Determining realistic scenarios for genome rearrangement events under biological constraints

by

Farhana Zaman Glory

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
January 2022

Thesis advisor

Olivier Tremblay-Savard

Author

Farhana Zaman Glory

Determining realistic scenarios for genome rearrangement events under biological constraints

Abstract

Genome organization is known to evolve by undergoing evolutionary events such as rearrangement operations, which essentially invert or translocate regions of the genomes. To measure similarities and differences between genomes, several algorithms have been proposed to calculate distances and to find parsimonious evolutionary scenarios. However, maximum parsimony approaches propose solutions that might not represent the true evolutionary distance between genomes, as evolution may not necessarily take the shortest path. In order to infer more realistic scenarios, more information must be considered.

In this thesis, I explore the use of intergenic regions in a model that uses Bayesian inference to produce realistic evolutionary scenarios between two genomes. The model presented here, which considers only reversal events, shows promising results and paves the way for the introduction of more events in the near future, such as translocations, duplications, deletions, fissions and fusions.

Contents

List of Figures

List of Tables

Acknowledgments

I am specially grateful to my thesis advisor, committee members, my parents, significant other, and the people who were there with me through the journey. I would like to express my gratitude to Professor Dr. Olivier for guiding me through this dissertation, always encouraging me and always being patient. I also extend my thanks to Professor Dr. Mike and Professor Dr. Parimala, whose comments on this dissertation were constructive. Furthermore, I will thank my parents for their constant encouragement. In the end, I would love to thank my husband, Miju, for being there through this entire ordeal and for always keeping me calm and inspiring me when I felt down. The love and support he has given me these past years are beyond words.

Dedicated to my parents and dear husband!

Chapter 1

Introduction

1.1 Genomes

In molecular biology, a genome, the blueprint for life, is the genetic material of an organism. It comprises nucleotide sequences of deoxyribonucleic acid (DNA). A genome is a complete chain of the nucleotide base pairs (Adenine, Cytosine, Guanine, and Thymine) that compose an individual's chromosomes. The genome of eukaryotes is formed by a set of chromosomes which are contained inside a nucleus. Prokaryotes, like bacteria, don't have a nucleus and typically their genomes are formed of a single circular chromosome. The genomes of viruses, which are technically not living organisms, can be made up of DNA or ribonucleic acid (RNA).

Human genomes contain approximately three billion chemical base pairs, whereas bacterial genomes such as *E. coli.* contain about five million base pairs. The number of genes, which are the regions of the genomes that are coding for proteins for example, also varies a lot between species. For example, the *E. coli.*, human, and rice genomes

have an estimated 4500, 20000 and 50000 genes respectively.

1.1.1 DNA structure, gene orientation and transcription

In the DNA double helix, two strands of DNA running in the inverse direction coil around each other. Thus they form the definitive spiral structure (see figure 1.1). Each strand's end portions are formed by deoxyribose sugar. The ends are called as the 5′ (five prime) end, and the 3′ (three prime) end.

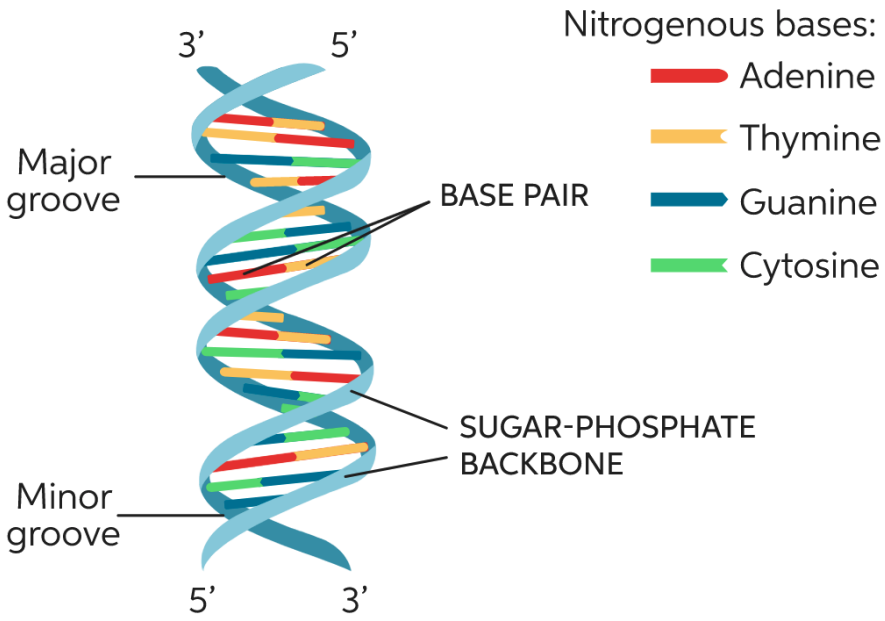

Figure 1.1: The image of DNA double helix (from [1])

Genes have an orientation on the chromosomes: they can be on any of the two DNA strands. In gene orders, signs (+ or -) are used to indicate the transcriptional orientation of each gene (+ is generally omitted). Minus (-) sign means the reverse orientation. In transcription process a DNA strand is copied into a messenger RNA

(mRNA). This mRNA then becomes a blueprint for the synthesis of protein in the course of another process named translation (see figure 1.2 for a representation of these last two processes). Transcription of the genes begins at the 5′ end of the genes to the 3′ end direction.

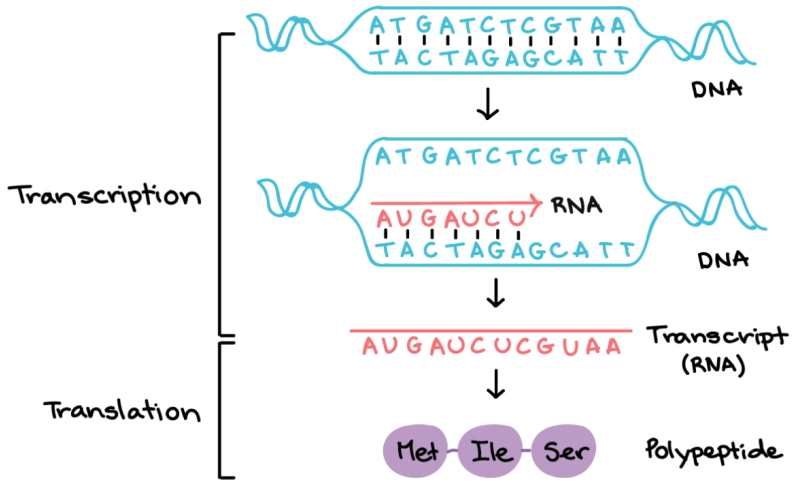

Figure 1.2: Gene transcription and translation (image taken from [2])

1.2 Genome evolution

Genome evolution is a biological process where a genome structure, sequence and/or size changes over time. Various mechanisms, i.e., duplications, gene insertions, deletions, inversions, transpositions, fissions, and fusions, contribute to genome evolution. These mechanisms are described briefly below.

Gene duplication or chromosomal duplication is a mechanism through which specific genetic material gets doubled during molecular evolution. In figure 1.3 we see that in the represented chromosome we have four (a, b, c, and d) coding genes. Only

genes c and d get doubled after the duplication event affecting those genes took place in that chromosome. The duplicated segment in this example (containing the copies c' and d') was placed right next to the original copies.

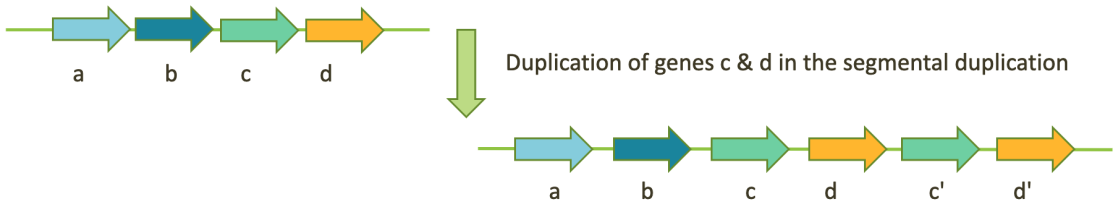

Figure 1.3: Example of a duplication event

The term transposition refers to when a chromosomal part is transferred to another position on the same or some other chromosome. For example, in figure 1.4, we can see that in a chromosome, we have three coding genes (a, b and e) sequentially. Then the transposition happens to gene b, which is transferred to the latter position of gene e, whereas it was situated before gene e.

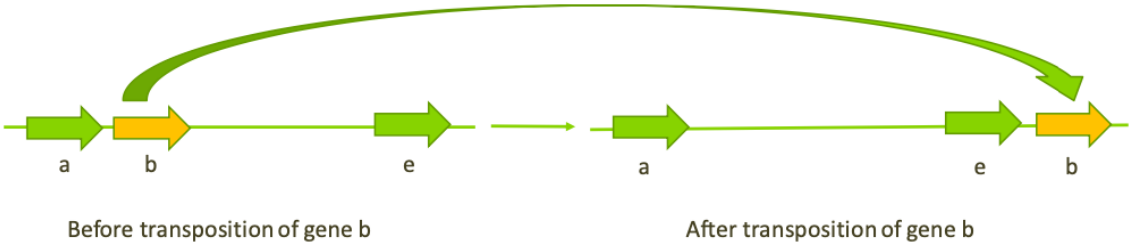

Figure 1.4: Genome Transposition

An insertion means adding one or more nucleotide base pairs into DNA sequences. This event can happen due to slipped strand mispairing during replication. A deletion is when a part of a chromosome or a sequence of DNA is left out during DNA replication, or when a gene has lost its function after accumulating mutations. In figure 1.5, we see that gene c has been removed from the chromosome after the deletion

event.

Figure 1.5: Gene Deletion

The term translocation refers to an exchange of genetic material between chromosome ends. In this event, two detached fragments of two different chromosomes are switched. In figure 1.6, we can see that in the first chromosome we have five coding genes (a, b, c, d, and e), and in the second chromosome, we have three different coding genes (f, g, and h). After the translocation happens to genes (d, e) of the first chromosome with gene h of the second chromosome, we can see that genes are shuffled in both chromosomes. In the first chromosome, we now have four coding genes (a, b, c, h). We have four coding genes (f, g, d, and e) in the second chromosome. Fission and fusion are two specific cases of genome translocation. Chromosomal fission and fusion are processes by which a whole chromosome is fissured into two separate chromosomes, and two chromosomes are fused into a whole chromosome, respectively.

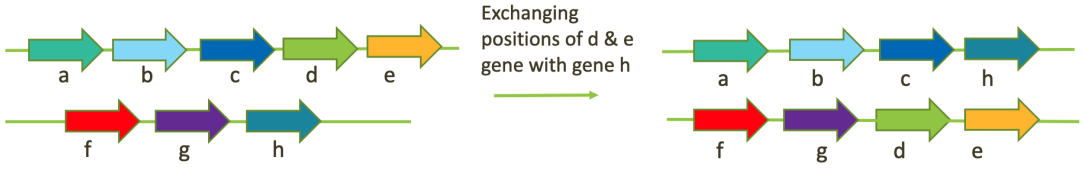

Figure 1.6: Genome Translocation

To understand the concept of gene inversions, we need to remember first how the genes are organized in the DNA on one of the two opposite strands. If an inversion

takes place on a gene, the orientation of that gene will be flipped. A condition in which a chromosome segment is reversed is called an inversion event. In addition to reversing the target segment, inversion events also change the orientation of the genes involved. For example in figure 1.7, the inversion of a chromosomal segment $X = a\ b\ c\ d$ produces the segment $-X = -d\ -c\ -b\ -a$.

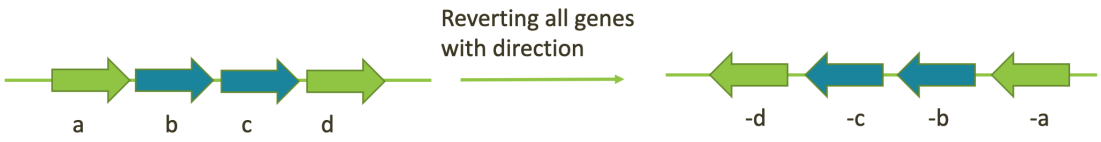

Figure 1.7: Genome inversion

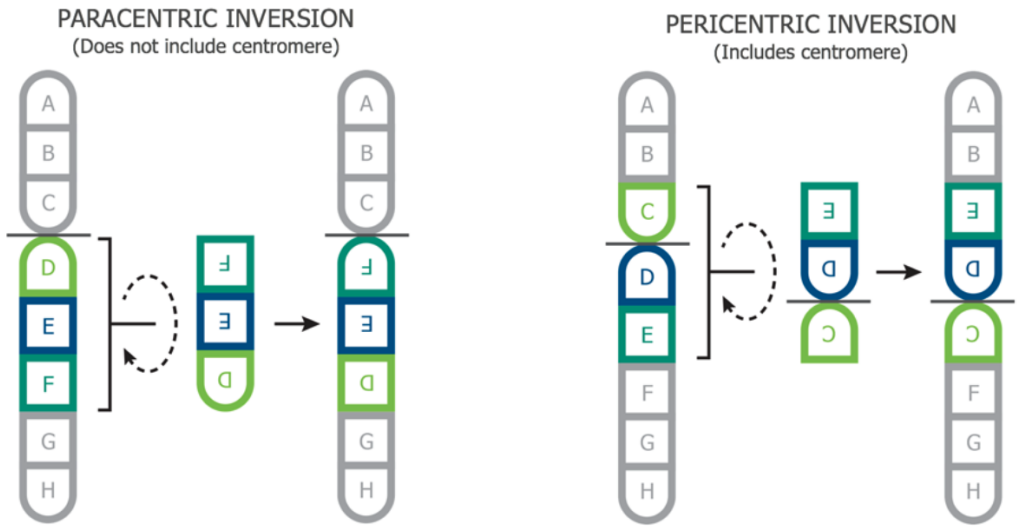

Figure 1.8: Two different types of inversions (image taken from [3])

A centromere is the central point of the chromosomes, where homologous chromosomes are attached. Inversions are of two types based on the location of the centromere. One type is called paracentric, and another type is pericentric. In paracentric inversions, the inversion occurs in one side of the centromere not including

the centromere. In pericentric inversions, the inversion includes the centromere with a breakpoint in every arm. In figure 1.8 we can see both types of inversions.

1.3 Distances and evolutionary scenarios

To find out the evolutionary scenario, scientists have to figure out the sequences of these events and also the number of events. Thus, the concepts of genetic distance and sorting genome problems have originated from this necessity. Calculating the genetic distance helps to know the number of events required for transforming one genome to another, which gives us a time estimation from when two biological sequences have diverged.

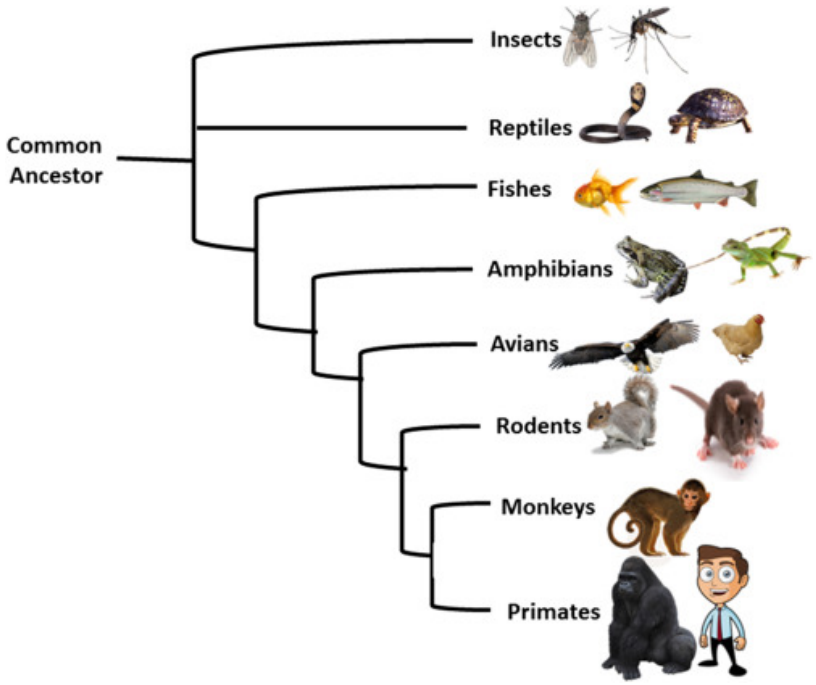

Figure 1.9: Phylogenetic tree (image taken from [4])

These measures of distances are also vital to build phylogenetic trees[5]. A phy-

logenetic tree (in figure 1.9) is a diagram or a tree having branches shows the evolutionary correspondence among various species. The field of study related to building phylogenetic trees is called phylogenomics. In other words, phylogenomics is the study of evolutionary correspondence among genes, individuals or species.

There are different ways and models to calculate distances between sequences or gene orders. Among them, the *edit distance* quantifies how dissimilar two biological sequences (DNA/RNA or protein sequences) are to one another by computing the lowest number of edit operations required to transform one into the other. The edit operations are: insertion, deletion, and substitution. This is also called *Levenshtein* distance. The edit operations for the edit distance are removal, insertion, or substitution of a single character in the nucleotides or amino acid sequences. Dynamic programming is applied to evaluate the edit distance efficiently, and the Wagner-Fischer algorithm is commonly used in this field. Nevertheless, this algorithm can only determine the count of insertions, deletions, and substitutions but can not determine the count of other evolutionary events, i.e., reversals, translocations, duplications that convert one genome into another. So the concept of the rearrangement distance has originated from this necessity.

The rearrangement distance refers to the lowest number of inversions, translocations, fusions and fissions required to convert one genome into the other. To calculate this distance efficiently for both circular and linear genomes, we need to use the breakpoint graphs. The breakpoint graph is a specific kind of diagram that represents the adjacencies of genes (located next to each other) in both genomes that are being compared. We need the breakpoint graph to get the number of rearrangement events

for mutating one genome into another. A breakpoint graph is drawn according to their gene adjacencies from the given genomes. Genes that are appearing at the end of a chromosome are said to be connected to the *telomere*, and telomere adjacencies are not represented in the breakpoint graph. Adjacencies in both genomes are represented by edges, using a different color for each genome. This results in a graph that contains some cycles and paths of alternating edge color, as seen in figure 1.10.

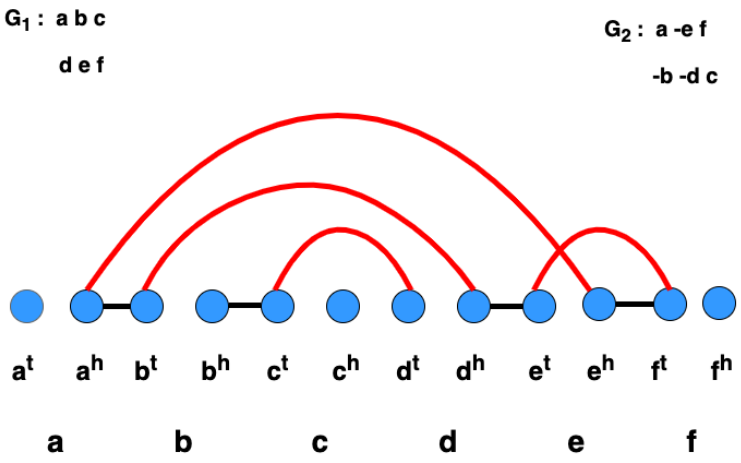

Figure 1.10: Breakpoint Graph

To evaluate the rearrangement distance between the couple of genomes, we need these factors, i.e., gene numbers in both genomes, the number of chromosomes in any genome, the number of cycles, and the number of pathways involved in the breakpoint graph. In figure 1.10, a breakpoint graph of two linear genomes has been shown where both genomes G_1 and G_2 have two chromosomes containing three genes each. Every gene has two ends: (a^h, b^h, c^h, d^h, e^h, f^h) for head and (a^t, b^t, c^t, d^t, e^t, f^t) for tail, and the tail (or head) represents the start (respectively the end) of the gene. This particular breakpoint graph shown in figure 1.10 has only one complete cycle starting from a^h to b^t and some paths.

Another problem related to genome rearrangements is called *sorting by reversals*, which is to find the specific order of reversal events that transforms one genome into another with the minimum number of events. This sorting by reversal problem is of two kinds: it can be signed or unsigned based on knowing or not the relative direction of gene transcription. Another version of this problem for genome rearrangement is sorting permutations by transposition, which is equivalent to finding out the sequence of transposition events between the given permutation and the identity permutation by sorting.

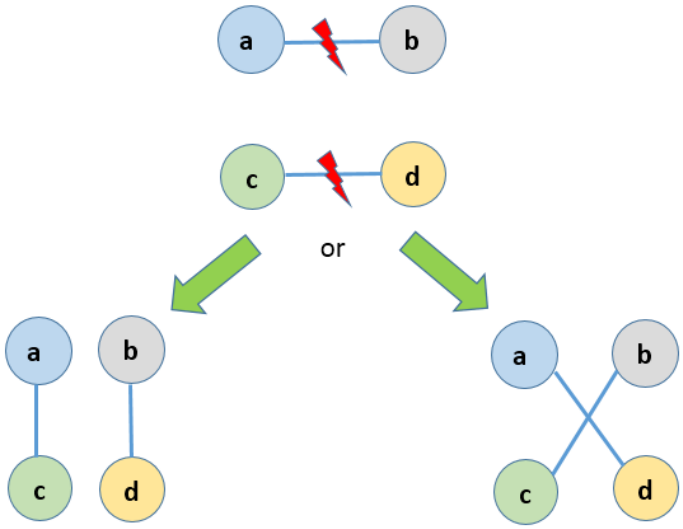

Figure 1.11: Double Cut and Join

After the rearrangement distance, a new linear genome reorganization model named *DCJ* was introduced [6]. The full form of *DCJ* is Double Cut and Join. In the *DCJ* model, genomes are modified by the following operation: two gene adjacencies are cut, and then the four open ends are pasted back together to form new adjacencies. This operation is shown in figure 1.11. Here, the adjacencies of (a and b) and (c and d) are broken by two cuts. Then it creates the possibility of two other

new adjacencies by two joinings: either ((a and c), and (b and d)) or ((a and d), and (b and c)). This model can represent inversions, transpositions, block interchanges (a generalized transposition event, where two segments swap positions), and translocations (including fusions and fissions). We can calculate DCJ distance in linear time using a breakpoint graph [7], and the distance formula is simpler than one of the rearrangement distance.

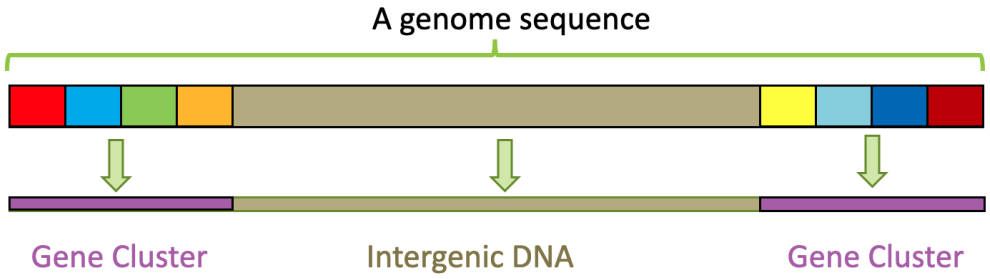

Figure 1.12: Intergenic regions

However, it has recently been argued that these models mentioned above could underestimate the accurate evolutionary distance. So additional genome features or biological information may need to be considered in the models to get around this problem. One way is to consider models that measure rearrangements according to their likelihood of happening. Another way can be to use intergenic regions along with the coding gene orders in the genomes. Now let us describe what these intergenic regions are. Particularly, consecutive genes in a genome are detached by intergenic regions, which are non-coding DNA sequences between genes having different sizes (in terms of the number of nucleotides). An intergenic region (IGR) is a span of DNA sequences or is a subset of non-coding DNA situated between coding genes (figure 1.12). These are located between the end and beginning of two consecutive coding

regions. For this reason, some information about the gene regulation processes can be present in the intergenic regions. Most of the intergenic DNA sequences have no known function. In humans, intergenic regions take in about 50% of the genome, whereas they are comparatively less in quantity in bacteria (15%) and yeast (30%)[8] genomes. Recently some researchers wanted to use biological factors to determine realistic rearrangement events. Fertin et al. [9], and Bulteau et al. [10] worked on intergenic size distribution, whereas Simonaitis et al. [11, 12] presented models that considered the 3D spatial distance between breakpoint regions. Fertin et al. [13] proved that their output genomes with intergenic regions were closer to the actual scenarios than the outputs with non-intergenic regions.

Researches for ancestral genome rearrangements by Bayesian approaches also opened a different gateway for solving rearrangement problems. Previously parsimony approaches (the idea that the smallest number of character changes is most likely to be exact) were very familiar. Then probabilistic methods got popular. Larget et al. [14], Miklós et al. [15] both used Bayesian inference methods to solve genome rearrangement problems. Though their rearrangement mechanisms were different (Larget et al. used phylogeny whereas Miklós et al. used genome reversals), both relied on statistical methods. Another work [16] got published in 2018, which also solved this problem using Phylogenetic trees based on the sequence alignments. They also used the Bayesian inference model. So we see that until now considering intergenic regions in this Bayesian approach is an open question for the researchers. Incorporating intergenic regions with this problem will be a significant contribution to the bioinformatics field.

To find out the more realistic evolutionary scenarios, my goal is to develop a model based on **Bayesian inference** handling genome rearrangement events (for now genome reversals only) in this thesis. Moreover, I integrated intergenic regions in the evolutionary model, which have never been used in other existing Bayesian models in the past.

1.4 Motivation

There are three specific research works that motivated us to do our research. One is Larget et al.'s work [14], the other two are Miklos et al.'s work [15] and Fertin et al.'s work [9]. Let me discuss how these works have paved the way to our research.

In 2005, Larget et al. [14] have developed a Bayesian framework **BADGER** to estimate phylogeny and ancestral rearrangements. In their framework, gene reversal was the sole mechanism of converting the source genome into the target one. As their framework was based on a Bayesian approach, they used the Markov Chain Monte Carlo method for the resampling process. Their method tried to find out the minimum reversal distance between the source and target genomes.

In 2009, Miklos et al. [15] proposed a method called **MC4Inversion** that gave the set of all minimum length-sorting paths as output for the genome rearrangement problem. Their method used a parallel Markov chain Monte Carlo formulation. They proved that their approach was faster than the **BADGER** method.

Some years later, researches found out that sometimes the existing models based on parsimony for genome rearrangement problems can underestimate the true evolutionary distance. So, Fertin et al. in 2017 [9] used a biological factor, i.e. intergene

size distributions, for finding a minimum weighted Double cut and Join scenario. They solved their weighted DCJ problem in polynomial time.

Like Fertin et al.'s work, we also are interested to find out realistic scenarios for the genome rearrangement problem, so we are using intergenic region sizes. Like Larget and Miklos et al.'s works, we are also interested in using a Bayesian Inference model for transforming one genome into another.

All of the three works were based on inferring minimum length scenarios, but we did not want to consider a parsimony approach. Rather than finding out the minimum rearrangement distance, we were more interested in finding out the realistic rearrangement distance. We believe that, in some cases, the longer reversal sequences also can be closer to the realistic scenarios based on the biological information. By finding more realistic scenarios of genome evolution, the goal is to better understand the characteristics of the most frequent evolutionary events that are affecting genomic sequences. This can ultimately shed light on which chromosomal regions are more likely to be affected by rearrangements, the average sizes of these events and how much the parsimony-based approaches are potentially underestimating the evolutionary distances.

1.5 Problem Statement

In the field of comparative genomics, parsimony is a fundamental principle used in many problems related to the inference of evolutionary histories. Indeed, the ability to infer optimal scenarios minimizing the total number of evolutionary changes allows to establish a lower bound on the distance between species and ancestors. However,

parsimony models do not necessarily represent the most likely evolutionary scenarios. Therefore, in this thesis, we are trying to explore the space of evolutionary scenarios that are not necessarily optimal but more realistic by taking into account an additional type of biological information.

More specifically, our goal is to infer a realistic inversion scenario between two genomes. To do so, we are considering the intergenic region size between coding genes in the genomes being compared.

Input:

Two gene orders (strings of numbers, in which each number represents a gene in a genome) which contain intergenic region sizes (integers) in between each gene. The two gene orders represent a source genome and a target genome, and the intergenic region sizes start with a preceding * sign in the input/output text format (see figure 1.13 for an example).

Output:

A set of most probable evolutionary scenarios, *i.e.* sequences of inversions that can transform the source gene order into the target one.

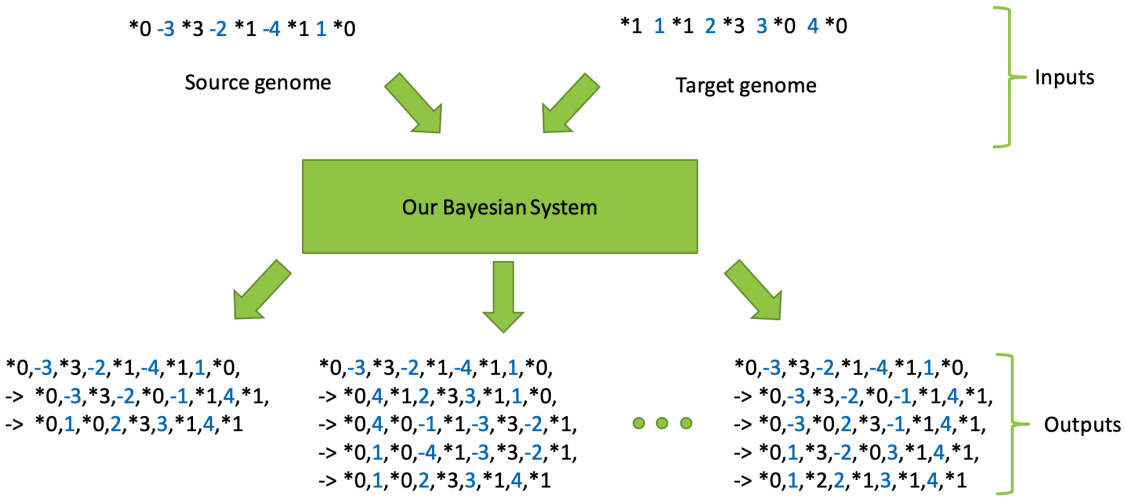

Figure 1.13: Inputs and outputs of our Bayesian System

The rest of this dissertation is ordered as follows: chapter 2 overviews various research works on genome rearrangement problems and their evolution over time. Then, in chapter 3 the system methodology is thoroughly discussed. In chapter 4, we have all the experimental results. Finally, I will bring to an end to this thesis in chapter 5.

Chapter 2

Literature Review

A brief summary of significant related works in the field of genome rearrangement is given in a tabular form:

Literature	Year	Topic					
		Rearrangements	Reversals	Permutations	DCJ		Probabilistic Approaches
					Without Biological factors	With Biological factors	
Felenstein [17]	1981	✗	✗	✗	✗	✗	✓
Yang [18]	1994	✗	✗	✗	✗	✗	✓
Hannenhalli et al. [19]	1995	✓	✗	✗	✗	✗	✗
Hannenhalli [20]	1996	✗	✓	✗	✗	✗	✗
Caprara [21]	1997	✗	✓	✗	✗	✗	✗
Christie et al. [22]	1998	✗	✓	✗	✗	✗	✗
Hannenhalli and Pevzner [23]	1999	✗	✓	✗	✗	✗	✗
Mau et al. [24]	1999	✗	✗	✗	✗	✗	✓
Bader et al. [25]	2001	✗	✓	✗	✗	✗	✗
Lin et al. [26]	2001	✓	✗	✗	✗	✗	✗
Huelsenbeck et al. [27]	2001	✗	✗	✗	✗	✗	✓
Tesler et al. [28]	2002	✓	✗	✗	✗	✗	✗
Kaplan et al. [29]	2003	✗	✓	✗	✗	✗	✗
Berard et al. [30]	2005	✗	✓	✗	✗	✗	✗
Yancopoulos et al. [7]	2005	✗	✗	✗	✓	✗	✗
Larget et al. [14]	2005	✗	✗	✗	✗	✗	✓
Feng et al. [31]	2007	✗	✗	✓	✗	✗	✗
Bender et al. [32]	2008	✗	✓	✗	✗	✗	✗
Swenson et al. [33]	2009	✗	✓	✗	✗	✗	✗
Tannier et al. [34]	2009	✗	✗	✗	✓	✗	✗
Farnoud et al. [35]	2012	✗	✗	✓	✗	✗	✗
Shao et al. [36]	2012	✗	✗	✗	✓	✗	✗
Thomas et al. [37]	2012	✗	✗	✗	✓	✗	✗
Huang et al. [38]	2014	✗	✗	✓	✗	✗	✗
Shao et al. [39]	2015	✗	✗	✗	✓	✗	✗
Swenson et al. [40]	2016	✗	✗	✗	✗	✓	✗
Bulteau et al. [10]	2016	✗	✗	✗	✗	✓	✗
Silva et al. [41]	2017	✗	✗	✗	✓	✗	✗
Fertin et al. [9]	2017	✗	✗	✗	✗	✓	✗
Simonaitis et al. [11]	2018	✗	✗	✗	✗	✓	✗
Bochkareva et al. [16]	2018	✗	✗	✗	✗	✗	✓
Simonaitis et al. [12]	2019	✗	✗	✗	✗	✓	✗
Bohnenkämper et al. [42]	2021	✗	✗	✗	✓	✗	✗

Table 2.1: Approaches for the inference of evolutionary scenarios over the years

2.1 Rearrangement distance – related work

Hannenhalli and Pevzner developed the first exact polynomial-time algorithm for the calculation of the rearrangement distance[19] between two multichromosomal genomes. They determined the genomic distance for transforming the human genome into mice genome by computing the most parsimonious rearrangement scenarios. They constructed a combined sequence of 131 reversals, translocations, fusions, and fissions events for human-mouse evolution.

In 2001, Lin et al. [26] presented a 2-approximation algorithm for solving genome rearrangement. They used transposition and reversal events to solve this problem.

In 2002, Glenn Tesler [28] reformulated an algorithm by handling only reversal events for both unichromosomal and multichromosomal genomes. This algorithm also handled circular, linear, signed, and unsigned gene data. This work improved the running time from polynomial to linear. Later on, they developed a system for analyzing genome rearrangements named GRIMM (**Genome Rearrangements In Man and Mouse**)[43] which now is available on the web [44]. GRIMM can determine the minimum number of rearrangement steps by handling reversals, translocations, fissions and fusions for multi chromosomal gene orders for transforming one genome into another.

2.2 Sorting genomes – related work

2.2.1 Sorting by reversals

2.2.1.1 Unsigned case

In the 90s, determining the minimum number of reversal events for converting a sequence into the identity sequence was conjectured to be an NP-hard problem. In 1997, Alberto Caprara [21] proved that sorting an *unsigned* permutation (unsigned means that gene orientation is not considered) by the lowest number of reversals is NP-hard.

In 1998, Christie [22] *et al.* presented a 3/2-approximation algorithm to sort unsigned permutations by reversals. After that, many works of research were conducted on this problem.

Among sorting by reversals related works, [32] is worth mentioning. Bender et al. [32] considered sorting by reversals (unsigned), by using an inversion cost that is a function of its size. They worked on both regular permutations and sequences of multiple 0s and 1s (in other words, only 2 gene labels but present in multiple copies), and they looked at different cost functions. The authors proved different lower and upper bounds on the runtime of different versions of the problems along with their approximation ratios. They also provided an exact $O(n^3)$ algorithm to determine the optimal reversal scenario for permutations of 0s and 1s.

2.2.1.2 Signed case

For the signed permutations firstly the reversal distance was solved by Hannenhalli and Pevzner [23] in quadratic time in 1999. Following their work, Bader et al. [25] solved the same problem in linear time in 2001. Their system is recognized as GRAPPA (Genome Rearrangements Analysis under Parsimony and other Phylogenetic Algorithms) and we can access their work from here [45]. In 2003, Kaplan et al. [29] solved the sorting by reversal problem for signed permutations in $O(n^{\frac{3}{2}}\sqrt{(logn)})$ time. In 2005, Berard et al. [30] solved the same problem in subquadratic time. Then this problem was solved in linearithmic $O(nlogn)$ complexity by some research works [33] handling only reversals in sorting *signed* permutations. The authors provided two algorithms for the problem. Their first algorithm was a fast ($n$log$n$) randomized heuristic that worked on most permutations, but could fail. Their other algorithm was a deterministic one that sorted all permutations and took $O(nlogn + kn)$ time, where k is a parameter which is not necessarily proportional to the distance but is bounded by it.

2.2.2 Sorting permutations by transpositions

Sorting permutations by transpositions is sorting a permutation into the identity permutation by finding out the lowest number of transposition events. This problem is well-studied in the applications of comparative genomics.

Farnoud et al. [35] solved this problem by describing two new polynomial-time 2-approximation algorithms for signed permutations.

Feng et al. [31] presented the permutation tree, a new data structure, that im-

proved the run time of sorting permutations by transpositions. This data structure also solved sorting permutations by block interchanges (a generalized transposition, which swaps two genomic regions). They improved their time complexity O($nlogn$) over the existing approximation algorithm for sorting permutation by transpositions.

Later, in 2014 this problem was solved by Huang et al. [38] in O(n) time. To solve this problem, they used a 2-approximation algorithm for a special kind of transposition named Block moves.

2.3 DCJ – related work

2.3.1 DCJ without biological factors

For solving the DCJ problem, many works of research were accomplished throughout the past. Yancopoulos et al. [7] described the DCJ operation and proposed a linear time algorithm to calculate the DCJ distance. The DCJ operations can represent inversion, translocation, fission, fusion, transposition and block interchange events, and as such, the DCJ distance represents an efficient alternative to the calculation of the rearrangement distance. However, it takes two DCJ operations to perform a transposition or a block interchange, which means that this distance does not necessarily correspond to a number of evolutionary events. Moreover, using DCJ operations to produce a transposition or a block interchange involves an intermediary step that forms a circular chromosome, which is not necessarily realistic.

When DCJ operations are used in the context of genomes with duplicated genes, the problem becomes NP-hard [37]. The problem of calculating DCJ distance with

duplicated genes was solved by Shao et al. [36] in 2012 by using an approximation algorithm. For their algorithm they found out an approximation ratio of $1.5 + \epsilon$. To calculate the DCJ distance between two genomes with duplicate genes, Shao et al. [39] formulated the maximum cycle decomposition problem as an integer linear programming (**ILP**) program. They assumed that, only DCJ took place after a speciation event. This assumption helped them to find out an exact solution by simplifying the problem. Moreover, their system was limited to balanced gene orders, where both genomes were assumed to have the same gene content. Another limitation was that their algorithm was very slow in runtime.

In 2017, Silva et al. [41] worked on the DCJ distance calculation with indels (insertion and deletion events) for the case where the indel cost is greater than the cost of DCJ operations. They proposed a linear time approach for this specific case, which could result in distances that disrupt the triangular inequality.

In 2021, Bohnenkämper et al. [42] was motivated from the work of Shao et al. [39]. They outlined an exceptionally efficient ILP for computing the distance of DCJ-indel . It considered arbitrary gene orders and the insertions, deletions as genome rearrangement operations. Their program handled real-sized genomes for both simulation and real data experiments.

2.3.2 DCJ with biological factors

To find out the realistic evolutionary scenarios, researchers got more interested in using the biological factors (physical proximity of the DNA segments, 3D spatial model, intergenic size distributions) to solve the DCJ problem. A polynomial-time

algorithm was approached in [40] for finding the minimum DCJ scenario using the physical proximity of the DNA segments (in 3D). Their model yielded a runtime of $O(N^5)$.

After Swenson et al.'s work [40], the authors carried on improving the lower bound of their algorithm in their later work [11, 12] using the 3D spatial modelling.

At the same time, Bulteau et al. [10] solved the DCJ problem in polynomial time using insertion and deletion scenarios and intergenic size distributions, optimizing the total sum of the sizes of the insertions and deletions.

After that, in 2016, Fertin et al. [13] solved Double Cut and Join problem for transforming one genome to another with arbitrary intergene size distributions. They solved the problem in polynomial-time by optimizing the aggregate number of indels (i.e. insertions + deletions). They proved that their output genomes with intergenic regions were closer to the true scenarios than the outputs with non-intergenic regions.

After that, in 2017, Fertin et al. [9] defined the concept of weighted genomes, in which adjacencies between genes also have a weight associated with them representing intergenic region sizes. They then defined wDCJ operations, which are DCJ operations acting on weighted genomes that also affect the resulting intergenic region sizes. The authors provided a formula for the calculation of the wDCJ distance and showed that calculating it is an NP-complete problem. Lastly, they presented three algorithms to calculate the wDCJ distance: a 4/3-approximation algorithm, a fixed-parameter tractable algorithm and an ILP algorithm.

2.4 Bayesian approaches for ancestral genome rearrangements inference

In the evolutionary studies, Bayesian inference has been a useful application over a broad array of research questions for its practical impact. Under Bayesian inference, many sophisticated algorithms and efficient models using Markov Chain Monte Carlo (MCMC) has been designed. Here are some research works involving Bayesian inference.

In 1999, Mau et al. [24] derived a Bayesian inference model for genomic information from the corresponding set of organisms. They used data on nine plant species, then extended to DNA sequences from thirty-two fish species for their experimentations. The program MRBAYES [27] performs Bayesian inference of Phylogeny. In 2005, Larget et al. developed a Bayesian framework **BADGER** software for phylogenetic inference using Markov chain Monte Carlo methods [14] to assess the reversal distance between genomes.

Later on, in 2009, Miklós et al. [15] proposed a theoretical method that uniformly samples all optimal sorting paths (the minimum length series of inversions). They used parallel Markov chain Monte Carlo which was faster than **BADGER**. Their proposed method has been named as **MC4Inversion**.

The main limitations of the above-mentioned works were that the researchers did not consider any biological factors (i.e., intergenic regions, genetic markers) in these statistical methods.

Chapter 3

Model Methodology

Larget et al. [14] used phylogenetic trees in their model of inference. The system of Miklós et al. [15] used reversal events in their model of inference and their system could output all optimal sorting paths (the minimum length series of inversions) by using parallel Markov chain Monte Carlo. In this thesis, we do not intend to only infer sequences of inversions of the minimum length. We believe, in some cases that the longer inversion sequences also can be closer to the realistic scenarios. So, in our methodology, we used the concept of Bayesian inference for reversal events only to identify the realistic scenarios for transforming one genome into another. So first, let me introduce what the Bayesian inference model and other related concepts are.

3.1 Preliminary concepts

3.1.1 Conditional Probability

By the term conditional probability (*e.g.* $P(A|B)$), we understand the probability of event A occurring, knowing that event B has already occurred.

Consider the following example. Suppose in the wonderland Alice was playing cards with Hatter. Hatter asked her to draw any card from the deck of cards. Alice drew a red card from the pile of 52 cards. Now Hatter asked Alice to calculate a conditional probability and that is: knowing that the card is red, then what is the probability of the card being number 7? Alice knew about conditional probability and she did the calculation right away. The probability of that card being seven is = (P(being seven|being red)) = 2/26 = 1/13. Remember that there are 26 red cards and among them two 7 cards (one of hearts and another of diamonds).

3.1.2 Bayes' Theorem

The methods of Bayesian statistics are based on **Bayes'** theorem, which is also referred to as the theorem of inverse probability. This is because it can be used to calculate the inverse of a conditional probability. The formal definition of the theorem is:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{3.1}$$

where $P(A|B)$ is the posterior probability of parameter A knowing that the data B

is observed, $P(A)$ is the prior probability of the parameter A, $P(B|A)$ is the likelihood of B given a parameter A and lastly $P(B)$ is the total probability of observing B without considering any other condition (which is used as a normalization factor).

Consider the following simple example to see Bayes' theorem in action. We want to calculate the probability $P(A|B)$ of someone having diabetes (A), knowing that they are already suffering from chronic kidney disease (B). Consider the following statistics:

- About 10% of the general population has diabetes: $P(A) = 0.10$
- The probability of developing chronic kidney disease when having diabetes is 33%: $P(B|A) = 0.33$
- About 15% of the general population is estimated to be suffering from chronic kidney disease: $P(B) = 0.15$

Using Bayes theorem, we can calculate the probability that is missing (the inverse probability):

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} = \frac{(0.10)(0.33)}{0.15} = 0.22 \tag{3.2}$$

In other words, someone who is suffering from chronic kidney disease has 22% chance of also having diabetes.

3.1.3 Bayesian inference

Bayes' theorem is often used in the context of Bayesian inference, where the goal is to calculate the probability distribution of a certain unknown parameter given a

sample of data. Different Bayesian inference techniques exist to estimate the posterior probability distribution without calculating it analytically, which can be quite complex. One of these techniques, called Markov Chain Monte Carlo, is explained in the following section.

3.1.4 Markov Chain Monte Carlo (MCMC) Sampling

Markov Chain Monte Carlo (MCMC) [46, 47] is a random sampling method used to estimate a probability distribution. It comprises of a Markov chain, which generates samples in a sequential manner in such a way that the next sampled state depends only on the present state of the chain. This property of Markov chains is called *memorylessness*, or simply the Markov property, and it means that all the history of previous moves has no influence on the next sampled state. The Metropolis-Hastings algorithm, briefly described below, is one of the most commonly used to build an MCMC chain.

3.1.4.1 Metropolis-Hastings

The algorithm of Metropolis-Hastings [48] does a random walk of the parameter space first by starting with an arbitrary sample, usually proposed from the prior distribution. Then it uses a transition model to propose a change to the current sample parameter value, but will only accept the change under certain conditions. If the new parameter value explains the data better than the previous parameter value, the change is always accepted. Otherwise, the change is only accepted with a probability that is equal to the ratio of posterior probabilities (new/current). Since

the normalization factor ($P(B)$ in equation 3.1) is the same for both new and current posterior calculations, it can be eliminated from the ratio. This is very useful, since it is normally the most difficult term to calculate. What is left is simply the ratio of prior and likelihood values, for the new and current parameters. In the context of Metropolis-Hastings, this ratio is often referred to as the *acceptance ratio*.

The strength of the approach is that if the sampling process is run for a large enough number of iterations, the frequency with which each parameter value is sampled will be directly proportional to its posterior probability. Thus the posterior probability distribution is estimated.

3.2 Our Proposed Model

Our methodology will consider inversion events for transforming one genome to another. In the future, we will extend our work by handling other rearrangement events (transpositions, duplications, deletions etc.).

3.2.1 Model properties and parameters

Our proposed model will work based on these parameters:

1. Total length of the reversal scenario:

   For transforming one genome into another we have to implement reversal events and we need to record the total number of reversals which was needed for the complete transformation.

2. Adjacent gene pairs in the chromosomes:

Adjacent gene pairs of the destination genomes play a vital role in our algorithm. When we apply a reversal on the source genome, we will check, after the reversal, how many gene pairs of the current genome are matched with the adjacent gene pairs of the destination genome. According to this checking, we will add a weight to this current reversal. Then we will do another reversal and will get another weight. The weights will be cumulatively added until the system reaches to the destination genome.

3. Ratio of the conservation of the intergenic regions in the source genome in comparison with the target one:

   After applying a reversal on the source genome, we will add all the ratios of the number of intergenic regions inside each gene pairs of the current genome and the destination genome. This added ratio will act as a weight to this current reversal. Then we will do another reversal and will get another weight. The weights will be cumulatively added until the system reaches to the destination genome. This weight will be used in the likelihood calculation of that complete reversal path.

4. Ratio of the current and new reversal path's length:

   As we know that our Bayesian system will do a random walk of resampling from current reversal path to a new candidate reversal path, so for the calculation of maximum likelihood for the new candidate we will need to calculate the ratio of the total reversal lengths of the current and new path.

3.2.2 Workflow

The flow of the full process can be divided into four steps (figure 3.1):

- Step 1: Input genomes processing
- Step 2: Creation of an initial inversion path
- Step 3: Resampling of the inversion path
- Step 4. Output processing

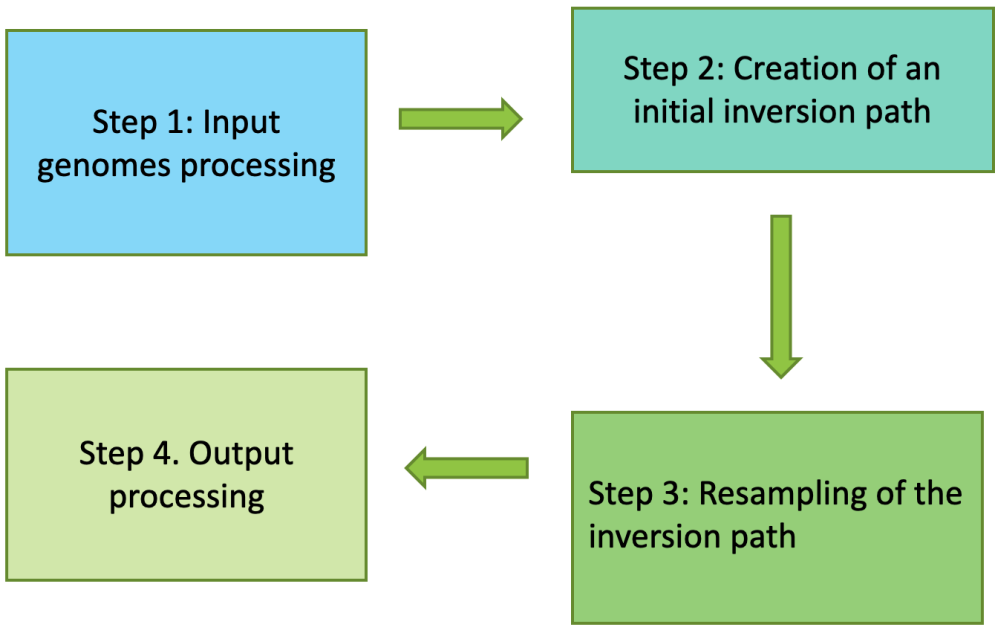

Figure 3.1: Four steps of the methodology of our Bayesian System

3.2.2.1 Step 1: Input genomes processing

Suppose, we have two genomes, one is the source genome, and another one is the target genome. The task of our system is to transform the source genome into the

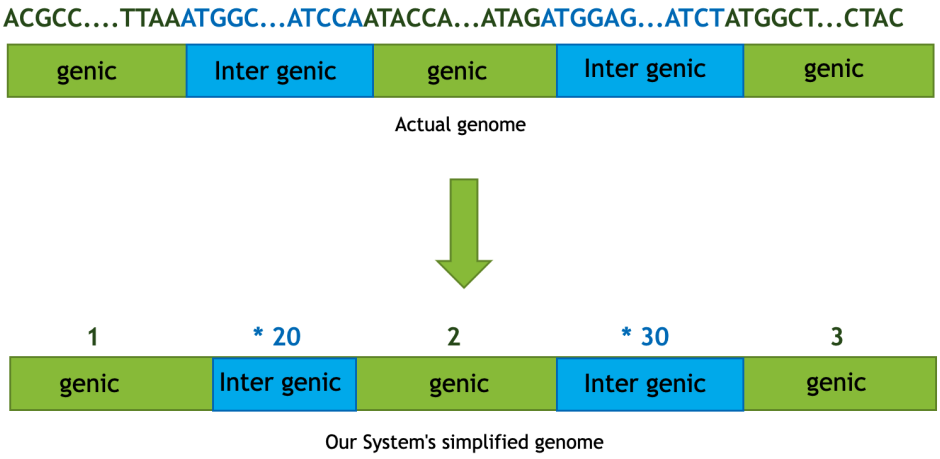

Figure 3.2: Input of our Bayesian System

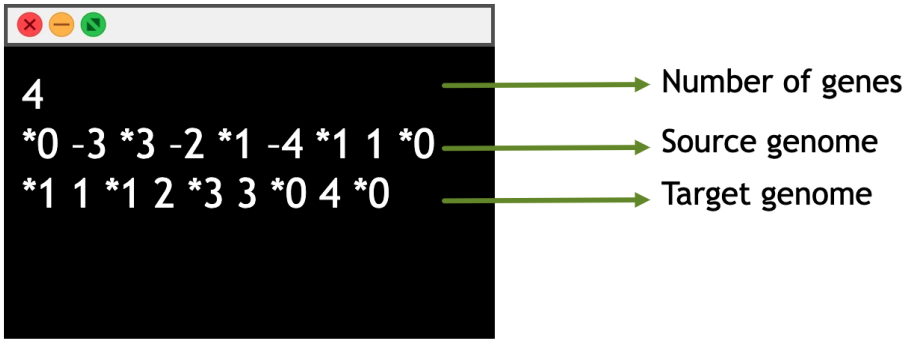

Figure 3.3: Input file

target genome using several reversals as evolutionary events. Generally, a genome sequence is a complete chain of nucleotide base pairs (Adenine (A), Cytosine (C), Guanine (G), and Thymine (T) in DNA). The genome includes both the coding genes and intergenic regions. To keep things simpler, we will use gene orders, in which genes are labeled as numbers, to represent the genomes rather than using DNA nucleotides. Both our Bayesian system and the random data generator (used for testing) use intergenic regions in between the coding genes of the source and destination genomes. Intergenic regions are non-coding DNA sequences which are

located in between the coding genes in the genome. In the introduction section, we have already learnt about the basics of intergenic regions. To avoid complexity, we have only used an integer number to represent the size of intergenic regions. In our input/output text format, the intergenic size is preceded by a * sign in between the labels of the coding genes, as shown in figures 3.2 and 3.3.

In the input file, sequentially we have these three entries:

- Number of genes of the source and target genomes (they both have the same number of genes)
- Source genome with coding genes and the size of intergenic regions
- Destination genome with coding genes and the size of intergenic regions

In figure 3.3, we see that in our input file in the first line, we have the number of genes contained in both the source and destination genomes. In this example, the source genome is found on the second line of the file: *0 -3 *3 -2 *1 -4 *1 1 *0. This means that we have coding genes -3, -2, -4, 1 with 3 as the intergenic region size between the (-3, -2) genes, 1 as the intergenic region size between the (-2, -4) and (-4, 1) genes and 0 as the intergenic region size at the beginning and end of the source genome. The target genome can be found on the third line of the file: *1 1 *1 2 *3 3 *0 4 *0. Here, we have the coding genes 1, 2, 3, 4 and intergenic regions size of 1 between the (1, 2) genes and at the beginning, 3 between the (2, 3) genes, and 0 between the (3, 4) genes and at the end of the target genome. To start the next step, we need to apply a pre-processings on the information of the coding genes and intergenic regions from the input file. This is to store all the adjacent coding genes

of both the genomes in a data structure.

3.2.2.2 Step 2: Creation of an initial inversion path

To generate an inversion path, our Bayesian system starts at the source permutation (genome) and applies one random inversion event at a time on each resulting permutation, until the target permutation is reached. The system goes through the same process to select the next inversion for each permutation traversed during the generation of the inversion path.

For each permutation, all the potential reversal moves are generated and a breakpoint graph is used to classify the inversions. Inversions that increase the number of cycles in the breakpoint graph are considered as good inversions because they reduce the distance between the permutation and the target genome. Inversions that reduce the number of cycles in the breakpoint graph are labeled as bad, because they increase the distance between the permutation and the target genome. Finally, some inversions might not change the cycle numbers in the breakpoint graph: these are labeled as neutral inversions. These three types reversals are stored in different vectors. Then, the system selects which type of inversion will be applied, using three fixed probabilities that can be changed by the user: $\mathbb{P}_{good}$, $\mathbb{P}_{neutral}$ and $\mathbb{P}_{bad}$, which should sum up to 1. These parameters can have a significant effect on the length of the inversion paths generated. For example, setting $\mathbb{P}_{good}$ to 1 and the others to 0 would produce scenarios of minimum length only. Conversely, using a probability for $\mathbb{P}_{bad}$ that is too high could result in very long inversion scenarios that might potentially never reach the target genome. This would make the Bayesian system at risk of never

exploring scenarios of high likelihood, and ultimately never converge. In our tests, we used the following values: $\mathbb{P}_{good} = 96\%$, $\mathbb{P}_{neutral} = 3\%$ and $\mathbb{P}_{bad} = 1\%$.

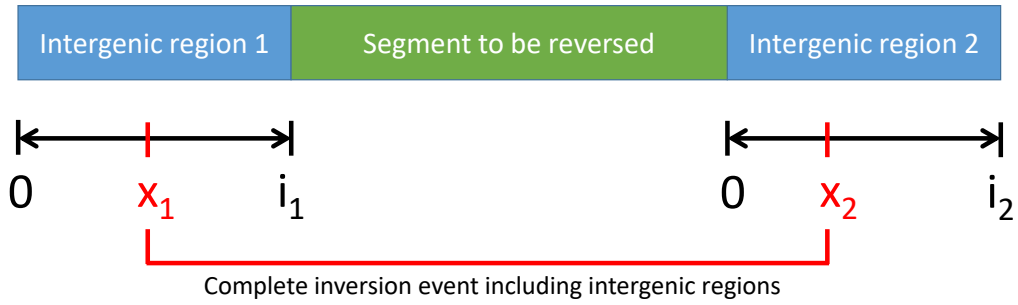

Figure 3.4: Example showing how the inversion events can affect the resulting intergenic region sizes. The inverted segment is shown in green, and the two flanking intergenic regions are shown in blue. The size of the intergenic region on the left is i_1 and (respectively on the right) is i_2. The breakpoints of the inversion are represented by x_1 and x_2

Once the next type of inversion is selected, our system randomly selects one reversal from the corresponding vector. The reversal is then applied to reach the next permutation. When applying the reversal, the intergenic regions on each side of the reversed segment are affected by the event, as described in the following. Let $i_1, i_2 \in \mathbb{Z}^*$ be the two intergenic region sizes flanking a segment of genes that is going to be reversed by the selected inversion. In the system, the breakpoints of the inversion are selected randomly anywhere between $[0, i_1]$ and $[0, i_2]$ (see Figure 3.4), and the resulting intergenic regions sizes will depend on the positions of those breakpoints. Let $x_1 \in [0, i_1]$ and $x_2 \in [0, i_2]$ be the positions of the breakpoints, the resulting intergenic sizes will be $i'_1 = x_1 + x_2$ and $i'_2 = (i_1 - x_1) + (i_2 - x_2)$ on each side of reversed segment.

After each reversal takes place, all the relevant changes in the intergenic regions get updated in the resulting permutation, and this process is repeated until the target

genome is reached and a complete inversion path is generated.

3.2.2.3 Step 3: Resampling on the inversion path:

Once the system has created the initial inversion path, the resampling process takes place. A random starting point is selected on the full inversion path, and the inversion sub-path defined by the starting point up to the end of the path is resampled. Basically, the origin point defines the starting permutation, and the target permutation stays the same. This way, the system can follow the same procedure described above in step 2 to produce a new resampled inversion sub-path (also called a *detour* below; see figure 3.5 for an example).

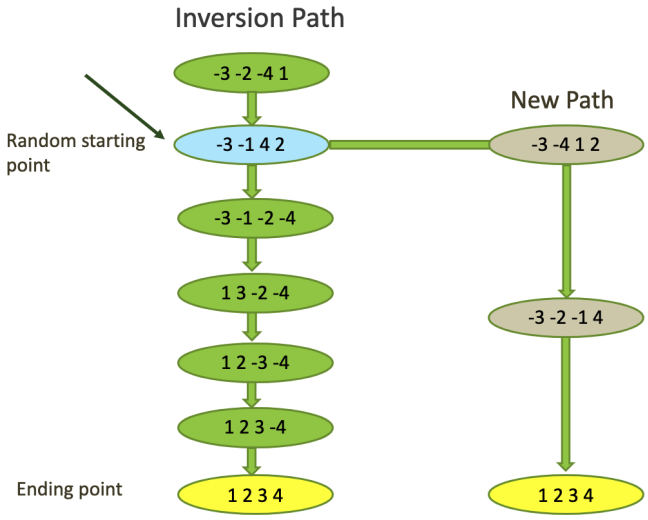

Figure 3.5: Generating a detour from a randomly selected starting position

Once the new sub-path is generated, its prior and likelihood and the ones of the previous sub-path are calculated in order to calculate the acceptance ratio. Essentially, if the new sub-path has a higher probability value (when considering the prior and likelihood) than the previous path, the change is always accepted. Otherwise,

the new sub-path is accepted with a probability equal to the ratio of likelihood and prior of the new sub-path, divided by the ones of the previous sub-path. If the new sub-path is accepted, the system will replace the current sub-path with it, and it will become part of the full inversion path (see figure 3.6 for an example).

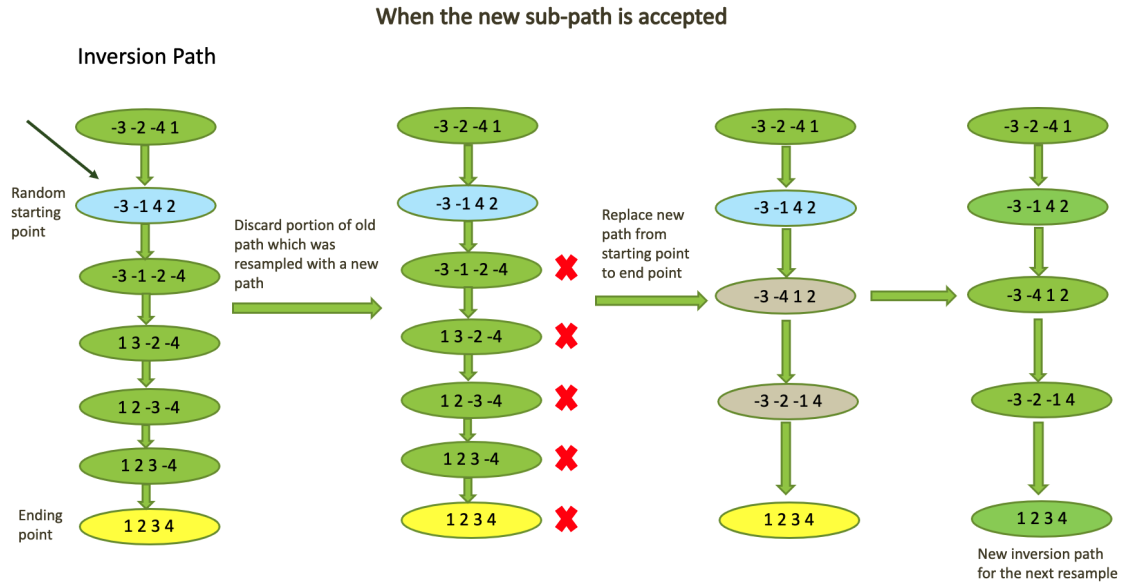

Figure 3.6: Replacing the current sub-path with the new sub-path (detour) when the change is accepted

Otherwise, if the new sub-path is not accepted, the previous inversion path will remain unchanged, and the new sub-path will be discarded (see figure 3.7). Each resampling process corresponds to one iteration of the Bayesian inference system, and the resampling continues until the total number of iterations defined by the user are completed.

A description of the prior and likelihood functions is presented below. Both of these can be calculated the same way on both the current sub-path and the new sub-path.

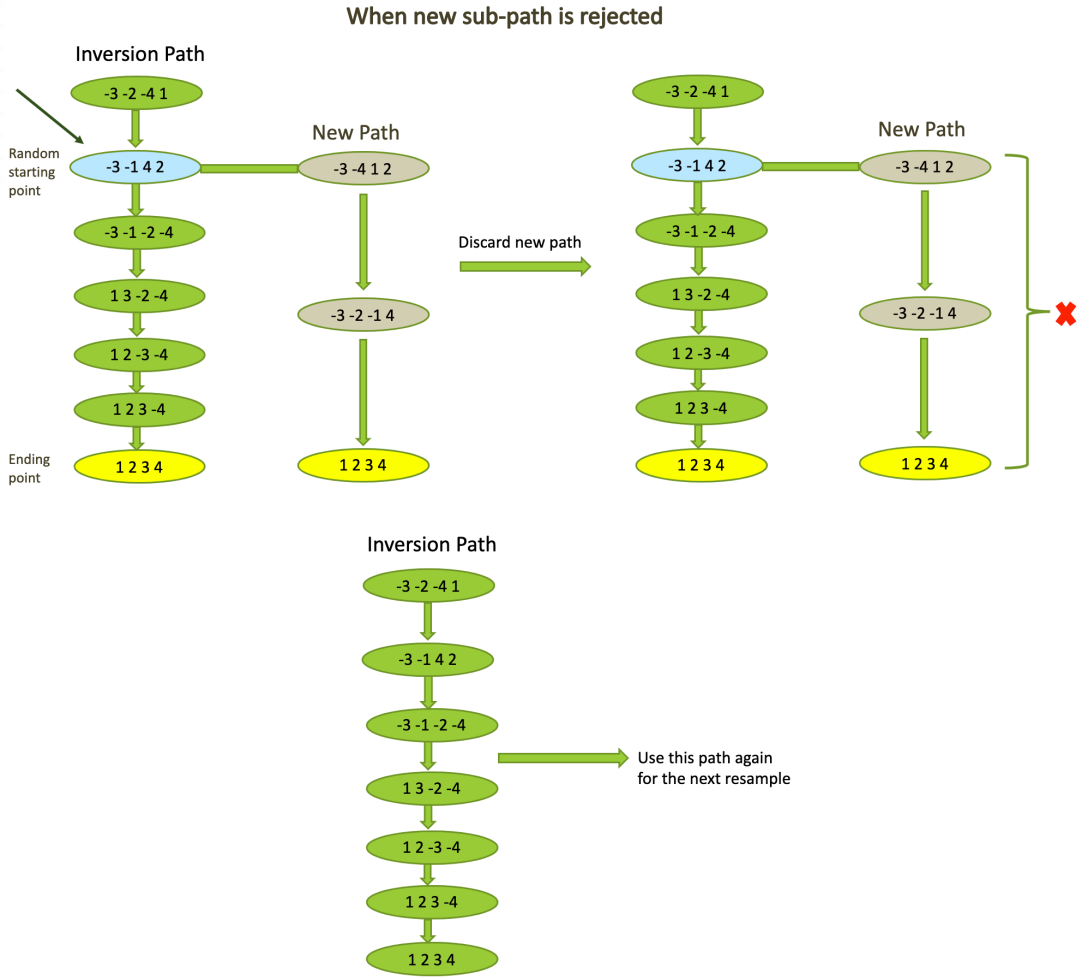

Figure 3.7: Discarding the new sub-path and keeping the full path intact when the change is not accepted

Prior

The prior that was used is uninformative in the sense that it gives the same uniform probability to each inversion used inside a path. Let IP be an inversion path of length $|IP|$ (*i.e.* containing $|IP|$ permutations). The number of inversion events involved in an inversion path IP is equal to $(|IP| - 1)$. Let P_i be a permutation at position $i \in [0, |IP|]$, and $|P_i|$ be the size of the permutation, *i.e.* the number of genes

contained in it. Note that since our model does not allow events that change the gene content, all the permutations have the same length, so we will just use $|P|$ instead.

The total number of possible inversions that can be applied to a permutation is given by the following equation:

$$|P|(|P|+1)/2 \tag{3.3}$$

Each inversion thus have a probability of occurring of:

$$\frac{1}{|P|(|P|+1)/2} = (|P|(|P|+1)/2)^{-1} \tag{3.4}$$

Consequently, the prior given to an inversion path IP of length $|IP|$ is:

$$(|P|(|P|+1)/2)^{-(|IP|-1)} \tag{3.5}$$

This prior effectively puts pressure on the inversion path length, since shorter inversion paths will have a higher probability than longer paths.

Likelihood

The likelihood calculation considers both the specific probability that each inversion in an inversion path was sampled by the system and the probability of observing each intergenic region size between each gene in the permutations of the path.

First, let $\mathbb{P}(r_i)$ be the probability that a reversal r_i was applied on permutation P_i in an inversion path, and let $\mathcal{G}$, $\mathcal{N}$ and $\mathcal{B}$ represent the sets of good, neutral and bad reversals that can be applied to P_i. $\mathbb{P}(r_i)$ is equal to:

$$\mathbb{P}(r_i) = \begin{cases} \mathbb{P}_{good}/|\mathcal{G}| & \text{if } r_i \text{ is good} \\ \mathbb{P}_{neutral}/|\mathcal{N}| & \text{if } r_i \text{ is neutral} \\ \mathbb{P}_{bad}/|\mathcal{B}| & \text{if } r_i \text{ is bad} \end{cases} \tag{3.6}$$

On a full inversion path IP, the total probability $\mathbb{P}(IP)$ that each inversion contained in it occurred is equal to:

$$\mathbb{P}(IP) = \prod_{i=0}^{|IP|-1} \mathbb{P}(r_i) \tag{3.7}$$

Second, each permutation P_i of an inversion path contains a total of $(|P| + 1)$ intergenic regions. For each intergenic region size int_j^i in P_i where $j \in [0, (|P| + 1)]$, we make a comparison with the corresponding intergenic region size $\overline{int_j^i}$ in the target genome, which corresponds to the intergenic region that is found between the same adjacent genes in the target, if the same adjacency exists. We calculate the following ratio of agreement M_j^i:

$$M_j^i = \begin{cases} \dfrac{min(int_j^i, \overline{int_j^i})}{max(int_j^i, \overline{int_j^i})} & \text{if } \overline{int_j^i} \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Note that by using the smallest value of the intergenic size as the nominator and the largest value as the denominator, the ratio is guaranteed to be ≤ 1.

To calculate the total level of agreement $A(P_i)$ for the intergenic regions of an entire permutation, we simply take the sum of all ratios of agreement and divide them by the total number of intergenic regions, and we multiply that by a weight $= i/|IP|$ that corresponds to the depth of the permutation in the path:

$$A(P_i) = \frac{i}{|IP|} * \frac{\sum_{j=0}^{|P|+1} M_j^i}{(|P|+1)} \tag{3.9}$$

The weight was added to give more importance to the intergenic regions that are found in the deeper permutations, since those would have a better chance to be closer to the intergenic regions found in the target permutation than the earlier permutations in the path.

The total level of agreement of an entire inversion path IP is:

$$A(IP) = \prod_{i=0}^{|IP|} A(P_i) \tag{3.10}$$

Lastly, the total likelihood of an inversion path IP is the product of equations 3.7 and 3.10: $\mathbb{P}(IP) * A(IP)$.

3.2.2.4 Step 4. Output processing

For each iteration of the Bayesian inference system, we store some information about the resampled inversion paths in two different output files. One file keeps track of the total reversal lengths of each resampled inversion path. This allows us to observe how the different parameters of the system are affecting the path length. Another file stores all the inversion paths (containing all the permutations with their coding genes and intergenic region sizes) that were resampled for each iteration of the system.

Since we are interested in finding the inversion paths of highest estimated posterior probability, we do a post-processing on the inversion paths of the output file. A script was built to go through the whole file and extract the 25 most frequently occurring inversion paths, along with the exact number of times they occur. The script also

calculates the estimate of the posterior probability by dividing the frequency of each inversion path by the total number of paths resampled (*i.e.* the total number of iterations of the Bayesian inference program). The output of this script is saved in another output file.

Chapter 4

Experimental Results

4.1 Developing a Random Reversal Scenario Generator

We have developed a Bayesian inference system that infers an inversion scenario converting a source genome into a target genome, which considers the intergenic size distributions between the genes. Now we want to show how accurate the algorithm of our system is at inferring realistic scenarios. Since there exists no curated real reversal scenario that can be used for accuracy evaluation, we needed to produce a generator that can simulate the evolutionary process of a genome (with intergenic regions) being affected by random reversal events.

In other words, the generator will randomly generate input genomes. At first we will pass the number of genes contained in the gene orders and the intergenic region sizes as parameters to our generator. We can control the size of the gene orders by

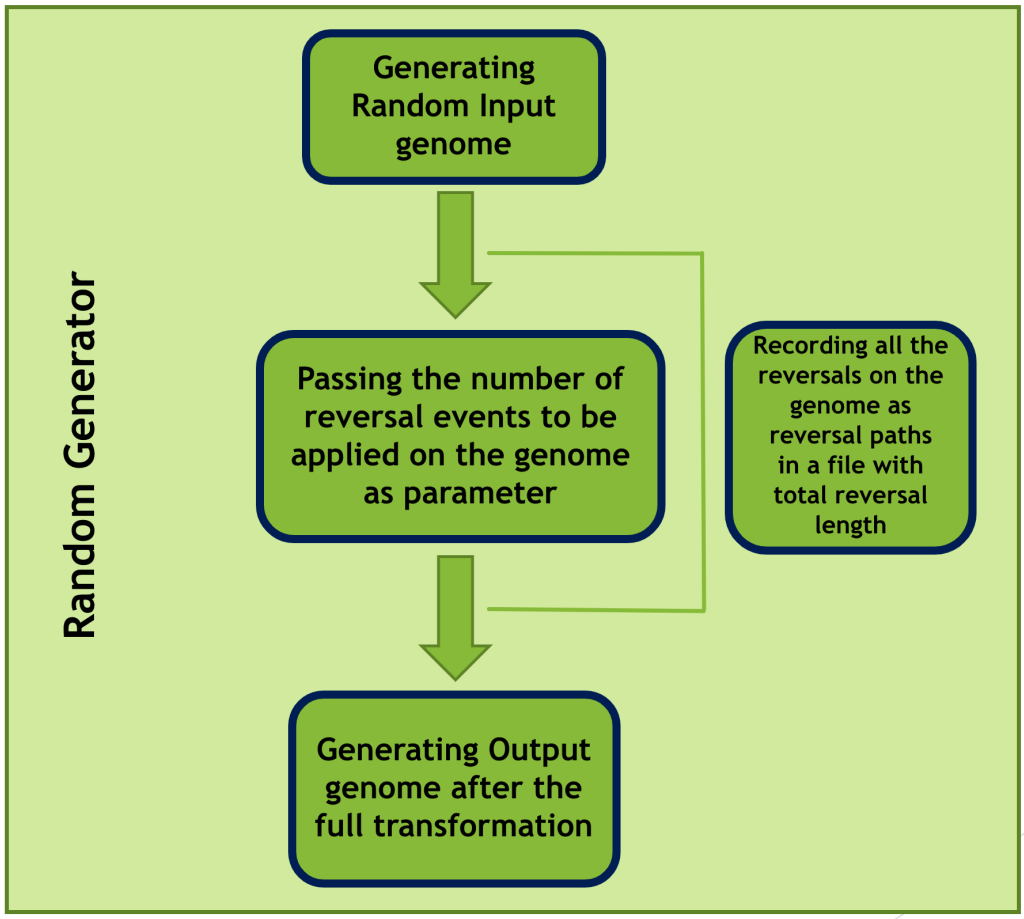

Figure 4.1: Working processes of the Random generator

these parameters. Our system will then generate a gene order of the required size. Then our system will apply some random shuffling on the genes of the gene order to generate the source gene order. Once the source genome is ready, our system needs another parameter to be passed to generate the destination gene order. That parameter is the number of reversal events to be applied on the source gene order. During every reversal event, gene orders and their flanking intergenic regions associated with the reversal revert their positions. So in each reversal, gene orders and intergenic region sizes get changed and updated by the generator. After all the reversals have

been applied on the source gene order, we now have the destination gene order ready. Then, for analysis, the source genome of the generator will act as the source genome for our Bayesian system, and the destination genome of the generator will act as target genome in our Bayesian system.

The generator also records all the reversal events applied in the simulated evolutionary scenario along with all the resulting permutations (with intergenic region sizes) after each reversal took place. This is required to contrast the results of the Bayesian inference system with the ones that were generated. The full process has been summarized in figure 4.1.

On the source genomes (derived from the generator), we will use our Bayesian system to infer evolutionary scenarios for transforming them into the target genomes. Then we will record the reversal path lengths (*i.e.* number of inversions) needed for the transformation. We will also conduct different types of experiments to check whether the results derived from two systems are similar or not, and how much the results are matching the simulated scenarios in terms of percentages. Our random reversal scenario generator is implemented in Python 3.

4.2 Experimental parameters

There are some parameters in our experiments. One parameter is how many Source-Target *genome sets* (or *replicates*) we will use in our experiments, since we want to run each test on different randomly generated genomes for each set of parameters, and plot the average of the obtained results. This is a common practice used in the evaluation of inference algorithms that is done to average out outlying input genomes,

which could by chance be either too favorable or unfavorable for the algorithm. Another parameter is used to control how many times we will run the inversion path resampling process in the Bayesian system for a particular input genome set. This parameter is called *repetitions*, and essentially represents the number of iterations of the Bayesian inference process. If we say, for a certain experiment, that we will have 200K repetitions, it means that we will run the resampling for 200,000 times. We also have another parameter called *burn-in*, which is used to set the number of times that we will resample the paths without recording them before running the selected number of repetitions. This is commonly used in MCMC sampling to let the inference process move to regions of higher probability before starting to save the sampled solutions.

Our Bayesian system records the reversal path lengths for each repetition in an output file. The system also keeps recording all the complete reversal paths from the source genome to the target genome for all repetitions in another output file. These files are eventually processed differently for various experiments. One type of processing is to find out the most frequent reversal paths among all the repetitions (topmost or top 1 path, top 5 paths, top 15 paths, top 25 paths) from the resultant file. Top 1 path is the best one as it has occurred most of the time in the MCMC sampling. Top 25 paths contain all the top 15, 5, 1 paths. We are not interested in implementing experiments on all of the paths in the output file, rather we are focusing on the top 1, 5, 15, 25 paths, which represent the paths of the highest posterior probability.

4.3 Classifications of the experiments

We need to evaluate our Bayesian system to measure its performance and accuracy in inferring the realistic evolutionary scenarios. For this reason, we would like to check multiple measures of accuracy. Some of the measures are flexible, and some are very strict. We will build graphs to plot all of these different measures, so that we can look at all of them. At first, let me explain these measures in details.

- Strict measures
  - Strict measures for the full reversal path
  - Strict measures for the destination genomes only
- Flexible measures
  - Percentage of agreement measures for the full reversal path
  - Percentage of accuracy measures for the destination genomes only
  - Percentage of frequency distribution

4.3.1 Strict measures

Our first measure is to check whether the inversion paths (or part of the paths) are exactly the same (considering both the gene orders and intergenic regions in the permutations) as the paths derived from the random generator or not. By the term same, we mean that the paths should have the same number of reversals, the same gene orders after each inversion, and the same intergenic region sizes in between the coding genes.

As we are not allowing any kind of small differences/changes in the coding genes and intergenic region distributions, we are calling this type of measure as strict, and the results are binary (0 or 1). If the entire path from the Bayesian system exactly matches with the full path from the random generator, the value is 1, otherwise 0. If in our result we do not have any 1, we can say that our system cannot find out the exactly the same output of the realistic scenario, otherwise we can say that our algorithm can determine the realistic scenario.

In our experiments, we used this strict measure in two ways, which are described below.

4.3.1.1 Strict measures for the full reversal path

This experiment checks whether the full paths are exactly the same of the paths derived from the random generator or not. This experiment is very strict as we are not allowing the slightest differences of coding genes and intergenic region sizes, for all the permutations present in the paths.

4.3.1.2 Strict measures for the destination genomes only

This experiment checks whether the intergenic region sizes of the target genomes inferred by our Bayesian system are the same as the ones of the target genomes produced by the generator. This measure is a bit more flexible than the previous one since it is considering only the destination/target genome, and not the full reversal path.

4.3.2 Flexible measures

Under this type of measure, the complete reversal paths of the Bayesian system do not need to be exactly the same as the full reversal paths derived from the random generator. For example, we can calculate the matching percentage of the target genomes (inferred vs generated), whether the reversal path lengths are the same or not. Another experiment can be to check the matching percentage of the complete reversal paths when both have the same reversal sizes. Another experiment can be to determine the percentage of the inferred inversion paths that have the same length as the ones that were generated. Let me describe the different flexible measures that we used in detail.

4.3.2.1 Percentage of agreement measures for the full reversal paths

With this measure, we will check the matching percentage of the top 1, 5, 15, 25 complete reversal paths of the Bayesian system with respect to the full paths derived from the random generator. In this measure, we have some flexibility in the sense that the inferred intergenic region sizes among the coding genes of the genomes may not be exactly the same as the simulated genomes. Also, if the total number of reversal events of both scenarios is not the same, we will skip that particular reversal path from our consideration. For example, if the path lengths of the top 1/5/15/25 do not match with the length of the simulation's reversal path, we discard them from this measure. We will only consider the paths which have the same number of reversal events for this measure.

In this measure, we calculate the percentage of match between the inferred solu-

tions and the generated simulations. We look at all the intergenic regions and gene orders for all the permutations of the paths and calculate the percentage of match. This measure will allow us to calculate how much our system has been successful in finding out the realistic inversion paths.

4.3.2.2 Percentage of accuracy measures for the target genomes only

In this measure, we will check the matching percentage of the target genomes in the top 1, 5, 15, 25 paths of the Bayesian system with respect to the target genomes derived from the random generator. Here, since the target gene orders are guaranteed to be obtained by the Bayesian inference process, we are actually focusing on calculating the percentage of match between the intergenic regions sizes of both the inferred and simulated target genomes. So we will calculate the percentage of how far the intergenic region size is from the simulated one for each of them situated in between the ordered coding genes. When we determine all the percentages for all the intergenic region sizes in the target genomes, then we will average out these values over all intergenic regions and get the overall matching percentage. This measure is flexible because the inferred and simulated scenarios can contain the same or a different number of reversal events.

4.3.2.3 Percentage of frequency distribution

In this measure, from the generator, first, we record the reversal path length that was simulated, and then we look for how many times that path length has occurred in the top 1, 5, 15, 25 results from the Bayesian system. Then we calculate the frequency percentage for each top category. For example, if the top 15 inferred paths have ten

same length paths, then the percentage of frequency will be $10/15 * 100$ or 66.67%. In this case, we could say that our system has a 66.67% accuracy in finding the expected number of reversals for the top 15 results. In this measure, it does not matter if we get or not the same intergenic region sizes within the paths because we are just concerned with the same path length. That is why this is another flexible measure.

4.4 Experiments in detail

We did experiments on various genome sizes (in terms of gene numbers) and reversal path lengths, from very small to large. We also varied the intergenic sizes in the genomes for different experiments. The same experiments have been replicated for 20 times and then we averaged out the results. We conducted every experiment using 200K repetitions of the Bayesian inference system and a burn-in of 75K.

Let us talk about the parameters of the experiments. We used varying genome sizes of 30, 40, 50, 60, 80, 100, and 120 genes for the experiments. We also changed other parameters, *i.e.*, intergenic region sizes (from 2, 4, 6, 8, 10, 12 to 14 in between each gene of the source genome) and reversal path lengths (from 4, 6, 8, 12, 15, 20 to 25) for each experiment. In the different experiments that were conducted, we tested multiple values for one of those parameters while the other two parameters were fixed. When fixed, the genome size is set to 60 genes, the intergenic region size is set to 6 and the reversal path length is set to 12. In the graphs shown below, on the Y-axis, percentages of accuracy are displayed and the parameter that is changing is plotted on the X-axis. In all the graphs, I am plotting the average values as lines for the top 1/5/15/25 paths. In the graphs, every line represents paths of one of the

top categories (the purple line represents top 1 path, the green line represents top 5 paths, the brown line represents top 15 paths, and the royal blue line represents top 25 paths). All the experiments are done in Python 3 and the plots have been drawn using the "matplotlib" and "Pandas" libraries of Python.

4.4.1 Experiments on the percentage of agreement measures for the full reversal paths

Let us first talk about the experiments on the percentage of agreement measures. There are three different graphs in figure 4.2 for each parameter that is changing (reversal path lengths, gene numbers and intergenic region sizes).

An interesting observation in figure 4.2 is that the percentage of agreement is decreasing when reversal path lengths are increasing, but the opposite is happening while the gene numbers and intergenic regions are increasing.

For all types of changes, we see that the top 1 path or the best solution from the Bayesian system has the highest percentage of agreement. From the graphs, we can observe that, almost all the reversal paths inferred from the Bayesian system have a minimum accuracy of 55% in comparison with the paths of the simulator for this experiment.

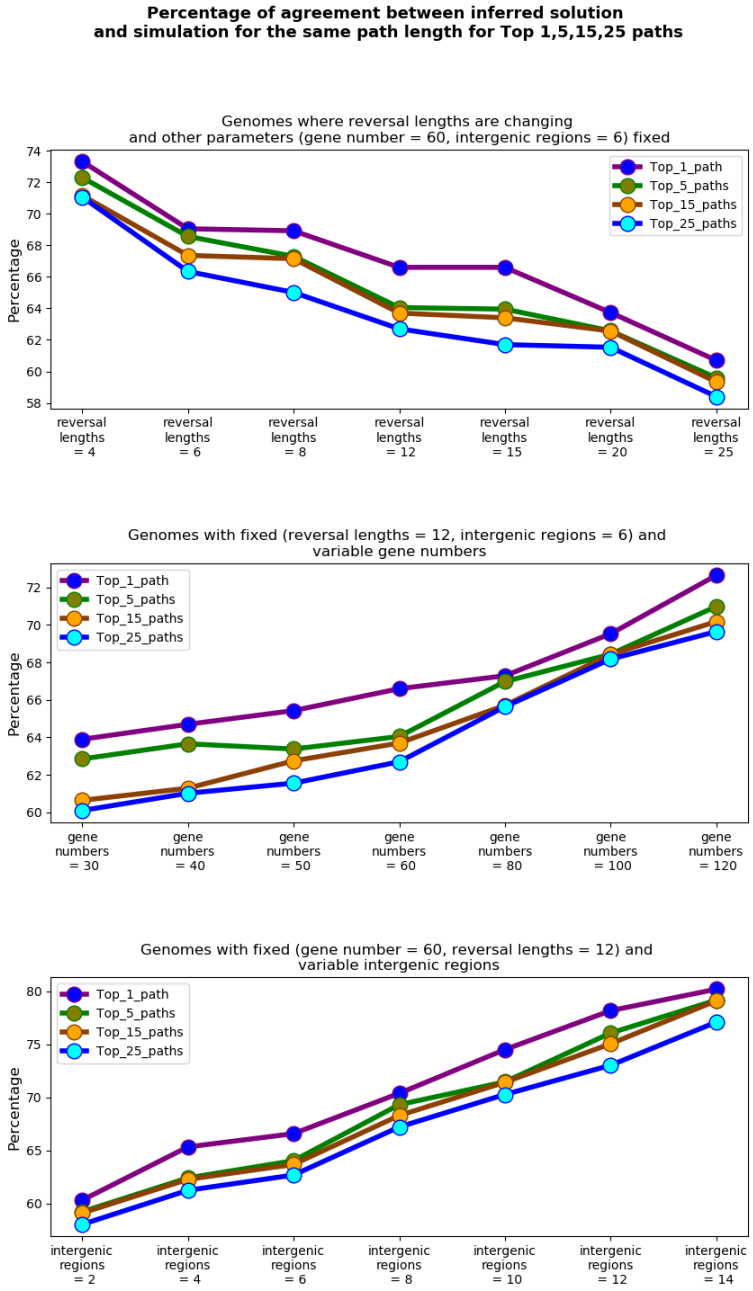

Figure 4.2: Graph of percentage of agreement measures where each parameter changing is shown in different graphs

4.4.2 Experiments on the percentage of accuracy measures for the target genomes

Let us talk about the experiments on the percentage of accuracy measures for the target genomes. There are three different graphs in figure 4.3 for each parameter that is changing (reversal path lengths, gene numbers and intergenic region sizes).

From the graphs, we can observe that, almost all the target genomes inferred by the Bayesian system have a minimum of 70% of accuracy. For all types of changes, we see that the top 1 path or the best solution from the Bayesian system has the best result. On the topmost graph, the percentage of accuracy decreases when reversal path lengths increase, confirming that increasing the number of inversions makes the problem harder. Moreover, we are again observing that when the gene numbers and intergenic region sizes are increasing, the percentage of accuracy hikes.

The reasons behind the interesting trends from the graphs of figure 4.2 and 4.3 is that when we are involving more biological information (larger intergenic regions or number of coding genes), it helps our method to infer more realistic scenarios. So the agreement of percentage just increases. In the top graph, when the reversal lengths are increasing, we are not increasing the amount of biological information. Instead, we are increasing the number of evolutionary events, which, as expected, is making the problem of inferring the correct evolutionary scenario harder. Indeed, the more events are simulated, the higher the chances of erasing the traces of some previous events.

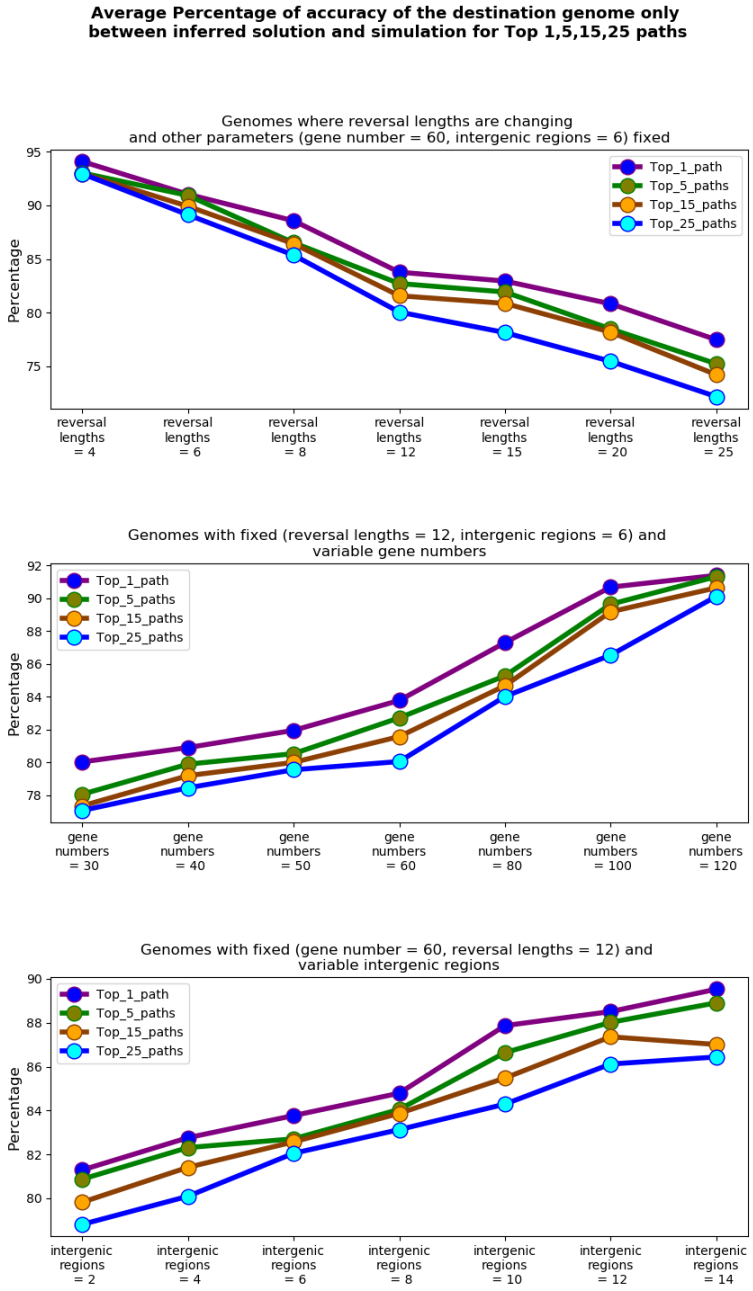

Figure 4.3: Graphs of accuracy of percentage measures for target genomes only for different parameters changing

4.4.3 Experiments on the percentage of frequency distribution

Here we present the results on the percentage of the top reversal paths inferred that have the exact same length as the simulated paths. There are three different graphs in figure 4.4 for each parameter that is changing (reversal path lengths, gene numbers and intergenic region sizes).

The common pattern we see from all of the graphs in figure 4.4 is that when the reversal path lengths, gene numbers and intergenic regions are increasing, the percentages of having the correct length in the top 1, 5, 15, and 25 paths are declining gradually. In all the graphs, the top 1 most probable path has the best result indicating that the top 1 path has minimum 80% accuracy of having the correct length (same as the simulated length).

When considering all the top 25 paths, the results are lower, meaning that up to 30-35% of the inferred paths can have an incorrect length. In other words, almost all the top results derived from the Bayesian system have a minimum of 65% capacity of finding out the same path length of the simulator for this experiment.

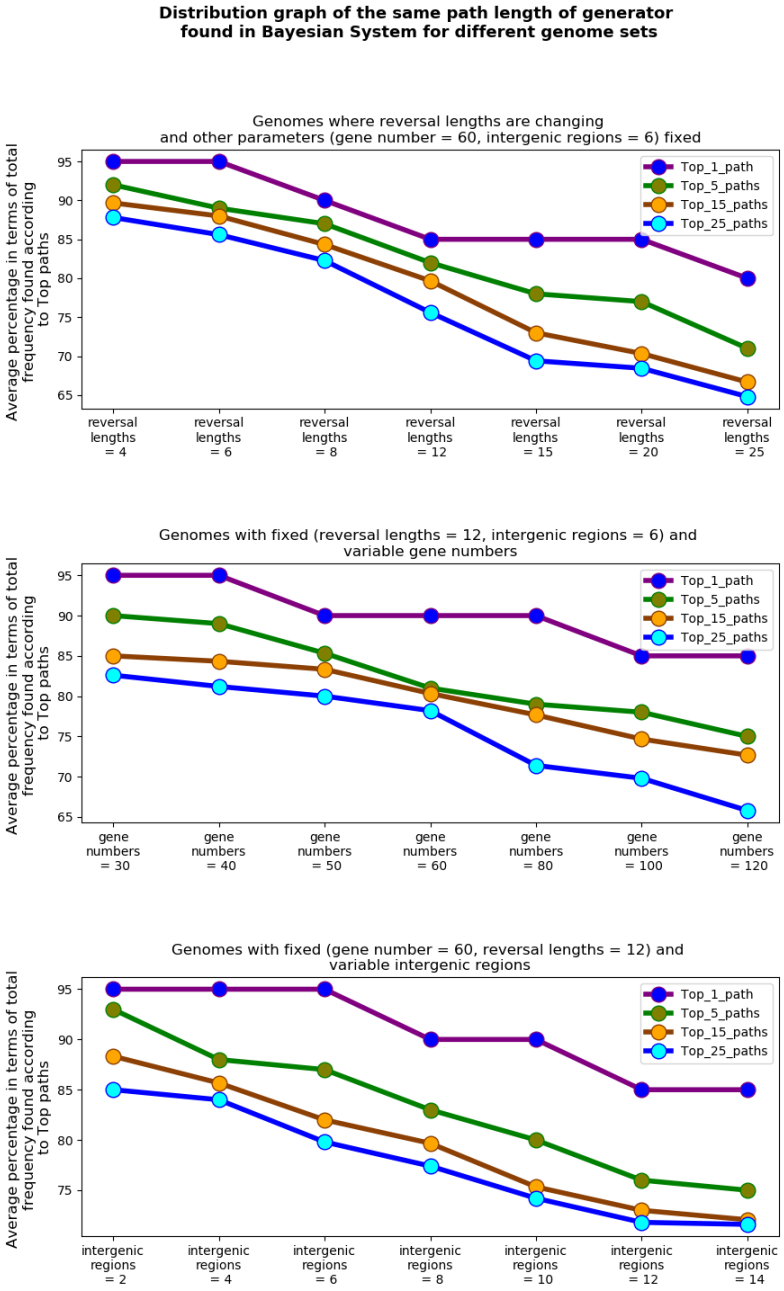

Figure 4.4: Distribution graph of the frequency distribution when parameters are changing

4.4.4 Experiments on the percentage of strict measures for target genomes only

Let us talk about the experiments on the percentage of strict measures for the target genomes only. Remember that the strict measures are binary (0 or 1), but since we are averaging the results for the different categories of top paths, and replicating the experiments 20 times, we are presenting average values, which have been transformed into percentages for ease of viewing. As with the previous experiments, there are three different graphs in figure 4.5 for each parameter that is changing (reversal path lengths, gene numbers and intergenic region sizes).

The interesting pattern we see from all of the graphs in figure 4.5 is that the percentage of finding out the exact target genome is decreasing when the reversal lengths, gene numbers, and intergenic regions are increasing. Once again, we see that the top 1 path or the best solution from the Bayesian system has the best result of up to 40% for the three experiments.

These results complement the ones of figure 4.3 and give us a clearer picture of how the intergenic sizes in the target genomes are affected by the different parameters. Even though increasing the genome and intergenic region sizes is reducing the proportion of perfect target genomes obtained (as seen in figure 4.5), it is still improving the overall accuracy percentage of all the individual intergenic regions (as seen in figure 4.3).

From the graphs, we can observe that, almost all the target genomes derived from the Bayesian system have a minimum of 12% chance and a maximum of 40% chance

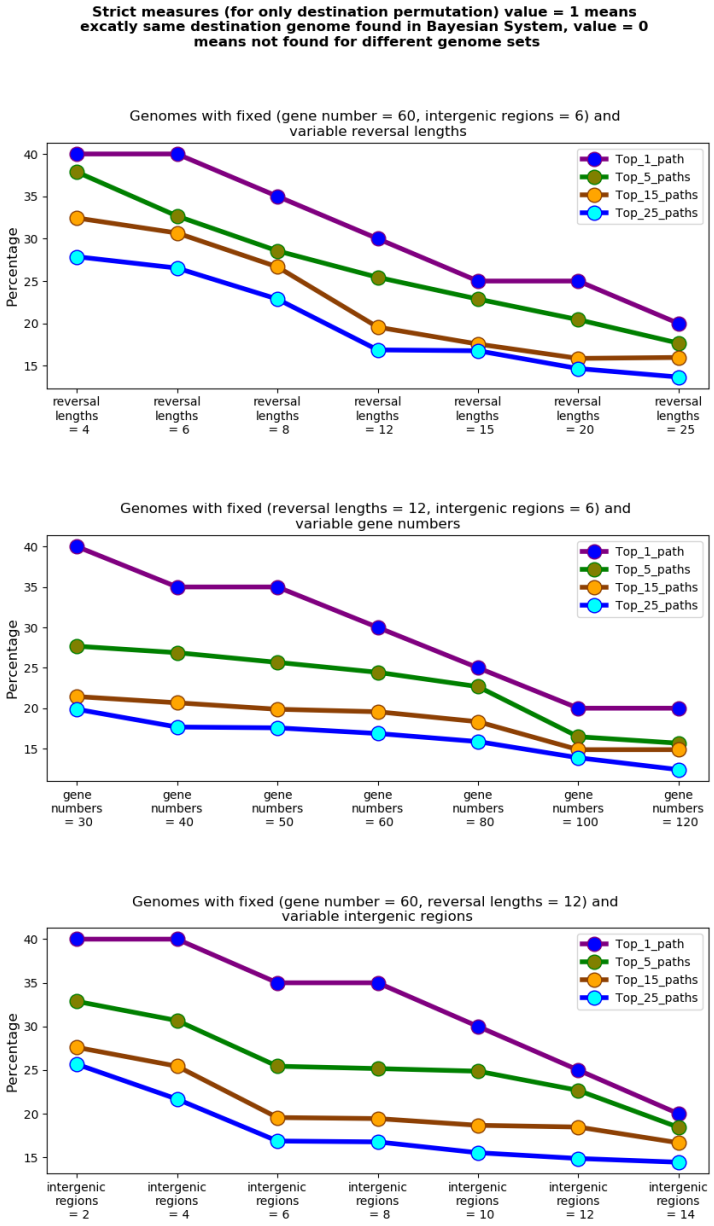

Figure 4.5: Average percentage graph of the strict measure for for target genomes only for different genome sets when parameters are changing

to have the exact same intergenic regions as the simulated ones.

4.4.5 Experiments on the percentage of strict measures for full reversal paths

Let us talk about the experiments on the percentage of strict measures for the full reversal paths. We are averaging the results for the different categories of top paths, and replicating the experiments 20 times. There are three different graphs in figure 4.6 for each parameter that is changing (reversal path lengths, gene numbers and intergenic region sizes).

The same pattern we see from all of the graphs in figure 4.6 like the previous experiment (figure 4.5) when the reversal path lengths, gene numbers, and intergenic regions are increasing. We see that the top 1 path or the best solution from the Bayesian system has the highest percentage in all the experiments and for all the different parameter values.

From the graphs, we can observe that, almost all the full paths derived from the Bayesian system have minimum 5% chance and maximum 30% chance to become exactly the same as the full paths of the simulator for these experiments. It is important to note that even though these values seem very low, the tests presented here are the most strict ones. It is generally very difficult for any inference method to infer exactly the same evolutionary history, with the exact same intergenic region changes throughout the scenario.

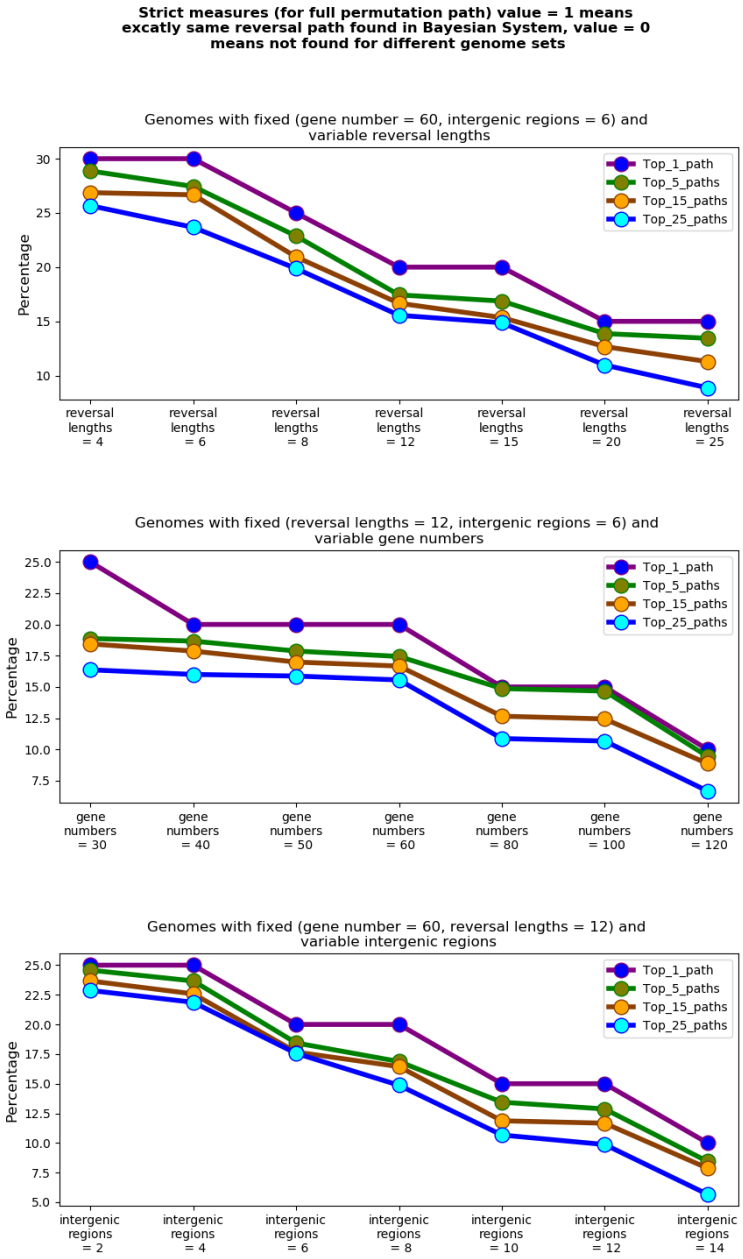

Figure 4.6: Average percentage graph of the strict measure for the full reversal path when parameters are changing

4.4.6 Experiments on the run times

We have also analyzed the run times (in seconds) for the experiments when the number of repetitions (or path resamples) and the genome size (number of genes) varies. Tables 4.1, 4.2 and 4.3 show the total and average run times (for a single resample) needed by the Bayesian system for small-sized genomes (containing 4 genes, intergenic sizes of 1 and 2 reversals), medium-sized genomes (containing 24 genes, intergenic sizes of 2 and 6 reversals) and large-sized genomes (containing 100 genes, intergenic sizes of 3 and 15 reversals) respectively. In each table, the run times are shown for 100, 1K, 10K, 100K, and 1000K repetitions.

In all three tables, we can observe that the total run time is increasing linearly with the number of resamples, and the average time needed for a single resample stays relatively constant when we use small to medium sized genome sets (see 4.1 and 4.2), but when we used large sized genome sets the average run time begins to increase slightly (see 4.3).

Table 4.1: Run time Vs number of samples (when genome size is small)

Small Genome size		
Input genome involves 4 genes		
Number of resamples	**Total run time (in seconds)**	**Average run time for single resample (in seconds)**
100	0.015	0.00015
1k	0.152	0.00015
10k	1.611	0.00016
100k	15.223	0.00015
1000k	149.845	0.00015

Table 4.2: Run time Vs number of samples (when genome size is medium)

Medium Genome size		
Input genome involves 24 genes		
Number of resamples	**Total run time (in seconds)**	**Average run time for single resample (in seconds)**
100	0.3931	0.0039
1k	3.494	0.0035
10k	34.808	0.0035
100k	351.157	0.0035
1000k	3499.78	0.0035

Table 4.3: Run time Vs number of samples (when genome size is large)

Large Genome size		
Input genome involves 100 genes		
Number of resamples	**Total run time (in seconds)**	**Average run time for single resample (in seconds)**
100	13.83	0.138
1k	140.05	0.140
10k	1545.68	0.154
100k	15639.6	0.156

We also checked the run times (in seconds) for the experiments when the number of genes in the genome varies for a fixed number of repetitions (1k). Table 4.4 shows the total run time and the average run time (for a single resample) required by the Bayesian system for various genome sizes (4, 15, 24, 50, 100, 200, and 300). We can observe that the total run times and the average run times increase exponentially with the genome size (from 0.15 second to 15514.15 seconds and from 0.00015 second to 15.51 seconds respectively for gene sizes = 4 to 300).

Table 4.4: Run time Vs different Input size

For 1k resamples		
Input Size (gene numbers)	**Total run time (in seconds)**	**Average run time for single resample (in seconds)**
4	0.15	0.00015
15	1.89	0.0019
24	3.49	0.0035
50	45.29	0.045
100	140.05	0.14
200	4033.1	4.03
300	15514.15	15.51

In terms of the effectiveness of the model and its applicability, we can say that our system runs reasonably fast when we consider genomes with small to average gene numbers, intergenic region sizes and reversal lengths. The results on run time confirm that the number of genes has a significant effect on the speed of the system. This was expected because all possible inversion events are considered at each resampling step to indentify good, neutral and bad reversals.

Chapter 5

Conclusion

5.1 Summary

Genome organization evolves by going through some rearrangement operations, which are primarily reversals. Many models have established the cornerstone for most algorithmic research works on genome comparisons over the last few years to know the similarities and dissimilarities between genomes. For example, the Double Cut and Join distance can be calculated in linear time for genomes which do not have duplicated genes. In contrast, this problem can be solved in polynomial time, considering arbitrary intergenic size distributions for insertion and deletion events only. A probabilistic evolution model solved the most recent work for the DCJ model [13]. This model considered the gene order alterations along with a biological factor (the intergenic regions size). It was proved that the output genomes with intergenic regions were closer to the actual scenarios than the outputs with non-intergenic regions. This is a promising discovery for the new researchers who can use this biological constraint,

the intergenic regions, in their researches to find out the realistic scenarios.

In my thesis work, I focused on the inference of realistic evolutionary scenarios of genome reversal events considering intergenic regions, a biological factor. Moreover, following the works of Larget et al. [14] and [15], by developing a **Bayesian probabilistic model**, I wanted to make my framework more realistic than the existing models.

5.2 Discussion

The main significance of our work is that it was able to give outputs closer to the realistic scenarios after involving a biological factor, "intergenic region sizes", to it. In the realistic scenario, it is possible that the evolutionary path may not necessarily take the shortest path. As in the reversal events, intergenic region sizes are entangled with the gene orders, the changes in the intergenic regions may not always lead to the shortest possible scenario. Rather it can lead to the most likely evolutionary scenarios resulting in creating paths longer than the shortest paths. So, our system is not always picking up the shortest path. Sometimes it gives the shortest paths as results and sometimes for some genome sets we get reversal paths that can be longer than the shortest path depending on the input genomes.

In general, inferring more realistic scenarios can allow us to better understand the size, frequency, location and types of evolutionary events that affect genomes during their evolution. Ultimately, producing more realistic evolutionary scenarios could help improve other evolutionary models and set appropriate costs for different events.

Another significance of our work is regarding the results. We see that in all the

experiments we have the best results for the most frequently iterated samples which is good in a sense that, the system is working correctly, and effectively finding out the paths of the highest likelihood.

Besides significance, we have some limitations in our work. One of the limitations is on our strict measures results. The results of these measures are not as strong in terms of accuracy whereas for other measures we have obtained good results. For the strict measures of target genomes, the accuracy is a maximum of 40% and for the full reversal paths, the accuracy is a maximum of 30% which is relatively low. It is really a very difficult problem to find out the exact evolutionary path. So in general it was expected that our system would not have that much good accuracy percentage in terms of finding out the exact same paths of the generator.

Another limitation is that our system is limited to be applied on real data-sets in which only inversion events occurred or genomes only had a linear and single chromosome. However, in many data sets, other evolutionary events (translocations, fissions, fusions) can happen and the genomes can have circular or multiple chromosomes. These would not be considered by the evolutionary model of our system.

The third limitation is that our system is based on a probabilistic approach. We know that probabilistic systems are generally a bit slower in nature. Our system is fast when we run resamples for a short number of repetitions, but for a large number of resamplings like 1000K, 10000K the system behaves slow. Also when the genome sets involve a large number of genes and intergenic region sizes the system gets slower to determine outputs.

5.3 Future Works

Right now, our system only handles linear genomes. It does not consider circular genomes. For circular genomes, some inversions can wrap around the end/start of the genome sequences. For example, say we have a genome sequence with four genes $G1$: $a\ b\ c\ d$, we can have an inversion of (d and a) if the genome is circular. This phenomenon is not possible for linear genomes as they have telomeres in their ends, and end-sided genes are not connected with each other like the circular ones. So, we will work on circular genomes in the future for reversal events.

There is another future scope to extend our work by modifying the input genome's structure. Our system now considers input genomes having one chromosome only. If our system can handle multiple chromosomes along with a single chromosome, then the gateway to consider other evolutionary events besides reversals may open. We know that genomes having multiple chromosomes tend to evolve by other rearrangement events (transpositions, translocations, fissions, fusions), and these should be considered in the new evolutionary model.

Another potential idea is to use some other biological information to reduce the exhaustive search in the search spaces. For example, along with the intergenic regions, we can also use genetic markers (a specific sequence of DNA at a known location on a chromosome). During a speciation event, nearby DNA segments on a chromosome tend to come into together. Genetic markers can be used to find out the inheritance of a closer gene which is not spotted yet but we already know the probable location. This can be another biological factor, and we can compare our present results with this

added factor to see whether the accuracy improves or not. If the accuracy improves, then we need to check how much they tend to act like the actual scenario.

There can be another exciting addition to our methodology. Suppose in both input and output genomes, we already know the intergenic region sizes of the genes between the coding genes. So, we can detect the pair of genes whose intergenic region sizes are conserved in both the genomes. Then during the selection of the realistic path, we can discard those reversals which can not preserve the listed intergenic region sizes in that particular pair of genes in both the genomes. However, there is a possibility that those discarded reversals may be good reversals, which our system chooses most of the time. So, we need to find out a way when discarding reversals would not affect much though they are good reversals, or we will discard only when they are not good reversals.

Moreover, with the help of intergenic regions and genetic markers, we are optimistic about identifying the duplicated genes in future. Furthermore, we have a plan to add other evolutionary events: insertions, deletions and duplications besides the inversion events. Thus our algorithm can guide us to determine the natural evolutionary history for every type of evolutionary event by utilizing biological information.

Bibliography

[1] https://www.chegg.com/learn/biology/introduction-to-biology/dna-double-helix-model. v, 3

[2] https://www.khanacademy.org/science/ap-biology/gene-expression-and-regulation/transcription-and-rna-processing/a/overview-of-transcription. v, 4

[3] https://www.coopergenomics.com/pgt-sr/. v, 7

[4] https://www.sciencedirect.com/topics/medicine-and-dentistry/phylogeny. v, 8

[5] F. S. Brinkman and D. D. Leipe, "Phylogenetic analysis," *Bioinformatics: a practical guide to the analysis of genes and proteins*, vol. 2, p. 349, 2001. 8

[6] A. Bergeron, J. Mixtacki, and J. Stoye, "A unifying view of genome rearrangements," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2006, pp. 163–173. 11

[7] S. Yancopoulos, O. Attie, and R. Friedberg, "Efficient sorting of genomic permutations by translocation, inversion and block interchange," *Bioinformatics*, vol. 21, no. 16, pp. 3340–3346, 2005. 12, 18, 22

[8] W. R. Francis and G. Wörheide, "Similar ratios of introns to intergenic sequence across animal genomes," *Genome biology and evolution*, vol. 9, no. 6, pp. 1582–1598, 2017. 13

[9] G. Fertin, G. Jean, and E. Tannier, "Algorithms for computing the double cut and join distance on both gene order and intergenic sizes," *Algorithms for Molecular Biology*, vol. 12, no. 1, p. 16, 2017. 13, 14, 18, 24

[10] L. Bulteau, G. Fertin, and E. Tannier, "Genome rearrangements with indels in intergenes restrict the scenario space," *BMC bioinformatics*, vol. 17, no. 14, p. 426, 2016. 13, 18, 24

[11] P. Simonaitis and K. M. Swenson, "Finding local genome rearrangements," *Algorithms for Molecular Biology*, vol. 13, no. 1, p. 9, 2018. 13, 18, 24

[12] P. Simonaitis, A. Chateau, and K. M. Swenson, "A general framework for genome rearrangement with biological constraints," *Algorithms for Molecular Biology*, vol. 14, no. 1, p. 15, 2019. 13, 18, 24

[13] L. Bulteau, G. Fertin, and E. Tannier, "Genome rearrangements with indels in intergenes restrict the scenario space," *BMC bioinformatics*, vol. 17, no. 14, pp. 225–231, 2016. 13, 24, 66

[14] B. Larget, J. B. Kadane, and D. L. Simon, "A bayesian approach to the estimation of ancestral genome arrangements," *Molecular phylogenetics and evolution*, vol. 36, no. 2, pp. 214–223, 2005. 13, 14, 18, 25, 26, 67

[15] I. Miklós and A. E. Darling, "Efficient sampling of parsimonious inversion histories with application to genome rearrangement in yersinia," *Genome biology and evolution*, vol. 1, pp. 153–164, 2009. 13, 14, 25, 26, 67

[16] O. O. Bochkareva, N. O. Dranenko, E. S. Ocheredko, G. M. Kanevsky, Y. N. Lozinsky, V. A. Khalaycheva, I. I. Artamonova, and M. S. Gelfand, "Genome rearrangements and phylogeny reconstruction in yersinia pestis," *PeerJ*, vol. 6, p. e4545, 2018. 13, 18

[17] J. Felsenstein, "Evolutionary trees from dna sequences: a maximum likelihood approach," *Journal of molecular evolution*, vol. 17, no. 6, pp. 368–376, 1981. 18

[18] Z. Yang, "Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: approximate methods," *Journal of Molecular evolution*, vol. 39, no. 3, pp. 306–314, 1994. 18

[19] 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*. IEEE, 1995, pp. 581–592. 18, 19

[20] S. Hannenhalli, "Polynomial-time algorithm for computing translocation distance between genomes," *Discrete applied mathematics*, vol. 71, no. 1-3, pp. 137–151, 1996. 18

[21] A. Caprara, "Sorting by reversals is difficult," in *Proceedings of the first annual international conference on Computational molecular biology*, 1997, pp. 75–83. 18, 20

[22] D. A. Christie *et al.*, "A 3/2-approximation algorithm for sorting by reversals." in *SODA*, 1998, pp. 244–252. 18, 20

[23] S. Hannenhalli and P. A. Pevzner, "Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals," *Journal of the ACM (JACM)*, vol. 46, no. 1, pp. 1–27, 1999. 18, 21

[24] B. Mau, M. A. Newton, and B. Larget, "Bayesian phylogenetic inference via markov chain monte carlo methods," *Biometrics*, vol. 55, no. 1, pp. 1–12, 1999. 18, 25

[25] D. A. Bader, B. M. Moret, and M. Yan, "A linear-time algorithm for computing inversion distance between signed permutations with an experimental study," in *Workshop on Algorithms and Data Structures*. Springer, 2001, pp. 365–376. 18, 21

[26] G.-H. Lin and G. Xue, "Signed genome rearrangement by reversals and transpositions: models and approximations," *Theoretical Computer Science*, vol. 259, no. 1-2, pp. 513–531, 2001. 18, 19

[27] J. P. Huelsenbeck and F. Ronquist, "Mrbayes: Bayesian inference of phylogenetic trees," *Bioinformatics*, vol. 17, no. 8, pp. 754–755, 2001. 18, 25

[28] G. Tesler, "Efficient algorithms for multichromosomal genome rearrangements," *Journal of Computer and System Sciences*, vol. 65, no. 3, pp. 587–609, 2002. 18, 19

[29] H. Kaplan and E. Verbin, "Efficient data structures and a new randomized approach for sorting signed permutations by reversals," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 2003, pp. 170–185. 18, 21

[30] S. Bérard, A. Bergeron, C. Chauve, and C. Paul, "Perfect sorting by reversals is not always difficult," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2005, pp. 228–238. 18, 21

[31] J. Feng and D. Zhu, "Faster algorithms for sorting by transpositions and sorting by block interchanges," *ACM Transactions on Algorithms (TALG)*, vol. 3, no. 3, pp. 25–es, 2007. 18, 21

[32] M. A. Bender, D. Ge, S. He, H. Hu, R. Y. Pinter, S. Skiena, and F. Swidan, "Improved bounds on sorting by length-weighted reversals," *Journal of Computer and System Sciences*, vol. 74, no. 5, pp. 744–774, 2008. 18, 20

[33] K. M. Swenson, V. Rajan, Y. Lin, and B. M. Moret, "Sorting signed permutations by inversions in o (nlogn) time," in *Annual International Conference on Research in Computational Molecular Biology*. Springer, 2009, pp. 386–399. 18, 21

[34] E. Tannier, C. Zheng, and D. Sankoff, "Multichromosomal median and halving problems under different genomic distances," *BMC bioinformatics*, vol. 10, no. 1, p. 120, 2009. 18

[35] F. Farnoud and O. Milenkovic, "Sorting of permutations by cost-constrained transpositions," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 3–23, 2012. 18, 21

[36] M. Shao and Y. Lin, "Approximating the edit distance for genomes with duplicate genes under dcj, insertion and deletion," in *BMC bioinformatics*, vol. 13, no. 19. BioMed Central, 2012, p. S13. 18, 23

[37] A. Thomas, A. Ouangraoua, and J.-S. Varré, "Tandem halving problems by dcj," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2012, pp. 417–429. 18, 22

[38] J. Huang and S. Roy, "On sorting under special transpositions," in *2014 IEEE International Conference on Bioinformatics and Bioengineering*. IEEE, 2014, pp. 325–328. 18, 22

[39] M. Shao, Y. Lin, and B. M. Moret, "An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes," *Journal of Computational Biology*, vol. 22, no. 5, pp. 425–435, 2015. 18, 23

[40] K. M. Swenson, P. Simonaitis, and M. Blanchette, "Models and algorithms for genome rearrangement with positional constraints," *Algorithms for Molecular Biology*, vol. 11, no. 1, p. 13, 2016. 18, 24

[41] P. H. da Silva, R. Machado, S. Dantas, and M. D. Braga, "Genomic distance with high indel costs," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 14, no. 3, pp. 728–732, 2017. 18, 23

[42] L. Bohnenkämper, M. D. Braga, D. Doerr, and J. Stoye, "Computing the rearrangement distance of natural genomes," *Journal of Computational Biology*, vol. 28, no. 4, pp. 410–431, 2021. 18, 23

[43] G. Tesler, "Grimm: genome rearrangements web server," *Bioinformatics*, vol. 18, no. 3, pp. 492–493, 2002. 19

[44] http://grimm.ucsd.edu/cgi-bin/grimm.cgi#report. 19

[45] https://www.cs.unm.edu/~moret/GRAPPA/. 21

[46] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995. 29

[47] C. J. Ter Braak, "A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces," *Statistics and Computing*, vol. 16, no. 3, pp. 239–249, 2006. 29

[48] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995. 29