehat{B}, \widehat{C}\}$ clade is optimal. However, the more clusters there are, the more precisely S can be inferred.

We do not know whether the STUC problem is NP-hard, although we suspect that is it. Nevertheless, we borrow ideas from (Mirarab *et al.*, 2014) in order to limit the space of possible species trees to those containing a predefined set of splits. For $X \subset \Sigma$, a *split* $\{A, B\}$ for X is a partition of X into two non-empty subsets. We say that S *contains* the split $\{A, B\}$ if there is a node v with children v_1, v_2 such that $A = \text{clade}(v_1)$ and $B = \text{clade}(v_2)$. Suppose that we are given a set of *allowable splits* $\mathbb{A} = \{A_i, B_i\}_{i=1}^{r}$, where for each i, A_i and B_i are disjoint subsets of Σ. Then we may ask for a species tree S that solves the STUC problem under the condition that every split

contained in S appears in $\mathbb{A}$. In the Supplementary Material, we show that this restricted version can be solved in polynomial time by dynamic programming, and then we discuss how a reasonable set $\mathbb{A}$ can be constructed.

THEOREM 7. The STUC problem with species trees restricted to the allowable splits $\mathbb{A}$ can be solved in time $O(|\mathbb{A}|^2|\mathscr{C}||\Sigma|)$.

2.3.3 Inference of inter-cluster orthology relations

In this final step, we assume that the orthology clusters $\mathscr{C}$ and the species tree S are known, but the orthologous gene pairs containing genes from distinct clusters are missing. Our goal is to recover these inter-cluster relations in a way to ensure S-consistency while preserving the properties of evolution under divergence after duplication. Note that unlike the last section, here we do not confound a cluster C and its set of species $\sigma(C)$.

Consider the orthology clusters C_1 and C_2 in Figure 3. As we have argued in the previous section, the ancestral gene of C_1 was born off a duplication node d satisfying $s(d) \geq s(C_1)$. One implication of this is that if $\sigma(g_2) < s(C_1)$, where $g_2 \in C_2$, then it is not be possible for a gene of C_1 to be orthologous to g_2. As an example, take the gene $a_2 \in C_2$ in Figure 3 and assume that S_1 and T_1 are the true species tree and gene tree, respectively. We have $\sigma(a_2) < s(C_1)$, and so a_2 is orthologous with no gene of C_1. However, this is not true for $d_2 \in C_2$, for example, because d_2 is a secondary ortholog with all of C_1. This is because $\sigma(d_2) \oplus s(C_1)$.

In our implementation, when inferring inter-cluster relations between C_1 and C_2, we simply make as many pairs of genes orthologous whilst preserving S-consistency. As earlier, we also require that if $\sigma(g_1) < s(C_2)$, then g_1 is paralogous to all the genes in C_2 (and vice-versa). This results in Algorithm 1. This works as described above in the case of $k = 2$ clusters. For larger k, we merge the clusters iteratively in a greedy manner.

More specifically, we first pick the cluster C for which $s(C)$ is the the lowest in the species tree, breaking ties arbitrarily. We then ask: what other cluster C_j could have spawned the C cluster? We call C_j the *favorite cluster* of C and assume that we are given a function *fav* $(C, \mathscr{C})$ that finds the favorite cluster of C among a set of clusters $\mathscr{C}$. We then merge C with C_j by making all possible orthologies - namely, we make $c_i \in C_j$ orthologous to every gene of C iff $\sigma(c_j) \oplus s(C)$. Afterwards, C_j is redefined as $C_j \cup C$, C is emptied and we proceed with the next cluster. We show that Algorithm 1 preserves S-consistency as desired.

Algorithm 1 Algorithm to infer inter-cluster orthologs.

1: **procedure** inferInterClusters($G, S, \mathscr{C} = \{C_1, \ldots, C_k\}, fav$) G is the initial orthology graph, S is the species tree, $\mathscr{C}$ is the set of clusters, and *fav* is a function that finds favorites. We assume that in G, two vertices share an edge iff they belong to the same cluster. The output is an orthology graph obtained from G.
2: **while** $\mathscr{C}$ has at least two non-empty clusters **do**
3: Let $C \in \mathscr{C}$ such that $s(C) \leq s(C_i)$ or $s(C) \oplus s(C_i)$ holds for every $C_i \in \mathscr{C}$
4: Let $C_j = fav(C, \mathscr{C})$.
5: **for** $g_j \in C_j$ such that $\sigma(g_j) \oplus s(C)$ **do**
6: Add an edge in G between g_j and every vertex in C.
7: $C_j \leftarrow C_j \cup C$ and $C \leftarrow \varnothing$

THEOREM 8. The orthology graph obtained from Algorithm 1 is S-consistent.

In our implementation, we used the following simple *fav* function: for each $x \in C$, find the gene $y \in \Gamma \backslash C$ of maximum score. Then x votes for the cluster that contains y. The favorite cluster of C is then the one with the highest number of votes. Only the members of the genes initially in C, as given in the input, are allowed to vote.

3 Results

We compare the default implementation of the HyPPO pipeline with two other orthology prediction methods: OMA-GETHOGS and OrthoMCL. To our knowledge, OMA-GETHOGS is the method that is the most similar to ours, as it outputs both the 1-to-1 orthology groups, and the pairwise orthology relations. The groups can be interpreted as the primary orthologs. As for OrthoMCL, it is a similarity-based clustering method that focuses on grouping orthologs, and so it is not expected to find all the secondary orthologs. Nevertheless, we include it for the sake of comparison, as OrthoMCL is one of the most popular orthology prediction tools. HyPPO was tested under two settings: one in which the species tree is unknown, and the other in which the true simulated species tree is provided. OMA-GETHOGS was provided the true species tree in all experiments. (The program does not require a species tree as input, but OMA-GETHOGS produced an error message when not given one—for reasons that are yet to be determined. Nevertheless, better results are expected when the true species tree is known.)

We performed our experiments on both simulated and real empirical datasets. In the simulated datasets, the true orthologs and the primary orthologs are known. This is not true for the real datasets. As in the study by Altenhoff *et al.* (2013), we used a set of gold standard, manually curated, gene trees from the SwissTree database (Wapinski *et al.*, 2007). These trees are annotated with duplication and speciation events, and so the underlying orthology/paralogy relations can be considered as 'gold standard' relations. We did not, however, include the analysis of the primary vs secondary orthologs, as those are not determined in the real datasets. Note that in both simulated and real datasets, the genes were already separated into homologous families. All the programs were executed on the separated gene families, thereby avoiding predictions between non-homologous genes. For HyPPO, the sequences were aligned with MAFFT (Katoh *et al.*, 2002), and the similarity values for the first step were obtained from the pairwise identity percentages computed from these alignments (number of identical base pairs divided by the length of the longest sequence, not counting gaps).

3.1 Performance metrics

In our experiments, we address two questions: is the set of all pairwise relations predicted correct? And, can the method distinguish the primary orthologs from the secondary orthologs? For the pairwise relations, we used precision, recall and accuracy as our performance metrics. The precision *Prec* and recall *Rec* values are calculated from the orthology relations as follows: $\text{Prec} = \frac{\text{TP}}{\text{TP+FP}}$ and $\text{Rec} = \frac{\text{TP}}{\text{TP+FN}}$. Here, *TP* means True Positive (the number of relations correctly predicted as orthology), *FP* means False Positive (a paralogy relation incorrectly predicted as orthology) and *FN* means False Negative (an orthology relation left undetermined or predicted as paralogy). As for accuracy, it consists in the total number of relations predicted correctly (including both orthologs and paralogs) divided by $\binom{n}{2}$, the total number of relations. (Note that

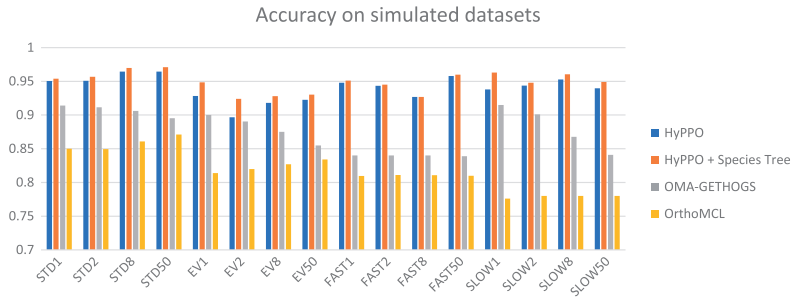

Fig. 4. Accuracy of the methods for each dataset and each branch multiplication factor $\ell \in \{1, 2, 8, 50\}$. STD stands for Standard and EV for Eventful

Ortho-MCL does not predict paralogs, aside from the recent ones. In our computation of accuracy, we interpreted as paralogy the set relations not predicted as orthology—otherwise, leaving these relations as undetermined made the accuracy too low for comparison.)

In the simulated datasets containing divergent duplication edges, we also evaluated the ability to recover the primary orthologs. The metric we used is the *cluster score* defined as follows: let P_1 and P_2 be two partitions of a given set of elements. Here, P_1 would be the predicted primary orthologs, and P_2 the 'true' simulated primary orthologs. Take a graph in which the vertices are $P_1 \cup P_2$ (i.e. one vertex for each subset), add an edge X_1X_2 between each pair $X_1 \in P_1, X_2 \in P_2$ and weigh this edge by $|X_1 \cap X_2|$. Then the cluster score between P_1 and P_2 is the value of a maximum matching in this graph, divided by the number of genes. Hence the cluster score is 1 iff P_1 and P_2 are equal. The idea behind this score is that this value yields the proportion of elements from P_1 that must be moved from one set to another in order to obtain P_2 (if the score is c, then a fraction of $1 - c$ elements from P_1 need to be moved).

3.2 Simulated datasets

For our simulations, we used SimPhy (Mallo *et al.*, 2016) to generate a set of species trees along with a set of gene trees evolving within these species trees. As SimPhy incorporates speciation and duplication events in its simulations, the true sets of orthologs and paralogs are known. We generated 40 species trees and 400 gene trees—10 gene families for each species tree. The number of species in each species tree was chosen uniformly at random between 30 and 50. The gene trees were subject to the events of speciation, duplication and losses. We then used the INDELiBLE (Fletcher and Yang, 2009) module provided with SimPhy to simulate the evolution of gene sequences on each gene tree. The nucleotide evolutionary model was selected at random for each gene family, each evolving under a general-time reversible (GTR) model with rates sampled from a six-dimensional Dirichlet distribution. The only input to the programs was the set of extant gene sequences and, depending on the method, the true simulated species tree.

In order to evaluate the impact of the rates of duplications, losses and substitutions, we used four sets of parameters for the simulations, yielding four datasets named as follows: Standard, Eventful, Fast and Slow. Standard uses default parameters, Eventful has an increased rate of duplications and losses, Fast has high mutation rates and Slow has low mutation rates. For each dataset, 10 species trees were generated. In both the Standard and Eventful datasets, the tree-wide substitution rate was set to 0.000005 (the default value in SimPhy). In the Default dataset, the duplication and loss rates were set to 0.0000005 each and to 0.000001 in the Eventful dataset. Higher duplication rates led to trees with too many genes in the trees (averaging in the thousands, making our analysis too computationally intensive) while lower values yielded gene trees with too few paralogs. In the Fast dataset, the duplication/loss rates were set as in the Standard setting, but the tree-wide substitution rate was increased to 0.00005, In the Slow dataset, this parameter took value 0.0000005. The Fast dataset led to difficulties in alignments, due to the presence of many substitutions and very large gaps, whereas the Slow dataset had a small amount of substitutions, resulting in sequences with fewer differences.

These simulations do not consider the phenomenon of divergence after duplication. That is, the children of $\mathbb{D}$ nodes in the generated gene trees do not differentiate from each other, and both copies continue to evolve under the same model. Therefore, we also simulated divergent edges by choosing, for every $\mathbb{D}$ node in the simulated gene trees, one child edge arbitrarily and multiplying its length by some factor ℓ. Thus for each gene tree and each $\ell \in \{1, 2, 8, 50\}$, we used INDELiBLE to generate a set of extant gene sequences obtained by multiplying the length each divergent edge by ℓ. Each gene family gave rise to four sets of sequences, resulting in a total of 1600 gene families to analyze. In the cases where $\ell > 1$, the primary orthologs can be deduced from the divergent duplication edges.

The average accuracy, precision, recall and cluster scores, taken over the 1600 simulated gene families, are presented in Table 1. Figure 4 shows the average accuracy per gene family, for each dataset and possible value of ℓ. HyPPO outperforms the other methods on every dataset, even when not given a species tree. Moreover, the accuracy of HyPPO always improves when the true species tree is known. We observe that the accuracy of OMA tends to decrease as ℓ is increased, whereas for HyPPO and OrthoMCL, the impact of changing ℓ seems to vary.

Figure 5 shows the average cluster scores per dataset. Note that the scores for the trees having multiplication factor $\ell = 1$ are not presented, as in these cases the primary orthologs cannot be determined with certainty. Here again, we see that HyPPO is better at identifying the primary orthologs. Observe that knowledge of the species tree is inconsequential here, as the species tree does not impact the gene clustering step in our pipeline. We also observe that for all methods, the best cluster scores are achieved at $\ell = 50$, which is expected since this creates a greater separation between the primary and secondary orthologs.

The comparison of precision and recall is shown in Figure 6 for $\ell = 8$. The other values of ℓ yielded similar results. One can see that OMA typically attains better precision values. This is at the cost of a much lower recall, as HyPPO outperforms the other methods in this

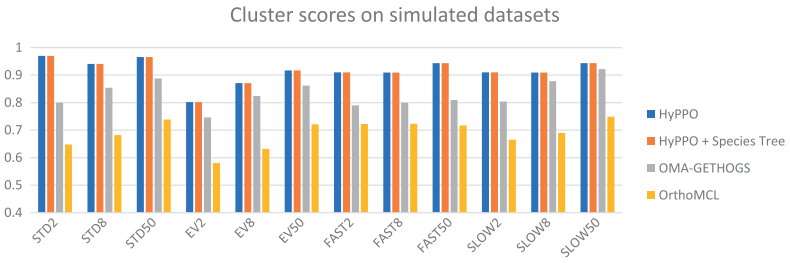

Fig. 5. Cluster scores of the methods for each dataset and each branch multiplication factor $\ell \in \{2, 8, 50\}$

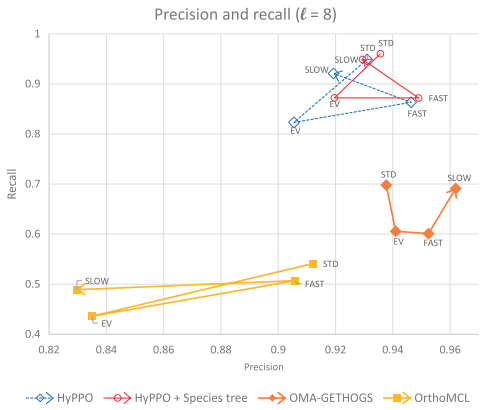

Fig. 6. Precision versus recall values of each method and each dataset for $\ell = 8$. Each method has four corresponding points, one for each dataset, and are ordered as follows: Standard—Eventful—Fast - Slow. For example the second point of each path for the analysis of the Eventful dataset

aspect. Therefore, the orthologs predicted by OMA are truly orthologs slightly more often, but the method tends to miss more of the orthology relations.

3.3 Empirical datasets

We used eight gene trees from the gold standard SwissTree database. This included the Popeye domain family (POP), the NOX 'ancestral-type' subfamily NADPH oxidases (NOX), the V-type ATPase beta subunit (VATB), the Serine incorporator family (SERC), the Sulfatase-modifying factor 1 family (SUMF), the HOX cluster genes family 9-14 (HOX), the Asterix family (ARX) and the Cited family (CITE). These gene families were chosen because they belong to Eukaryotes and, according to the curators, are not suspected to have undergone horizontal gene transfer.

Figure 7 shows the accuracy of each method for each gene tree. Contrary to the simulated datasets, HyPPO does not outperform OMA on every single dataset, as it performs poorly on the VATB and SERC families. Notably, Ortho-MCL attains almost perfect accuracy on the VATB family (0.96), whereas HyPPO is not much more than half as accurate (0.59). The reason appears to be that most gene pairs are orthologous (95.7%), since the duplications are low in this tree. HyPPO infers multiple orthology clusters joined together by duplications, resulting in many false paralogs (we predict 47% pairs as orthologous), whereas OrthoMCL puts almost all the genes into one big orthology cluster, making its predictions more accurate. A similar phenomenon appears to occur with SERC. Nevertheless, HyPPO achieves similar or superior accuracy on the other trees, and so it remains competitive. Observe that as in the simulations, when HyPPO is given the species tree (provided by SwissTree), the accuracy increases.

As for precision and recall, the four methods attained similar values. Once again, OMA offers slightly better precision at the cost of lower recall. The averages over all eight trees are presented in Table 2.

4 Discussion

Although the results of HyPPO are promising, there is still room for improvements. Each step of the pipeline could be implemented in a more sophisticated manner, and the DAD model around which HyPPO is built could be more realistic. When a duplication node is not followed by accelerated mutations, there is no special post-duplication divergent edge, and it could be argued that the orthology clusters should contain in-paralogs. In the future, we might thus consider merging the clusters that are share a high enough degree of similarity. We have also ignored the effects of convergent evolution, even though it is possible that two independent divergent duplication events lead to similar genes. If this occurs, one cannot guarantee that similar genes are orthologs, and that primary orthologs form cliques.

On the practical side, we assume that the genes are already partitioned into homologous families, which allow the computation of a multiple sequence alignement and eases the clustering phase. Future versions of the framework should be able to predict orthologs and paralogs from whole proteomes, as OrthoMCL and OMA-GETHOGS do. However, one could also argue that partitioning genes into homologous families and predicting orthology are two different tasks. The aforementioned programs perform both simultaneously. A better approach might be to use methods dedicated to homology clustering and then refine the orthology/paralogy relations within each family using HyPPO.

Also, the reconstruction of a species tree from the clusters is the main bottleneck of HyPPO in terms of time. To give a rough idea, in the worst case, some datasets resulted in a set of about 3000 allowable splits, which took our algorithm 4–5 h to solve. This motivates the need to improve the complexity of the dynamic programming formulation given in this paper. Also, in the near future, we plan to evaluate the quality of the predicted species tree. Note however that our current algorithm may take many arbitrary decisions, and we must refine this process before performing a careful species tree analysis. Finally, it will be interesting to include a wider range of orthology prediction software in a future comparative study and to establish a procedure to analyze primary orthologs on real

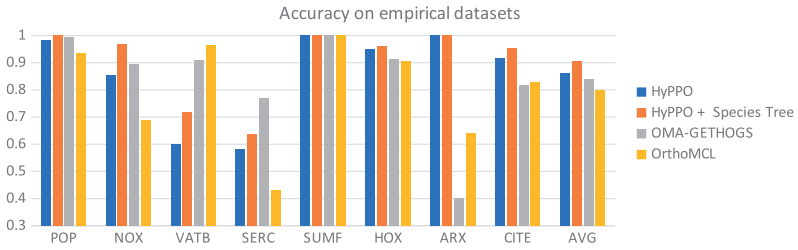

Fig. 7. Accuracy of the methods for each tree in our empirical dataset. The last column shows the average accuracy of the methods

Table 2. Average accuracy, precision and recall over the set of eight trees from SwissTree

	Accuracy	Precision	Recall
HyPPO	0.860	0.941	0.785
HyPPO + Species Tree	0.905	0.945	0.881
OMA-GETHOGS	0.837	0.950	0.711
OrthoMCL	0.800	0.859	0.680

datasets—for instance, by predicting these orthologs by comparing the ratio of branch lengths following a duplication.

5 Conclusion

In this work, we have introduced the notion of orthology in the presence of divergence after duplication and have addressed the problem that some orthologs may behave as paralogs from a functional point of view. We therefore propose to make a distinction between the orthologs that have or have not been separated by a divergent duplication event. As we have shown, the HyPPO framework is able to distinguish paralogs, primary orthologs and secondary orthologs with better accuracy than its competitors. This is true even if divergence after duplication does not always occur. Our work is far from over though, as HyPPO offers many opportunities to improve these results even further. Notably, it remains to evaluate alternative algorithms for each step in our pipeline, and to identify which ones yield the best results. Finally, our work raises some interesting questions from a theoretical standpoint. The complexity of inferring a species tree from gene clusters is left open, and it remains to evaluate how many clusters are necessary to reconstruct a high quality species tree.

Funding

M.L. was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Conflict of Interest: none declared.

References

Altenhoff,A.M. *et al.* (2012) Resolving the ortholog conjecture: orthologs tend to be weakly, but significantly, more similar in function than paralogs. *PLoS Comput. Biol.*, **8**, e1002514.

Altenhoff,A.M. and Dessimoz,C. (2012) Inferring orthology and paralogy. *Methods Mol. Biol.*, **855**, 259–279.

Altenhoff,A.M. *et al.* (2013) Inferring hierarchical orthologous groups from orthologous gene pairs. *PLoS One*, **8**, e53786.

Böcker,S. *et al.* (2009) Going weighted: parameterized algorithms for cluster editing. *Theor. Comput. Sci.*, **410**, 5467–5480.

Böcker,S. *et al.* (2011) Exact algorithms for cluster editing: evaluation and experiments. *Algorithmica*, **60**, 316–334.

Cardoso-Moreira,M. *et al.* (2016) Evidence for the fixation of gene duplications by positive selection in drosophila. *Genome Res.*, **26**, 787–798.

Chen,X. and Zhang,J. (2012) The ortholog conjecture is untestable by the current gene ontology but is supported by rna sequencing data. *PLoS Comput. Biol.*, **8**, e1002784.

Dondi,R. *et al.* (2017a) Approximating the correction of weighted and unweighted orthology and paralogy relations. *Algorithms Mol. Biol.*, **12**, 4.

Dondi,R. *et al.* (2017b) Orthology correction for gene tree reconstruction: Theoretical and experimental results. *Proc. Comput. Sci.*, **108**, 1115–1124.

Doyle,M.A. *et al.* (2010) Drug target prediction and prioritization: using orthology to predict essentiality in parasite genomes. *BMC Genomics*, **11**, 222.

Doyon,J.-P. *et al.* (2011) Models, algorithms and programs for phylogeny reconciliation. *Brief. Bioinformatics*, **12**, 392.

Emms,D.M. and Kelly,S. (2015) Orthofinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. *Genome Biol.*, **16**, 157.

Fitch,W.M. (2000) Homology: a personal view on some of the problems. *Trends Genet.*, **16**, 227–231.

Fletcher,W. and Yang,Z. (2009) Indelible: a flexible simulator of biological sequence evolution. *Mol. Biol. Evol.*, **26**, 1879–1888.

Gabaldón,T. and Koonin,E.V. (2013) Functional and evolutionary implications of gene orthology. *Nat. Rev. Genet.*, **14**, 360–366.

Hartung,S. and Hoos,H.H. (2015). Programming by optimisation meets parameterised algorithmics: a case study for cluster editing. In *International Conference on Learning and Intelligent Optimization*, pp. 43–58. Springer.

Hellmuth,M. *et al.* (2013) Orthology relations, symbolic ultrametrics, and cographs. *J. Math. Biol.*, **66**, 399–420.

Hellmuth,M. *et al.* (2015) Phylogenomics with paralogs. *Proc. Natl. Acad. Sci.USA*, **112**, 2058–2063.

Hernandez-Rosales,M. *et al.* (2012) From event-labeled gene trees to species trees. *BMC Bioinformatics*, **13**, S6.

Innan,H. and Kondrashov,F. (2010) The evolution of gene duplications: classifying and distinguishing between models. *Nat. Rev. Genet.*, **11**, 97.

Jones,M. *et al.* (2016) On the consistency of orthology relationships. *BMC Bioinformatics*, **17**, 416.

Jordan,I.K. *et al.* (2004) Duplicated genes evolve slower than singletons despite the initial rate increase. *BMC Evol. Biol.*, **4**, 1–22.

Jothi,R. *et al.* (2006) Coco-cl: hierarchical clustering of homology relations based on evolutionary correlations. *Bioinformatics*, **22**, 779–788.

Katoh,K. *et al.* (2002) Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.

Koonin,E.V. (2005) Orthologs, paralogs, and evolutionary genomics. *Ann. Rev. Genet.*, **39**, 309–338.

Kristensen,D.M. *et al.* (2011) Computational methods for gene orthology inference. *Brief. Bioinformatics*, **12**, 379–391.

Lafond,M. *et al.* (2016) The link between orthology relations and gene trees: a correction perspective. *Algorithms Mol. Biol.*, **11**, 4.

Lafond,M. and El-Mabrouk,N. (2014) Orthology and paralogy constraints: satisfiability and consistency. *BMC Genomics*, **15**, S12.

Lechner,M. *et al*. (2011) Proteinortho: detection of (co-) orthologs in large-scale analysis. *BMC Bioinformatics*, **12**, 124.

Li,L. *et al*. (2003) Orthomcl: identification of ortholog groups for eukaryotic genomes. *Genome Res*., **13**, 2178–2189.

Lynch,M. and Conery,J.S. (2000) The evolutionary fate and consequences of duplicate genes. *Science*, **290**, 1151–1155.

Mallo,D. *et al*. (2016) Simphy: phylogenomic simulation of gene, locus, and species trees. *Syst. Biol*., **65**, 334–344.

Mirarab,S. *et al*. (2014) Astral: genome-scale coalescent-based species tree estimation. *Bioinformatics*, **30**, i541–i548.

Nehrt,N.L. *et al*. (2011) Testing the ortholog conjecture with comparative functional genomic data from mammals. *PLoS Comput. Biol*., **7**, e1002073.

O'Brien,K.P. *et al*. (2004) Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Res*., **33**, D476.

Powell,S. *et al*. (2012) eggnog v3. 0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. *Nucleic Acids Res*., **40**, D284.

Roth,A.C. *et al*. (2008) Algorithm of oma for large-scale orthology inference. *BMC Bioinformatics*, **9**, 518.

Soria,P.S. *et al*. (2014) Functional divergence for every paralog. *Mol. Biol. Evol*., **31**, 984–992.

Stolzer,M. *et al*. (2012) Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics*, **28**, i409–i415.

Studer,R.A. and Robinson-Rechavi,M. (2009) How confident can we be that orthologs are similar, but paralogs differ? *Trends Genet*., **25**, 210–216.

Swenson,K.M. and El-Mabrouk,N. (2012) Gene trees and species trees: irreconcilable differences. *BMC Bioinformatics*, **13**, S15.

Tatusov,R.L. *et al*. (2003) The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, **4**, 41.

Thomas,P.D. *et al*. (2012) On the use of gene ontology annotations to assess functional similarity among orthologs and paralogs: a short report. *PLoS Comput. Biol*., **8**, e1002386.

Train,C.-M. *et al*. (2017) Orthologous matrix (oma) algorithm 2.0: more robust to asymmetric evolutionary rates and more scalable hierarchical orthologous group inference. *Bioinformatics*, **33**, i75–i82.

Ullah,I. *et al*. (2015) Integrating sequence evolution into probabilistic orthology analysis. *Syst. Biol*., **64**, 969–982.

Van der Heijden,R.T. *et al*. (2007) Orthology prediction at scalable resolution by phylogenetic tree analysis. *BMC Bioinformatics*, **8**, 83.

Wapinski,I. *et al*. (2007) Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics*, **23**, i549–i558.

Woods,S. *et al*. (2013) Duplication and retention biases of essential and non-essential genes revealed by systematic knockdown analyses. *PLoS Genet*., **9**, e1003330.

Zhang,J. (2003) Evolution by gene duplication: an update. *Trends Ecol. Evol*., **18**, 292–298.

Zheng,C. *et al*. (2011) Omg! orthologs in multiple genomes–competing graph-theoretical formulations. In *International Workshop on Algorithms in Bioinformatics*. Springer, pp. 364–375.

Supplementary material

Proof of Proposition 3

It suffices to show that for any triple of distinct vertices u, v, w, $uv \in E(G)$ and $vw \in E(G)$ imply that $uw \in E(G)$. Let E_{uv}, E_{vw} and E_{wu} denote the set of edges on the unique $u-v$, $v-w$ and $w-u$ paths in T, respectively. In T, one can get from w to u by first going from w to v, then from v to u. This implies that $E_{wu} \subseteq E_{vw} \cup E_{uv}$. As none of the edges in $E_{vw} \cup E_{uv}$ is marked as divergent, by definition, we have that u and w are similar, and thus uw is an edge of G.

An algorithm for STUC with allowable splits

We describe how to solve the STUC problem with allowable splits using dynamic programming. Given $X \subseteq \Sigma$, a set of clusters $\mathscr{C}$ and allowable splits $\mathbb{A}$, denote by $OPT(X, \mathscr{C}, \mathbb{A})$ the minimum number of losses in a species tree S, but only counting the lost clades that are strict subsets of X, and subject to the conditions that (1) $X \in clades(S)$ and (2) S has only splits from $\mathbb{A}$. To rephrase more formally, let $l_S(C, X)$ be the number of maximal C-free nodes v of the species tree S_C with the additional condition that $clade(v) \subset X$. Then $OPT(X, \mathscr{C}, \mathbb{A})$ is the minimum of $\sum_{C \in \mathscr{C}} l_S(C, X)$ over all possible species trees that contain the X clade (and only splits from $\mathbb{A}$). Define $OPT(X, \mathscr{C}, \mathbb{A}) = \infty$ if no such species tree exists. Clearly, $OPT(\Sigma, \mathscr{C}, \mathbb{A})$ yields a solution to our restricted STUC instance, since every loss occurs at a clade that is a strict subset of Σ, and any species tree contains the Σ clade. We would like to devise a recurrence formula for $OPT(\Sigma, \mathscr{C}, \mathbb{A})$.

Towards this goal, first let $\{A, B\}$ be a split of $X \subseteq \Sigma$ and let $C \in \mathscr{C}$. Define the cost $l_{A,B}(C)$ of C with respect to $\{A, B\}$ as

$$l_{A,B}(C) = \begin{cases} 1 & \text{if } C \cap (\Sigma \setminus X) \neq \emptyset, \text{and exactly one of} \\ & C \cap A \text{ or } C \cap B \text{ is empty} \\ 0 & \text{otherwise} \end{cases}$$

In words, we count a loss for $l_{A,B}(C)$ is C has something outside $X = A \cup B$, and C loses all of A or all of B. We can now state our recurrence:

Lemma 1. *If $|X| = 1$, then $OPT(X, \mathscr{C}, \mathbb{A}) = 0$. If $|X| > 1$, then either $OPT(X, \mathscr{C}, \mathbb{A}) = \infty$, or $OPT(X, \mathscr{C}, \mathbb{A})$ is equal to*

$$\min_{\{A,B\} \in spl(X) \cap \mathbb{A}} \left(OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} l_{A,B}(C) \right)$$

where $spl(X)$ is the set of all possible splits of X.

Proof. We prove the Lemma by induction on $|X|$. As a base case, suppose $|X| = 1$. Then for any species tree S and any cluster C, $l_S(C, X)$ must be zero since S does not contain a C-free node v such that $clade(v) \subset X$ (as otherwise, $clade(v) = \emptyset$). Now suppose that the Lemma holds for any strict subset of X, and assume $OPT(X, \mathscr{C}, \mathbb{A}) \neq \infty$. Let S be a species tree containing a node v satisfying $clade(v) = X$ and having every split in $\mathbb{A}$, chosen such that S minimizes $\sum_{C \in \mathscr{C}} l_S(C, X)$. Let v_1, v_2 be the children of v and let $\{A, B\} = \{clade(v_1), clade(v_2)\}$. Then $\{A, B\} \in spl(X) \cap \mathbb{A}$. Since in the recurrence, the range of splits over which we take the minimum contains $\{A, B\}$, it suffices to prove that $OPT(X, \mathscr{C}, \mathbb{A}) = OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} l_{A,B}(C)$.

For some $C \in \mathscr{C}$, let F_A^C (resp. F_B^C) be the set of maximal C-free nodes w of S_C such that $clade(w) \subset A$ (resp. $clade(w) \subset B$). Moreover, let F^C be the set of maximal C-free nodes that belong to $\{v_1, v_2\}$. Then $OPT(X, \mathscr{C}, \mathbb{A}) = \sum_{C \in \mathscr{C}} l_S(C, X) = \sum_{C \in \mathscr{C}} (|F_A^C| + |F_B^C| + |F^C|)$. To prove the Lemma, we will show that $\sum_{C \in \mathscr{C}} |F_A^C| = OPT(A, \mathscr{C}, \mathbb{A})$, $\sum_{C \in \mathscr{C}} |F_B^C| = OPT(B, \mathscr{C}, \mathbb{A})$ and $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$.

To achieve this, we first show that the subtrees of S rooted at v_1 and v_2 must incur $OPT(A, \mathscr{C}, \mathbb{A})$ and $OPT(B, \mathscr{C}, \mathbb{A})$ losses, respectively. Consider a species tree S' that contains a node v' satisfying $clade(v') = A$, that has every split in $\mathbb{A}$, and such that $\sum_{C \in \mathscr{C}} l_{S'}(C, A) = OPT(A, \mathscr{C}, \mathbb{A})$. Note that S' exists since, in particular, S is a possible solution. Let $S'_{v'}$ be the subtree of S' rooted at v'. In S, replace the S_{v_1} subtree rooted at v_1 by the $S'_{v'}$ subtree. Observe that this new tree obtained has cost $OPT(A, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} |F_B^C| + \sum_{C \in \mathscr{C}} |F^C|$, implying that $\sum_{C \in \mathscr{C}} |F_A^C| = OPT(A, \mathscr{C}, \mathbb{A})$. By a similar reasoning, $\sum_{C \in \mathscr{C}} |F_B^C| = OPT(B, \mathscr{C}, \mathbb{A})$.

So far, we have shown that $OPT(X, \mathscr{C}, \mathbb{A}) = OPT(A, \mathscr{C}, \mathbb{A}) + OPT(B, \mathscr{C}, \mathbb{A}) + \sum_{C \in \mathscr{C}} |F^C|$. It remains to argue that $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$. Fix some $C \in \mathscr{C}$. If $s(C) < v$ or $s(C) \oplus v$, then neither of v_1 and v_2 can be a maximal C-free node. If $s(C) = v$, the same holds, because $s(C) = v$ implies that C contains an element from both $clade(v_1)$ and $clade(v_2)$. In both these cases, $F^C = \emptyset$. Accordingly, $l_{A,B} = 0$ in these cases (because when $s(C) \leq v$, we have $C \cap (\Sigma \setminus X) = \emptyset$, and when $s(C) \oplus v$, both $C \cap A$ and $C \cap B$ are empty). Assume that $s(C) > v$. It follows that $C \cap (\Sigma \setminus X) \neq \emptyset$, and also that $C \cap X \neq \emptyset$. The latter implies that v_1 and v_2 cannot both be maximal C-free nodes. If only v_1 is C-free maximal, then $F^C = \{v_1\}$. Accordingly, $l_{A,B}(C) = 1$, since $C \cap A = \emptyset$. The same holds if only v_2 is maximal C-free. Finally, if none of v_1 and v_2 are maximal C-free, then $F^C = \emptyset$, and accordingly $l_{A,B}(C) = 0$. In every case, we have equality between $|F^C|$ and $l_{A,B}(C)$. This shows that $\sum_{C \in \mathscr{C}} |F^C| = \sum_{C \in \mathscr{C}} l_{A,B}(C)$, concluding the proof. □

Observe that $spl(X) \cap \mathbb{A}$ can be computed in time $O(|\mathbb{A}||X|)$, as it suffices to iterate over every element of $\mathbb{A}$ and check whether the split partitions X.

In the next section, we prove Theorem 7, i.e. that the above dynamic programming formulation leads to an algorithm that runs in time $O(|\mathbb{A}|^2 |\mathscr{C}||\Sigma|)$.

We construct the set $\mathbb{A}$ in a heuristic fashion as follows. We build a collection D of subsets of Σ, and $\mathbb{A}$ consists of all the pairs $\{D_1, D_2\}$ where $D_1, D_2 \in D$ and $D_1 \cap D_2 = \emptyset$. To construct D, sort the species of Σ by non-increasing order of *frequency*, where the frequency of x is the number of clusters of $\mathscr{C}$ that contain x. This results in an ordering $(s_1, \ldots, s_m)$ of Σ. Then for any $i, j \in [m]$ with $i < j$, add the set $\{s_i, s_{i+1}, \ldots, s_j\}$ to D. For good measure, we also add every cluster $C_i \in \mathscr{C}$ to D, concluding the construction. The idea behind the ordering is that if a set of species have a similar frequency, it is possibly because they always appear together in the same clusters. This set may thus be part of some split of S.

Proof of Theorem 7

Consider the algorithm that starts computing $OPT(\Sigma, \mathscr{C}, \mathbb{A})$ and recursively computes $OPT(X, \mathscr{C}, \mathbb{A})$ for the necessary values of X according to the formulation of Lemma 1. Assume that $OPT(X, \mathscr{C}, \mathbb{A})$ is computed once for each value of X, and stored in memory if this value is required later on. The value of $OPT(X, \mathscr{C}, \mathbb{A})$ only needs to be computed if $\mathbb{A}$ contains a pair of the form $\{X, Y\}$. Therefore, at most $2|\mathbb{A}|$ computations of $OPT(X, \mathscr{C}, \mathbb{A})$ are necessary. The time required to compute $OPT(X, \mathscr{C}, \mathbb{A})$, without counting the time for the recursive calls, corresponds to the time required to compute $l_{A,B}(C)$ for each $C \in \mathscr{C}$ and each $\{A, B\} \in spl(X) \cap \mathbb{A}$. The value $l_{A,B}(C)$ only needs to compute intersections over subsets of Σ and thus takes time $O(|\Sigma|)$. Since there are at most $|\mathbb{A}|$ splits in $spl(X) \cap \mathbb{A}$, the time required for

$OPT(X, \mathscr{C}, \mathbb{A})$ is $O(|\mathbb{A}||\mathscr{C}||\Sigma|)$. As mentioned before, X can take $2|\mathbb{A}|$ values, and the overall complexity is $O(|\mathbb{A}|^2|\mathscr{C}||\Sigma|)$.

Proof of Theorem 8

First note that if $\sigma(x) = \sigma(y)$, then no edge will ever be added between x and y, due to the condition in line 5. Hence no two genes from the same species are ever made orthologs. We must prove that the graph output by the algorithm has no forbidden P_3 and no P_4. Beforehand we make an important observation. Consider an iteration, when some cluster C is merged with its favorite cluster C_j. Let $z \in C_j$. Then either z becomes orthologous with every member of C, or with none of them. Moreover, these relations between z and C remain until the end of the algorithm.

Now, suppose that vertices x, y, z introduce a P_3 in G (with edges xy and yz) at the i-th iteration, for some $i \in [k-1]$. Assume that $\sigma(x), \sigma(y), \sigma(z)$ are all distinct. We must prove that $\sigma(x)\sigma(z)|\sigma(y)$ is a triplet of S, as required by S-consistency. Let the clusters $C_1, \ldots, C_k$ be the clusters at the beginning of the i-th iteration (in particular, there are $i-1$ empty clusters due to line 7). To avoid ambiguity, if we need to refer to some cluster C_l at the beginning of the h-th iteration, we will write $C_l^{(h)}$ (hence, under our notation, we use C_l as a shorthand for $C_l^{(i)}$). Now, let C be the cluster chosen at the i-th iteration, and let C_j be the favorite cluster of C chosen at this iteration. Since only edges between C and C_j are added at this i-th iteration, C and C_j must both contain one of $\{x, y, z\}$.

Assume that x, y and z all belong to a distinct cluster at the i-th iteration, say the clusters C, C_j and C_h. Then there is an edge going from C_h to at least one of C or C_j. This edge was added in an iteration prior to i, implying that one of C, C_j or C_h must have been emptied, a contradiction. We may therefore assume that C and C_j contain all of $\{x, y, z\}$. We check all the possible locations of x, y and z:

- If $x, z \in C$ and $y \in C_j$, then $\sigma(y) \oplus s(C)$ (because every $y-C$ edge must have been added by the algorithm). Moreover, $\sigma(x), \sigma(z) \leq s(C)$ (by the definition of $s(C)$). This implies $\sigma(x)\sigma(z)|\sigma(y)$ is a triplet of S, as required.
- If $x, y \in C$ and $z \in C_j$, z is a neighbor of y but not x. But z should be a neighbor of every member of C, or of none of them. Hence z cannot form a P_3 with x and y. Similarly, we cannot have $z, y \in C$ and $x \in C_j$.
- If $x \in C$, $y, z \in C_j$, then $\sigma(y) \oplus s(C)$, $\sigma(x) \leq s(C)$ and $\sigma(z) \leq s(C)$ (otherwise zx would be an edge). This implies the $\sigma(x)\sigma(z)|\sigma(y)$ triplet in S, as required. The same idea works if $z \in C$ and $x, y \in C_j$.
- Finally, suppose that $y \in C$ and $x, z \in C_j$. Then $\sigma(x) \oplus \oplus s(C)$ and $\sigma(z) \oplus s(C)$. For the sake of contradiction, assume w.l.o.g. that $\sigma(y)\sigma(x)|\sigma(z)$ is a triplet of S (the other contradicting triplet is $\sigma(y)\sigma(z)|\sigma(x)$, and the same proof applies). Since x and z do not share an edge, they did not belong to the initial set of given clusters. Then there must have been some iteration $h < i$ such that $C_j^{(h)}$ was the favorite of some cluster $C_l^{(h)}$, with either $x \in C_j^{(h)}, z \in C_l^{(h)}$ or vice-versa. Suppose $x \in C_j^{(h)}, z \in C_l^{(h)}$. As no edge was added between x and z, we had $\sigma(x) \leq s(C_l^{(h)})$. Thus $s(C_l^{(h)})$ is a common ancestor of $\sigma(x)$ and $\sigma(z)$. Since $\sigma(y)\sigma(x)|\sigma(z)$ is a triplet of S and $\sigma(x), \sigma(z) \oplus s(C)$, we have $s(C_l^{(h)}) > s(C)$. However, the algorithm iterates over the clusters in non-decreasing order with respect to S, a contradiction since C should have been handled before C_h at the h-th iteration. The same argument holds for the case $z \in C_j^{(h)}, x \in C_l^{(h)}$.

This proves that every P_3 satisfies the requirement of Theorem 6.

Now, suppose that vertices w, x, y, z introduce a P_4 at the i-th iteration (with edges wx, xy, yz), at which point C is merged with its favorite C_j. Observe that as above, we may assume that C and C_j contain all of $\{w, x, y, z\}$. Notice also that C_j cannot contain only one vertex from this P_4: this vertex must either be adjacent to the 3 others, or to none of them, both cases preventing a P_4. Also, C_j cannot contain exactly two of these vertices. Indeed, if $w, x \in C_j$ and $y, z \in C$, then since xy is an edge, x shares an edge with every member of C. This includes the xz edge, a contradiction. One can check that the type of argument holds for the other five cases $w, y \in C_j$, $w, z \in C_j$, $x, y \in C_j$, $x, z \in C_j$ and $y, z \in C_j$ in which C_j has exactly two vertices of the P_4.

It follows that C_j must have 3 vertices. In particular, we either have both $x, z \in C_j$, or only one of x or z is in C_j. Suppose $x, y \in C_j$ and w.l.o.g. $w \in C, z \in C_j$. Then $\sigma(y) \leq s(C), \sigma(z) \leq s(C)$ and $\sigma(x) \oplus s(C)$. Note that this implies $\sigma(x) \neq \sigma(z)$. Moreover, $\sigma(x) \neq \sigma(y) \neq \sigma(z)$, since xy and yz are edges. These fact imply that $\sigma(y)\sigma(z)|\sigma(x)$ is a triplet of S, and that the x, y, z P_3 should be forbidden, contradicting the fact that every P_3 is consistent. It remains to cover the case when only one of x or y is in C_j. Suppose, w.l.o.g. that $x \in C$, $w, y, z \in C_j$. Note that $\sigma(w) \oplus s(C)$, $\sigma(y) \oplus s(C)$ and $\sigma(z) \leq s(C)$. As we did before, consider the iteration h at which w and z both ended up in C_j. Then either $w \in C_j^{(h)}$ and $z \in C_l^{(h)}$ for some l, or vice-versa. If $w \in C_j^{(h)}$ and $z \in C_l^{(h)}$, then $\sigma(w) \leq s(C_l^{(h)})$. Thus $C_l^{(h)}$ is a common ancestor of $\sigma(w)$ and $\sigma(z)$, which implies $s(C_l^{(h)}) > s(C)$ since $\sigma(w) \oplus s(C)$ and $\sigma(z) \leq s(C)$. This contradicts the ordering of the iterations of the algorithm. If $z \in C_j^{(h)}$ and $w \in C_l^{(h)}$, then $\sigma(z) \leq s(C_l^{(h)}), s(C_l^{(h)})$ is a common ancestor of $\sigma(w)$ and $\sigma(z)$, and we reach the same contradiction. This concludes the proof.

Chapter 6

Conclusion

Apart from some enumerative calculations about overlapping reversals by Sturtevant and Novitski [34] in 1941, the mathematical study of breakpoints and genome rearrangements operations dates from Watterson *et al.* in 1982 [36]. This was some 60 years after the discovery of experimental demonstration of reversals [33] by Sturtevant himself following the establishment of the chromosome theory of heredity by Thomas Hunt Morgan in 1910 [26]. The Watterson article was neglected for some years, while the algorithmic study of rearrangements was taken up in the Sankoff lab [31] and in the the Pevzner lab [2] in the early 1990. After its discovery in 1995, the Hannenhalli-Pevzner approach dominated mathematical approaches to genome rearrangement for a decade [13], at which point it was supplanted by DCJ [37], [6].

The combinatorial optimization approach of this work was not conducive to probability modeling as an analytical way of calculating genomic distance, because of the nature of the rearrangement operations. Although, such models are readily simulated. Thus the escape from parsimony can be observed and measured, for various values of the model parameters, but not proved mathematically. The breakthrough by Beresticki and Durret [4] showed that this could be overcome, at least for very simple rearrangements consisting only of pairwise transpositions of genes. This operation, however, does not have any biological counterpart. This motivates our research, namely to explore the Berestycki and Durret insight, and to try to adapt it to biologically well-motivated problems.

In the model studied by Berestycki and Durret, the distance measure between a genome and its ancestor depends on the cyclic decomposition of the permutation representing the genome, which approximates the number of tree components of the Erdös-Rényi graph constructed during the random process.

By leveraging Beresticki and Durret's insight and adopting their method to biologically more related problems (such as DCJ, HP, IRT, etc., including multichromosomal versions) and constructing the random graph based on a different definition of the graph representation of the genomes, we proved the number of cycles in the

breakpoint graph of two genomes for our models also depend on the a random graph, which is an Erdös-Rényi for genomes with large number of genes. Moreover, we proved some of the the other parameters in the distance measures applied in our models are negligible for large genomes.

In this thesis, we have achieved our goals with respect to both restricted and unrestricted DCJ, IRT and HP and have made some progress with translocation, two models of weighted reversal, and maximum length reversals.

Future work should try to complete these latter analyses, and investigate the applications of our result as to the median problem [20], the small phylogeny problem and other problems previously studied in the combinatorial optimization context.

Genomes have more than one copy of some genes, which gives rise to the "orthology detection" problem. In parallel to the permutation-based investigations described in Chapters 2-4, in Chapter 5 we also undertook an analysis of the orthology problem under a particular model of contrasting mutation rates. We constructed an algorithm, implemented software, and illustrated this problem with realistic examples.

Appendix A

Algorithms

Algorithm 1 Reversal Random Walk

Input: Two integers n and R representing the number of genes belonging to a linear unichromosomal genome, and the number of reversal operations, respectively.

Output: Three integers, the reversal distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, and the number of tree components of the random graph G.

Define the identity genome g_0 with $2n+2$ vertices, and for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $j = 1 : R$ **do**
    Choose two edges $e = (a, b)$ and $e' = (c, d)$ from genome g
    Remove e and e' from genome g
    Add (a, c) and (b, d) to genome g
    if g has two components **then**
        Remove (a, c) and (b, d) from genome g
        Add (a, d) and (b, c) to genome g
    end if
    Let $x = min\{a, b, c, d\}$
    Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
    Join the two vertices labelled by e and e' in G if they are not already connected
    Report the number of tree components of G
    Translate g into its equivalent representation

17: Via UniMoG report the reversal distance between genome g and the identity genome g_0
18: Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

1. $g' \leftarrow g$
2. Remove all $(2i + 1, 2i + 2)$ edges in g'
3. Add $(2i, 2i + 1)$ edges to g'

19: Report the number of cycles in g'
20: **end for**

Algorithm 2 Maximum Length Reversal Random Process

Input: Three integers n, R, and M $(1 \leq M \leq n)$ representing the number of genes belonging to a linear unichromosomal genome, the number of reversal operations, and the maximum length of reversals, respectively.

Output: Three integers, the reversal distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, and the number of tree components of the random graph G.

Define the identity genome g_0 with $2n+2$ vertices, and for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $j = 1 : R$ **do**
    Choose an edge $e = (a, b)$ and from genome g
    Find the shortest distance of all black edges from e i.e. the number of red edges separating them in g
    Choose an edge $e' = (c, d)$ such that there are at most M red edges separating e and e'
    Remove e and e' from genome g
    Add (a, c) and (b, d) to genome g
    if g has two components **then**
        Remove (a, c) and (b, d) from genome g
        Add (a, d) and (b, c) to genome g
    end if
    Let $x = min\{a, b, c, d\}$
    Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
    Join the two vertices labelled by e and e' in G if they are not already connected
    Report the number of tree components of G
    Translate g into its equivalent representation
    Via UniMoG report the reversal distance between genome g and the identity genome g_0
    Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

    1. $g' \leftarrow g$
    2. Remove all $(2i+1, 2i+2)$ edges in g'
    3. Add $(2i, 2i+1)$ edges to g'

    Report the number of cycles in g'
end for

Algorithm 3 Weighted Reversal Random Process: Step Model

Input: Two integers n and R representing the number of genes belonging to a linear unichromosomal genome, and the number of reversal operations, respectively.

Output: Three integers, the reversal distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, and the number of tree components of the random graph G.

Define the identity genome g_0 with $2n+2$ vertices, and for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $j = 1 : R$ **do**
    Choose an edges $e = (a, b)$ from genome g
    Find the shortest distance of all black edges from e that is the number of red edges separating them from e
    Remove e from genome
    Find the maximum distance and denote it by m_e
    Let p_k for $1 \leq k \leq m_e$ be the probability of choosing an edge with distance k from the edge e
    Make a probability vector for the distance $\{1, 2, \ldots, m_e\}$ such that the greater the distance the smaller is the probability
    Define a distribution function F based on the probability vector
    Choose an edge $e' = (c, d)$ randomly using distribution F from genome g, in case there exist two edges with the same distance we randomly pick one
    Remove e' from genome g
    Add (a, c) and (b, d) to genome g
    if g has two components **then**
        Remove (a, c) and (b, d) from genome g
        Add (a, d) and (b, c) to genome g
    end if
    Let $x = min\{a, b, c, d\}$
    Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
    Join the two vertices labelled by e and e' in G if they are not already connected
    Report the number of tree components of G
    Translate g into its equivalent representation

Via UniMoG report the reversal distance between genome g and the identity genome g_0
Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

1. $g' \leftarrow g$
2. Remove all $(2i+1, 2i+2)$ edges in g'
3. Add $(2i, 2i+1)$ edges to g'

Report the number of cycles in g'
end for

Algorithm 4 Weighted Reversal Random Process: Uniform Model

Input: Two integers n and R representing the number of genes belonging to a linear unichromosomal genome, and the number of reversal operations, respectively.

Output: Three integers, the reversal distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, and the number of tree components of the random graph G.

Define the identity genome g_0 with $2n+2$ vertices, such that for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $j = 1 : R$ **do**
Choose an edges $e = (a, b)$ from genome g
Find the shortest distance of all black edges from e that is the number of red edges separating them from e
Remove e from genome
Find the maximum distance and denote it by m_e
Let p_k for $1 \leq k \leq m_e$ be the probability of choosing an edge with distance k from the edge e
Make a probability vector for the distance $\{1, 2, \ldots, m_e\}$ such that the probability increase up to distance $\frac{m_e}{2}$ and then decrease, such that $p_1 = p_{m_e}$ and are the lowest
Define a distribution function F based on the probability vector
Choose an edge $e' = (c, d)$ randomly using distribution F from genome g, in case there exist two edges with the same distance we randomly pick one
Remove j from genome g
Add (a, c) and (b, d) to genome g
if g has two components **then**
Remove (a, c) and (b, d) from genome g
Add (a, d) and (b, c) to genome g
end if
Let $x = min\{a, b, c, d\}$

Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
Join the two vertices labelled by e and e' in G if they are not already connected
Report the number of tree components of G
Translate g into its equivalent representation
Via UniMoG report the reversal distance between genome g and the identity genome g_0
Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

1. $g' \leftarrow g$
2. Remove all $(2i + 1, 2i + 2)$ edges in g'
3. Add $(2i, 2i + 1)$ edges to g'

Report the number of cycles in g'
end for

Algorithm 5 Unrestricted DCJ Random Process with Probability p (UniChromosomal Identity Genomes)

Input: 2 integers n and D, representing the number of genes belonging to a linear unichromosomal genome, and the number of DCJ operations, respectively. A real number p, where $0 \leq p \leq 1$, representing the probability of producing circular chromosomes.

Output: The DCJ distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, the number of tree components of the random graph G, the number of circular chromosomes. The maximum length of the circular chromosomes.

Define the identity genome g_0 with $2n+2$ vertices, such that for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $k = 1 : D$ **do**
    Choose two edges $e = (a, b)$ and $e' = (c, d)$ from genome g
    $Comp(g)$ = number of components of g
    Remove e and e' from g
    coin $\leftarrow$ uniformly pick a random number in [0,1]
    Decision $\leftarrow 0$
    if $e = (a, b)$ and $e' = (c, d)$ are in the same component of g **then**
        $g' \leftarrow g$
        Add (a, c) and (b, d) to g'
        $Comp(g')$ = number of components of g'
        if $Comp(g') > Comp(g)$ **then**
            Decision $\leftarrow 1$
        else
            Decision $\leftarrow 2$
        end if
    end if
    if (Decision $!= 0$ and $0 \leq$ coin $\leq p$) **then**
        if Decision $== 1$ **then**
            Add (a, c) and (b, d) to g
        else
            Add (a, d) and (b, c) to g
        end if
    else if (Decision $!= 0$ and $p <$ coin ≤ 1) **then**
        if Decision $== 2$ **then**
            Add (a, c) and (b, d) to g
        else
            Add (a, d) and (b, c) to g
        end if

else if coin ≤ 0.5 **then**
    Add (a, c) and (b, d) to genome g
else
    Add (a, d) and (b, c) to genome g
end if
Let $x = min\{a, b, c, d\}$
Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
Join the two vertices labelled by e and e' in G if they are not already connected
Report the number of tree components of G
Translate g into its equivalent representation
Via UniMoG report the DCJ distance between genome g and the identity genome g_0
Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

1. $g'' \leftarrow g$
2. Remove all $(2i + 1, 2i + 2)$ edges in g''
3. Add $(2i, 2i + 1)$ edges to g''

Report the number of cycles in g''
Report the maximum length of the circular chromosomes
end for

Algorithm 6 Restricted DCJ Random Process with Probability p (UniChromosomal Identity Genomes)

Input: 2 integers n and D', representing the number of genes belonging to a linear unichromosomal genome, and the number of restricted DCJ operations, respectively. A real number p, where $0 \leq p \leq 1$, representing the probability of producing circular chromosomes.

Output: The DCJ distance of the genome g from the identity genome g_0, the number of cycles of the breakpoint graph g and g_0, the number of tree components of the random graph G.

Define the identity genome g_0 with $2n+2$ vertices, such that for $i \in \{0, 1, \ldots, n\}$:

1. Connect two vertices $2i$ and $2i+1$ in g_0 by a black edge and label it $i+1$
2. Connect two vertices $2i+1$ and $2i+2$ in g_0 by a red edge

Translate genome g_0 to its equivalent representation
Define the random graph G with no edge on the vertex set $\{1, 2, \ldots, n+1\}$
for $k = 1 : D'$ **do**
    $Comp(g)$ = number of components of g
    coin $\leftarrow$ uniformly pick a random number in [0,1]
    if $Comp(g) == 1$ **then**
        Choose two edges $e = (a, b)$ and $e' = (c, d)$ from genome g
        Remove e and e' from g
        Decision $\leftarrow 0$
        $g' \leftarrow g$
        Add (a, c) and (b, d) to g'
        $Comp(g')$ = number of components of g'
        if $Comp(g') > Comp(g)$ **then**
            Decision $\leftarrow 1$
        else
            Decision $\leftarrow 2$
        end if
        if (Decision $!= 0$ and $0 \leq$ coin $\leq p$) **then**
            if Decision $== 1$ **then**
                Add (a, c) and (b, d) to g
            else
                Add (a, d) and (b, c) to g
            end if
        else if (Decision $!= 0$ and $p <$ coin ≤ 1) **then**
            if Decision $== 2$ **then**
                Add (a, c) and (b, d) to g
            else
                Add (a, d) and (b, c) to g
            end if
        end if
    else if $Comp(g) > 1$ **then**
        Choose $e = (a, b)$ from the linear chromosome and $e' = (c, d)$ from the circular chromosome

Remove e and e' from g
if coin ≤ 0.5 **then**
Add (a, c) and (b, d) to genome g
else
Add (a, d) and (b, c) to genome g
end if
end if
Let $x = min\{a, b, c, d\}$
Assign two label e and e' to the two new edges in g such that the one incident to x is labelled by e
Join the two vertices labelled by e and e' in G if they are not already connected
Report the number of tree components of G
Translate g into its equivalent representation
Via UniMoG report the DCJ distance between genome g and the identity genome g_0
Construct the breakpoint graph of g and g_0, such that for $i \in \{0, 1, \ldots, n\}$

1. $g'' \leftarrow g$
2. Remove all $(2i + 1, 2i + 2)$ edges in g''
3. Add $(2i, 2i + 1)$ edges to g''

end for

Algorithm 7 DCJ Random Process with Probability p (Multichromosomal Identity Genomes)

Input: N and D, representing the number of linear chromosomes, and the number of DCJ operations, respectively. A set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes in each chromosome, and a real number $p \in [0, 1]$ representing the probability of producing circular chromosomes.

Output: The DCJ distance of the genome g from the identity genome g_0, the number of circular chromosomes, the maximum length of the circular chromosomes.

Construct the identity genomes g_0 with N linear chromosomes, whose numbers of genes in its chromosomes are $\{n_1, \ldots, n_N\}$
Translate genome g_0 to its equivalent representation
for $k = 1 : D$ **do**
    Choose two edges $e = (a, b)$ and $e' = (c, d)$ from genome g
    $Comp(g)$ = number of components of g
    Remove e and e' from g
    coin $\leftarrow$ uniformly pick a random number in [0,1]
    Decision $\leftarrow 0$
    if $e = (a, b)$ and $e' = (c, d)$ are in the same component of g **then**
        $g' \leftarrow g$
        Add (a, c) and (b, d) to g'
        $Comp(g')$ = number of components of g'
        if $Comp(g') > Comp(g)$ **then**
            Decision $\leftarrow 1$
        else
            Decision $\leftarrow 2$
        end if
    end if
    if (Decision $!= 0$ and $0 \leq$ coin $\leq p$) **then**
        if Decision $== 1$ **then**
            Add (a, c) and (b, d) to g
        else
            Add (a, d) and (b, c) to g
        end if
    else if (Decision $!= 0$ and $p <$ coin ≤ 1) **then**
        if Decision $== 2$ **then**
            Add (a, c) and (b, d) to g
        else
            Add (a, d) and (b, c) to g
        end if
    else if coin ≤ 0.5 **then**
        Add (a, c) and (b, d) to genome g
    else
        Add (a, d) and (b, c) to genome g
    end if
    Translate g into its equivalent representation

37: Via UniMoG report the DCJ distance between genome g and the identity genome g_0
38: Report the number of circular chromosomes
39: Report the maximum length of circular chromosomes
40: **end for**

Algorithm 8 Graph Representation of the Multichromosomal Identity Genome

Input: An integer N, representing the number of chromosomes, and a set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes in each chromosome.

Output: Graph representation of the identity genome g_0.

1: $L = [n_1, \ldots, n_N]$
2: $n \leftarrow n_1 + \ldots + n_N$
3: Define a temporary graph g_0' with $2(n + N)$ vertices labelled by $\{0, 1, \ldots, 2(n + N) - 1\}$:

   1. Connect two vertices $2i$ and $2i + 1$ in g_0' by a black edge and label it $i + 1$
   2. Connect two vertices $2i + 1$ and $2i + 2$ in g_0' by a red edge

4: Let E be the set representing the adjacencies among telomers of g in g_0'
5: **for** i in range $(N - 1)$ **do**:
6: E.append(2*(sum(L[:i+1])+i+1)-1),2*(sum(L[:i+1])+i+1)))
7: **end for**
8: $g_0 \leftarrow g_0'/E$

Algorithm 9 HP Random Process

Input: N and H, representing the number of linear chromosomes, and the number of HP operations, respectively. And a set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes in each chromosome.

Output: The HP distance of the genome g from the identity genome g_0.

Construct the identity genomes g_0 with N linear chromosomes, whose numbers of genes are $\{n_1, \ldots, n_N\}$
Translate genome g_0 to its equivalent representation
for $k = 1 : H$ **do**
    Choose two edges $e = (a, b)$ and $e' = (c, d)$ from genome g
    $Comp(g)$ = number of components of g
    if $e = (a, b)$ and $e' = (c, d)$ are in the same component of g **then**
        Remove e and e' from g
        $g' \leftarrow g$
        Decision $\leftarrow 0$
        Add (a, c) and (b, d) to g'
        $Comp(g')$ = number of components of g'
        if $Comp(g') > Comp(g)$ **then**
            Decision $\leftarrow 1$
        end if
        if Decision $== 0$ **then**
            Connect (a, c) and (b, d) to g
        else
            Connect (a, d) and (b, c) to g
        end if
    else
        Remove e and e' from g
        coin $\leftarrow$ uniformly pick a random number in [0,1]
        if coin ≤ 0.5 **then**
            Add (a, c) and (b, d) to genome g
        else
            Add (a, d) and (b, c) to genome g
        end if
    end if
    Translate g into its equivalent representation
    Via UniMoG report the HP distance between genome g and the identity genome g_0
end for

Algorithm 10 IRT Random Process

Input: N and I, representing the number of linear chromosomes, and the number of IRT operations, respectively. And a set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes in each chromosome.

Output: The HP distance of the genome g from the identity genome g_0.

Construct the identity genomes g_0 with N linear chromosomes, whose numbers of genes are $\{n_1, \ldots, n_N\}$
Translate genome g_0 to its equivalent representation
for $k = 1 : I$ **do**
    Choose two non-telomeric edges $e = (a, b)$ and $e' = (c, d)$ from genome g
    $Comp(g)$ = number of components of g
    if $e = (a, b)$ and $e' = (c, d)$ are in the same component of g **then**
        Remove e and e' from g
        $g' \leftarrow g$
        Decision $\leftarrow 0$
        Add (a, c) and (b, d) to g'
        $Comp(g')$ = number of components of g'
        if $Comp(g') > Comp(g)$ **then**
            Decision $\leftarrow 1$
        end if
        if Decision $== 0$ **then**
            Connect (a, c) and (b, d) to g
        else
            Connect (a, d) and (b, c) to g
        end if
    else
        Remove e and e' from g
        coin $\leftarrow$ uniformly pick a random number in [0,1]
        if coin ≤ 0.5 **then**
            Add (a, c) and (b, d) to genome g
        else
            Add (a, d) and (b, c) to genome g
        end if
    end if
    Translate g into its equivalent representation
    Via UniMoG report the HP distance between genome g and the identity genome g_0
end for

Algorithm 11 Translocation Random Process

Input: N and T, representing the number of linear chromosomes, and the number of Translocation operations, respectively. And a set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes in each chromosome.

Output: The DCJ distance of the genome g from the identity genome g_0. And a set of integers $\{n_1, \ldots, n_N\}$ representing the number of genes belonging to each chromosome.

Construct the identity genomes g_0 with N linear chromosomes, whose numbers of genes are $\{n_1, \ldots, n_N\}$
Translate genome g_0 to its equivalent representation
for $k = 1 : T$ **do**
    Choose two edges $e = (a, b)$ and $e' = (c, d)$ from different chromosomes of genome g
    coin $\leftarrow$ uniformly pick a random number in [0,1]
    if coin ≤ 0.5 **then**
        Add (a, c) and (b, d) to genome g
    else
        Add (a, d) and (b, c) to genome g
    end if
    Translate g into its equivalent representation
    Via UniMoG report the DCJ distance between genome g and the identity genome g_0
end for

Bibliography

[1] V. Bafna and P.A. Pevzner. Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of x chromosome. In *Mol. Biol. and Evol.* Citeseer, 1995.

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

[3] S. Bérard, A. Chateau, C. Chauve, C. Paul, and E. Tannier. Computation of perfect dcj rearrangement scenarios with linear and circular chromosomes. *Journal of Computational Biology*, 16(10):1287–1309, 2009.

[4] N. Berestycki and R. Durrett. A phase transition in the random transposition random walk. *Probability theory and related fields*, 136(2):203–233, 2006.

[5] A. Bergeron, J. Mixtacki, and J. Stoye. On sorting by translocations. In *Annual International Conference on Research in Computational Molecular Biology*, pages 615–629. Springer, 2005.

[6] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *International Workshop on Algorithms in Bioinformatics*, pages 163–173. Springer, 2006.

[7] P. Biller, L. Guéguen, and E. Tannier. Moments of genome evolution by double cut-and-join. *BMC Bioinformatics*, 16(14):1–10, 2015.

[8] B. Bollobás. *Random graphs*. Cambridge university press, 2001.

[9] G. Bourque and P.A. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Research*, 12(1):26–36, 2002.

[10] M.D. Braga, E. Willing, and J. Stoye. Double cut and join with insertions and deletions. *Journal of Computational Biology*, 18(9):1167–1184, 2011.

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

[12] T. Chen and S. Skiena. Sorting with fixed-length reversals. *Discrete Applied Mathematics*, 71(1-3):269–295, 1996.

[13] G. Fertin, A. Labarre, I. Rusu, S. Vialette, and E. Tannier. *Combinatorics of genome rearrangements*. MIT press, 2009.

[14] A. Hagberg, D. Schult, and P. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[15] M. Haghighi. *Application of Combinatorial Optimization Techniques in Genomic Median Problems, PhD thesis*. University of Ottawa, 2012.

[16] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71(1-3):137–151, 1996.

[17] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of IEEE 36th annual foundations of computer science*, pages 581–592. IEEE, 1995.

[18] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM (JACM)*, 46(1):1–27, 1999.

[19] R. Hilker, C. Sickinger, C.C. Pedersen, and J. Stoye. Unimog—a unifying framework for genomic distance calculation and sorting based on dcj. *Bioinformatics*, 28(19):2509–2511, 2012.

[20] A. Jamshidpey and D. Sankoff. Phase change for the accuracy of the median value in estimating divergence time. In *BMC Bioinformatics*, volume 14, pages 1–6. Springer, 2013.

[21] G. Jean and M. Nikolski. Genome rearrangements: a correct algorithm for optimal capping. *Information Processing Letters*, 104(1):14–20, 2007.

[22] J.D. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Symposium on Discrete Algorithms*, volume 604, page 613. Citeseer, 1995.

[23] J.D. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. In *Annual Symposium on Combinatorial Pattern Matching*, pages 87–105. Springer, 1993.

[24] M. Lafond, M. Meghdari Miardan, and D. Sankoff. Accurate prediction of orthologs in the presence of divergence after duplication. *Bioinformatics*, 34(13):i366–i375, 2018.

[25] M. Meghdari Miardan, A. Jamshidpey, and D. Sankoff. Escape from parsimmony of a double-cut-and-join genome evolution process. *Journal of Computational Biology*, 2021, submitted.

[26] T.H. Morgant. Chromosomes and heredity. *The American Naturalist*, 44(524):449–496, 1910.

[27] M. Ozery-Flato and R. Shamir. Two notes on genome rearrangement. *Journal of Bioinformatics and Computational Biology*, 1(01):71–94, 2003.

[28] P.A. Pevzner. *Computational molecular biology: an algorithmic approach*. MIT press, 2000.

[29] R.Y. Pinter and S. Skiena. Genomic sorting with length-weighted reversals. *Genome Informatics*, 13:103–111, 2002.

[30] H. Pishro-Nik. *Introduction to probability, statistics, and random processes*. Kappa Research LLC, 2014. `https://www.probabilitycourse.com`.

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

[32] J.C. Setubal and J. Meidanis. *Introduction to computational molecular biology*. Number 04; QH506, S4. PWS Pub. Boston, 1997.

[33] A.H. Sturtevant. A case of rearrangement of genes in drosophila. *Proceedings of the National Academy of Sciences of the United States of America*, 7(8):235, 1921.

[34] A.H. Sturtevant and E. Novitski. The homologies of the chromosome elements in the genus drosophila. *Genetics*, 26(5):517, 1941.

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

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

[37] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.