RICARDO HENRIQUE GRACINI GUIRALDELLI

ALGORITHMIC GRAPH UNIFICATION AND SIMULATIONS FOR HAPLOTYPE NETWORKS

A thesis submitted to the Polytechnic School of University of São Paulo in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering.

São Paulo
2012

RICARDO HENRIQUE GRACINI GUIRALDELLI

ALGORITHMIC GRAPH UNIFICATION AND SIMULATIONS FOR HAPLOTYPE NETWORKS

A thesis submitted to the Polytechnic School of University of São Paulo in partial fulfillment of the requirements for the Degree of Master of Science in Electrical Engineering.

Concentration Area:

Digital Systems

Advisor:

Ricardo Luis de Azevedo da Rocha

São Paulo
2012

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 21 de maio de 2012.

Assinatura do autor Guiraldelli

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Guiraldelli, Ricardo Henrique Gracini
Algorithmic graph unification and simulations for haplotype networks / R.H.G. Guiraldelli. -- ed.rev. -- São Paulo, 2012.
87 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Teoria dos grafos 2.Bioinfomática 3.Filogenética 4.Genética de populações 5.Computação adaptativa 6.Simulação de sistemas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

To my lovely family: Luiz, Maria Helena and Francisco.

ABSTRACT

This research dealt with the unification of graphs applied to the problem of haplotype networks. From the low reliability found in networks generated by the traditional algorithms, it was necessary to introduce a new proposition to improve the outcome. For proper evaluation of the results obtained by the new algorithm, a formal-theoretical framework for generalization of haplotype networks related solutions, making use of parameterized functions easily represented through LISP-like functional languages. Seeking for real case application of the theory developed, a structure of simulation and testing were developed to build genetic code strings randomly or even parameterized.

The generated tests have allowed to observe the expected improvement in the algorithm, especially for cases of low mutation. The framework for simulation and testing was extremely useful and easy to use, making itself a product to be distributed to the academic Biology community.

RESUMO

Esta pesquisa lidou com a unificação de grafos aplicada ao problema das redes haplotípicas. A partir da baixa confiabilidade encontrada nas redes geradas pelos algoritmos tradicionais, foi necessário introduzir uma nova proposição para melhorar o resultado obtido. Para esta avaliar o resultado obtido pelo novo algoritmo, desenvolveu-se um arcabouço teórico-formal possibilitando a generalização de soluções relativas a redes haplotípicas através de aplicação de funções parametrizáveis, facilmente representada através de linguagens funcionais no estilo LISP. Para aplicação em problemas reais, desenvolveu-se estrutura de simulação e testes que constrói cadeias genéticas de maneira aleatória ou ainda parametrizável.

Os testes gerados permitiram observar a melhoria esperada no algoritmo, especialmente para os casos de baixa mutação. O arcabouço de simulação e testes mostrou-se extremamente propício e de fácil utilização, o que o torna, em si mesmo, uma ferramenta a ser disponibilizada para a comunidade acadêmica da área de Biologia.

CONTENTS

LIST OF FIGURES

SYMBOLS

(G, w)	weighted graph,	p. 34
Φ	space of elementary adaptive actions,	p. 26
Ψ	general selection function,	p. 55
Σ	alphabet,	p. 27
$\emptyset$	empty set,	p. 27
ϵ	empty string,	p. 27
$\langle rule \rangle$	pattern of a rule,	p. 26
$\mathbb{G}$	set of graphs,	p. 46
$\mathcal{F}$	take *first* element function,	p. 55
$\mathcal{R}$	take *rest* of the elements function,	p. 55
∂	set of rules of the computational device,	p. 27
ϕ	elementary adaptive function,	p. 26
ψ	selection function,	p. 54
AA	posterior adaptive function,	p. 26
AD	adaptive device,	p. 26
AM	adaptive mechanism,	p. 26
BA	anterior adaptive function,	p. 26
E	set of edges,	p. 33
f^+	foreign-transition addition operator,	p. 28
f^-	proper-transition remotion operator,	p. 28
G	graph,	p. 33
M	computational device (machine),	p. 26
MST	minimum spanning tree,	p. 39
ND_i	non-adaptive subjacent computational device,	p. 26
NR	non-adaptive rules,	p. 26
Q	set of states,	p. 27
S	proper-search operation,	p. 27
S	search function,	p. 53
$S_{\mathbb{G}}$	general search function,	p. 54
V	set of vertices,	p. 33
w	weight function,	p. 34

ABREVIATIONS

ABC	approximate Bayesian computation,	p. 11
BNF	Backus-Naur Form,	p. 65
CSV	comma-separated values,	p. 67
DNA	deoxyribonucleic acid,	p. 10
GUI	graphical user interface,	p. 64
mtDNA	mitochondrial DNA,	p. 16
NCPA	nested clade phylogeographic analysis,	p. 11
RNA	ribonucleic acid,	p. 17
SNP	single-nucleotide polymorphism,	p. 17

THANKS

After three (very intesive) years, it's time to admit that the present work, although references just one author — in this case, me — actually is a composition from ideas of dozens gifted minds that, somehow decisively, guided the research to its final results; in the next few lines some of them are remembered, but many will left in anonymity simply due to my failing memory; for these, my sincere apologies.

First of all, I would like to thank my parents, Luiz Antonio Guiraldelli and Maria Helena César de Camargo Bellaz Guiraldelli, and my brother, Francisco Augusto César de Camargo Bellaz Guiraldelli, which raised me with strong values and taught me that *knowledge* is the most valuable asset, one that can never be stolen from its "owner". They also supported me in the bad moments and kept me motivated to continue the graduate studies.

Following, I thank professor Ricardo Luis de Azevedo da Rocha, Reginaldo Inojosa da Silva Filho, professor Cristina Yumi Miyaki and Fábio Sarubbi Raposo do Amaral for guiding me in the whole research process, frequently and continuosly discussing every and each detail that arose in the development of the work as well as yielding valuable insights in those moments that any progress seemed impossible; *advising* has taken a higher sense under their guidance.

I could not forget the contributions of Ana Paula Brambila, Átila Madureira Bueno, Antonio Henrique Della Colleta Herbst, Antonio Mauro Saraiva, Bárbara Dariano Silva, Celso Vital Crivelaro, Cristiano Magalhães Panazio, Daniel Assis Alfenas, Danilo Picagli Shibata, Diego Paolo Ferruzzo Correa, Flávio Nishikawa Vilani, Francisco Javier Ramirez Fernandez, Geovandro Carlos Crepaldi Firmino Pereira, Graziela Martins Pedro Dias, Gregory John Chaitin, Isabela Rocha Schelim, João José Neto, Joaquim Aparecido Machado, José Roberto Castilho Piqueira, L. Lacey Knowles, Leia Sicilia, Lina Alexandrino dos Santos, Luis Eduardo Soares Netto, Luís Emilio Cavechioli Dalla Valle, Marie-Anne Van Sluys, Paulo Sérgio Licciardi Messeder Barreto, Pedro Bruno Pereira de Brito, Rafael Augusto Ricco, Raphael Nunes da Motta, Rodolpho Henrique Orlovsky Eckhardt, Rubens Chamtob Levy and many others that, unfortunately, my mind has failed to remember but they will be immortalized in the permanent thoughts derived from this work.

1 INTRODUCTION

Since the invention of the *"code of life"* — the deoxyribonucleic acid (DNA)— by James Watson and Francis Crick in the 1950s, the emphasis of the biologists and other researchers on the seek for comprehension of its properties and operation has been increasing considerably, driving to the emergence of many ideas about the information that could be extracted from the sequences of nucleobases. The scientific community, soon, has got the possession of the principal knowledge of this molecule — the *inheritance* capability — and started to pursuit all relationship associated with this characteristic, as well as deducting others because of this information flow through generations (CALLADINE et al., 2004; PEREIRA, 2005).

The profile of genetic modification and the information flow from ancestors to descendants was promptly perceived as an useful way for extracting the historical outline of generations and, with slight improvements, for the entire community. The improvement of the comparison techniques, the addition of new variables in the analysis, the increase of acquired genetic sequences from field explorations and, most important, the demand for state-of-the-art methods for comprehension of environment and the evolutionary trail has created new research fields in biology such as *phylogenetics* (DAWKINS, 2004) and, recently, *phylogeography*(AVISE, 2007).

Working with a large amount of data for proper comparison, these new de-

veloped areas started to require computational efficient algorithms[1] for providing fast results on the correlation of the data, moving away from qualitative only analysis and providing a quantitative one(AVISE, 2007; KNOWLES, 2002), as well as developing optimized visualization approaches for the number of sequences and values in possession. In an attempt to fill this technological gap, concurrent methods such as *nested clade phylogeographic analysis (NCPA)* (TEMPLETON; BOERWINKLE; SING, 1987; TEMPLETON, 1998) and *approximate Bayesian computation (ABC)* (BEAUMONT; ZHANG; BALDING, 2002) were designed and rapidly established themselves as tools to support research (TEMPLETON, 2008).

As a side effect to the extensive application of the algorithmic techniques in the research fields, a rivalry among them has grown and their (statistical) validity were put to test (BEAUMONT; PANCHAL, 2008; KNOWLES, 2008; PETIT, 2008b, 2008a; POSADA; CRANDALL; TEMPLETON, 2006; TEMPLETON, 2010, 2008) and the scientific community, now splitted in different fronts, each defending their preferred procedure, presented, in general, uncomfortable with the existing approaches, requiring evolution in the available ones, as well as the development of new ones (KNOWLES, 2002).

Following this trend, some initiatives have risen from diverse areas, from the expected one, biology (PANCHAL, 2008; POSADA; CRANDALL; TEMPLETON, 2000; TEACHER; GRIFFITHS, 2011; TEMPLETON et al., 2005; WOOLLEY; POSADA; CRANDALL, 2008), to pure mathematics (MANOLOPOULOU, 2008), and knowledge areas as *statistics*, *probabilities*, *Bayesian networks*, *graph theory* and *theoretical computer science* have been evoked for improving the analysis tools in these

[1] An algorithm is a mathematical representation (GRAHAM; KNUTH; PATASHNIK, 1994; LEWIS; PAPADIMITRIOU, 1997), under a specific language, of some repetitive set of operations. In this sense, it may be considered a formal representation of specific thoughts. In the context of the current work, though, it is considered as the representation of a software in a general way. A formal representation will be considered the one based on common mathematical notation excepting algorithmically, such as those used on function, graph, (adaptive) automata theories and etc...

prominent research fields in the frontier of *genetics* and *evolution*.

1.1 Objectives

Although the acquisition and processing of genetic code has evolved significantly in the last years, the information extraction from the "raw data" is largely limited by the incipient methods for its analysis; even more severe, it has been diagnosed the way data is modeled may negatively influence the analysis and inference on the collected data (JOLY; STEVENS; VUUREN, 2007).

Thus, in an attempt to minimize the concerns of the community (JOLY; STEVENS; VUUREN, 2007; WOOLLEY; POSADA; CRANDALL, 2008), the present work has the objective of improving the data treatment before the inference algorithms, during the process of composition of the *haplotype network*, including the foundation concepts of *algorithmic information theory* (FORTNOW, 2001; GRÜNWALD; VITÁNYI, 2010; LI; VITÁNYI, 1997; NANNEN, 2003) such as *Occam's razor* and *Epicurus' principle*, as some *graph theory* concepts (BOLLOBAS, 1998; BONDY; MURTY, 2008; DIESTEL, 2010).

1.1.1 General Objectives

To introduce other models for haplotype modification using haplotype network, driving to a new network topology that will provide more (and previously hidden) information to algorithms such as NCPA or ABC for proper inference of the historical evolutionary events.

A formal (and, hence, mathematical) definition of the network will also be considered, intending to establish a permanent bridge between this segment of Biology and Mathematics.

1.1.2 Specific Objectives

In order to demonstrate the validity of the research developed, a number of software were developed for data simulation, execution of the proposed technique and comparison of the final results, both simulated and real ones.

For this objective, both the algorithm and the formal specification of the main application make use of *adaptive technology*(FILHO; ROCHA, 2011) for the handling of the dynamic entities (such as graphs) that may be assembled by the methods proposed in this work.

2 PHYLOGEOGRAPHY

Phylogeography is a research field from biology that studies the biological, geographical and historical processes from particular species (*i.e.* intra-specifically). It derives from or maintains strong intersection with coalescent theory, phylogenetics, population biology, biogeography and (molecular) ecology.

Particularly, Avise (AVISE, 2007) defines phylogeography as

> [...] a field of study concerned with the principles and processes governing the geographical distributions of genealogical lineages, especially those that occur within a species (*i.e.*, intra-specifically).

The term was first referenced — and, also, its formalization as research field — in the 1987 article (AVISE et al., 1987), being classified as a recent biology area; notwithstanding, the phylogeography origins are dated to the end of 1970's (AVISE, 2007).

The phylogeography field was naturally developed from the necessity of understanding why some populations of determined species have a number of distinct biological features than others of the same species. Although many environmental factors motivate these distinctions, the geographical ones are prominent — as *latitude* and *longitude*, among others.

Once the study subject is the variation among populations of particular species, it is also necessary to consider the temporal axis since the most accepted

evolutionary argument (DAWKINS, 2004) has this component in its structure; this way, phylogeography contemplates geo-temporal events.

As an illustrative example, the *Homo sapiens* species is taken. It is (natively) distributed among five of the six continents[1] of the Earth with very distinct characteristics. Analyzing the DNA of sample populations from each localization, there is the possibility to trace the geographical origin of the species, as well as its emigration route, and understand the evolutionary characteristics that made possible the survival of the species under the particular environmental restrictions. In fact, a similar study found the *Mitochondrial Eve* and traced the human origin to Africa (CANN; STONEKING; WILSON, 1987).

2.1 Techniques

As a scientific field — according to the scientific method — phylogeography follows certain set of rules in the process of studying the target populations. The recurrent and systematic use of these rules led to development of a range of techniques, some of them still used nowadays, others abandoned.

The techniques evolved not only following the expansion of knowledge, such as biological, chemical or even computational, but also as a requisite for proper and detailed analysis of the population history. In this sense, two different branch of techniques evolved separately: data *extraction* and data *analysis*.

The first is concerned to collect data with as much information about the population as the current technology is able to analyze. Thus, the extraction of information started as the detailed phenotypic observation (TEMPLETON; BOERWINKLE; SING, 1987) but reached the complex molecular analysis involving several molecular markers and clocks (AVISE, 2007)(FREELAND, 2005); the ease of DNA

[1] It is known that Antarctic continent do not have native humans. (ANISIMOV; FITZHARRIS, 2011)

sequencing, in particular, has been a great evolutionary force in data extraction techniques using molecular analysis.

The latter branch, data analysis, comprises procedures such as nested clade phylogeographic analysis (NCA or NCPA) (TEMPLETON, 2008, 2004, 1998; TEMPLETON; BOERWINKLE; SING, 1987; TEMPLETON; SING, 1993) and haplotype networks and trees(TEMPLETON et al., 2005). This class of methods are used to model and evaluate data in such a structure that the draw of conclusions over the data set is made systematically and straightforward by an actor with a body of knowledge sufficient for inference[2].

2.1.1 Molecular Markers

Molecular markers are known portions of DNA that belong to a specific genome. It means that a molecular marker is a DNA sequence or gene with distinguish characteristics that is used to identify a group as a cell, a tissue or a species.

Molecular markers are used because their sequences and functionality — as well as other particular properties — are known and easily recognized in the whole species samples.

In phylogeography, the most used molecular markers are the mitochondrial DNA, also known as mtDNA, and the nuclear markers.

Mitochondrial DNA (mtDNA) Mitochondrion is a cytoplasmic organelle existent at all eukaryote cells, *i.e.*all cells with complex structures and membranes, as an organized nucleus. It is mainly responsible for cellular respiration (oxidative

[2] *Actor*, in the preceding context, is the one who performs an action(BOOCH; RUMBAUGH; JACOBSON, 2005). By this definition, an actor may be a human, but not indispensably; in many science fields, such as computer science's artificial intelligence subject, the actor is not a human, but a computer program.

metabolism), but also for other activities, as cellular differentiation and death (apoptosis).

Differently from the other organelles, mitochondrion presents — along with chloroplast and the nucleus — in its inner structure, some DNA strand called *mitochondrial DNA (mtDNA)*. This has circular shape (figure 1), similar to the bacterial DNA (JUNQUEIRA; CARNEIRO, 2008).

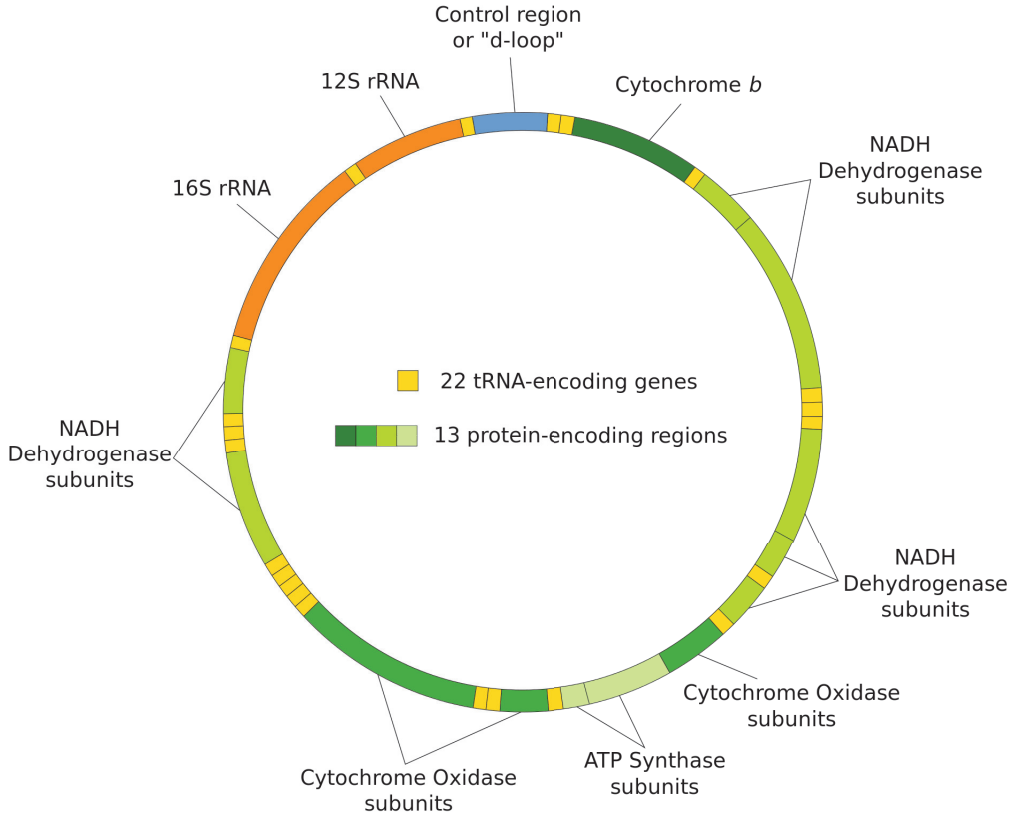


Figure 1: Mitochondrial DNA, in its circular shape. (WIKIPEDIA, 2012)

Under the phylogeographical perspective, the mtDNA is very interesting because of its diverse features. Firstly, mitochondria are transmitted only by maternal lineage[3], enabling a more reliable and robust construction of genetic descent tree; these properties are reached because of the low mtDNA recombination rate[4].

[3]It is known, though, that some species transmit their mitochondria through the paternal lineage; *e.g.* some conifers (MOGENSEN, 1996) and mussels (CAO; KENCHINGTONA; ZOUROS, 2004).

[4]It is important to note that the low mtDNA recombination rate is due the fact that mtDNA comes uniquely from the maternal lineage and, for this reason, any recombination that may

Nevertheless, the excess of DNA strand stability is harmful for *genetic variability* analysis, the study subject of phylogeography; but mtDNA is considerably susceptible to mutations due to spacial proximity shared with oxygen, a highly reactive chemical compound, and also the lower capability of self-regeneration compared with nuclear DNA (FREELAND, 2005)[33].

At last, mtDNA is significantly smaller than nuclear DNA — 180,000 times smaller, approximately — with only 16,569 nitrogenous base[5] pairs (bp) in opposition to the almost 3×10^9 bp of nuclear DNA (ANDERSON et al., 1981).

2.1.2 Haplotype Networks

From all the data collected from phylogeographers, there are special DNA sequences called *haplotypes* that ease the research work. These special sequences are very similar among them except by some particular single-nucleotide polymorphisms (SNPs) that are statistically associated (The International HapMap Consortium, 2003); this similarity provides (i) genetic stability (ii) with sample differentiation through minor mutations (iii) along with smaller nucleotides sequences to analyze, desired properties for a detailed but feasible phylogeography study.

The haplotypes are fundamental data in phylogeography study, allowing a wide range of conclusions provided directly from the molecular data (*i.e.* DNA strands) of the sampled populations.

Although the haplotypes have a great potential of providing important insights on data, their proper analysis can be compromised if immediate and rustic methods are applied; among a set of reasons, the variable size of the haplotypes

occur is not visible because all of them happen with the same genetic sequences, not a foreign one (*e.g.* paternal mtDNA).

[5]The nitrogenous bases are the fundamental composing elements of DNA and ribonucleic acid (RNA). there are five different kinds of nitrogenous bases, named *adenine*, *cytosine*, *guanine*, *thymine* (only presented in DNA molecules), and *uracil* — the latter, exclusively for RNA structures — following specific interconnection patterns with the objective of joining the DNA double helix.

Haplotype	Sequence				
H1	A	T	T	G	G
H2	A	C	T	G	G
H3	A	C	T	G	C
H4	A	T	T	C	G
H5	A	T	T	A	G
H6	A	T	C	C	G
H7	A	C	C	C	C
H8	A	T	T	A	C
H9	A	C	T	A	C

Table 1: Set of nine 5-bp long haplotypes (named H1 to H9).

base pairs sequences, that can be of few dozen to thousands, is a remarkable one: considering a not so long haplotype sequence of 400 base pairs in a set of 20 haplotypes, there would be 8×10^3 base pairs to be compared every time an information should be extracted.

With the desire to simplify the data analysis, a graphical approach was proposed, objectively representing the relationship among its internal elements. This graph is called *haplotype network*.

Haplotypes networks are graphical representations of the sampled haplotypes in a study and of the number of mutations that differs one haplotype of another; typically, the haplotypes are represented by circles whose diameters are proportional to the sampled frequency of each haplotype and an in-circle label representing the haplotype; a number n of mutations that differ two haplotypes[6] are represented as an edge interconnecting two circles with $n - 1$ dashes perpendicular to the edge.

The concept of network is associated to existence of cycles (also called *loops* by biologist) in the graphical representation.

The greatest advantage of a graphical representation over string analysis is

[6]The mutations that differs two haplotypes is tighted with the *edition distance* in bioinformatics. (GUSFIELD, 1997)

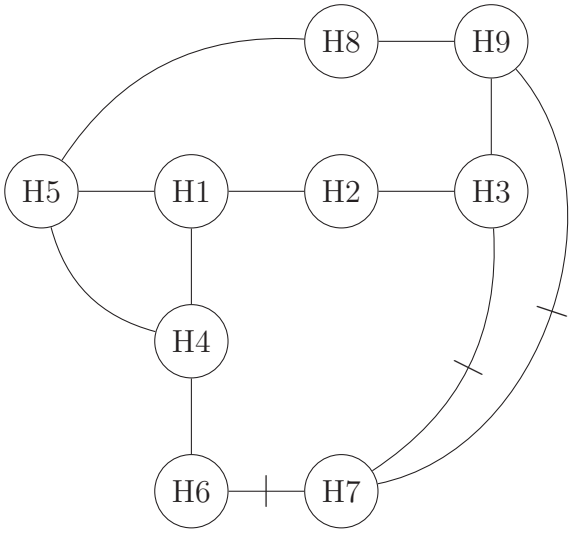


Figure 2: Haplotype network built from data of table 1 showing four loops.

that the relationship information among the haplotypes are faster and easily highlighted; still, frequency information can also be contrasted through the circle diameter. However, the loops that may exist in the graph brings complications to data analysis since it may signal existence of genetic recombination and, hence, a direct history of population in a cladogram style.

In an attempt to transform the haplotype in a cladogram, algorithms that extracts trees from graphs are applied, like Prim's algorithm (TEACHER; GRIFFITHS, 2011; CORMEN et al., 2010); so, a historical perspective is extracted from the haplotype network.

Methods There are a number of methods for haplotype networks construction (LABARRE, 2007), with extensive benchmark evaluations among them (WOOLLEY; POSADA; CRANDALL, 2008).

One of the most used methods is the *statistical parsimony* (TEMPLETON, 2008, 2004, 1998; TEMPLETON; BOERWINKLE; SING, 1987; TEMPLETON; SING, 1993), implemented on the TCS software (CLEMENT; POSADA; CRANDALL, 2000); laid on the concept of finding the most simple relationship among the haplotypes

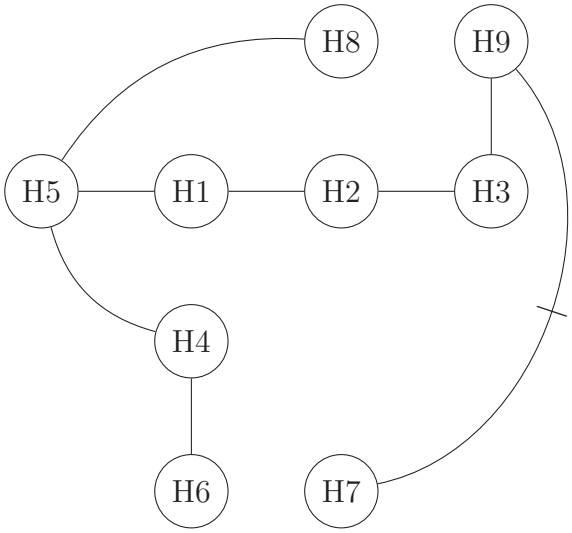


Figure 3: One of the possible haplotype trees built from data of table 1 and figure 2.

— a statistical attempt to apply the *Occam's razor* concept in the networks building — *statistical parsimony* follows four basic steps (TEMPLETON; BOERWINKLE; SING, 1987):

1. Calculate, for every haplotype pair, the probability parsimony $P_{J=1}$ that differ by only one site (substitution).

2. Based on the results of the previous step (1-step network), search for single recombination that can resolve homoplasies.

3. Increase j by one and calculate P_j and merge the (j-1)-step networks with j-step networks until all haplotypes are in a single haplotype network or two or more non-overlapping networks are left.

4. If two or more networks are left without being connected, estimate the minimum number of 95% (or more) probability non-parsimonious changes and connect them.

The statistical parsimony method is, generally, the best method available for haplotype networks and tree building (WOOLLEY; POSADA; CRANDALL, 2008);

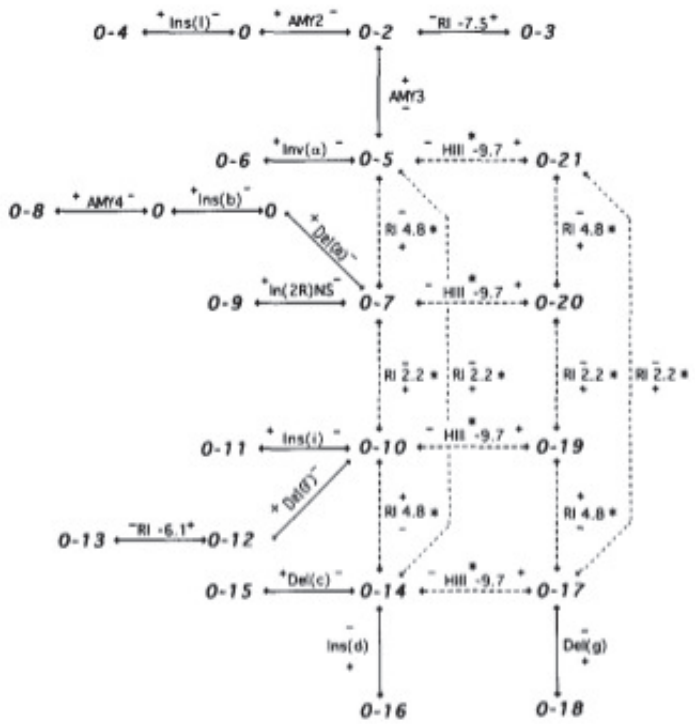


Figure 4: A 1-step network built by statistical parsimony method (TEMPLETON; BOERWINKLE; SING, 1987).

however, it still presents weakness, particularly at sequences with high frequency of recombination appearance.

3 ADAPTIVITY

There are numbers of different computational models, each with particular characteristics that ease the modeling or resolution of certain kinds of problems; among those, one could cite the finite automata, pushdown automata and Turing machines (LEWIS; PAPADIMITRIOU, 1997).

Although these models are widespread used in most computational applications, they are rigidly and formally defined to give solutions to well-defined problems; any deviation to the expected inputs, if not foreseen and pre-programmed, leads to a malfunction of the system, compromising results and efforts — computational or not. There are computational contexts, typically from machine learning but not restricted to, that require runtime modifications in their internal structure; in these subjects, the *training step* is the one where unexpected conditions are verified for posterior parameters reconfiguration with the system not running — realizing the *learning*, from the machine, of previously unknown inputs (RUSSELL; NORVIG, 2003).

The kind of model stated above has been working and in use for years, some of them with computational power for solving problems in the limit of the human capacity of understanding(TURING, 1936; LEWIS; PAPADIMITRIOU, 1997; SIPSER, 2005); however, their notation are not sufficiently adequate to express self-modifying (at runtime) machines, creating the need of a new modeling system. Observing this need, some formalisms were proposed (SHUTT, 1995)(NETO,

1993b), among them the *adaptive automata*(NETO, 2001), the first of the *adaptive devices*.

3.1 Adaptive Devices

An adaptive device is the one that can change its internal structure in run-time, adapting its behavior to the different stimuli it receives — usually external ones. It differs from the standard computational devices since these are fixed instances of pre-determined algorithm. Such self re-modeling property provides the adaptive device the ability to shape itself up to broader classes of problems and, hence, be used to solve problems with dynamic properties and higher model complexity (ROCHA; NETO, 2000).

Clearly a strong formalism permeates the definition of an adaptive device. But, for proper comprehension of the concept, an adaptive framework equivalent model was developed to didactically and correctly explain the concept.

3.1.1 Adaptive Framework

An equivalent model for adaptive devices is a conceptual adaptive framework, surrounding a standard computational device (*e.g.* deterministic finite automaton), compounded of two elements with the ability to modify the subjacent (standard) device. These elements, representing modification functions called *adaptive functions*, act in the structural elements and rules of the inner device in well-defined circumstances: *before* a standard device computational step or *after* it.

During the execution of a computational device, there are transitions among states marking the computational progress; these transitions follow a certain set of rules usually defined over the relation of states and stimuli. An instance of

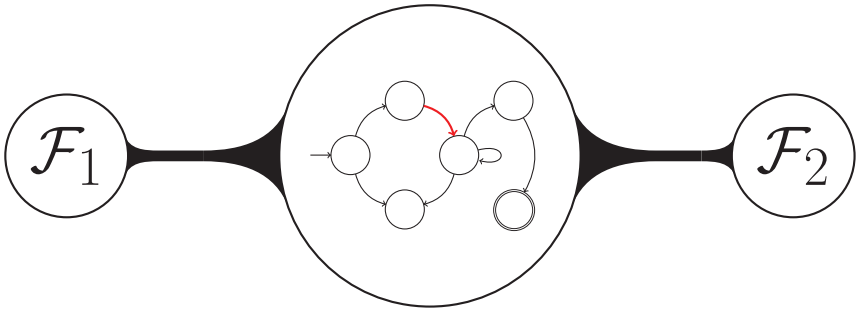


Figure 5: A previous $\mathcal{F}_1$ and a posterior $\mathcal{F}_2$ adaptive functions attached to a traditional subjacent device (with a transition highlighted in red).

this relation — in general, defined as a tuple *(actual state, stimuli)* — is called a *configuration*, that can be understand as a picture of the actual situation of the device.

The transition between one configuration $c_1 = (q_1, w)$ to one $c_2 = (q_2, x)$, represented as $c_1 \vdash c_2$ or $(q_1, w) \vdash (q_2, x)$ (LEWIS; PAPADIMITRIOU, 1997)(HOPCROFT; ULLMAN; MOTWANI, 2003), is called a *computational step* and represents the basic unit of progress in the computational process of the device.

Attached to each computational step, there are adaptive functions that can alter the structural elements of the device with the goal to adapt it to new stimuli. These modifications happen before or after the computational step for maintenance of the traditional behavior of the underlying device under the formal concepts it is defined.

The potential adjustments of adaptive functions concern the states and the rule elements of one device, making possible the recognizement and treatment of and adaption for new inputs and situations(NETO, 2001); these results are possible through the composition of the adaptive functions using addition and remotion actions over the set of structural elements (states or rules) along with a search function to find the specific elements in the sets.

The application of the adaptive functions, as described above, constructs, at

each alteration of the inner device, a new standard computational device within the same class of the previous subjacent one, only with slight differences on their sets of structural elements — but with a traditional concept of algorithm still being executed.

In this sense, the adaptive functions do not act as modifying functions, but solely as selection functions in a space of computational devices(ROCHA, 2008; KAUFFMAN, 1993), defining the best ones to execute a particular computational step based on the triggering stimuli[1].

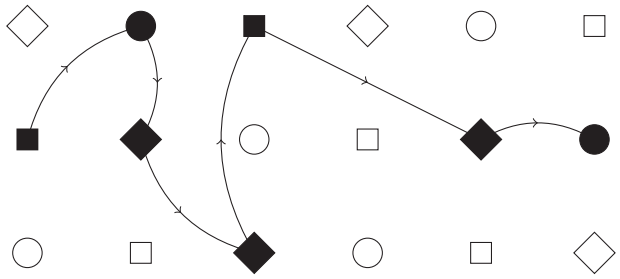


Figure 6: A representation of a traditional computational devices space with many different instances. Highlighted nodes represent visited subjacent devices.

3.1.2 Mathematical Formalism

The adaptive devices were proposed, originally, in (NETO, 1993a) with a well-spring mathematical formalism that has been evolving in two different branches: the classical formalism (NETO, 2001), based on set and automata theory, and a direct heritage of the first formalism proposed; and the algebraic one (FILHO; ROCHA, 2011), influenced by mathematical algebra, modeling ideas and control theory.

For historic and clarification reasons, both formalisms are explained; since they require a previous knowledge on formal languages theory, supportive references on this subject are provided at (LEWIS; PAPADIMITRIOU, 1997; SIPSER,

[1]This concept of "walk on a space" is also present in biology (in the context of adaptivity (KAUFFMAN, 1993)) and complex systems areas.

2005; ROZENBERG; SALOMAA, 1997).

3.1.2.1 Classical Formalism

Definition 1. *Let an adaptive device* $AD = (ND_0, AM)$*, which* ND_0 *is a non-adaptive initial subjacent computational device;* $AM \subseteq BA \times NR \times AA$ *is an adaptive mechanism;* BA *(anterior adaptive function) and* AA *(posterior adaptive function) are adaptive functions; and* NR *is the set of all non-adaptive rules for* AD*. (NETO, 2001)*

Elementary adaptive actions are primitive editing functions that compose every adaptive function. There are three different elementary adaptive actions acting on parameter pattern matching (as formally defined in definition 2): (i) inspection (or rule-searching); (ii) deletion (or rule-erasing); and (iii) insertion (or rule-inserting). (NETO, 2001)

Definition 2. *Let* $\langle rule \rangle$ *be the pattern of a rule defined for the ordinary encapsulated device* ($\langle rule \rangle \in \delta(M_i)$) *, then the elementary adaptive actions are:*

(i) *inspection actions, represented by* $?[\langle rule \rangle]$*, which returns all the rules of the computing function matching the pattern rule;*

(ii) *deletion actions, represented by* $-[\langle rule \rangle]$*, which deletes all the rules of the computing function matching the pattern rule;*

(iii) *insertion actions, represented by* $+[\langle rule \rangle]$*, which inserts the pattern rule to the set of rules of the computing function of the device.*

(ROCHA, 2009)

Definition 3. *Let* Φ *be a space of elementary adaptive actions and* $\mathbb{M}$ *be the one of ordinary devices. An* adaptive function *is defined as a set* $\{\phi_0, \ldots, \phi_n\}$*, with*

$\phi_i \in \Phi$, $0 \leq i \leq n$ and $n \in \mathbb{N}$, that, when instanced and executed as an adaptive action, changes the underneath device M_i to M_j, $\{M_i, M_j\} \subseteq \mathbb{M}$. (ROCHA, 2009)

The application of the concepts above were first to define adaptive systems and are enough (as shown in definition 1) to specify an adaptive computational device, as an adaptive automata. Nonetheless, the way it represents an adaptive device may become uncomfortable for newcomers or, even, be too complicate to express some kinds of adaptive systems. This way, following the paradigms opening that exists in theoretical computer science (LEWIS; PAPADIMITRIOU, 1997; SIPSER, 2005; CARNIELLI; EPSTEIN, 2009), a new representation pattern was proposed as detailed in section 3.1.2.2.

3.1.2.2 Algebraic Adaptivity

An alternative formalism, more elegant and clear, was proposed in (FILHO; ROCHA, 2011), strongly influenced by (mathematical) algebra and theory of computation, turning the adaptive formalism easier to implement and understand. Nonetheless, the algebra consider solely the adaptive automata devices.

In a functional composition notation, this branch of adaptivity theory reaches a formal closure not defined before in the field and intrinsically presents software development guidelines for functional programming such as LISP.

The algebraic adaptivity is based on external and internal elements. The external elements are those used to assist the operations and functions; the internal ones are intrinsic to the algebraic approach.

External Elements Before the presentation of the adaptive algebra, an "external framework" of functions is defined to support some operation of the former, those not concerning the modification of the subjacent device.

Definition 4. *Let S be the* proper-search operation *defined as* $S : Q \cup \emptyset \times \Sigma \cup \{\epsilon\} \times Q \cup \emptyset \times \mathcal{M}^0 \mapsto \{Q \times \Sigma \cup \{\epsilon\} \times Q\} \subseteq \{\partial \cup \emptyset\}$.

The proper-search operation has eight possibilities:

$$S(q_i, a, q_j, M^0) = \{(q_i, a, q_j) : (q_i, a, q_j) \in \partial\} \tag{3.1a}$$

$$S(q_i, q_j, M^0) = \{(q_i, \alpha, q_j) : (q_i, \alpha, q_j) \in \partial\} \tag{3.1b}$$

$$S(q_i, M^0) = \{(q_i, \alpha, q_j) : (q_i, \alpha, q_j) \in \partial\} \tag{3.1c}$$

$$S(q_j, M^0) = \{(q_i, \alpha, q_j) : (q_i, \alpha, q_j) \in \partial\} \tag{3.1d}$$

$$S(q_i, a, M^0) = \{(q_i, a, q_j) : (q_i, a, q_j) \in \partial\} \tag{3.1e}$$

$$S(a, q_j, M^0) = \{(q_i, a, q_j) : (q_i, a, q_j) \in \partial\} \tag{3.1f}$$

$$S(a, M^0) = \{(q_i, a, q_j) : (q_i, a, q_j) \in \partial\} \tag{3.1g}$$

$$S(M^0) = \partial \tag{3.1h}$$

Definition 5. *A* non-pertinence identification operation *is an external element of the adaptive algebra such that:*

$$N(\delta, M^0) = \begin{cases} \delta & \Leftrightarrow \delta \notin \partial \\ \emptyset & \Leftrightarrow \delta \in \partial \end{cases}$$

Definition 6. *A* start identification function *is the one that returns the start state of a transition, i.e. for a transition $\delta = (q', \alpha, q'')$, $A(\delta) = q'$.*

Definition 7. *A* finish identification function *is the one that returns the end state of a transition, i.e. for a transition $\delta = (q', \alpha, q'')$, $\Omega(\delta) = q''$.*

Adaptive Algebra The adaptive algebra is focused exclusively in the modification operations

Definition 8. *A proper-transition remotion operator is defined as* $f^-_{\delta_{pro_k}} M^0 = f^-(\delta_{pro_k}, M^0) = (Q, q_0, E, \Sigma, rem(\partial, \delta_{pro_k}))$, *with* $\delta_{pro_k} \in \hat{\delta}_{pro}$ *and* $\hat{\delta}_{pro} =$

$(\delta_{pro_1}, \ldots, \delta_{pro_m})$.

Definition 9. *A foreign-transition addition operator is defined as* $f^+_{\delta_{for_k}} M^0 = f^+(\delta_{for_k}, M^0) = (ins(ins(Q, q'), q''), q_0, E, \Sigma, ins(\partial, \delta_{for_k}))$*, with* $\delta_{for_k} \in \hat{\delta}_{for}$ *and* $\hat{\delta}_{for} = (\delta_{for_1}, \ldots, \delta_{for_n})$.

Definition 10. *Let* $F = (f^-, f^+)$ *and* $\phi = (\hat{\delta}_{pro}, \hat{\delta}_{for})$*. An adaptive function is defined as* $\mathbb{F}_\phi M^0 = \mathbb{F}(\phi, M^0) = F^-_{\hat{\delta}_{pro}} F^+_{\hat{\delta}_{for}} M^0$.

The application of operation F into machine M^0 (definition 10) follows the function composition below, describing the *remove* (equation 3.2) *insertion transformations* (equation 3.3).

$$F^-_{\hat{\delta}_{pro}} M^0 \triangleq F^-(\hat{\delta}_{pro}, M^0) = (f^-_{\delta_{pro_m}} \circ f^-_{\delta_{pro_{m-1}}} \circ \ldots \circ f^-_{\delta_{pro_2}} \circ f^-_{\delta_{pro_1}}) M^0 \tag{3.2}$$

$$F^+_{\hat{\delta}_{for}} M^0 \triangleq F^+(\hat{\delta}_{for}, M^0) = (f^+_{\delta_{for_n}} \circ f^+_{\delta_{for_{n-1}}} \circ \ldots \circ f^+_{\delta_{for_2}} \circ f^+_{\delta_{for_1}}) M^0 \tag{3.3}$$

3.1.2.3 Models' Equivalence

A full-equivalence between the classical and algebraic formalisms exists and is formally presented at (FILHO; ROCHA, 2011), enabling the application of known theoretical results reached from a model to another.

Runtime Dynamics The runtime self-adaptive and, as a consequence, dynamic behavior of the adaptive devices is the greatest advantage of this class of computational devices against Turing machines, pushdown automata, finite state automata or other traditional devices.

This property is equally inherited by both formalisms, along with the additional easy and powerful expressiveness characteristic of both of them.

One Adaptive Function per Transition The classical adaptive definition, as represented in figure 5 and stated in definition (NETO, 2001), declares the existence of two adaptive functions — pre- and post-transition adaptive functions — for proper modifications of the subjacent device.

However, it was demonstrated that any adaptive device with pre- or post-transition adaptive function has an equivalent one with only one adaptive function for each transition, with the property of execution either before or after the subjacent machine transition (ROCHA; NETO, 2005).

That result is used in the core concepts of the algebraic adaptive model, since an unique adaptive function per subjacent transition is allowed in that model (FILHO; ROCHA, 2011).

Turing Powerful The device proposed by Alan Turing (TURING, 1936) — the so-called *Turing machine* — is the theoretical foundation for computation, presenting widely recognized interesting characteristics; among many of them, its most prominent is the ability to recognize (and, in particular cases, only accept) the broadest set of languages among all the computational devices. Over Turing machine fundamental concepts as *effectively calculable (or computable) procedure* and *computational complexity* are defined (TURING, 1936; LEWIS; PAPADIMITRIOU, 1997; SIPSER, 2005; CARNIELLI; EPSTEIN, 2009); particularly, the *Turing-Church thesis* also derives from this conceptual device, stating that if some algorithm exists to calculate some method, it can be executed by a Turing machine and, equivalently, by λ-calculus or recursively-defined functions.

From the information stated by the Turing-Church thesis (and the knowledge

that this device accepts all classes of languages in the Chomsky hierarchy (LEWIS; PAPADIMITRIOU, 1997)) follows the conclusion that Turing machine is the most "powerful" computational device and every machine with the same computational power is called **Turing powerful** or **Turing complete**.

Any device modified by the adaptive formalism presented in sections 3.1.2.1 and 3.1.2.2, even the simplest one as a deterministic finite state automaton, has the same computational power of a Turing machine (ROCHA; NETO, 2000)(ROCHA, 2011); *i.e.* recognizes (or accepts) the same languages a Turing machine can recognize (or accept), being, hence, a *Turing powerful* device.

4 GRAPHS

When the seven bridges of Königsberg problem dared the mathematicians in the 18th century, Leonhard Euler presented an elegant negative resolution to it, as well as presented to the mathematics community a new study field composed of simple visual structures with high potential of representativeness called *graph theory* (HAWKING, 2005).

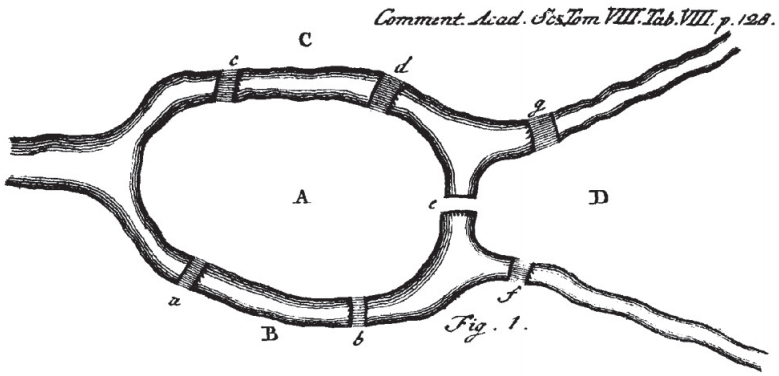


Figure 7: Euler's drawing of the Königsberg bridges.(EULER, 1736)

Graphs are visual representations made of vertices and edges, simple and natural components present in many scratches of modeling. Through these two elements, unimaginable equivalences between real world problems and drawings become possible; with some characteristics extension, as vertices shape and edged labels, the number of equivalences grows exponentially.

The expressiveness power of graphs are increased with some modifications in their compounding elements, as adding labels to edges, color edges and vertices,

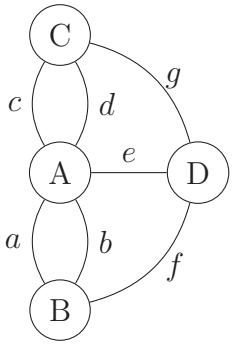


Figure 8: Graph representation of the Königsberg bridges.

among others.

Since graph theory is a discrete mathematics instance, it is well defined through rigorous formalisms presented below.

Definition 11. *A graph is an ordered pair $G = (V, E)$, where V is a set of vertices and E is one of edges, with $E \subseteq [V]^2$ and $V \cap E = \emptyset$. (DIESTEL, 2010)*

As an example, let G be the graph presented in figure 9. At the example, the set of vertices $V = \{A, B, C, D, E\}$ and the set of edges is $E = \{\{A, B\}, \{A, C\}, \{A, E\}, \{B, C\}, \{C, D\}, \{C, E\}\}$.

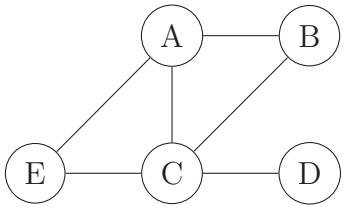


Figure 9: Graph $G = \{V, E\}$.

It is important to note that the names of the vertices are not restricted to single letters of the alphabet. In fact, for expressiveness increase, they are labeled in easy-for-understanding way such as figure 10, where the graph $G' = \{V', E'\}$ has $V' = \{\text{Campinas}, \text{Ribeirão Preto}, \text{Rio de Janeiro}, \text{São Paulo}, \text{Sorocaba}\}$ and $E' = \{\{\text{São Paulo, Sorocaba}\}, \{\text{São Paulo, Campinas}\}, \{\text{São Paulo, Rio de Janeiro}\}, \{\text{Sorocaba, Campinas}\}, \{\text{Campinas, Ribeirão Preto}\}, \{\text{Campinas, Rio de Janeiro}\}\}$.

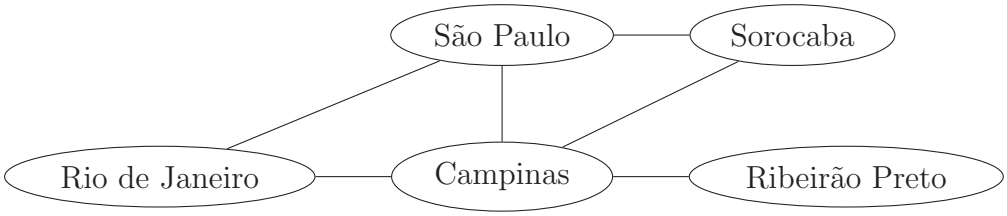


Figure 10: Graph $G' = \{V', E'\}$.

All the graphs presented so far are *undirected*[1] *and unweighted graphs* (BONDY; MURTY, 2008)[31](DIESTEL, 2010)[28], *i.e.* the edges do not flow in a certain ordering as well as there are no real number associated to them.

Definition 12. *A weighted graph, denoted* (G, w)*, with* $w : E \mapsto \mathbb{R}$*, is one which a real value (weight) is attributed to each edge of the graph. (BONDY; MURTY, 2008)*

The usage of weight in the edges add new information in the graph data pool, clarifying new relations among the elements, as priority, probability, required constraint or others; it enriches the model, turning possible the information extraction in an optimized way.

An instructive example derives from figure 10. In this figure, each city is represented as a vertex and the relation among them through the set E' of edges. It is clear, looking at this graph, that a person leaving from *Sorocaba* reaches *Rio de Janeiro* either going to *São Paulo* or *Campinas*, but no information is provided regarding which one is the shortest path to go.

If a function w that weights each edge with the distance, in kilometers, between the vertices is attached to the graph G' — as can be seen in figure 11 — the acquisition of the shortest path information is rapidly provided. At the *Sorocaba–Rio de Janeiro* travel, the shortest path is *Sorocaba–São Paulo–Rio de Janeiro* which is 12 kilometers shorter than *Sorocaba–Campinas–Rio de Janeiro* one by the graph in figure 11.

[1]The notion of directed graph is not relevant in the scope of this research work.

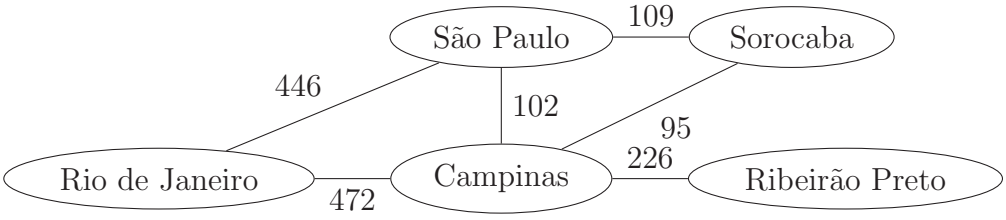


Figure 11: Graph G' with weighted edges.

The figure 10 (or figure 11) presents a graph containing some vertices not connected to others. Notwithstanding, there is a special type of graph called *complete graph* in which every vertex is connected to the others, except itself.

Definition 13. *A complete graph is an undirected one which* $G = (V, E)|\{e_i, e_j\} \in E$ *and* $e_i \neq e_j$. *If* $|V| = n$, G *is represented as* K_n. *(BONDY; MURTY, 2008)*

Corollary 1. *Given a* K_n *complete graph, its number of edges* $|E| = \frac{n!}{2! \times (n-2)!} = \frac{n \times (n-1)}{2}$.

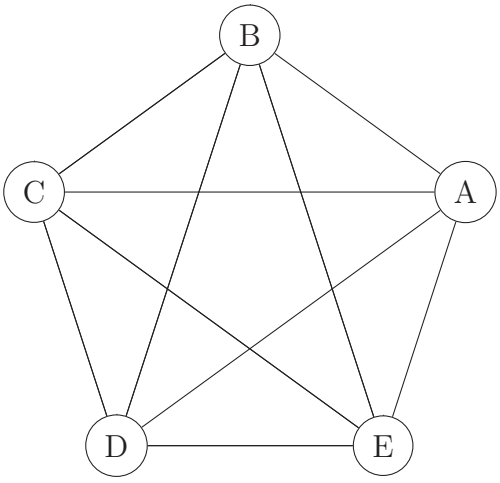


Figure 12: A K_5 graph.

Definition 14. *In a graph* $G = (V, E)$, *two vertex* $\nu', \nu'' \in V$ *are* adjacent *if* $\exists\{\nu', \nu''\} \in E$. *(BONDY; MURTY, 2008)*

Definition 15. *A* vertex degree $d_G(\nu')$ *of a graph* G, *for any* $\nu' \in V$, *is the quantity of edges* $\{\nu', \nu''\} \in E, \forall \nu'' \in V$. *If there is an edge in which* $\nu'' = \nu'$, *that edge is counted twice.*

A complete graph, by definitions 14 and 15, can be described as the one which every vertex is adjacent to all other vertices (except itself) and, hence, any $d_G(\nu) = |V| - 1$.

Definition 16. *Given a sequence* $(\nu_0, \nu_1, \cdots, \nu_k)$ *of non-repeating vertices and* $k \geq 2$*, for* $\nu_i, 0 \leq i \leq k$*. Let* ν_i *be adjacent to* ν_{i+1} *for* $0 \leq i < k$ *and* ν_k *be adjacent to* ν_0*. A graph* $C = (V, E)$ *with* $V = \bigcup\{\nu_i\}, 0 \leq i \leq k$*, is called a* cycle. *(BONDY; MURTY, 2008)*

Definition 17. *A graph* T *without cycles (acyclic) and connected is called a* tree. *(BONDY; MURTY, 2008)(DIESTEL, 2010)*

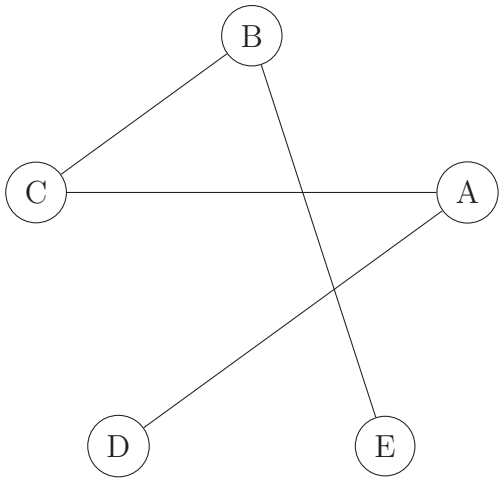


Figure 13: Tree T derived from figure 12.

5 GRAPHS UNIFICATION

There is so much skepticism over the results of automatic methods for data analysis in the biological field, specially because there are lots of unknown variables even for (human) experts and many relationships usually show up after long studies, experiments, field and laboratory work. In phylogeographic studies, it may get even worse since the data under analysis are highly (and necessarily) attached with geographical, geological, evolutive and genetics events.

On the other side, with the improvement in and extensive use of data collecting techniques, the massive amount of raw data to analyze in a current routine research becomes intractable under the treatment of a (or even several) human specialist; in this situation, an automated information extraction system could work several times faster than human beings and point out hidden relationships among data collected through extensive application of diverse algorithms supported by its architecture, *i.e.* its big memory and processor speed.

Seeking for a solution intersecting both worlds, some algorithms were proposed using, in most cases, statistical approaches for data analysis and some sort of "expert knowledge" through some artifices, such as (i) inference keys (TEMPLETON, 1998); (ii) Bayesian networks (BEAUMONT; ZHANG; BALDING, 2002); (iii) model selection score.

Notwithstanding these efforts, their contribution is limited and, in some sort, come to negatively interfere in the ongoing research progress and conclusions;

some of them have been publicly attacked in the scientific community (KNOWLES, 2008), what has driven the computational biology community to seek the development of algorithms to solve this particular problem (KNOWLES, 2002)(KNOWLES, 2003)(BEAUMONT; ZHANG; BALDING, 2002).

The solutions proposed, nonetheless, still present some weak aspects; in the general case, they eliminate concurrent hypothesis with little confidence degree (if compared with the one automatically selected by the algorithms) because they are unable to handle multiple explanations or ensemble of models. It is even true for those methods in which high human interference in the system analysis exist: in such cases, an automated algorithm scores the models that probably fit the data and, through an ordered list, outputs the model the scientist should use to direct the explanations need in his research. This kind of algorithm, although of some help, hurts some scientific, philosophical and common sense principles, specially the *Epicurus' principle of multiple explanations* (LI; VITÁNYI, 1997), restricting the result to a single hypothesis based uniquely in a numerical value scored by probabilistic — or even by some "pseudo-expert" knowledge source.

In the search of a broader, more complete and less sensitive algorithm to the variations of Nature, an unification algorithm is proposed, mainly acting in the input data.

As shown in figure 14, haplotype networks are input data in the processing algorithms, determining primary relationship among the several haplotype sequences extract from living beings in field work. Nonetheless, it is known that incorrect or missing data in these networks compromise the correct analysis of phylogeographic events (JOLY; STEVENS; VUUREN, 2007), leading to incorrect conclusions (or even no conclusion at all) about the ongoing research because of propagation of errors, also known as *butterfly effect*(LORENZ, 1963). Since the haplotype networks are fundamental input data for phylogeographic inference,

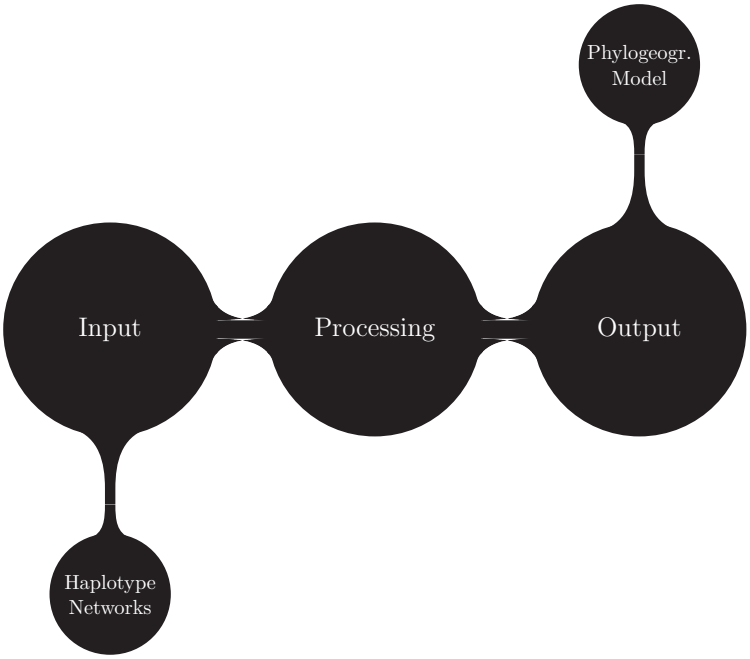


Figure 14: Data-flow of data analysis algorithms (left to right).

the proposed algorithm is totally focused in the treatment of this particular data.

By historical reasons(AVISE, 2007), mitochondrial DNA has been used in phylogeographic research because of some properties as high mutation rate, low recombination rate and traceable maternal origin, besides being smaller than nuclear DNA. However, the extensive usage of this material has reached the limitation of useful information it provides, driving researchers to the exploration of nuclear DNA (KNOWLES, 2002)(KNOWLES, 2003) where new properties become available (*e.g.* recombination rate, significantly higher if compared if mitochondrial one). Still, transformations over haplotype networks are applied — such as *minimum spanning tree* (MST) extraction — reducing the amount of information this representation provides and, hence, restricting the hypothesis space for inference.

In face of new contexts (as those stated above), new data analysis methodology comes out necessary since the old ones are unable to deal with current requirements and particularities (WOOLLEY; POSADA; CRANDALL, 2008). With

the intention to increase the amount of available information from the same input data set, an algorithm of graph composition (or graph unification) is proposed.

From definition 11, the isomorphism between a haplotype network and a graph appears naturally; this property is also highlighted in some other publications (JOLY; STEVENS; VUUREN, 2007; TEMPLETON, 1998). The haplotypes in a haplotype network compound the vertices set of the graph and their relationships are represented by the edges of the latter; this way, independent of the algorithm which will synthesize the haplotype network, the set of vertices is previously known and depends uniquely on the raw input data from field work, being the edges set the one dependent of algorithm implementation.

Under this perspective, the diverse range of existing methods for haplotype network (WOOLLEY; POSADA; CRANDALL, 2008), each of them highlighting a specific attribute of the input data, could be used in parallel to produce many (different) haplotype networks representing an (different) instance of the hypothesis space and, hence, increasing the number of alternatives an inference algorithms can analyze.

Obtaining all the haplotype networks separately and performing individual analysis of each of them is, solely, a reinterpretation of the original problem, requiring, the same way, a human specialist to interpret and unify the intermediate results. For this reason, a new complete graph $G = (V, E)$ is constructed, with $V = \bigcup V_i$ and each edge $\{v', v''\} \in E$ selected among from E_i under certain rules depending on the weight of the vertices (algortihm 1). The remaining $\{v', v''\}$ necessary to construct the complete G graph and not in E are added with suitable information expliciting their non-existence in the original haplotype networks.

The graph G, with all the possible (and most probable) relationships among the haplotypes, presents a hypothesis space denser than any individual one presented by each of the methods used to compound this new graph (and space),

reintroducing the Epicurus' principle to the universe of data analysis.

5.1 Algorithm

For correct treatment of the data (haplotype networks) and widen hypothesis space, a formal definition is required in order to standardize the independent processing of incoming stimuli; under a computational view, an algorithm is the best and straightforward solution to achieve this feat. This way, the existence of a general algorithm for haplotype network processing is only possible under the definition of data standards that guarantees the optimal computation functionality.

5.1.1 Algorithm Input Restrictions

Seeking an uniform data processing, the haplotype networks must follow the same constructions rules, keeping a visual identity among them.

In general manner, three main restrictions are applied:

1. Nonexistence of zero-type haplotype.
2. Edition distance between haplotypes are represented as edge weight.
3. There are no two isomorphic graphs.

It is important to note that the restrictions do not diminish the generalist aspect of the algorithm; actually, these restrictions act as a transformation to a common representation where all data are in such a structure they are optimally manipulated.

5.1.1.1 Nonexistence of Zero-Type Haplotype

Haplotype networks are visual representations with high expressiveness and easiness of use, commonly used in phylogeographic community to represent relationship among sampled haplotypes. Nevertheless, the evolutive history of haplotype networks are dynamical, with variations being added through published contributions and no standardization defined along the years; therefore, the most used drawing identities became the *de facto* standard representations.

Some of these representations (CLEMENT; POSADA; CRANDALL, 2000; TEACHER; GRIFFITHS, 2011) use the concept of "zero-type haplotype" to keep the connectivity of the graph. The zero-type haplotype is a virtual node added to the haplotype network so every edge represents an edition distance of one and there is a path between any two haplotypes, with one of the sampled haplotypes always being reached through the walking over the zero-type haplotype.

The name *zero-type haplotype* derives from the notation used in the Templeton's works (CLEMENT; POSADA; CRANDALL, 2000; TEMPLETON, 1998), where every zero-type haplotype is labeled after the number zero ("0") while the sampled ones are named after the natural numbers greater than zero. Other references, though, do not label de zero-type haplotypes, leaving them without label (TEACHER; GRIFFITHS, 2011).

The absence of zero-type vertices in the graph avoids label and definition conflicts, since every $v \in V$ has a distinct name and no element repetitions are allowed in V since it is a set; for multiple zero-type vertices as shown in figures 15 and 16, a type system should exist, defining different vertices in the set V, but some with common properties, being the zero-type of these specific types, sharing the property of being virtual and not sampled. This activity, though, is not a central one for the current purposes and, hence, is out-of-scope.

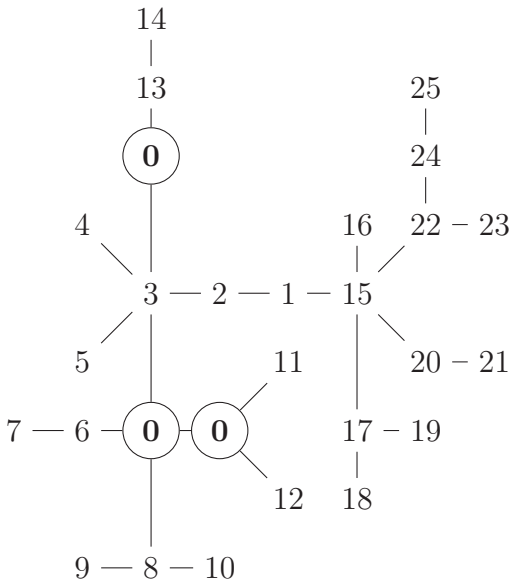


Figure 15: Haplotype network presenting a *zero-type haplotype* highlighted (TEMPLETON; BOERWINKLE; SING, 1987).

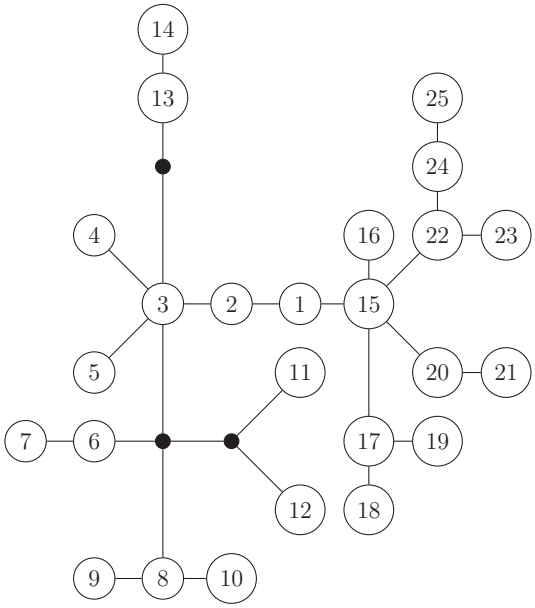


Figure 16: Haplotype network presenting a *zero-type haplotype* as black vertices (TEMPLETON; BOERWINKLE; SING, 1987; TEACHER; GRIFFITHS, 2011).

Also, as a complement feature, the lack of zero-type vertices explicits the relationship among the sampled haplotype, demanding the information of editing distance be clearly annotated in the drawing for proper graph comprehension. This way, an automatic process can easily extract this information for further use.

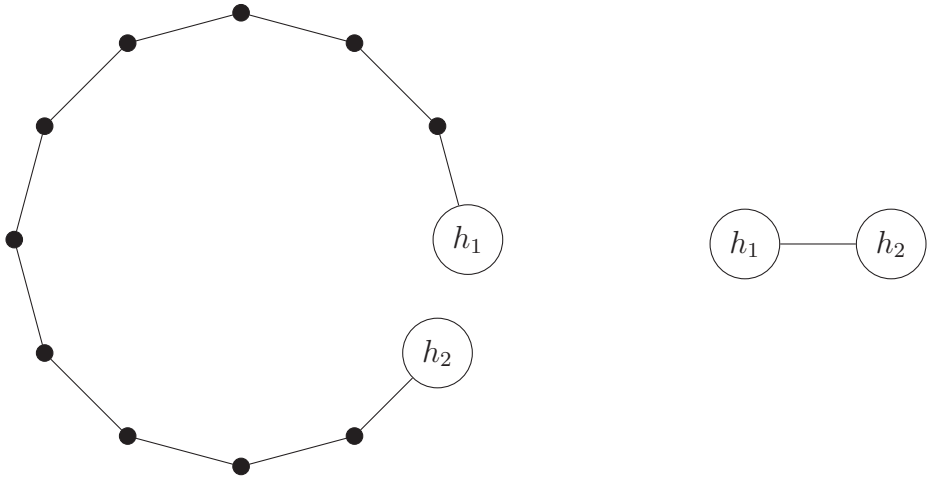
5.1.1.2 Edition Distances Between Haplotypes Are Represented As Edge Weight

An usual application of graphs as haplotype networks is the representation of "mutation distance" of one haplotype to another one, interconnecting them through an edge if a single nucleotide (SNP) between the haplotypes are different. In phylogeographic studies, particularly those based on mitochondrial DNA, that kind of haplotype network usage is very common.

However, a phylogeographic study is not limited to mutation analysis and may include other kind of "edition over the genetic sequences, *e.g.* recombination. In these new situations the graphical representation can also be used and is, actually, desirable since highlights relationships harder to see than SNPs.

As an example, if an haplotype h_1 has 11 SNPs of recombination origin different from a haplotype h_2, the former would be separated through 11 edges from the latter in a mutational graph, drawing a very polluted and confusing picture, allowing errors in data interpretation; in the other hand, the former would be separated by a single edge, representing an unique recombination (or editing operation), from the other in a recombination graph.

For interpretation and future comparison purposes, every connection between haplotypes is made through a single edge with the proper editing distance in that graph between the vertices represented as the weight of the edge. In the example cited above, the connection between h_1 and h_2, in both cases, would



(a) Unweighted mutational graph (b) Unweighted recombination graph

Figure 17: Example of unweighted haplotype networks representations

(a) Weighted mutational graph

(b) Weighted recombination graph

Figure 18: Example of weighted haplotype networks representations

be represented by a single edge, one with weight equals to 11 and the other with weight equals to 1, keeping the clearness and all information provided by graphical representation.

Another advantage of this modeling is the straightforward connection between two distinct haplotypes, optimizing the search operations of automatic algorithms and ease of application of well-known ones such as Dijkstra's algorithm (CORMEN et al., 2010); under this perspective, the construction of a complete weighted graph in this model and the application of the latter algorithm can be used to extract new and valuable information in a different way than proposed by the current, most used, methods based, solely, on Prim's algorithm (CORMEN et al., 2010).

5.1.1.3 There Are No Two Isomorphic Graphs

The existence of isomorphic graphs as product of application of series of graph synthesis methods is quite reasonable and should be expected in a context like

this. Nonetheless, in circumstances where automatic algorithms process an input set of graphs, the presence of isomorphic graphs increases the complexity of a general solution since features as verification of already processed graphs and performance optimization must be implemented.

For the purpose of this research, hence, it is considered that no two isomorphic graphs exists in the input data set.

5.2 Algorithmic Steps

The development of the algorithm gets easier with the input data restrictions presented in sub-section 5.1.1; though, a functional fragmentation is still necessary for proper picture the scenarios and understand the procedures.

This way, the algorithm is divided in three different functional steps, enough for the correct construction of the proposed algorithm. The steps are defined as:

1. **main algorithm**, where the haplotype networks act as input data and the main procedures are executed to the proper treatment of these data, producing an unified graph with the desired properties presented in the beginning of this chapter as output;

2. **vertices function**, where a set V of all the vertices contained in the input data (haplotype networks) is defined;

3. **minimum weighted edge function**, which seeks among the many input graphs in $\mathbb{G} = \{G_1, G_2, \ldots, G_N\}$ for the edge $e^i_{v,v'} = \{v, v'\}$, fixed v and v', that minimizes the weight function w such that $w\left(e^i_{v,v'}\right) = \min\left\{w\left(e^1_{v,v'}\right), w\left(e^2_{v,v'}\right), \ldots, w\left(e^N_{v,v'}\right)\right\}$, with $e^i_{v,v'} \in V_i \in G_i \in \mathbb{G}$.

5.2.1 Main Algorithm

The idea of the main algorithm, as state above, is to manipulate the input data producing a single output graph, particularly a *network* in the phylogeographic jargon, with the main characteristics of each input graph according to Epicurus' principle of multiple explanations; for this accomplishment, a series of requirements must be satisfied.

First of all, a graph is a pair that consists of a set of vertices V and a set of edges E (as defined in chapter 4) and although, formally, V and E are allowed to be an empty set, V is always known (to be not empty) because it is the set of haplotypes; the same, though, is not true for the edges set E and, for this reason, it is considered initially empty in the output graph.

Since a haplotype can not derive from itself because it does not make any sense, a vertex $\{v, v\}$ has no possibility to exist in the input data or the output graph. On the other side, a complete graph may exist — even the output one — and, this way, every edge $\{v, v'\}$, with $\{v, v'\} \subseteq V$ and $v' \neq v$, must have its existence verified against the set $\mathbb{G}$ of input graphs. Therefore, this procedure is realized through the usage of a special *search function* covering the elements of the set $\mathbb{G}$.

Even if an edge $e^i_{v,v'}$ belonging to a $G_i \in \mathbb{G}$ is found, it is not a guarantee that it will be included as an element of the set E: the found edge must be a representative of the most relevant property in the relation between haplotypes v and v'; within this context, this representative, which will be called $e_{v,v'}$, is the edge with the minimum associated value for the weight function w.

Once the edge $e_{v,v'}$ is found, it is added to the set E, representing the optimal (under the set $\mathbb{G}$) relation between the vertices v and v'. Otherwise, an edge is created with $w\left(e_{v,v'}\right) = +\infty$.

The process of finding edges with desired properties, selecting the best representative and adding it to the set of edges of the output graph resembles the elementary adaptive actions presented in the adaptive automata theory. Under a mathematical view, the adaptive automata theory is desirable since it has many theoretical results already proved and known, as well as particular and appropriate notation for self-modifying systems.

The morphism between graph and automata exists, derived from category theory (PIERCE, 1991), and is used in a range of applications (FILHO; ROCHA, 2011), also in the current proposal through the usage of adaptive automata theory for mathematical formalism presented in the notation introduced in section 3.1.2.2.

Concerning the algorithm 1, the process of adding an $e^i_{v,v'}$ edge to the set E of edges is the unique adaptive action explicited in the following pseudo-code. Formally, it represents the *insertion function* (equation 3.3), which is defined and mathematically described (for this context) in section 5.3.

Algorithm 1 Main algorithm

Require: $\mathbb{G} = \{G_1, G_2, \ldots, G_N\}$
Ensure: Composition of graph G from every $g \in \mathbb{G}$

```
V ← Vertices(𝔾)
E ← ∅
for i = 1 to |V| do
    v ← v_i ∈ V
    for j = i + 1 to |V| do
        v' ← v_j
        e ← MinimumWeightedEdge(v, v', 𝔾)
    end for
    E ← E ∪ {e}
end for
```

5.2.2 Vertices Function

The algorithm described in section 5.2.1 presumes the access to the whole set of vertices (haplotypes) available from the input data. While it may be true in

a monolithic application responsible for the full treatment of the field collected data, the proposed algorithm cannot assume this hypothesis as a tautology since it has a generalist aspect and must keep the possibility of being implemented in a wide range of software, including specific haplotype networks unification ones; in this sense, to define a function for gathering all the vertices in a single set is justified.

The $Vertices$ function has the objective of visiting all the graphs G_i ($1 \leq i \leq |\mathbb{G}|$) belonging to $\mathbb{G}$, the argument of the function, and, for each vertex v_j^i found in the set V_i, with $1 \leq j \leq |V_i|$, add it to the general set of (all) vertices V.

It is important to note that the set V is, initially, empty ($V = \emptyset$) and the addition of the vertices in this set is done through the set operation of *union* ($\cup$), what prevents any vertex to be added two (or more) times in the general vertices collection V — actually, defined as a *set*.

Although the $Vertices$ function may appear an (insertion) adaptive action, it is solely a formalism that guarantees the existence of fundamental components (vertices) for the graphs unification operation.

Algorithm 2 Function $Vertices(\mathbb{G})$

Require: $\mathbb{G} = \{G_1, G_2, \ldots, G_N\}$
Ensure: A set $V \in G$ containing all vertices presented in the graphs $g \in \mathbb{G}$
1: $V \leftarrow \emptyset$
2: **for** $i = 1$ **to** $|\mathbb{G}|$ **do**
3: $V' \leftarrow V_i \in G_i$
4: **for all** $v \in V'$ **do**
5: $V \leftarrow V \cup \{v\}$
6: **end for**
7: **end for**
8: **return** V

5.2.3 Minimum Weighted Edge

In algorithm 1, the find and selection process of the edges to compose the final haplotype network is majority hidden under the function name $MinimumWeightedEdge$; its objective is finding, for an unordered pair of vertices, the edge connecting these vertices with most expressiveness of the characteristics desired in the final graph G — in this case, the edge with minimum weight associated to it.

For successful reach of this goal, the $MinimumWeightedEdge$ function must analyze every graph G_i belonging to $\mathbb{G}$, with $1 \leq i \leq |\mathbb{G}|$, and verify the existence of the edge $e_{v,v'} = \{v, v'\}$ against the set of edges $E_i \in G_i$ and then whether it is the minimum weighted edge among all existing edges $e_{v,v'}$ (in the graphs of $\mathbb{G}$).

Algorithmically, the definition of a M (for *minimum value*) variable, with infinity initial value, and an e (for *minimum edge*) variable, with empty value, easy the comparison among the many $e_{v,v'}$ since they keep information of the fittest found edge (under the minimum weight metric). At the end of the algorithmic process, the e variable itself is returned to the main algorithm, adding the minimum weighted edge $e = e^i_{v,v'} = \min\left\{w\left(e^1_{v,v'}\right), w\left(e^2_{v,v'}\right), \ldots, w\left(e^N_{v,v'}\right)\right\}$ to the graph G.

Theoretically, however, $MinimumWeightedEdge$ represents more than a function for finding minimum weighted edges, but the kernel of the adaptive theory applied in the current research project: while the main algorithm (section 5.2.1) represents the insertion adaptive action and the vertices function (section 5.2.2) does not represent any adaptive action, the minimum weighted edge function represents the *search adaptive function* and the special designed *selection* and *first functions*, all supporting functions for the insertion adaptive action.

Since the theoretical and algorithmical aspects are all represented, it is

straightforward to conclude that the remotion adaptive action is not used in the context of the current work; this property, though, was expected since the algorithm has a synthesizing essence, aggregating desired properties from many graphs and with no need to modify existing data structures — even in a constructive way.

Algorithm 3 Function $MinimumWeightedEdge(v, v', \mathbb{G})$

Require: $\{v, v'\} \subseteq V$, with $V \in G$, and $\mathbb{G} = \{G_1, G_2, \ldots, G_N\}$
Ensure: Find the edge $e^i_{v,v'} = \{v, v'\}$ belonging to $G_i \in \mathbb{G}$ such that $\min\left\{w\left(e^1_{v,v'}\right), w\left(e^2_{v,v'}\right), \ldots, w\left(e^N_{v,v'}\right)\right\} = w\left(e^i_{v,v'}\right)$

```
1: e ← ∅
2: M ← ∞
3: for i = 1 to |𝔾| do
4:    E' ← E_i ∈ G_i
5:    if e_i = {v, v'} ∈ E' then
6:       if w(e_i) < M then
7:          M ← w(e_i)
8:          e ← e_i
9:       end if
10:   end if
11: end for
12: return e
```

5.3 Applied Adaptive Theory

The development of the algorithm, although presented previously in the text, is thought in parallel to and immediately derives from a formal-theoretical framework. The requirements, same for both, are cleared defined and specified because of the tough constraints imposed by the latter.

In this sense, the adaptive algebra (and the adaptive theory) imposes a series of (strong) restrictions to a free (and unstructured) algorithm development, demanding an algorithmic design strongly tighted with the theoretical fundamentals and, hence, generalist and still mathematically consistent.

Reminding the concepts on adaptive automata theory from chapter 3 and the

adaptive algebra depicted in section 3.1.2.2, the application of these are designed for automata structures only, demanding the definition of an equivalence between graphs and (finite state) automata. Theorem 1 deals with the graph-automaton equivalence, while the automaton-graph one naturally rises from the fact that automata are graphically represented by directed graphs (MENEZES; HAEUSLER, 2008).

Theorem 1. *Let (G, w) be a directed weighted graph, according to definitions 11 and 12. There is a non-deterministic finite state automaton $A_G = (\Sigma, Q, F, q_0, \partial)$ that is visually represented by (G, w) and is considered equivalent to the latter.*

Proof Sketch.

Graph Finiteness A graph (G, w) which seeks an equivalent non-deterministic finite automaton A_G must have its set of vertices V and, hence, its set of edges E as finite ones since the compounding sets of A_G are all finite ones by definition.

Definition of the States Set In the directed weighted graph (G, w), the set of vertices V belonging to G can be directly mapped to the set of states Q in such that $\forall v_i \in V, \exists q_i \in Q$ such that $q_i = \tau_\circ(v_i)$, where $\tau_\circ : V \mapsto Q$ and implying that $i = |V| = |Q|$.

Setting Start State From the $\tau_\circ$-transformation, if V is an ordered set, $q_0 = \tau_\circ(v_0)$; else, $q_0 = \tau_\circ(v')$, with v' being any element of V.

Definition of the Final States Set F, the set of final states, is allowed to be void in a non-deterministic finite state automaton and defines a special group of states from this. If (G, w) do not have any special vertices that need to

be marked as final states in an automaton, $F = \emptyset$; else, $F \cap Q \neq \emptyset$ and every final state q_i belongs to F as well to Q.

Definition of the Alphabet The image of the weighting function w applied to the set of edges E, represented as $w[E]$, is a set containing (as elements) all the weights applied to each edge of E which, graphically, are commonly represented as the graph edges' labels. Since $w[E]$ is a finite set (because E is a finite set) containing symbols (weights) related to edges (or relations among the vertices) of the graph (G, w), this set is considered equivalent to the alphabet Σ of the A_G.

Definition of the Transition Function Let $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$. A transition function ∂ of a non-deterministic finite state automaton A_G is defined as $\partial : Q \times \Sigma_\epsilon \mapsto 2^Q$. Once $V \mapsto Q$ and $w[E] \mapsto E$, the transition function ∂ of A_G is defined based on composing sets of the graph (G, w).

□

Since an equivalence is established between graph and automaton, a proper adaptive theory for graphs is developed using (through some specializations for the context) the adaptive algebra presented in section 3.1.2.2. Under this algebraic structure, there are *external elements* supportive for the algebra operations and the *internal elements* which define de algebra itself and possessing, solely, the ability to change the underlying device.

Among the external elements, the *search function* is of central interest since it selects desired relations[1] through pattern matching which are future used for the other external elements, as the internal ones to compose the final result.

[1]When the subject is of an automaton type, these relations are the ones from the transition function. However, in the context of the present work, it is an edge and its weight value.

The proper-search function proposed in definition 4 is widely defined concerning its parameters for sake of generality of adaptive automata theory. For the present context (using graphs), a proper-search particularization S (definition 18) over the tuple *(vertex v', vertex v'', graph G')* returns the existing edge, in the graph G', connecting the vertices v' and v''.

Definition 18. *Let the* search function *S, defined as $S : V \times V \times \mathbb{G} \mapsto E$, be a particularization of the* proper-search operation *(definition 4).*

However, the present analysis occurs in a space of graphs, and a search in an unique graph does not represents the entire space. Therefore a new, and higher level, function $S_{\mathbb{G}}$ is developed and defined in definition 19, which seeks for and retrieves all the edges connecting two vertices v' and v'' belonging to the graphs of the graph-space in study. This way, through the application of $S_{\mathbb{G}}$, every edge $\{v', v''\}$ in this space, possessing different attributes or not, is gathered in a specific set for future evaluation against their properties.

It is important to note the explicit operation over the graph-space[2] $\mathbb{G}$, made clear through the necessity of the definition of the space as a parameter for the function $S_{\mathbb{G}}$ (in the place of the specific graph G') and the set union operation over the elements of this space.

Definition 19. *Let $S_{\mathbb{G}}$ be a* general search function*, functionally defined as $S_{\mathbb{G}} : V \times V \times 2^{\mathbb{G}} \mapsto E$, represents $S_{\mathbb{G}}(v, v', \mathbb{G}) = \bigcup_{1}^{|\mathbb{G}|} S(v, v', G_i) = S(v, v', G_1) \cup S(v, v', G_2) \cup \cdots \cup S(v, v', G_{N-1}) \cup S(v, v', G_N)$.*

Possessing the set built of all the existing edges $\{v', v''\}$ in $\mathbb{G}$, with v' and v'' representing two distinct parametrized vertices, it is possible to seek for those edges that highlight in the set because of its particular properties. Nonetheless the

[2]Formally, $\mathbb{G}$ is a set of graphs, with no particularization. In the present context of this work, $\mathbb{G}$ is the set of all graphs G_i sharing the same set of vertices V, a set of haplotypes.

definition of the interesting properties is completely connected to the definition of a *selection function* ψ which, as suggested by its name, *selects* exclusively the elements with desired properties. In the present context, as shown in definition 20, the elements in analysis are each of the individual edges $\{v', v''\}$ belonging to the set built from the application of $S_{\mathbb{G}}(v, v', \mathbb{G})$ and the property desired is the minimum edge weight value $min\left\{w\left(e^i_{v,v'}\right)\right\}$, $\forall e^i_{v,v'}$ belonging to a G_i in the graph-space $\mathbb{G}$ and $i = |\mathbb{G}|$.

Definition 20. *Let* ψ *be a* selection function *defined as*

$$\psi\left(e^k_{v,v'}\right) = \begin{cases} e^k_{v,v'} & , \textit{ if } w\left(e^k_{v,v'}\right) = min\left\{w\left(e^i_{v,v'}\right)\right\} \\ \emptyset & , \textit{ otherwise} \end{cases}$$

Following a generalization procedure similarly to the ones on definitions 18 and 19, a *general selection function* Ψ (definition 21) is developed as an application of the selection function ψ over all the individual elements of the set built from $S_{\mathbb{G}}$, composing a new set of edges $\Psi(S_{\mathbb{G}}(v', v'', \mathbb{G})) \subseteq S_{\mathbb{G}}(v', v'', \mathbb{G})$ with all of its elements owning the desired property stated in definition 20.

Definition 21. *Let* Ψ *be a* general selection function *defined as* $\Psi\left(\left\{e^1_{v,v'}, e^2_{v,v'}, \ldots, e^{N-1}_{v,v'}, e^N_{v,v'}\right\}\right) = \psi\left(e^1_{v,v'}\right) \cup \psi\left(e^2_{v,v'}\right) \cup \cdots \cup \psi\left(e^{N-1}_{v,v'}\right) \cup \psi\left(e^N_{v,v'}\right)$.

The successive application of the previously defined functions through the mathematical device called function composition established a subset of all edges $\{v', v''\}$ belonging to the graphs of $\mathbb{G}$, all of them sharing the same minimum weight edge value, the desired property — under the Occam's razor perspective — to compose the best representative as a final graph G.

Nonetheless, only one of these edges is necessary to be allocated in the set of edges of the final graph; since all of them are equivalent, the one with lower

index in the indexed and ordered set produced by Ψ function is taken through the usage of the $\mathcal{F}$ function defined below (definition 22).

It is important to observe that the first element of the Ψ-set is taken for algorithmical simplicity reasons: since the elements are equivalent under the constraints imposed by selection functions S and ψ, any of them could perfectly fit the objective of composing G.

Definition 22. *Let $\mathcal{F}$ (named after* first*) and $\mathcal{R}$ (*rest*) be functions defined, for $w = \{a\} \cup w'$ and $w' = \{b, c, \ldots\}$, as*

$$\mathcal{F} : 2^{V \times V} \mapsto V \times V \quad | \quad \mathcal{F}(w) = a$$
$$\mathcal{R} : 2^{V \times V} \mapsto 2^{V \times V} \quad | \quad \mathcal{R}(w) = w'$$

Finally, after, the selection of the best edge candidate, the application of the foreign-transition addition operation f^+ over the edge and the final graph G characterizes the *edge insertion adaptive operation*, adding the edge to the set E of G and keeping it associated to its edge weight value, $min\left\{w\left(e^i_{v,v'}\right)\right\}$.

Although not cited in the previous passages, when no edge is found by the selection functions, they return an empty value (empty set) and, hence, no edge is added to G.

Definition 23. *The* edge insertion adaptive operation*, composed by a composition of functions (definitions 19, 21 and 22) and adaptive operations (definition 9), is defined as*

$$f^+\left(\mathcal{F}\left(\Psi\left(S_{\mathbb{G}}\left(v, v', \mathbb{G}\right)\right)\right), G\right)$$

where G = (V, E) is the general output graph having edges added to.

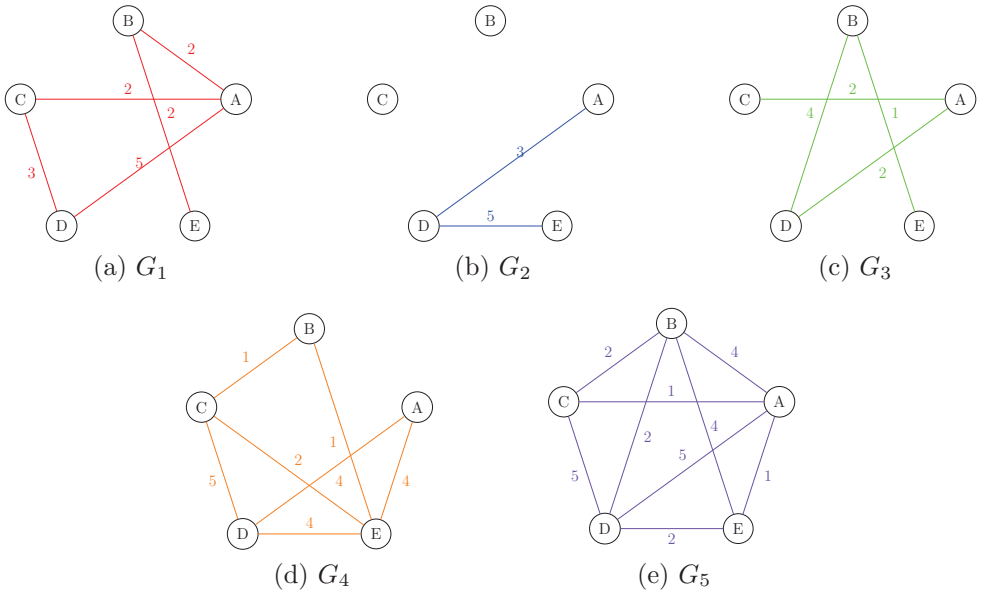
5.4 Final Graph G

The final graph G owns the most expressive properties of each graph (belonging to the universe $\mathbb{G}$) that compose it under the restrictions imposed by selection function ψ (definition 20). Hence, it absorbs, for every pair of vertices, the most relevant edge with all its associated properties, like the weight of the vertices. Since it analyzes every existing vertex in the universe of $\mathbb{G}$ and $V \times V$ space, G is potentially a complete graph (definition 13) once it may aggregate every edge $\{v', v''\}$ with $v' \neq v''$ whether all of them exists in $\mathbb{G}$. (The $v' \neq v''$ imposition exists because, in haplotype networks context, a hypothetical event — as a mutation — that transforms a haplotype v' to the same haplotype v' do not bring much information about the overall evolution of the species under study. Therefore, this information is not taken in account when studying haplotype networks.)

The presentation of the final G as a graph with cycles (definition 16) provides a wide range of information, including central haplotypes (BONDY; MURTY, 2008), most connected ones (see definition 15) and many evolutionary paths. Nonetheless, an unique evolutionary path is reasonable (and common) for phylogeographic and phylogenetics studies, investigating the reasons and events related to the species under the selected and solely evolutionary path context; this path, though, is represented by a tree (definition 17) which can be extracted from a cycled graph through the usage of a series of algorithms, such as Prim's algorithm (BONDY; MURTY, 2008). However, it is important to note that better algorithms for tree extraction from G can be developed, exploring the ϕ-selected properties for the best path choice.

5.5 Example

Let a graph space $\mathbb{G} = \{G_1, G_2, G_3, G_4, G_5\}$ be constituted of five graphs, all of them composed by the set $V_{\mathbb{G}} = V_i = \{A, B, C, D, E\}$, for $1 \leq i \leq |\mathbb{G}| = 5$, of five vertices and represented by figure 19.



(a) G_1 (b) G_2 (c) G_3

(d) G_4 (e) G_5

Figure 19: Graphs belonging to the space $\mathbb{G}$ of graphs.

For the composition of a final graph G under the characteristics of *Epicurus' principle* and *Occam's razor* and also keeping the main properties of the graphs members of $\mathbb{G}$, it will be gradually constructed following the algorithms presented in section 5.2. For sake of space, repetitions on the algorithmic steps will be avoided without loss of comprehension of the example execution.

Composition of sets V and E The first line of algorithm 1 is the attribution of the sets of vertices, V, of G through the functional call of $Vertices(\mathbb{G})$ (algorithm 2), which initially defines $V = \emptyset$. For $i = 1$, V_1 is mapped to V' and, in lines 4–6, every vertex belonging to $V' = V_1$ is added to V through the *set union operation* (line 5); this way, every vertex is added only one time even if it is presented in other sets of vertices (which, in fact, is true for every vertex of V_1

since $V_1 = V_2 = V_3 = V_4 = V_5$). Finally, the composed $V = V_1 = \ldots = V_5 = V_{\mathbb{G}}$ returns (line 8 of algorithm 2) to the main algorithm, proceeding the operation with $E = \emptyset$.

Covering all possible edges An extensive search of all possible edges in the graphs of $\mathbb{G}$ is made, seeking the edges with the desired property of minimum weight edge. Since the graphs are undirected, the number of possible ones to analyze is

$$\frac{(|V|)!}{2! \times (|V| - 2)!} = \frac{|V| \times (|V| - 1)}{2} \tag{5.1}$$

which, in the present example, is

$$\begin{aligned} \frac{5!}{2! \times (5-2)!} &= \frac{5 \times 4 \times 3!}{2! \times 3!} \\ &= \frac{5 \times 4}{2} \\ &= \frac{20}{2} \\ &= 10 \end{aligned}$$

In algorithm 1, the combination of the vertices in pairs to compose the possible edges to analyze is represented in the the loops and their variable attributions described in lines 3 to 6. In particular, line 5 guarantees that every edge is taken only one time.

In the example, the algorithm searches for the edges belonging to the following set E_{search}:

$$\begin{aligned} E_{search} = \{ & \{A,B\}, \{A,C\}, \{A,D\}, \{A,E\}, \\ & \{B,C\}, \{B,D\}, \{B,E\}, \\ & \{C,D\}, \{C,E\}, \\ & \{D,E\} \} \end{aligned}$$

Finding a minimum weighted edge The responsibility for finding the edge $\{v, v'\}$ (a generic representation of the edges belonging to E_{search}) with minimum weight value is attributed, by the main algorithm (line 7), to the function $MinimumWeightedEdge(v, v', \mathbb{G})$. This function, defined in algorithm 3, uses two auxiliary variables, e and M, to structure and correctly search for the minimum weighted edge; the former variable represents the minimum weighted edge found so far, while the latter represents the value $w(e)$ (for that edge). For this reason (lines 1 and 2), these variables are initialized with values ($e \leftarrow \emptyset$ and $M \leftarrow \infty$) that represent the absence of found edges under the desired property.

In the loop of line 3, the set of edges of the i^{th} graph of $\mathbb{G}$ is mapped to E' (*i.e.* $E' \leftarrow E_1$, $E' \leftarrow E_2$, etc...) and the existence of the edge $\{v, v'\}$ is verified in E'. If it is positive (line 5) and its weight is less than the registered up to the moment (line 6), the present edge is considered the minimum weighted edge (lines 7 and 8) until the exhaustive search finishes or a new edge with smaller $w(e_i)$ is found. Finally, the minimum weighted edge found, e, is returned to the main function.

Table 2 summarizes, as simulation tables, the internal states of $MinimumWeightedEdge$ when executed for the first for the vertices $(\{A, B\}, \{A, C\}, \{A, D\}, \{A, E\})$ in graph space $\mathbb{G}$, exhibiting the selected minimum weighted edge at each execution.

Composing the final graph When an edge with minimum value of $w(e)$ is found (line 7 of algorithm 1), its addition to the set E of edges (line 9) is, at this point of the algorithmic process, the solely operation needed to compose the final graph $G = (V, E)$ once $\forall\{v', v''\} \in E_i \implies \{v', v''\} \subseteq V_i$ and $\forall V_i, V_i \subseteq V \mid G_i = (V_i, E_i) \wedge G_i \in \mathbb{G}$. When this repetitive operation ends at line 10, the final graph is presented complete as represented by figure 20. At figure 20b, the edges of

v	v'	$\|G\|$	i	E'	$w(e_i)$	M	e
A	B	5				∞	$\emptyset$
			1	G_1	2	2	e_1
			2	G_2			
			3	G_3			
			4	G_4			
			5	G_5	4		
returns $e = e_1$ with $w(e) = 2$							

(a) $MinimumWeightedEdge(A, B, \mathbb{G})$

v	v'	$\|G\|$	i	E'	$w(e_i)$	M	e
A	C	5				∞	$\emptyset$
			1	G_1	2	2	e_1
			2	G_2			
			3	G_3	2		
			4	G_4			
			5	G_5	1	1	e_5
returns $e = e_5$ with $w(e) = 1$							

(b) $MinimumWeightedEdge(A, C, \mathbb{G})$

v	v'	$\|G\|$	i	E'	$w(e_i)$	M	e
A	D	5				∞	$\emptyset$
			1	G_1	5	5	e_1
			2	G_2	3	3	e_2
			3	G_3	2	2	e_3
			4	G_4	4		
			5	G_5	5		
returns $e = e_3$ with $w(e) = 2$							

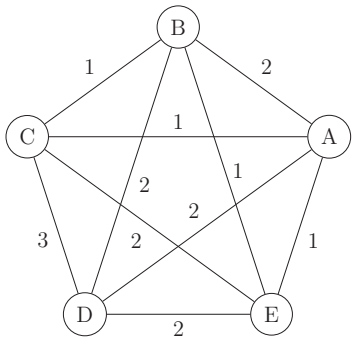
(c) $MinimumWeightedEdge(A, D, \mathbb{G})$

v	v'	$\|G\|$	i	E'	$w(e_i)$	M	e
A	E	5				∞	$\emptyset$
			1	G_1			
			2	G_2			
			3	G_3			
			4	G_4	4	4	e_4
			5	G_5	1	1	e_5
returns $e = e_5$ with $w(e) = 1$							

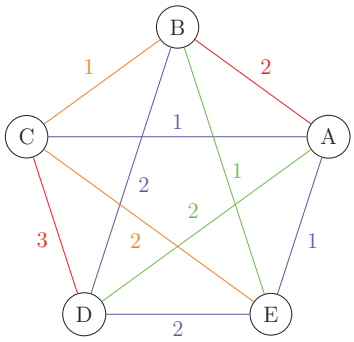
(d) $MinimumWeightedEdge(A, E, \mathbb{G})$

Table 2: Simulation of the function $MinimumWeightedEdge(v, v', \mathbb{G})$ for the first four edges.

G are colored according to the graphs of figure 19, emphasizing the origins of each edge, as well as the unification and best features selection properties of the proposed algorithm.



(a) G



(b) G with colored edges

Figure 20: Final graph G in normal representation (20a) and with edges colored representing the original graphs they come from (20b).

6 DATA ANALYSIS AND RESULTS

The present work has been exploring theoretical aspects of biology, mathematics and computer science for the development of a still theoretical abstraction that unifies an arbitrary range of hypothesis through the usage of user-defined selection function — which, in general, is a *minimum weight function* $\min\{w_1, w_2, \ldots\}$. And although strongly based on well-established theories and concepts from the aforementioned study fields, it is important to stress the proposed framework over a series of controlled tests and examples of its application in an attempt to expose its strengths over the existing methods, diagnose critical points for improvements as specific behavioral patterns.

Concerning the last two hypothesis (stress tests and critical point analysis), they are not conceived in the project and development of the scientific work, opposed to the former one; nonetheless, product of the insuccess (under certain metrics) of the first versions of the model tested against a set of known data, they allow the refinement of the proposed theory, what generally represents its improvement, presenting neglected aspects in the initial analysis or positive or negative side effects from the design choice. Therefore, the application of a failure analysis based on extensive set of tests is, opposed to the first and naive idea, critical to the success of the proposed solution.

For the exhaustive execution of the tests, nonetheless, a reasonable computational effort is necessary once it is not possible to demonstrate the algorithm

effectiveness in a simple and straight way, but solely through theorems demonstrations.

6.1 Developed Tools

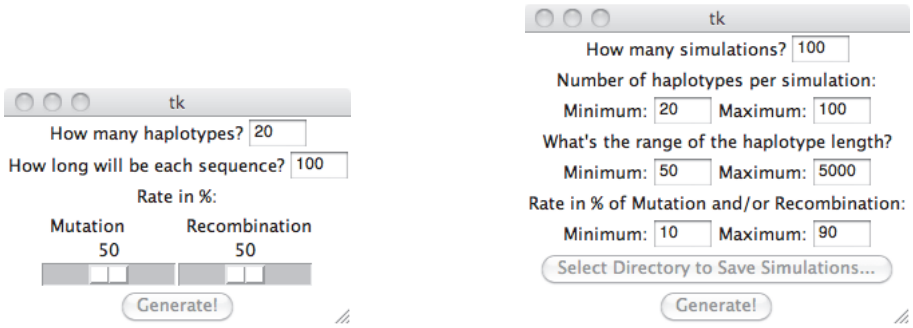
Under this perspective, a set of applications were developed with the unique intention to validate the proposed method in the present work. A total of two software were built covering the activities of artificial data generation for testing and analysis of the results produced by the data processing through the application of the proposed algorithm.

6.1.1 Data Generation Tool

Since the confrontation of the existing algorithm against the ones already used by the scientific community, under the *real data evolved naturally*, is inconclusive because of the lack of knowledge of the *real behavior* of Nature and its relationship with the genetic (and environment) data. For this reason, artificial data are generated under certain known evolutionary models, which can be used to compare the expected results against the models used to guide the generation of the data (KNOWLES, 2008).

For this purpose, a *data generation tool* was developed, assembling DNA strands in a random way, but observing some pre-defined rules which can be preconfigured by the user (figure 21a), interactively, or arbitrarily specified by the software (figure 21b) seeking to build a valid statistical corpus of artificial DNA data.

There are two common functions for building the sets of haplotypes, one being responsible for creating the group of haplotypes and the other for assembling the DNA code of a specific haplotype.



(a) Interactive version (b) Corpus version

Figure 21: Graphical User Interface for both versions of the data generation tool.

The general idea of these functions is:

Group of Haplotypes Function

1. generates an *original haplotype* (*i.e.*, the first randomly generated haplotype by the software);
2. creates empty lists of *mutated and recombined haplotypes*, with each list length related to the percentage ratio of mutation and recombination defined in the graphical user interface (GUI);
3. if the list of mutated haplotypes is not full, generates a *mutated haplotype*;
4. if the list of recombined haplotypes is not full, generates a *recombined haplotype*;
5. iterates through the steps 3 and 4 until both lists get full.

Single Haplotype Generator Function

1. verifies whether the requested haplotype to generate is an *original*, *mutated* or *recombined haplotype*;
2. if an *original haplotype*, then generates an haplotype through the random sampling of DNA nucleobases;

3. else if an *mutated haplotype*, define the number of mutation randomly but under the exponential distribution (ROSCHE; FOSTER, 2000) and change the nucleobases arbitrarily (related to its position in the haplotype sequence);

4. else, a *recombined haplotype* is requested and one haplotype is chosen, randomly, to be copied in a new haplotype which will become the *recombined* one. Other haplotype is selected at chance and a continuous segment of nucleobases (said *i*-*j* segment, as a reference to the indexes where the segment starts and ends) is replicated from this haplotype to the copied one, exactly in the same indexes (i and j) delimiting the recombined segment, forming a new and *recombined haplotype*.

As the result of these processing activities, the *data generation tool* outputs a NeXus file (MADDISON; SWOFFORD; MADDISON, 1997) with a list of named haplotypes under the following pattern in Backus-Naur Form (BNF)(BACKUS et al., 1964):

```
<haplotype name>           ::=  original
                                | <modified haplotype kind><index>
<modified haplotype kind>  ::=  mutated | recombined
<index>                    ::=  <number>
<number>                   ::=  <non-zero algarisms>
                                | <non-zero algarisms><number suffix>
<number suffix>            ::=  <algarisms>
                                | <algarisms><number suffix>
<algarisms>                ::=  0 | <non-zero algarisms>
<non-zero algarisms>       ::=  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Table 3: BNF representation of the *simulated haplotype name* grammar

where `<haplotype kind>` may be substituted by *original*, *mutated* or *recombined* and `<index>` is the numerical representation of the index of the haplotype in the generation process.

6.1.2 Data Analysis Tool

The usage of the *data generation tool* (described in section 6.1.1) for statistical reasoning demands (and produces) a great volume of *raw data* in a special file format called NeXus (MADDISON; SWOFFORD; MADDISON, 1997); each of these files, in turn, contains a considerable number of haplotypes composed of a long sequence of nucleobases, resulting in an intractable corpus of raw data for human analysis. Since the application of the present work (as detailed in chapter 5) produces three times more data (because of the tests under *mutation*, *recombination* and the *proposed technique*), the task becomes utterly time-consumable even for entire laboratory staff. Therefore, a software tool was developed, seeking for the automation of the repetitive task of data analysis.

The information that can be retrieved from an automatic analysis, though, is limited to numerical one because of the inability of digital systems to deal with other kinds of data; interpretation on visual data, *e.g.* spatial structure of the graphs, are inferred from relations among the quantitative data extracted autonomously.

Thus, two distinct sets of properties may be defined, one concerning the fundamental parts composing a graph ($\mathcal{F}$ set) and other related to the relation among the diverse number of haplotype modification ($\mathcal{R}$ set). In the present software developed, $\mathcal{F}$ is composed of the *nodes*, *edges* and *edges' weight* and $\mathcal{R}$ of *mutation*, *recombination*, *final*, which represented the final graph after the application of the techniques proposed in this work, as well as all the combinations of the previous elements, *i.e. mutation* $\wedge$ *recombination*, *mutation* $\wedge$ *final*, *recombination* $\wedge$ *final* and *mutation* $\wedge$ *recombination* $\wedge$ *final*.[1] Then, a complete relation $\mathcal{P} = \mathcal{F} \times \mathcal{R}$ may be defined, creating properties that match graph

[1] The $\alpha \wedge \beta$ notation, in this context, has the interpretation of "α *and* β *has the same value*" for some property.

"building blocks" against operations over haplotypes. For example, the element $(node, mutation \wedge recombination)$ of $\mathcal{P}$, which may be represented as $N(M = R)$, means "*the property which specific* ***nodes*** *are the same for the* ***mutation and recombination*** *graphs*".

Finally, 18 properties were defined for data analysis:

$$\begin{aligned}\boldsymbol{\mathcal{P}} = \{ \quad & N(M), N(R), N(F), \\ & N(M=R), N(M=F), N(R=F), N(M=R=F), \\ & E(M), E(R), E(F), \\ & E(M=R), E(M=F), E(R=F), E(M=R=F), \\ & W(M=R), W(M=F), W(R=F), W(M=R=F) \quad \}\end{aligned}$$

For the proper calculation of these properties, the required data ($\mathcal{F}$ and $\mathcal{R}$) are extracted from the output file of the main software — the one that processes the proposed method on the NeXus files; the total number of *nodes* and *edges*, as well as each *edge weight*, are parsed directly from the file, following the grammar represented at table 4; the intersection information among the graphs, *i.e.* the $\mathcal{R}$ set, are individually calculated using the file name suffix that indicates the used method for graph construction (table 5) and automatic comparison.

```
<file>          ::=  <nodes><edges><common edges>
<nodes>         ::=  Nodes:  <number><new line>
<edges>         ::=  Edges:  <number><new line>
<common edges>  ::=  (('<haplotype name>','<haplotype name>'),
                     <number>)<new line><next element>
<next element>  ::=  <common edges> | ϵ[2]
<new line>      ::=  CR | LF | CR LF [3]
```

Table 4: BNF representation of the *output file of the main software* grammar

The *data analysis tool* (figure 22) process the outputted files from a single NeXus or even from multiples ones, as in case of the multiple simulations; as result, it outputs a single *comma-separated values (CSV)* file (SHAFRANOVICH,

```
<file name>          ::=  <user input>_<processing method>.txt
<user input>         ::=  <printable> | <printable><user input>
<processing method>  ::=  mutation | recombination | final
<printable>          ::=  <letter> | <number> | <valid symbols> [4]
<valid symbols>      ::=  _ | - | .  | ( | )
```

Table 5: BNF representation of the *output file name* grammar

2005) with all properties $\mathcal{P}$ calculated for each of the several files selected (or even for the single one, if it is the case). The CSV file is not valid as a final analysis of the results, but enhances the understanding over the relationship among the diverse existing methods tested and directs deeper studies highlighting important variables, for example.

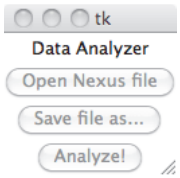


Figure 22: Graphical User Interface for the data analysis tool.

At last, the online repository (GUIRALDELLI, 2012) contains the source code of the *data analysis tool* (as well as the other software) for full understanding of the realized calculations performed by the software.

6.2 Results

With a real (CABANNE; SANTOS; MIYAKI, 2007)(CABANNE et al., 2008) haplotype information available, along with a hundred simulated ones, as well as all properties $\mathcal{P}$ accessible through the processed CSV file, a series of interesting relationship emerged from the existing data, either by the numerical properties or graphical ones.

[2] ϵ is used as a representation of the null character. (SIPSER, 2005)

[3] *Carriage return* (`CR`), *line feed* (`LF`) or a combination of both are standard ways to represented new line in a plain text file in the main operating systems.

[4] For sake of brevity, the definition of the non-terminal symbol `<letter>` was omitted without loss of comprehension.

6.2.1 Numerical Analysis

The numerical analysis on the processed CSV file, at first, reveals a strong relation between the number of edges of the *mutation* and *final graphs*, emphasized by the fact that these number are always the same; this natural relationship rises from the fact that a mutational connection can always be established between two haplotypes, even if they are completely different — what would define a mutational distance of the length of the haplotype sequence — and, then, a complete graph can be assembled (what, in fact, happens).

Conversely, the edges derived from recombination relationship predominate as the most used ones by the final graph, with all of them "copied" to the former because their (edges) weights are smaller than their mutated counterparts. The origin of the smaller weights resides in the fact that a recombination distance of 1 may be equivalent to a mutation distance of $1, 2, \ldots$ up to half of the haplotype length, *i.e.*, the probability of $n > 1$ modified nucleobases have mutational origin is smaller than the probability of recombinational origin, if the recombination and mutation rate are equal.

As a consequence, the *final (and complete) graph* $G = (V, E)$ has all the edges of the *recombination graph* $G_R = (V_R, E_R)$ and the remaining ones from the *mutational graph* $G_M = (V_M, E_M)$; in mathematical notation, $E - E_R \subseteq E_M$. For instance, in the studied corpus, an average of 73.2% of the elements in E belongs exclusively to E_R ($E_R - E_M$ set), in opposition to 15.6% exclusively to E_M ($E_M - E_R$ set); the remaining 11.2% of the edges belongs to the set $E_M \cap E_R$.

Owing to the *redefinition* of the edges' weight of recombination root, the structure and relationships of the graphs and its subgraphs are deeply altered; accordingly, a MST_G (*minimum spanning tree of the graph* G) derived from the complete final graph G is topologically different from one, namely MST_M, derived

from a complete mutational graph M used to compose G, due to the fact that the minimum spanning tree algorithm (CORMEN et al., 2010) assembles the tree based on the edges with minimum weights.

Finally, the effect of the false positives and negatives was diagnosticated, with a number of mutational edges recognized as recombinational ones. These side effects, yet, are diminished when the reference mutational rate is lower than 50%.

6.2.2 Graphical Analysis

The pure numerical analysis of graphs, although reveals interesting properties, if fails to clearly expose these facts. In this sense, plotting the graphs clarifies many of these "numerically uncovered properties", particularly the *topological* ones.

Under this perspective, an analysis of the final graph G along the mutational graph G_M does not aggregate any information, since both of them are complete graphs and their differences are promptly known, the set E_R of the recombination graph G_R. For this reason, the graphical analysis is densely benefited if applied over the *minimum spanning trees* MST_G (figure 23), MST_M (figure 24) and MST_R (figure 25).

As the following figures manifest, the spatial disposition of the elements are pretty different, even with the graphs composed exactly by the same nodes. As already stated in section 6.2.1, this property rises from the behavior of the *minimum spanning tree* algorithm, seeking for the least effort to travel between nodes.

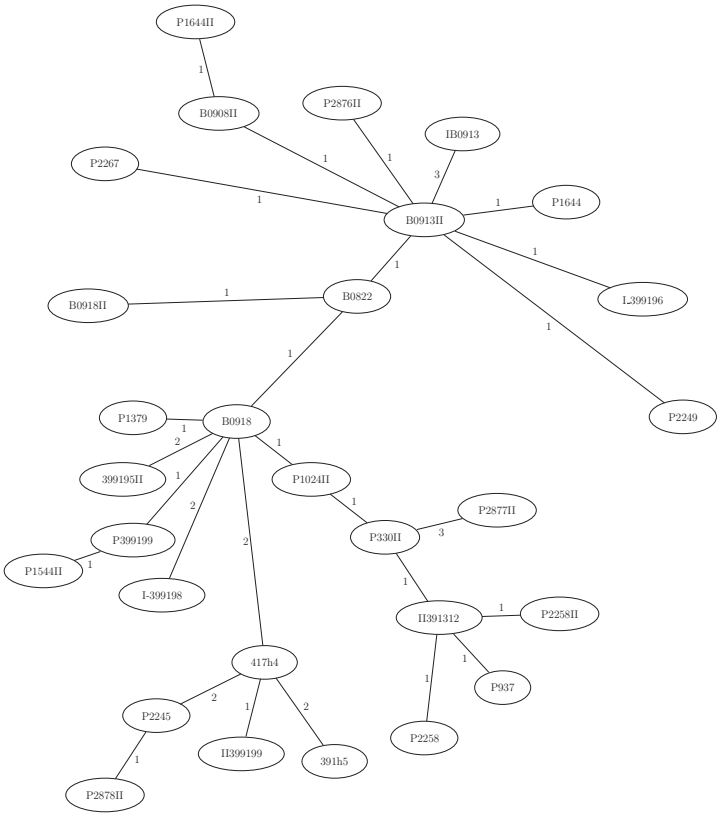


Figure 23: Minimum spanning tree extract from the final graph G of the real data from (CABANNE; SANTOS; MIYAKI, 2007; CABANNE et al., 2008).

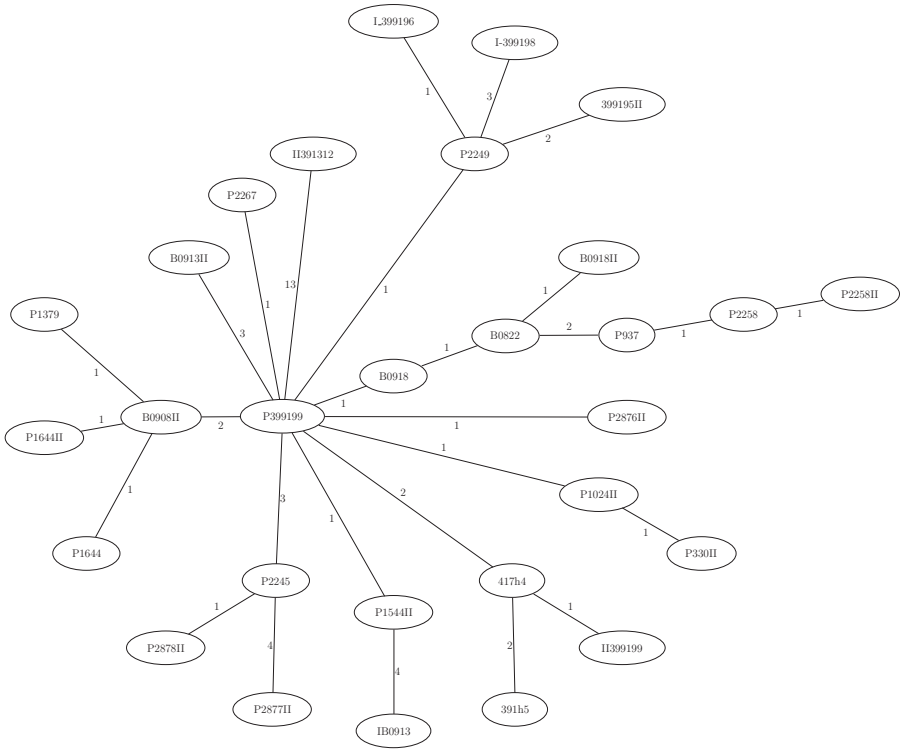


Figure 24: Minimum spanning tree extract from the mutation graph G_M of the real data from (CABANNE; SANTOS; MIYAKI, 2007; CABANNE et al., 2008).

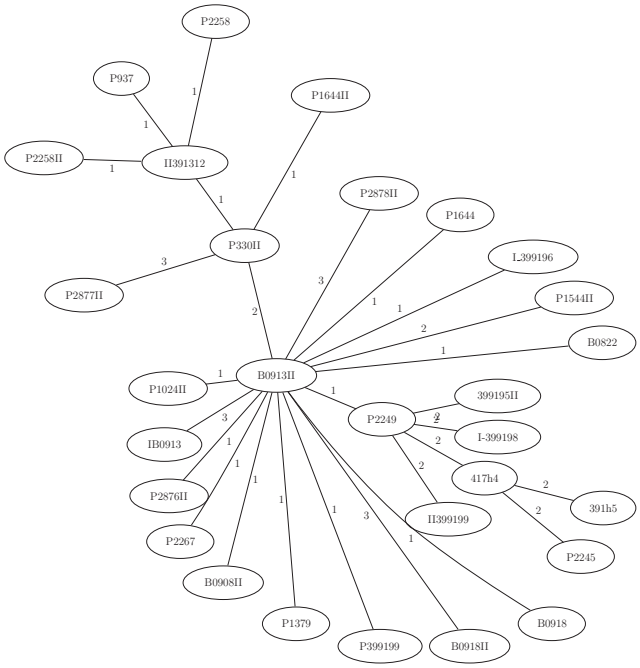


Figure 25: Minimum spanning tree extract from the recombination graph G_R of the real data from (CABANNE; SANTOS; MIYAKI, 2007; CABANNE et al., 2008).

7 CONCLUDING REMARKS

The usage of graphical notation for representing knowledge follows the scientific community for long time in a number of fields and, in Biology, it is not different. Nonetheless, not long past years (TEMPLETON; BOERWINKLE; SING, 1987) it has been used for phylogeographic studies and a definitive technique is far from established (BEAUMONT; PANCHAL, 2008; KNOWLES, 2008; PETIT, 2008b, 2008a; TEMPLETON, 2008), with many requirements still pending such as error rates (KNOWLES, 2008) and the deliverance of mutational only analysis (WOOLLEY; POSADA; CRANDALL, 2008).

Sharing the same concerns of many biologists, the present work extends the nowadays biological usage of graphical representation, attentive exclusively to the notational didactic and explanatory aspect, to the graph theory formalisms originated in the discrete mathematics field, presenting a clear connection (and unification) between these two fields through the loan of concepts from one to another (chapter 5).

There are some remarkable advantages in joining these subjects, specially the ones derived from exact sciences. As an almost three centuries old field, *graph theory* comprises knowledge and techniques for solving most of the usual (and even uncommon) barriers involving graphs, as algorithms for extracting the minimum trees (CORMEN et al., 2010), graph rewriting techniques (ROZENBERG; SALOMAA, 1997; ROZENBERG, 1997) and graph algebras (ROZENBERG; SALOMAA,

1997; ROZENBERG, 1997); furthermore, it is a still developing mathematical field with recent important discoveries (BOLLOBAS, 1998).

The graph theoretical perspective brought to the haplotype network topic allows the application of algorithmically — and, hence, computationally — structured thoughts to the area, moving the graphical notation away from the biological concerns and bringing some abstractions on; as an example, dynamic adaptations of the graph topology as new data is collected or inputed in the system independently of a preconception model. In fact, based on the *graph rewriting* concepts, *set operations* on graphs (BOLLOBAS, 1998) and *adaptive automata* (FILHO; ROCHA, 2011; NETO, 1993b, 2001), the present work now allows the inclusion of many DNA modification (as *mutation*, *recombination* and *horizontal genetic transference* (GOLDENFELD; WOESE, 2007), as any other) in the haplotype analysis and, finally, adapts the final result to the **best one** (under a specific standpoint defined by a particular *selection function* as presented in definition 20) dynamically, compounding a network scenario hardly seen and functionally efficient thanks to the computation and comparisons performed.

7.1 Contributions

The above adaptive formalism applied to the proposed technique, explicitly because of the chosen notation ((FILHO; ROCHA, 2011) and section 3.1.2.2), provides an algorithmic framework in *functional programming style* (HUDAK, 1989) which enables easy implementations of a haplotype network analysis software, as well as their maintenance, since the compounding "movable parts" (functions $S_{\mathbb{G}}$ and Ψ in definition 23) may be substituted for ones that better suit the desired purposes.

Two other features also highlight as results of the current research: (i) new

haplotype network topologies; (ii) and new error rates information. The former characteristic emerges from the inheritance of the many graphs properties aggregated in the final graph and their processes for inference of the most realistic (and, as the final purpose, truly representative of the Nature's historical behavior) haplotype modification scattering; as straight benefit, the new topologies provides new insights — not seen before because of the limitation in the number of tested genetic code modifications hypothesis — over phylogenetic related researches since new connections are revealed.

The later one, in the same trend, revealed the importance and influence of choosing the multiple algorithms for raw data analysis instead of relying solely on the mutational one; the positively impressive statistics for low mutation simulations reveals the efficiency of the multiple algorithm, state kept (with quality decay) up to half-half mutation-recombination simulations. Above this threshold, the number of *false positives* (which, in this case, means the number of mutational haplotypes recognized as recombinational ones) grows considerably, as well as the error rates.

7.2 Future Work

This way, as a future strategy for improvements, other mathematical abstractions should also be tested, particularly as composition element of the *selection function* Ψ, as initially proposed and experienced in (GUIRALDELLI; ROCHA, 2011). Elements like conditional probabilities, mutation and recombination rates and some information theory, as entropy (COVER; THOMAS, 2006) and Kolmogorov complexity (LI; VITÁNYI, 1997; WALLACE, 2005), sound like reasonable initial attempts because of their similar usage, in other but similar situations, for concurrency resolution.

7.3 Conclusion

Finally, it is important to state the incipience of the subject and its growth and significant existing potential as genetic information is gaining central importance in a diverse number of studies, like biological, medical or even historical fields. Thus, the application of the presented technique may directly and immediately benefit some neglected areas like hospital, particularly infectology reasoning, as well as support other phylogenetics applications such as species distribution modeling.

BIBLIOGRAPHY

ANDERSON, S. et al. Sequence and organization of the human mitochondrial genome. *Nature*, v. 290, p. 457–465, 1981.

ANISIMOV, O.; FITZHARRIS, B. Climate change 2001: Impacts, adaptation and vulnerability. In: _____. [S.l.]: Intergovernmental Panel on Climate Change, 2011. cap. Polar Regions (Arctic and Antarctic), p. 801–841.

AVISE, J. C. *On Evolution*. [S.l.]: The Johns Hopkins University Press, 2007. 95–102 p.

AVISE, J. C. et al. Intraspecific phylogeography: the mitochondrial DNA bridge between population genetics and systematics. *Annual Review of Ecology and Systematics*, v. 18, p. 489–522, 1987.

BACKUS, J. et al. *Revised Report on the Algorithmic Language Algol 60*. [S.l.], 1964.

BEAUMONT, M.; PANCHAL, M. On the validity of nested clade phylogeographical analysis. *Molecular Ecology*, v. 17, n. 11, p. 2563–2565, June 2008.

BEAUMONT, M. A.; ZHANG, W.; BALDING, D. J. Approximate bayesian computation in population genetic. *Genetics*, v. 162, n. 4, p. 2025–2035, December 2002.

BOLLOBAS, B. *Modern Graph Theory*. [S.l.]: Springer, 1998. 408 p.

BONDY, A.; MURTY, U. *Graph Theory*. 3rd. ed. [S.l.]: Springer, 2008. 666 p. (Graduate Texts in Mathematics).

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *The Unified Modeling Language User Guide*. [S.l.]: Addison-Wesley Professional, 2005.

CABANNE, G. S. et al. Nuclear and mitochondrial phylogeography of the atlantic forest endemic *Xiphorhynchus fuscus* (aves: Dendrocolaptidae): Biogeography and systematics implications. *Molecular Phylogenetics and Evolution*, v. 49, p. 760–773, 2008.

CABANNE, G. S.; SANTOS, F. R.; MIYAKI, C. Y. Phylogeography of *Xiphorhynchus fuscus* (passeriformes, dendrocolaptidae): vicariance and recent demographic expansion in southern atlantic forest. *Biological Journal of the Linnean Society*, v. 91, p. 73–84, 2007.

CALLADINE, C. et al. *Understanding DNA — The Molecule and How It Works*. [S.l.]: Elsevier, 2004.

CANN, R. L.; STONEKING, M.; WILSON, A. C. Mitochondrial DNA and human evolution. *Nature*, v. 325, p. 31–36, January 1987.

CAO, L.; KENCHINGTONA, E.; ZOUROS, E. Differential segregation patterns of sperm mitochondria in embryos of the blue mussel (*Mytilus edulis*). *Genetics*, v. 166, p. 883–894, 2004. Disponível em: <http://www.genetics.org/cgi/content/full/166/2/883>.

CARNIELLI, W.; EPSTEIN, R. *Computabilidade, Funções Computáveis, Lógica e Fundamentos da Matemática*. [S.l.]: UNESP, 2009. 415 p.

CLEMENT, M.; POSADA, D.; CRANDALL, K. TCS: a computer program to estimate gene genealogies. *Molecular Ecology*, v. 9, n. 10, p. 1657–1660, 2000.

CORMEN, T. H. et al. *Introduction to Algorithms*. [S.l.]: PHI Learning, 2010. 1312 p.

COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. 2. ed. [S.l.]: John Wiley & Sons, 2006.

DAWKINS, R. *The Ancestor's Tale: A Pilgrimage to the Dawn of Life*. [S.l.]: Houghton Mifflin, 2004. 673 p.

DIESTEL, R. *Graph Theory*. 4th. ed. [S.l.]: Springer, 2010. (Graduate Texts in Mathematics).

EULER, L. P. Commentarii academiae scientiarum imperialis petropolitanae. In: _____. *Commentarii academiae scientiarum imperialis Petropolitanae*. [S.l.: s.n.], 1736. v. 8, cap. Solutio problematis ad geometriam situs pertinentis, p. 128–140.

FILHO, R. I. d. S.; ROCHA, R. L. d. A. d. International conference on adaptive and natural computing algorithms, ICANNGA 2011, Ljubljana, Slovenia, april 14-16, 2011, revised papers. In: _____. [S.l.]: Springer-Verlag, 2011. (Lecture Notes in Computer Science, to be published), cap. Adaptive Finite Automaton: a New Algebraic Approach, p. 1–10.

FORTNOW, L. Kolmogorov complexity. In: DOWNEY, R.; HIRSCHFELD, D. (Ed.). *Aspects of Complexity, Minicourses in Algorithmics, Complexity, and Computational Algebra*. [S.l.]: de Gruyter, 2001. cap. Kolmogorov Complexity, p. 73–86.

FREELAND, J. R. *Molecular Ecology*. [S.l.]: John Wiley & Sons, 2005. 155–199 p.

GOLDENFELD, N.; WOESE, C. Biology's next revolution. *Nature*, v. 445, n. 7126, p. 369–369, January 2007. ISSN 0028-0836. Disponível em: <http://dx.doi.org/10.1038/445369a>.

GRAHAM, R. L.; KNUTH, D. E.; PATASHNIK, O. *Concrete Mathematics: A Foundation for Computer Science*. [S.l.]: Addison-Wesley Professional, 1994. 672 p.

GRÜNWALD, P.; VITÁNYI, P. Shannon information and Kolmogorov complexity. *IEEE Transactions on Information Theory*, July 2010. Submitted. Disponível em: <http://www.cwi.nl/ paulv/papers/info.pdf>.

GUIRALDELLI, R. H. G. *MSc softwares source code in Python*. February 2012. Disponível em: <https://github.com/guiraldelli/MSc>.

GUIRALDELLI, R. H. G.; ROCHA, R. L. de Azevedo da. Adaptive unification of graphs applied to haplotype networks. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, v. 9, n. 6, p. 950–955, October 2011.

GUSFIELD, D. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. [S.l.]: Cambridge University Press, 1997.

HAWKING, S. *God Created the Integers*. [S.l.]: Running Press, 2005.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. *Introdução à Teoria de Autômatos, Linguagens e Computação*. [S.l.]: Elsevier, 2003.

HUDAK, P. Conception, evolution, and application of functional programming languages. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 21, p. 359–411, September 1989. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/72551.72554>.

JOLY, S.; STEVENS, M. I.; VUUREN, B. J. van. Haplotype networks can be misleading in the presence of missing data. *Systematic Biology*, v. 56, n. 5, p. 857–862, 2007. Disponível em: <http://sysbio.oxfordjournals.org/content/56/5/857.short>.

JUNQUEIRA, L. C.; CARNEIRO, J. *Histologia Básica*. [S.l.]: Guanabara Koogan, 2008.

KAUFFMAN, S. A. *The Origin of Order: Self-Organization and Selection in Evolution*. [S.l.]: Oxford University Press, 1993.

KNOWLES, L. L. Statistical phylogeography. *Molecular Ecology*, v. 11, p. 2623–2635, 2002.

_____. The burgeoning field of statistical phylogeography. *Journal of Evolutionary Biology*, v. 17, p. 1–10, 2003.

_____. Why does a method that fails continue to be used? *Evolution*, v. 62, n. 11, p. 2713 to 2717, November 2008.

LABARRE, A. *Minimum Common Supergraphs and Haplotype Networks*. [S.l.], 2007.

LEWIS, H.; PAPADIMITRIOU, C. *Elements of the Theory of Computation*. [S.l.]: Prentice-Hall, 1997.

LI, M.; VITÁNYI, P. *An introduction to Kolmogorov complexity and its applications*. Second. [S.l.: s.n.], 1997.

LORENZ, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.*, American Meteorological Society, v. 20, n. 2, p. 130–141, mar. 1963. ISSN 0022-4928. Disponível em: <http://dx.doi.org/10.1175/1520-0469(1963)020¡0130:DNF¿2.0.CO;2>.

MADDISON, D. R.; SWOFFORD, D. L.; MADDISON, W. P. NeXus: An extensible file format for systematic information. *Systematic Biology*, v. 46, n. 4, p. 590–621, 1997. Disponível em: <http://sysbio.oxfordjournals.org/content/46/4/590.abstract>.

MANOLOPOULOU, I. *A Bayesian approach to Nested Clade Analysis*. Tese (Doctor of Philosophy) — University of Cambridge, September 2008.

MENEZES, P. B.; HAEUSLER, E. H. *Teoria das Categorias para Ciência da Computação*. [S.l.]: Bookman, 2008.

MOGENSEN, H. L. The hows and whys of cytoplasmic inheritance in seed plants. *American Journal of Botany*, v. 83, n. 3, p. 383–404, 1996.

NANNEN, V. *A Short Introduction to Kolmogorov Complexity*. [S.l.], April 2003. Disponível em: <http://volker.nannen.com/work/mdl/>.

NETO, J. J. *Contribuições à metodologia de construção de compiladores*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 1993.

NETO, J. J. *Contributions to Compiler Design Methology*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 1993. In Portuguese.

NETO, J. J. Adaptive rule-driven devices — general formulation and case study. In: *Pre-Proceedings of the CIAA'2001 Sixth International Conference on Implementation and Application of Automata*. [S.l.: s.n.], 2001. p. 158–176.

PANCHAL, M. *ANeCA: User Guide*. 1.2. ed. [S.l.], July 2008.

PEREIRA, L. d. V. *Seqüênciaram o Genoma Humano... E Agora?* [S.l.]: Editora Moderna, 2005.

PETIT, R. J. The coup de grâce for the nested clade phylogeographic analysis? *Molecular Ecology*, v. 17, p. 516–518, 2008.

_____. On the falsifiability of the nested clade phylogeographic analysis method. *Molecular Ecology*, v. 17, n. 6, p. 1404, 2008.

PIERCE, B. C. *Basic Category Theory for Computer Scientists*. [S.l.]: MIT Press, 1991.

POSADA, D.; CRANDALL, K. A.; TEMPLETON, A. R. Geodis: A program for the cladistic nested analysis of the geographical distribution of genetic haplotypes. *Molecular Ecology*, v. 9, n. 4, p. 487–488, 2000.

_____. Nested clade analysis statistics. *Molecular Ecology Notes*, v. 6, p. 590–593, 2006.

ROCHA, R. L. A. The $\mathcal{A}$-FA, its properties, and a Walk through a FA-Space. 2008.

_____. *Adaptatividade: Um método de escolha automática de soluçoes*. [S.l.]: Blucher, 2011. 192 p.

ROCHA, R. L. d. A. d. An Attempt to Express the Semantics of the Adaptive Devices. *Advances in Technological Applications of Logical and Intelligent Systems - Selected Papers from the Sixth Congress on Logic Applied to Technology*, IOS Press, Lansdale, PA, v. 186, p. 13–27, 2009.

ROCHA, R. L. d. A. d.; NETO, J. a. J. Adaptive automaton, limits and complexity compared to the Turing machine - in Portuguese Autômato Adaptativo, Limites e Complexidade em Comparação com Máquina de Turing. In: FACULDADE SENAC DE CIêNCIAS EXATAS E TECNOLOGIA. *Proceedings of the I Congress of Logic Applied to Technology - LAPTEC 2000*. São Paulo, 2000. p. 33–48.

_____. An Adaptive Finite-State Automata Application to the Problem of Reducing the Number of States in Approximate String Matching. In: *XI Congreso Argentino de Ciencias de la Computación - CACIC 2005*. Concordia, Entre Ríos, Argentina: [s.n.], 2005. v. 1, n. 1, p. 17–21.

ROSCHE, W. A.; FOSTER, P. L. Determining mutation rates in bacterial populations. *Methods*, v. 20, n. 1, p. 4 – 17, 2000. ISSN 1046-2023. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1046202399909015>.

ROZENBERG, G. (Ed.). *Handbook of Graph Grammars and Computing by Graph Transformation*. [S.l.]: World Scientific Pub Co Inc, 1997.

ROZENBERG, G.; SALOMAA, A. (Ed.). *Handbook of Formal Languages*. [S.l.]: Springer, 1997.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence – A Modern Approach*. 2. ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2003.

SHAFRANOVICH, Y. *RFC 4180 – Common Format and MIME Type for Comma-Separated Values (CSV) Files*. October 2005. Disponível em: <http://tools.ietf.org/html/rfc4180>.

SHUTT, J. N. Self-modifying finite automata: Power and limitations. *Technical Report*, Worcester Polytechnic Institute, n. WPI-CS-TR-95-4, 1995.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Course Technology, 2005.

TEACHER, A. G. F.; GRIFFITHS, D. J. HapStar: automated haplotype network layout and visualization. *Molecular Ecology Resources*, Blackwell Publishing Ltd, v. 11, n. 1, p. 151–153, 2011. ISSN 1755-0998. Disponível em: <http://dx.doi.org/10.1111/j.1755-0998.2010.02890.x>.

TEMPLETON, A. R. Nested clade analyses of phylogeographic data: testing hypotheses about gene flow and population history. *Molecular Ecology*, Blackwell Science Ltd., v. 7, n. 4, p. 381–397, 1998. ISSN 1365-294X. Disponível em: <http://dx.doi.org/10.1046/j.1365-294x.1998.00308.x>.

_____. Statistical phylogeography: methods of evaluating and minimizing inference errors. *Molecular Ecology*, v. 13, n. 4, p. 789–809, April 2004.

_____. Nested clade analysis: an extensively validated method for strong phylogeographic inference. *Molecular Ecology*, v. 17, n. 8, p. 1877–1880, April 2008.

_____. The diverse applications of cladistic analysis of molecular evolution, with special reference to nested clade analysis. *International Journal of Molecular Sciences*, v. 11, n. 1, p. 124–139, 2010.

TEMPLETON, A. R.; BOERWINKLE, E.; SING, C. F. A cladistic analysis of phenotypic associations with haplotypes inferred from restriction endonuclease mapping. i. basic theory and an analysis of alcohol dehydrogenase activity in drosophila. *Genetics*, v. 117, p. 343–351, October 1987.

TEMPLETON, A. R. et al. Tree scanning: A method for using haplotype trees in phenotype/genotype association studies. *Genetics*, v. 169, p. 441–453, January 2005.

TEMPLETON, A. R.; SING, C. F. A cladistic analysis of phenotypic associations with haplotypes inferred from restriction endonuclease mapping. iv. nested analyses with cladogram uncertainty and recombination. *Genetics*, v. 134, p. 659–669, June 1993.

The International HapMap Consortium. The international hapmap project. *Nature*, v. 426, p. 789–796, 2003.

TURING, A. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, v. 42, p. 230–265, 1936.

WALLACE, C. S. *Statistical and Inductive inference by Minimum Message Length*. [S.l.]: Springer, 2005.

WIKIPEDIA. 2012. Disponível em: <http://en.wikipedia.org/wiki/MtDNA>.

WOOLLEY, S. M.; POSADA, D.; CRANDALL, K. A. A comparison of phylogenetic network methods using computer simulation. *PLoS ONE*, Public Library of Science, v. 3, n. 4, p. e1913, 04 2008.