Academic Year 2014

Thesis submitted to

Université Claude Bernard Lyon 1

in fulfilment of the requirements for the degree of

Doctor of Philosophy

Defended on 12 November 2014

by

Beatrice Donati

Graph models and algorithms in (co-)evolutionary contexts

Directeur: Marie-France Sagot
Co-Directeur: Pierluigi Crescenzi

Jury:	Pierluigi Crescenzi,	Directeur
	Christian Gautier,	Examinateur
	Linda Pagli,	Rapporteur
	Marco Pellegrini,	Rapporteur
	Marie-France Sagot,	Directeur
	Charles Semple,	Rapporteur
	Cecilia Verri,	Examinateur

❧ A Matilde e Luca

I would like to thank all the people, especially my professors Marie and Pilu, who have proved to be sufficiently crazy to believe that a philosopher could successfully develop mathematical models...for biologists. I really want to thank my family and my friends for supporting me. I am also in debt with David for making me feel less alone in all the travels of the last three years.

UNIVERSITE CLAUDE BERNARD - LYON 1

Président de l'Université	**M. François-Noël GILLY**
Vice-président du Conseil d'Administration	M. le Professeur Hamda BEN HADID
Vice-président du Conseil des Etudes et de la Vie Universitaire	M. le Professeur Philippe LALLE
Vice-président du Conseil Scientifique	M. le Professeur Germain GILLET
Directeur Général des Services	M. Alain HELLEU

COMPOSANTES SANTE

Faculté de Médecine Lyon Est – Claude Bernard	Directeur : M. le Professeur J. ETIENNE
Faculté de Médecine et de Maïeutique Lyon Sud – Charles Mérieux	Directeur : Mme la Professeure C. BURILLON
Faculté d'Odontologie	Directeur : M. le Professeur D. BOURGEOIS
Institut des Sciences Pharmaceutiques et Biologiques	Directeur : Mme la Professeure C. VINCIGUERRA
Institut des Sciences et Techniques de la Réadaptation	Directeur : M. le Professeur Y. MATILLON
Département de formation et Centre de Recherche en Biologie Humaine	Directeur : Mme. la Professeure A-M. SCHOTT

COMPOSANTES ET DEPARTEMENTS DE SCIENCES ET TECHNOLOGIE

Faculté des Sciences et Technologies	Directeur : M. F. DE MARCHI
Département Biologie	Directeur : M. le Professeur F. FLEURY
Département Chimie Biochimie	Directeur : Mme Caroline FELIX
Département GEP	Directeur : M. Hassan HAMMOURI
Département Informatique	Directeur : M. le Professeur S. AKKOUCHE
Département Mathématiques	Directeur : M. Georges TOMANOV
Département Mécanique	Directeur : M. le Professeur H. BEN HADID
Département Physique	Directeur : M. Jean-Claude PLENET
UFR Sciences et Techniques des Activités Physiques et Sportives	Directeur : M. Y.VANPOULLE
Observatoire des Sciences de l'Univers de Lyon	Directeur : M. B. GUIDERDONI
Polytech Lyon	Directeur : M. P. FOURNIER
Ecole Supérieure de Chimie Physique Electronique	Directeur : M. G. PIGNAULT
Institut Universitaire de Technologie de Lyon 1	Directeur : M. C. VITON
Ecole Supérieure du Professorat et de l'Education	Directeur : M. A. MOUGNIOTTE
Institut de Science Financière et d'Assurances	Directeur : M. N. LEBOISNE

Contents

Abstract

Résumé de thèse

Beatrice Donati

Cette thèse s'inscrit dans le cadre de la bioinformatique. Les outils mathématiques les plus utilisés dans ce travail relèvent de la théorie des graphes, des statistiques, de la théorie des ensembles et des mathématiques discrètes. Ces mathématiques ont permis de développer des modèles de systèmes biologiques ainsi que des algorithmes efficaces dans l'étude concrète de ces modèles. La nécessité d'analyses de jeux de données de très grande taille a rendu critique dans notre démarche cette notion d'efficacité des algorithmes. Il faut enfin remarquer que le champ biologique qui a servi de support à cette thèse nous a conduit à explorer un domaine particulier au sein de la théorie de la complexité, à savoir le développement et l'analyse des algorithmes d'énumération.

Le texte se compose de deux parties qui regroupent des résultats qui dérivent du même problème biologique. Dans chaque partie est présentée une introduction mathématique et une biologique, ainsi qu'une exposition détaillée des résultats que nous avons obtenus. Dans la première partie, la théorie des graphes est utilisée afin de modéliser l'information phylogénétique ainsi que les relations symbiotiques entre organismes. Cela conduit à l'analyse simultanée de plusieurs arbres, désignée sour le terme de co-phylogénie. Ces analyses sont importantes sur le plan fondamental par leur apport à la connaissance des mécanismes évolutifs mais aussi sur le plan plus appliqué dans le cadre des relations hôtes/pathogènes (la course aux armements), voire dans celui de l'émergence des pathologies nouvelles. Dans le premier chapitre, nous fournissons les principes mathématiques et biologiques nécessaires pour comprendre les résultats obtenus. En plus, nous donnons des informations sur l'état de la recherche dans le domaine de la reconstruction co-phylogénétique. En particulier, nous nous sommes intéressés à l'aspect énumératif de son coté énumératif centré autour de la possibilité d'expliciter toutes les solutions optimales pour une question donnée. Ce problème avait été déjà abondamment traité dans la littérature au moment où nous avons commencé ce travail. Cependant nous nous sommes très tôt rendu compte que non seulement aucun logiciel ne l'abordait d'une façon efficace et correcte, mais que de plus les limites, pratiques et théoriques, de cette approche étaient mal connues. C'est avec ce double objectif

que nous avons développé et amélioré un nouvel algorithme, appelé Eucalypt, qui, n'étant pas seulement un outil efficace et innovant pour la reconstruction phylogénétique , nous a permis d'étudier le comportement du modèle basé sur les évènements, en termes de nombre et qualité des solutions sur des données réelles. Nous avons largement comparé notre méthode avec les logiciels qui étaient disponibles. Les résultats de l'expérimentation conduite sur Eucalypt, nous ont permis de mettre en évidence les avantages et les difficultés d'une des approches classiques de la co-phylogénie. Le logiciel développé est accessible à l'addresse: http://eucalypt.gforge.inria.fr/. La méthode et les résultats correspondants sont présentés dans le deuxième chapitre. Cette partie de nos études est présentée dans l'article : "B. Donati, C. Baudet, B. Sinaimeri, P. Crescenzi, and M.-F. Sagot. Eucalypt: Efficient tree reconciliation enumerator", accepté par la revue Algorithms for Molecular Biology. Les études conduites avec Eucalypt, montrent que l'approche du scénario le plus parcimonieux présente des limites, qui ne peuvent pas être ignorés et dont un est constitué par la façon arbitraire avec laquelle on assigne des coûts aux différents événements ce qui influence profondément les résultats. Un deuxième point faible demeure le fait que sur des jeux de donnés réels d'une certaine ampleur, le nombre de solutions équivalentes est tellement élevé que toute réconciliation est absolument non justifiée. Pour répondre, au moins en partie à certains aspects négatifs emergés de notre analyse, nous avons avant tout défini, une nouvelle version du problème, dans laquelle les transferts ont une distance maximale fixée : le « k-bounded-All-MPR ». Eucalypt donne des solutions a cette version du problème en les énumérant avec un délai polynomial. Le deuxième pas pour éviter les faiblesses de la technique dite basée sur les évènements, a été le développement d'un deuxième algorithme, nommé Coala, basé sur un modèle Bayésien approximé. Les bénéfices de cette méthode sont doubles : il permet à la fois d'inférer un ensemble de coûts ad hoc pour un certain jeu de données, et de fournir une estimation de la fréquence de chaque évènement. Cela est particulièrement utile lorsqu'il n'est pas possible d'appliquer la règle de la parsimonie.

Cette partie de nos études est présentée dans l'article : "C. Baudet, B. Donati, B. Sinaimeri, P. Crescenzi, C. Gautier, C. Matias, and M.-F. Sagot. Co-phylogeny reconstruction via an Approximate Bayesian Computation, article en révision dans Systematic Biology. Dans la partie 2, l'application biologique change, même si les outils mathématiques utilisés restent toujours la théorie des graphes

et l'optimisation combinatoire. Le problème biologique que nous avons traité se situe dans le domaine du séquençage génomique. Plus spécifiquement, il s'agit d'ordonner et d'orienter un ensemble de fragments de même longueur, appelés contigs. On appelle ce processus effectuer un scaffolding. Celui-ci est introduit dans le quatrième chapitre, avec le pré-requis mathématiques nécessaire pour l'aborder. Le développement des méthodes de séquençage massives (NGS) a conduit à la nécessité du développement d'algorithmes et d'approches expérimentales pour terminer le séquençage complet d'un génome. Nous avons développé une nouvelle méthode avec le logiciel Medusa. Cet algorithme présenté dans le cinquième chapitre, résout efficacement le problème du scaffolding en utilisant beaucoup moins de mémoire que les procédures les plus utilisées.

En fait, si la majorité des logiciels de scaffolding nécessite d'une grande quantité d'informations, provenant des démarches précédentes du processus du séquençage, Medusa exploite la comparaison avec un nombre variable d'organismes similaires, ce qui permet de séparer complètement la phase du scaffolding de l'assemblage et de travailler avec des files d'entrée sensiblement plus légeress.

Avec Medusa, le problème du scaffolding est formalisé en terme d'optimisation combinatoire sur descgraphes et résolu grâce à un algorithme d'approximation avec un facteur constant. Contrairement aux autres méthodes actuellement utilisées, il ne nécessite ni d'une connaissance « a priori » des relations phylogénétiques qui existent entre l'organisme cible et les organismes de comparaison, ni de librairires de reads provenant d'un assembleur. Tout cela implique facilité d'utilisation et vitesse sont deux caractéristiques importantes de notre méthode. Benchmark et tests montrent aussi que Medusa est précis, et obtient souvent une meilleure performance que les scaffolders traditionnels. Medusa peut être utilisé localement ou à travers une interface web: http://combo.dbe.unifi.it/medusa/, et est présenté dans l'article « E. Bosi, B. Donati, M. Galardini, S. Brunetti, M.-F. Sagot, P. Lio, P. Crescenzi, R. Fani, et M. Fondi. Medusa: a multi-draft based scaffolder », en révision pour la revue Bioinformatics. Durant le développement de ce dernier algorithme nous avons rencontré un ensemble très intéressant de problèmes, purement mathématiques. En particulier, nous nous sommes intéressés a à un problème appelé Implicit Hitting Set qui n'a jamais été étudié en termes de complexité d'énumération.

Ce problème a trouvé sa première application dans le cadre de la biologie computationnelle. Cependant, nous croyons qu'il est également intéressant d'un point de vue théorique parce que, grâce à son caractère très abstrait, il peut être considéré comme un cadre à l'intérieur duquel on peut redéfinir la plupart des problèmes combinatoires. Puisque le problème peut être formulé de différentes façons, nous proposons d'abord un ensemble de définitions et ensuite une démonstration de NP-complétude pour la plus générale de ces définitions. De nombreuses questions restent encore ouvertes et, en fait, nous aimerions étudier le problème dans deux directions: Top-bottom, d'identifier les conditions dans lesquelles le problème d'énumérer les solutions cesse d'être difficile; et bottom-top en définissant un ensemble de conditions pour lesquelles le problème est résoluble efficacement. Nous savons que certains de ses sous-problèmes sont polynomialement résolubles, ce qui garantit que cette condition existe. Les résultats et les lignes de recherche actuellement ouvertes sont présentées dans le chapitre 6.

Introduction

The relationship between graph theory and biology has a long history, from the very first metaphor of *the tree of life* to the advanced techniques of genome sequencing. Perhaps it is possible to catch a glimpse of this link in the famous phylogenetic tree sketched in Darwin's notebook , back in 1837 and shown in Figure 0.1. The simplicity of this structure, drawn in a period when biology was starting to be mathematically formalized, well represents the fact that biological concepts can be translated in graph-theoretical terms. We have to wait until the middle of the twentieth century for the birth of *bioinformatics / computational biology*[1] as a properly defined field, but since then the history of biology and the history of graph theory have been closely intertwined.

Many famous biological problems have been described in terms of graphs, strings, sets and all the equivalent formulations that discrete mathematics provides to the mathematician. Just as an example, think of the *Needleman-Wunsch* algorithm to align protein or nucleotide sequences, dated from 1970, that led to a series of results in the field. On the other hand, biology has often provided the inspiration for purely theoretical research in computer science. In this introduction, we want to clarify, without trying to be exhaustive, the reasons for which biology seems to be such a fertile ground for computationally difficult (meaning also interesting!) problems and at the same time why discrete mathematics, together with other types of mathematical approaches, is important to model biological problems.

Graphs are simple and elegant structures that are suitable to model data of

[1] The two are sometimes given different meanings; henceforth, the term computational biology will be preferred.

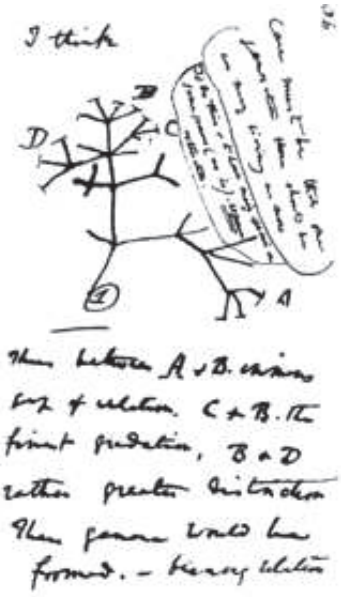


Figure 0.1: Darwin's first sketch of an evolutionary tree, dated around July 1837.

different nature. However, biology has exploited graph models since a long time; we have already recalled that trees allow to represent an evolutionary process, but every branch of biology uses its own specific type of graph and the examples are countless. They range from *metabolic networks*, where one possible model has nodes corresponding to compounds and edges to chemical reactions, to *de Brujin* graphs that may be used to encode the relationships between the nucleotide sequences of a set of DNA fragments. The ease with which biological processes are represented in graph-theoretical terms is only one reason for their wide use in bioinformatics and computational biology. Another one is that, since these fields are characterised by great amounts of data that have to be analysed, efficiency is a crucial issue. Graph theory has a long and rich history of development and analysis of algorithms. Problems on graphs are well categorized in terms of computational complexity and a long series of theoretical results are available.

It is no wonder then that graphs have often been adopted as an instrument for biological modelling; less obvious, and perhaps more interesting, is to ask whether and how a theoretician can be encouraged to work in computational biology. Are there purely mathematical reasons that make biological problems

particularly attractive for algorithm developers? There are various interesting aspects that are specific to this field and are not common in other applications. The first aspect is the need to consider the multiplicity of solutions: in most optimisation problems, the optimal solution is not unique in general, but usually a single solution is enough as long as it meets the conditions of consistency and optimality defined by the problem. For example if you are looking for the fastest car route between two cities, you have no interest in enumerating all the equivalent paths. In biology instead this is rarely true; the purpose of the biologist is often to reconstruct some natural fact and not, for example, to design the best strategy. This means that the biologist prefers to obtain all the equivalent solutions and then to evaluate this set of possibilities in a qualitative or experimental way. In mathematical terms, this means that the computational biologist needs to develop *enumeration algorithms*. Some enumeration problems are already studied and classified in in terms of computational complexity but the results relative to them are far less numerous than the results about simple optimisation problems. Moreover, in general, polynomial algorithms to enumerate an entire set of solutions are difficult to find. In Section 1.1, we will define formally the concept of efficiency for enumeration techniques. For the moment, we can say that the difficulty of the problems, together with the great number of open problems in enumeration theory, make this field interesting for the mathematician. Another challenging aspect in biological modelling is that the size of the data is often very big. This makes efficiency a real issue and not only a theoretical vagary: answering to biological questions requires a precise design and an accurate implementation. Usually, we are confronted to NP-hard problems and a direct approach is not feasible. Different trade-offs can be adopted to solve in an efficient way a computationally hard problem: approximation algorithms can be developed or the optimality requirement can be substituted by other, computationally less expensive constraints. This happens, for example, when minimal (not redundant) solutions are selected instead of minimum ones. In our work, we used all such types of approaches in different situations, and the present manuscript can provide to the reader a good overview of the challenges encountered in the attempt to give to the biologist the models and methods that can help him/her in the analysis of natural data.

To help the reader in following our series of results, the manuscript is divided in two parts, each one characterised by a common mathematical background

and by the biological application for which the algorithms were designed. For each Part, the first chapter is dedicated to the definition of this background and the others to our original results. At the end of each chapter dedicated to our research, future perspectives and open problems are presented. Finally, a common appendix is presented that contains additional material.

Part I groups a set of results designed for phylogenetics analysis, and in particular for reconstructing the co-evolution of two groups of organisms. First the mathematical structures involved are defined (Section 1.1), then the biological background and the state of the art of the *co-phylogeny reconstruction problem* is given (Section 1.2), and finally our original contributions are extensively presented. In particular two new algorithms – Eucalypt (Chapter 2) and Coala (Chapter 3) – were developed.

A similar structure is maintained in Part 2, where another set of studies is presented. We developed a model and an algorithm for helping in the finishing steps of dna sequencing. The mathematical and biological background is presented in Section 4.1 and 4.2 respectively. Our original model for the *contig scaffolding problem*, and our algorithm MeDuSa, are presented in Chapter 5.

During the development of this method we encountered some pure theoretical open problems and we decided to dedicate part of our job to their analysis. Chapter 6 is dedicated to the formal definition of a set of problems, all related to the Implicit Hitting set enumeration problem. After some formal definitions, an original NP-completeness result is presented in Section 6.2 and the future directions of our work are finally described in Section 6.3.

The material contained in Chapter 2 is presented in the paper: " B. Donati, C. Baudet, B. Sinaimeri, P. Crescenzi, and M.-F. Sagot. *EUCALYPT: Efficient tree reconciliation enumerator.*", accepted for publication on the journal *Algorithms for Molecular Biology*.

The material contained in Chapter 3 is instead presented in the paper: "C. Baudet, B. Donati, B. Sinaimeri, P. Crescenzi, C. Gautier, C. Matias, and M.-F.Sagot. *Co-phylogeny reconstruction via an approximate Bayesian computation.*", and currently under revision by the journal *Systematic Biology*.

Finally, material contained in Chapter 5 is presented in: "B. Donati, E.Bosi, M. Galardini, S. Brunetti, M.-F. Sagot, P. Lió, P. Crescenzi, R. Fani and M. Fondi. *MeDuSa: a multi-draft based scaffolder.*", and currently under revision by the journal *Bioinformatics*.

Part I

Tree composition

Chapter 1

Reconciliation of phylogenetic trees.

In the set of results contained in Part I, graph theory is used to model phylogenetic information, and in particular to investigate the interactions among different organisms at a phylogenetic level. In this first chapter, we give all the preliminaries that lead to the definition of our problem, together with a state of the art.

In Section 1.2, we present the specific biological problem we will address and define a formal model for its computational treatment.

1.1 Mathematical background

In this section, we introduce some concepts extensively used in Part I. A first section is dedicated to *phylogenetic trees*, while a second introduces the concept of *enumeration problems* and of *efficient enumeration algorithms*.

Phylogenetic Trees

Phylogenetic trees, also called *Dendrograms* or *Evolutionary Trees*, are mathematical objects encoding the evolutionary interrelations of a group of organisms, derived from a common ancestral. Since, depending on the framework inside which they are used, phylogenetic trees can present different mathematical characteristics, it is useful to provide here the definition that will be used throughout the thesis.

In the present work, we focus on trees that are rooted, binary, full (every vertex other than the leaves has two children) and labelled at the leaves.

The interpretation is the following:

- An arc of the tree represents the life of a single species.
- An inner vertex corresponds to a speciation event, where the incoming arc corresponds to the old species and the two outgoing arcs to the new ones.
- The root represents the *least common ancestor* of all the *taxa*.
- The leaves are the current taxa.

Usually the least common ancestor is putative but necessary to give a temporal direction to the tree, from the root to the leaves, coherent with the evolution of these organisms. From a mathematical point of view, the structure of a rooted tree induces a partial order on the vertices that, in the case of phylogenetic trees, is interpreted as being temporal: each speciation (represented by a given vertex) happens strictly after any ancestor and strictly before any of its descendants. A relative order between two incomparable vertices is not specified. In some cases, the available temporal information is more precise and it is possible to enrich this structure by adding labels to the arcs that indicate a period of existence for the corresponding species. However, this kind of information is rarely available and may not be reliable. We therefore decided not to take it into consideration.

The following notation will be used in the remaining of the work: by a phylogenetic tree T, we thus mean a rooted tree with labelled leaves and where the root, $r(T)$, has in-degree 0 and out-degree 2, the leaves have out-degree 0 and in-degree 1 and every other vertex has in-degree 1 and out-degree 2. For such a tree T, the set of vertices is denoted by $V(T)$, the set of arcs by $A(T)$, and the set of leaves by $L(T)$. The root of T is denoted by $r(T)$. Given an arc $a = (v, w) \in A(T)$, going from v to w, we call its *head*, denoted by $h(a)$, the vertex w and its *tail*, denoted by $t(a)$, the vertex v. For a vertex $v \in V(T)$, we define the set of *descendants* of v, denoted by $Des(v)$, as the set of vertices in the subtree of T rooted at v. Similarly, the set of *ancestors* of v, denoted by $Anc(v)$, is the set of vertices in the unique path from the root of T to v, including the root and v. For a vertex $v \in V(T)$ different from the root, we call its *parent*, denoted by $par(v)$, the vertex x for which there is the arc $(x, v) \in A(T)$. We denote by $\texttt{lca}(v, w)$ the least common ancestor of v, w in T. Finally, we denote by $\geq$

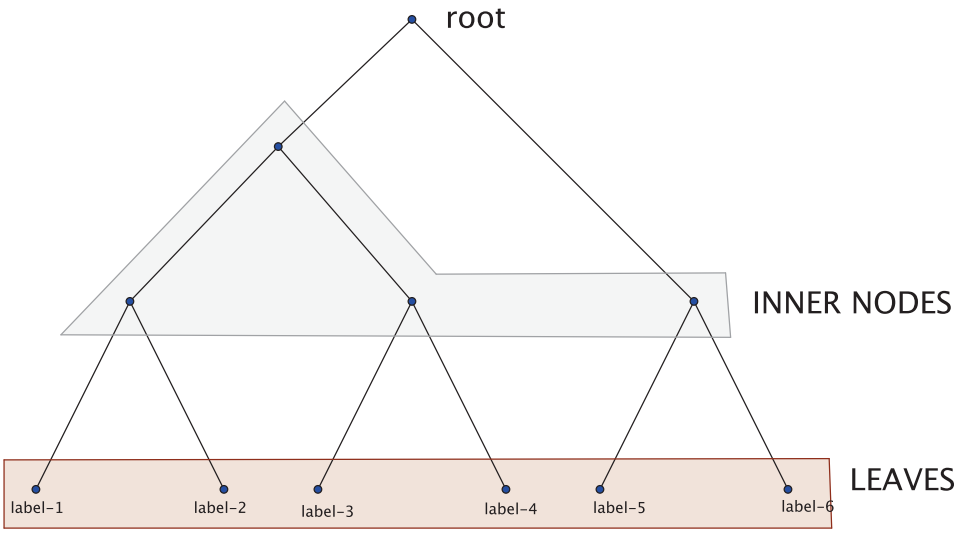


Figure 1.1: A binary labelled tree.

the partial order induced by the ancerstorship relation in the tree. Formally, for $x, y \in V(T)$, we say that $x \geq y$ if $x \in Anc(y)$. If neither $x \in Anc(y)$ nor $y \in Anc(x)$, the vertices are said to be *incomparable*.

Observe that the phylogenetic tree for a given set of related organisms is not unique. In our work, we do not consider the process of obtaining the phylogenetic trees. We assume instead that the most reliable trees have already been inferred.

Measures on trees

It is useful, for many purposes, to compare two different trees and to define measures of similarity between them. There is a wide literature on distances for phylogenetic trees (Felsenstein, 2003), we will recall some example here.

One of the oldest and efficient ones is the Robinson-Foulds distance (Robinson and Foulds, 1981) which can be calculated in linear time (Day, 1985).

For any $v \in V$, let the *cluster* of v, denoted by $C(v)$, be equal to $L(T(v))$, the set of leaves which are the descendants of v. We denote by $C(N) = \{C(v) : v \in$

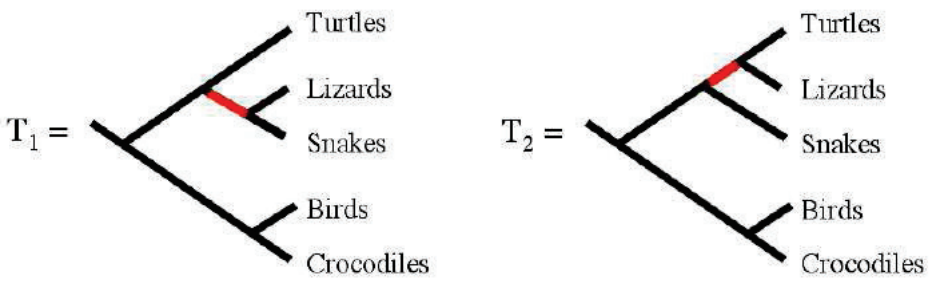


Figure 1.2: An example of a single difference that Robinson-Foulds distance takes in account

$V)\}$ the *cluster collection* of T. Observe that, since repetitions are allowed in the labels of the leaves, $C(N)$ can be a multi-set.

The Robinson-Foulds distance d_{rf} between two phylogenetic trees T_1 and T_2 is defined as follows.

Definition 1
Robinson-Foulds distance

$$d_{rf}(T_1, T_2) = \frac{|C(T_1)\ C(T_2| + |C(T_2)\ C(T_1|}{2}$$

A second approach to measure the similarity of two trees is to define a finite set of local operations that transform a tree in another, and to assign a cost to each operation. Given two trees, T_1 and T_2, the most parsimonious set of operations that transform T_1 in T_2 is computed. The total cost of this optimal transformation is given as the distance between the two trees. These kind of measures are called *edit tree distances*, and are usually $NP - complete$ to compute for general cases. However, many polynomial time algorithms are known for some special classes of trees as *bounded degree* tress or *ordered* trees. See Bille, 2005 for a survey about tree edit distances. In Figure 1.3 an example of this edit operation is shown: the bisection-reconnect operation consists in choosing an edge to cut and then reconnect the two separate components with a new edge. This operation is already sufficient to define a distance, called *TBR distance*, equivalent to the number of operation needed to transform the first tree in the second (D. Swofford and Hillis, 1996).

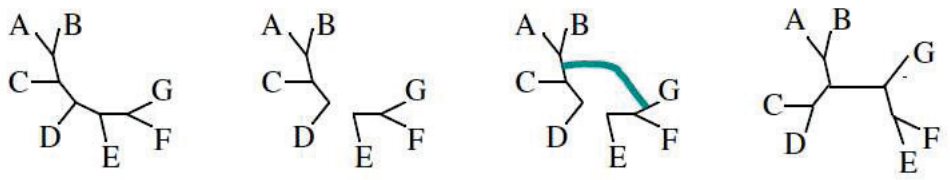


Figure 1.3: An example of the basic operation in the *Tree Bisection Reconnect* distance.

Many of the distances used in biology are unfortunately NP-hard to compute (Waterman and Smith, 1978; Hein, 1990; Baroni et al., 2005), and this holds also for *TBR* distance. This difficulty partially explains also the fortune of the Robinson-Foulds one that can be computed in linear time.

Although in practice, the efficiency of the method is an important requirement, it is also true that the Robinson-Foulds distance has some drawbacks for the analysis of real data. This measure is indeed poorly distributed and thus is not a good enough discriminator (Steel and Penny, 1993; Bryant and Steel, 2009). Moreover, many efficient to compute distances are not robust to even small changes (such as in the position of a single leaf) in one of the two trees. Depending on the context, this can be a problem.

For our purposes (see Chapter 3), we need to define a similarity measure between multi-labelled trees that discriminates better than Robinson-Foulds but can still be calculated in polynomial time. An interesting concept in this sense is the *maximum agreement area cladogram* (MAAC) (Ganapathy et al., 2006). This is a generalisation for multi-labelled trees of the well-known *maximum agreement subtree* (Finden and Gordon, 1985; Farach-Colton, Przytycka, and Thorup, 1995). It corresponds to the number of leaves in the largest isomorphic subtree that is common to two trees. Clearly this isomorphism takes into account the labels of the trees.

Given two trees T_1, T_2, an *agreement cladogram* is a labelled tree that is a subtree of both T_1, T_2, with isomorphic leaf labelling. The size of an agreement cladogram is given by the number of its leaves. We denote the agreement cladogram having maximum size by $maac(T_1, T_2)$.

A distance, d_{maac}, based on the concept of agreement cladogram can be defined as follows.

Definition 2
MAAC *distance*

$$d_{maac}(T_1, T_2) = max(|L(V_1)|, |L(V_2)|) - maac((T_1, T_2))$$

It can be calculated in $O(n^2)$ time, where n is the size of the largest input tree (Ganapathy et al., 2006).

For our porpoises (see Chapter 3) we are looking for an efficient distance that can discriminates among tress better then d_{rf}. We then defined a normalised version of d_{maac} that takes into account also the size of the leaves in common between the two trees. More formally, for two trees T and T', we define the measure $maacN(T, T')$ as follows:

$$maacN(T, T') = \begin{cases} 1 - \dfrac{MAAC(T, T')}{|L(T) \cap L(T')|} & \text{if } L(T) \cap L(P') \neq \emptyset \\ 1 & otherwise. \end{cases}$$

Observe that the intersection operation involves multi-sets. We recall that a multi-set is a generalisation of a set where the elements are allowed to appear more than once, hence the operations take into account their multiplicity.

Enumeration Algorithms

Computational complexity characterises problems in terms of the number of atomic steps needed to solve them. The first kind of problems that has been analysed are the so called *decision* problems, where the solution consists of a boolean variable. A very natural generalization of this boolean problems are the *optimization* problems where the possible solutions are ranked with respect of a given quality function and the aim of the problem consists in find an optimal solution. In both these cases the mathematical community agree in consider *feasible* or *tractable* a problem for which a solution can be found in polynomial time with respect to the size of the input.

Some of the results contained in the present work concern the so called *enumeration* techniques (the first enumeration algorithm will be presented in Chapter 2). In this case we are not interested in producing one single solution but to enumerate all the equivalent solutions. Since the space of optimal solutions

can be exponentially large in the size of the input, in most of the cases we have no hope to list its elements in polynomial time. The definition of efficiency indeed appears to be too restrictive for enumeration problems and we need to define new concepts that are better adapted to the context. We provide now some definitions in order to clarify what is exactly an enumeration algorithm and how the concept of *efficiency* has been adapted for such algorithms.

Definition 3
Enumeration problem

Given a combinatorial problem P, we denote by $\mathcal{S}(P)$ its set of solutions. The enumeration version of P consists in explicitly listing, without repetitions, all the elements of $\mathcal{S}(P)$.

If P is an optimisation problem, the enumeration version of P consists in listing all and only the optimal solutions.

There are more then one property that can characterise enumeration algorithm in terms of efficiency; here we will define the most important ones:

Definition 4
Polynomial total time

An enumeration problem P is said to be *Polynomial total time* solvable if and only if an algorithm exists that solves it and the number of steps required to complete the algorithm for a given input is polynomial both in the size of the input and in the number of solutions.

The property defined above does not give any guarantee about the behaviour of the algorithm in the intermediate steps of the enumeration. We define then two other, stronger, properties that involve the time of outputting a single solution in the middle of the listing process.

Definition 5
Incremental Polynomial solvable

An enumeration problem P is said to be*Incremental Polynomial solvable* if and only if it is Polynomial total time solvable and, given a set of solutions already generated, the number of steps required for the output of anew solution is polynomial in the size of the input and in the size of the set of solutions already generated.

Definition 6
Polynomial delay solvable

An enumeration problem P is said to be*Polynomial delay solvable* if and only if it is Polynomial total time solvable and the number of steps required between the

output of a solution and the output of the next one is polynomial in the size of the input.

It easy to see that Definition 4 is included in Definition 5 that is included in Definition 6.

1.2 Biological application: The co-phylogeny reconstruction problem.

The biological framework.

The term *co-evolution* is used in biology to describe the fact that different organisms affect each other's evolution. It is well known that organisms that live in a close ecological relationship may act as agents of natural selection for each other, and that this pressure may be responsible for much of the genetic diversity seen in normal populations. This interaction is possible between species belonging to very different taxonomical categories, such as plants and animals, or animals and bacteria. Observe that evolution in response to abiotic factors, such as climate, is not considered coevolution, since climate itself does not undergo evolution.

The concept of co-evolution is quite old in evolution theory. The first historical example dates back to the nineteenth century. In the study "Fertilisation of Orchids" (Darwin, 1862), C. Darwin describes a species of orchid from Madagascar characterised by a nectary so deep that no known species of moth could pollinate it. A few years later, Alfred Russell Wallace made a precise hypothesis of what this study suggested, predicting the existence of a particular moth with a proboscis long enough to reach the nectar:

> "That such a moth exists in Madagascar may be safely predicted; and naturalists who visit that island should search for it with as much confidence as astronomers searched for the planet Neptune,–and they will be equally successful!"

Indeed, in 1903, a population of *Xanthopan morganii* with an unusually long proboscis was discovered in Madagascar, and it was named subspecies *praedicta* in honor of Wallace's prediction. Nowadays, the study of the interactions between different species, and in particular of the resulting evolutionary pressure, has

become an active field on its own. Moreover, the birth of computational biology has deeply changed our approach to the study of co-evolution. In particular, more recently, a number of models and algorithms have been developed that may help provide mathematical evidence in favour or against some important biological hypotheses.

Starting from the general framework of co-evolution, ecology distinguished different co-evolutionary relationships depending on the role of the two agents.These are:

- The predator/prey interaction: In this case, the prey evolves in order to survive the predator and, vice versa, the predator adapts to these changes. Observe that, even in the case when the prey does not evolve enough to survive and simply stops inhabiting the same environment, the passive response of the prey induces in the predator an evolutionary change. The predator is indeed forced to react, looking for other food resources.

- The interaction among two different competitors: two species that share the same food or the same living space influence each other's evolution. The development of new hunting techniques or the adaptability to change one's feeding habits become a fundamental feature to be pursued by both agents.

- The parasite/host relationship: It is possible to speak properly of a parasitic relationship only if the parasite lives *on* or *in* the body of the host in a way that is dangerous for the host.

  Since they not only share a same environment but the host in a way *is* the environment of the parasite, this makes the parasite very sensible to the evolution of its host even when the changes are not aimed at the elimination of the parasite. For this reason, the host-parasite relationship is usually considered to be asymmetric with the evolution of the parasite following the one of the host. However, the parasite may impose some evolutionary pressure on its host by forcing the latter to get rid of it or to neutralise the negative effect of its presence.

- The mutualistic relationship: This takes place when two different species both benefit from the interaction. One of the two species can help the other one by protecting it from enemies or helping in the reproductive process.

Examples of such relationship are the birds that live with the rhinoceros, eating the insects that affect the big mammal, or the bees that pollinate the flowers.

- The commensalistic relationship: When one species benefits from the proximity of a second one which, however, does not enjoy nor is damaged by this coexistence, this is called commensalism. It is indeed possible for an organism (typically a big animal) to create feeding opportunities for smaller creatures or help them to move carrying them on his body.

The phenomenon of co-evolution is so complex that even this general subdivision in macro-categories is restrictive. A parasitic relationship could evolve into mutualism if one of the two species neutralized the negative effect of this relationship. Inversely, a species interacting positively with another could became its predator if other sources of food are not available.

An analytical study of these interconnections can be useful not only to validate biological hypotheses of past evolution history, but also to predict the evolutionary response of some organisms of interests (for example a pathogen) to the presence of other organisms in a same ecosystem (for example a host or a competitor). One must however pay attention to the difficulties inherent in any computational approach to the problem. One of the factors that must be taken in consideration is that any choice of a subset of organisms from an entire ecosystem is always artificial, since all the co-existent ones interact at a same time. Moreover, the phylogenetic information can be partial or wrong and, most importantly, any *a posteriori* validation is almost impossible in this field.

Keeping in mind all these delicate aspects, it is also true that, beyond any qualitative consideration, the large amount of quantitative data forces to look for automatic methods and mathematical models that can help during the analysis. From a mathematical point of view, to design a framework for a co-evolution study is an interesting challenge since the amount of data to analyse can be very large and also because the problems are sufficiently general to open this model to many different applications: it is worth observing that the concept of co-evolution has been applied by analogy to many different fields such as computer science (Paredis, 1995) (Jong, 2005), sociology (Hird, 2010) and linguistics (Deacon, 1997). In the next section, we present an approach to a

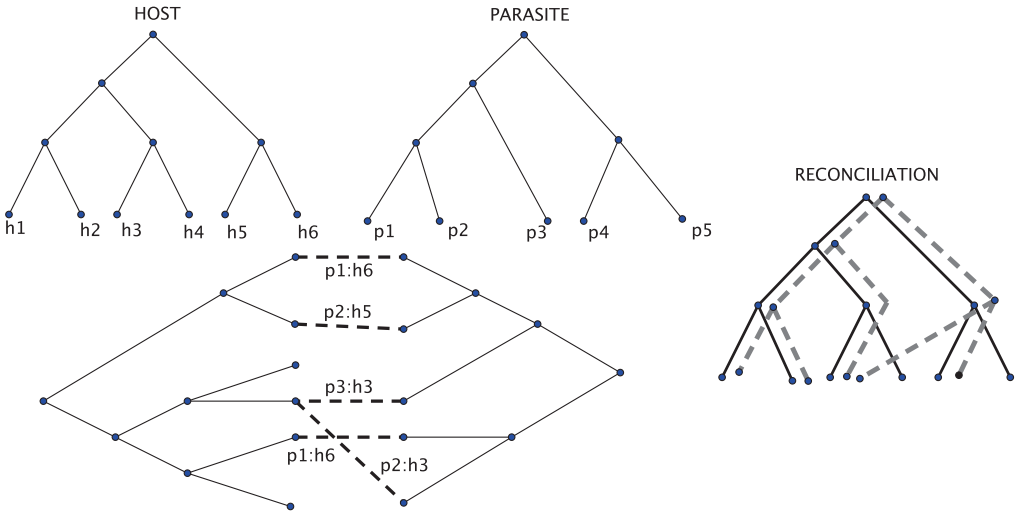


Figure 1.4: A schematic representation of a phylogenetic reconciliation

numerical analysis of co-evolution. In particular, our attention will focus on the parasite-host relationship, but the model can be adapted also to other situations.

Phylogenetic Scenarios and the DTL model.

Phylogenetic trees can be used to describe the evolutionary histories of a group of organisms. Suppose we have identified two groups of organisms that are assumed to interact, and that an accurate phylogeny has been built for each group. If an interaction between the two has driven their evolutionary histories, this fact has to be somehow reflected in the trees.

The aim of the model we are going to present here is to reconstruct the common history of two groups of organisms by comparing their phylogenetic trees. As mentioned, we will focus on parasite-host relationships. In this case, the two groups play different roles: the host evolves and the parasite may, or may not, respond to this evolutionary event. Each speciation event of the parasite (represented by a vertex of its tree), has to be placed inside the host phylogeny. This placement tells us when this event took place (relatively to the host history) and if this event happened in response to the host evolution or is independent from it. Mathematically this corresponds to define a mapping between the nodes of the parasite and the nodes of the host. The reconstruction

of this common history is called *phylogenetic reconciliation* and will be formally defined later in this Section.

A phylogenetic tree reconciliation has been the approach of choice for investigating the co-evolution of sets of organisms such as hosts and parasites (Charleston, 1998; Charleston, 2003; Merkle and Middendorf, 2005). As already pointed out in the previous section, one advantage of this mathematical framework is that it is applicable to different types of data. For instance, it is extensively used for analysing the associations between genes and species (Doyon et al., 2011b; Hallett and Lagergren, 2001; Tofigh, Hallett, and Lagergren, 2011), and between species and geological history (Rosen, 1978). The similarity between all three classes of problems was pointed out by Page already in 1994 (Page, 1994) and further considered in (Maddison, 1997; Ronquist, 2002). More recently, a unique generalised formal model appeared in (Wieseke, Bernt, and Middendorf, 2013). In this work, we focus on the host/parasite associations but we want to call attention to the fact that, due to the similarity of the models, our algorithm can be straightforwardly applied to the other problems as well.

The model we are going to present now belongs to the so-called *event-based* methods. In this approach, a finite number of evolutionary events are taken in account and for each of them a formal representation inside the model is defined and a cost is associated. When the mapping of the vertices of the parasite tree in the vertices of the host tree is defined, it is possible to determine the event associated to each vertex by looking at the mappings of its children (see Figure 1.5). This means that a reconciliation can be uniquely associated to a multi-set of events and a total cost of a reconciliation can be calculated. A parsimonious solution (or simply a *reconciliation*) seeks to minimise the total cost of the mapping.

If timing information is available, i.e. if we happen to know the total order in which speciation events occurred in the host phylogeny, then any proposed reconciliation must also respect the temporal constraints imposed by the available timing information. In this case, the reconciliation problem can easily be solved using dynamic programming, in time polynomial in the size of the trees (Libeskind-Hadas and Charleston, 2009). However, timing information may not be available or may be insufficiently reliable to be used with enough confidence. In such a case, the reconciliation problem is NP-hard (Hallett and Lagergren, 2001; Conow et al., 2010; Tofigh, Hallett, and Lagergren, 2011). In this case, an

efficient algorithm is able to generate optimal solutions in polynomial time but without guaranteeing the time feasibility constraint.

The choice of the set of events is not unique but, for the moment we will focus on the more used and accepted set of events (Page and Charleston, 1998; Charleston, 1998). These are: *co-speciation* (this happens when both host and parasite speciates), *duplication* (when the parasite speciates but not the host, both new parasite species remaining associated with the host), *loss* (when the host speciates but not the parasite, leading to the loss of the parasite in one of the two new host species), and *host-switch* (when the parasite speciates, one species remaining with its current host while the other switches, that is jumps to another).(See Figure 1.5) In the context of gene-species associations, this model is known as the *DTL* (for "Duplication, Transfer, and Loss") model for the reconciliation problem and has been extensively studied (see, for example, (Doyon et al., 2011b; Hallett and Lagergren, 2001; Tofigh, Hallett, and Lagergren, 2011; Bansal, Alm, and Kellis, 2012; Stolzer et al., 2012)). In particular the model of host-parasite evolution we will rely on in this work is presented by Tofigh *et al.* (Tofigh, Hallett, and Lagergren, 2011), and later further analysed by Bansal *et al.* (Bansal, Alm, and Kellis, 2012).

We can now define the model in a more formal way: Let H, P be the phylogenetic trees for the host and parasite species respectively. We define ϕ as a function from the leaves of P to the leaves of H that represents the association between currently living host species and parasites. Such association is an input of our algorithm, together with the trees themselves. In our model, we allow each parasite to be related to one and only one host, while a host can be related to zero, one, or more than one parasite. More formally, ϕ is thus a function which may be neither surjective nor injective.

Definition 7
Phylogenetic Reconciliation

A *reconciliation* γ is a function $\gamma : V(P) \to V(H)$ that is an extension of ϕ. In particular γ partitions the set $V(P)$ into three sets Σ, Δ, and Θ, and a subset Ξ of $A(P)$, for which the following hold Tofigh, Hallett, and Lagergren, 2011:

1. For any $p \in L(P)$, $\gamma(p) = \phi(p)$ (γ extends ϕ).

2. For any internal vertex $p \in V(P) - L(P)$ with children p_1 and p_2:

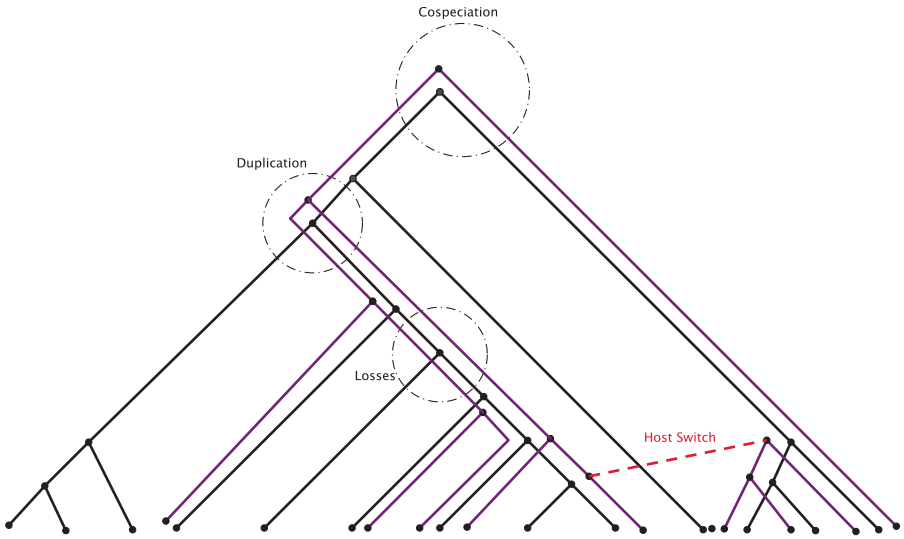


Figure 1.5: Schematic representation of a phylogenetic scenario with co-speciation, duplication, host-switch and loss events.

a) $\texttt{lca}(\gamma(p), \gamma(p_i)) \geq \gamma(p_i)$, for $i = 1, 2$ (a child cannot be mapped in an ancestor of the father).

b) $\texttt{lca}(\gamma(p), \gamma(p_1)) = \gamma(p)$ or $\texttt{lca}(\gamma(p), \gamma(p_2))) = \gamma(p)$ (one of the two children is mapped in the subtree rooted at the father).

3. For any $(p_1, p_2) \in \Xi \Leftrightarrow \texttt{lca}(\gamma(p_1), \gamma(p_2)) \notin \{\gamma(p_1), \gamma(p_2)\}$ (the arc (p_1, p_2) is an arc denoting a switch event).

4. For any $p \in V(P) - L(P)$ with children p_1 and p_2:

   a) $p \in \Theta \Leftrightarrow (p, p_1) \in \Xi$ or $(p, p_2) \in \Xi$ (p is associated to a switch event.

   b) $p \in \Delta \Leftrightarrow \texttt{lca}(\gamma(p_1), \gamma(p_2)) \in \{\gamma(p_1), \gamma(p_2)\}$ (the children are mapped to comparable vertices and p is associated to a duplication event).

   c) $p \in \Sigma \Leftrightarrow \texttt{lca}(\gamma(p_1), \gamma(p_2)) = \gamma(p)$ and $\gamma(p_1)$ and $\gamma(p_2)$ are incomparable and p is associated to a co-speciation event.

The sets Σ, Δ, and Θ correspond to the vertices of P associated to, respectively, co-speciations, duplications, and host-switches, while the set Ξ corresponds to

the arcs associated to host-switches. Finally, losses are identified by a multi-set $\Lambda \subseteq V(H)$ containing all the vertices $h \in V(H)$ that are in the path from the image of a vertex in $V(P)$ and the image of one of its children.

The triple $S = \langle H, P, \gamma \rangle$ is said to be a *scenario* or simply a reconciliation. Given a *vector* $\langle c_c, c_d, c_s, c_l \rangle$ of non negative real values that correspond to the cost of each type of event, the *cost* of a reconciliation is equal to $c_c|\Sigma| + c_d|\Delta| + c_s|\Theta| + c_l|\Lambda|$.

Host switches can introduce an incompatibility due to the temporal constraints imposed by the host and parasite trees, as well as by the reconciliation itself. Determining whether a reconciliation is time-feasible can be done in polynomial time Stolzer et al., 2012. It is common to refer to a time-feasible (unfeasible) solution as acyclic (cyclic).

Given a riconciliation $\gamma : V(P) \to V(H)$ we build the digraph $G = (V_G, E_G)$ where: $V_G = V_H$ and the set of edges E_G is defined as follows:

- $E_G = E_H \cup$ for all the couples of transfer edges $(u, v)(u', v')$ for which $u \in Ancestors(u')$:
  - $(p(\gamma(u)), \gamma(u'))$
  - $(p(\gamma(u)), \gamma(v'))$
  - $(p(\gamma(v)), \gamma(u'))$
  - $(p(\gamma(v)), \gamma(v'))$

Note that when $(u, v) = (u', v')$ the edges to add are:

- $(p(\gamma(u)), \gamma(u))$ redundant
- $(p(\gamma(u)), \gamma(v'))$ (it creates a path from the ancestors of the donor to the recipient)
- $(p(\gamma(v)), \gamma(u'))$ (it creates a path from the ancestors of the recipient to the donor)
- $(p(\gamma(v)), \gamma(v))$ redundant

Definition 8
Feasible Reconciliation

The reconciliation is time-feasible if the graph G does not contain any directed cycle. Intuitively the constraints are given by the following facts:

- From the ancestry induced by the parasite tree we know that u' happened strictly after u.
- The contemporary of $\gamma(u)$ and $\gamma(v)$ is induced by the transfer (u, v)
- Everything that has happened before $\gamma(u)$ has happened before $\gamma(v)$.

We are finally able to formally define the optimization problem which will be discussed in the next chapter. Observe also that this problem belongs to the family of enumeration problems, defined in Section 1.1.

Problem 1 *All-MPR problem*

All Most Parsimonious Reconciliation problem: Given two phylogentic trees H and P and a vector of costs $\langle c_c, c_d, c_s, c_l \rangle$, generate all reconciliations of minimum cost.

Chapter 2

Eucalypt

Although the All-MPR problem was already treated in the available literature, and the DTL model often cited in the biological papers dealing with co-evolution, there wasn't a tool that solves such problem in a complete and efficient way. We thus developed and implemented a new one, called Eucalypt , with this purpose in mind. This not only provides a novel and usable software for co-phylogeny reconstruction but also allows to investigate how the event-based model performs in practice in terms of the number and quality of the solutions obtained. We compared our method to the available software that are somewhat similar and we tested it on many real datasets. By looking at the results obtained, some interesting considerations about the advantages and disadvantages of the commonly accepted mathematical model could be drawn. Finally, we introduced a new version of the problem where the host-switches are distance bounded: the k-bounded-All-MPR problem. Eucalypt solves both problems in polynomial delay. Eucalypt is available at `http://eucalypt.gforge.inria.fr/`.

In the context of a DTL reconciliation, there are two main issues that must be taken into account. The first is time-feasibility and the second the multiplicity of the solutions. Providing a single optimal solution is not a good option since, even when a parsimony filter is applied, an exponential number of optimal

reconciliations is possible. Thus, even when restricting to time-feasible solutions, this number can remain huge. For this reason, the capacity to enumerate all optimal solutions becomes a crucial given that an *a posteriori* validation of which among the equivalent solutions is relevant requires further input and a biological expertise.

The reconciliation algorithms that try to deal with more than one optimal solution are CoRe-Pa (Merkle, Middendorf, and Wieseke, 2010) Mowgli (Doyon et al., 2011a), Jane 4 (Conow et al., 2010), Notung (Stolzer et al., 2012), and Ranger-DTL (Bansal, Alm, and Kellis, 2013). However, Mowgli assumes that the host and parasite trees are fully dated and computes just the number of optimal reconciliations without generating them. Jane 4 uses a heuristic based on a genetic algorithm for finding one or a number of solutions (not all and not necessarily of optimal cost). Ranger-DTL can handle both dated and undated trees and can compute the total number of optimal reconciliations. However, the currently available version of Ranger-DTL outputs only one optimal reconciliation. CoRe-Pa and Notung are the only publicly available algorithms that claim to generate all optimal reconciliations. However for most instances, CoRe-Pa enumerates only a proper subset of all optimal solutions. Notung was designed for a more general event model that includes duplications, losses, transfers, and incomplete lineage sorting (ILS). In particular, the DTL model is a special case when the species tree is binary. However, the algorithm imposes some restrictions on the cost values. Indeed, the cost of a co-speciation is always assumed to be equal to 0 and the cost of a loss positive.

We provided an algorithm that, given a cost model for the events, efficiently generates all the optimal solutions for the reconciliation problem. It is also possible for the algorithm to generate only optimal reconciliations that are time-feasible. Eucalypt requires no assumption concerning the cost values: it thus allows negative ones while co-speciation and loss may have any arbitrary cost. In addition, the algorithm can efficiently handle distance-bounded host-switches, *i.e.* cases where the host-switches are allowed to happen only between species that are within some fixed distance along the host tree. Observe that this is not an artificial requirement. The significance of a host-switch distance has already been pointed out in several studies (Vienne, Giraud, and Skyhoff, 2007; Poulin and Mouillot, 2003). Indeed, if parasites had switched only between closely related hosts, this would lead to a higher degree of congruence between the

parasite and host trees. When this information is available, it should thus be taken into account in the reconciliation process. Moreover, it can happen that for some datasets and cost vectors, there is no optimal time-feasible solution. One way to overcome this problem can be by varying the length of the furthest allowed switch until at least one time-feasible solution is obtained. On the contrary, in the case where the number of optimal time-feasible reconciliations is high, one can decrease their number by selecting a subset of them. This can be done by decreasing the value of the maximum allowed distance of a switch while still maintaining the same optimal total cost. The complexity of the bounded-switch problem remains open. It could be that this constraint makes the reconciliation problem solvable in polynomial time, which in turn is of both theoretical and practical interest.

We showed that the algorithm we developed , EUCALYPT (For "EnUmerator of Co-evolutionary Associations in PoLYnomial-Time delay" with one switch, of "P" before "LY"), loses no optimal solution, and is able to list all of them in linear-time delay: the time required for getting from one solution to the next one is indeed $O(m)$ for m the number of vertices in the parasite tree, while finding the first solution requires $O(n^3m)$ time for n the number of vertices in the host tree. The space complexity for the whole enumeration process is also $O(n^3m)$.

2.1 Method

Finding one solution.

EUCALYPT uses a dynamic programming approach to find one or to enumerate all optimal reconciliations. In this approach, each (p, h) cell of the m by n dynamic programming matrix, let us denote it by D, contains a single (real or integer) number that represents the cost of an optimal (sub-)reconciliation mapping vertex p in the parasite tree to vertex h in the host tree. The matrix is filled following a post-order traversal of P and H.

Bansal *et al.* provided an algorithm that finds the cost of one optimal reconciliation in time and space $O(nm)$ (Bansal, Alm, and Kellis, 2012). We adapted the algorithm for solving the more general *k-bounded-All-MPR problem*. More precisely, let k be the maximum allowed host-switch distance. Adding to the dynamic programming procedure a test for checking the distance of a host-switch, we obtain an algorithm whose time complexity is $O(nm2^k)$. However, for a

constant value of k, this complexity remains in $O(nm)$. Observe that if we do not require the k bound on the distance of the switches, one could easily replace the algorithm by the theoretically faster method of Bansal *et al.*

Finding one optimal solution requires keeping trace of a path in the matrix leading to the minimum cost. This is easily done by keeping in each $D(p,h)$ cell not only the cost associated to it but a pair of pointers to one mapping for the children p_1, p_2 of p having led to such cost.

As for Bansal *et al.* (Bansal, Alm, and Kellis, 2012), we make use of some auxiliary structures apart from the dynamic programming matrix D. We thus use another matrix of size m by n that will contain the optimal solutions of the subtrees and that we denote by D_{ST}. Formally, $D_{ST}(p,h)$ holds the cost of an optimal solution in which p is mapped to some vertex i in the host subtree rooted in h. We also use two vectors. One is denoted by $Switch_k$ gives, for each vertex h in H the set of vertices that are incomparable with h and are at a distance at most k from it. Formally, we have that $Switch_k(h) = \{g \in V(H) | d(h,g) \leq k \wedge lca(h,g) \notin \{h,g\}\}$ where $d(h,g)$ is the distance of vertex h to vertex g in H. The second vector, denoted by $cost_{switch}$, holds the cost of a switch to a given vertex h in H. The pseudocode for this procedure is given in Algorithm 1.

Enumerating all optimal solutions.

To enumerate all solutions, we need to keep more information. This can be done using $O(n^3m)$ space instead of $O(nm)$. Consider a cell $c = D(p,h)$ of the dynamic programming matrix D. Besides the numerical value corresponding to the cost of an optimal solution obtained by mapping p to h, the cell now also contains a list of pairs of pointers, one to each of the mappings of the children p_1 and p_2 of p having led to the cost of an optimal sub-solution that mapped p to h. Clearly the size of such a list is $O(n^2)$ in the worst case. The set of all pointers for D naturally form a DAG-like structure that is driven by the topology of the parasite tree. Figure 2.1 shows the information contained in cell $c = D(p,h)$ of the matrix (left side). This may be also visualised in the form of a local tree (right side of the same figure) with the parent vertex c as the root which corresponds to the mapping of p to h (denoted in the figure by $p : h$) and one child for each alternative solution leading to that mapping (rectangle vertices in the figure). Each such alternative solution in turn corresponds to a pair of pointers, to two circle vertices which represent, in each case, a different pair of

Algorithm 1 Finding the cost of an optimal solution

Input: $< H, T, \phi >$ and a cost vector $\langle c_c, c_d, c_s, c_l \rangle$
Output: Optimal cost of the *k-bounded-All-MPR problem*

for $p \in V(P)$ and $h \in V(H)$ **do**
  Initialise $D(p,h), D_{ST}(p,h)$ to ∞
  Compute $Switch_k(h)$
end for
for $l \in L(P)$ **do**
  Initialise $D(l, \phi(l)) = 0$
  for $a \in Anc(\phi(l))$ **do**
    $D_{ST}(l,a) = c_l * d(a, \phi(l))$
  end for
end for
for $p \in V(P) - L(P)$ in post-order with children p_1, p_2 **do**
  for $h \in V(H)$ in post-order with children h_1, h_2 **do**
    if $h \in L(H)$ **then**
      $\delta_d \leftarrow c_d + c(p_1,h) + c(p_2,h)$
      for $g \in Switch_k(h)$ **do**
        $cost_{switch}(g) = c_s + min\{D(p_1,g) + D_{ST}(p_2,h), D(p_2,g) + D_{ST}(p_1,h)\}$
      end for
      $\delta_s \leftarrow min\{cost_{switch}(g) | g \in Switch_k(h)\}$
      $D(p,h) = min\{\delta_d, \delta_s\}$
      $D_{ST}(p,h) = D(p,h)$
    else
      $\delta_c \leftarrow min\{(c_c + D_{ST}(p_1,h_1) + D_{ST}(p_2,h_2)), (c_c + D_{ST}(p_1,h_2) + D_{ST}(p_2,h_1))\}$
      $$\delta_d \leftarrow min \begin{cases} D(p_1,h) + D(p_2,h) \\ D(p_1,h) + D_{ST}(p_2,h_1) + c_l \\ D(p_1,h) + D_{ST}(p_2,h_2) + c_l \\ D(p_2,h) + D_{ST}(p_1,h_1) + c_l \\ D(p_2,h) + D_{ST}(p_1,h_2) + c_l \\ D_{ST}(p_1,h_1) + D_{ST}(p_2,h_1) + 2c_l \\ D_{ST}(p_1,h_2) + D_{ST}(p_2,h_2) + 2c_l \end{cases}$$
      for $g \in Switch_k(h)$ **do**
        $cost_{swicth}(g) = c_s + min\{D(p_1,g) + D_{ST}(p_2,h), D(p_2,g) + D_{ST}(p_1,h)\}$
      end for
      $\delta_s \leftarrow min\{cost_{switch}(g) | g \in Switch_k(h)\}$
      $D(p,h) = min\{\delta_c, \delta_d, \delta_s\}.$
      $D_{ST}(p,h) = min\{D(p,h), c_l + D_{ST}(p,h_1), c_l + D_{ST}(p,h_2)\}$
    end if
  end for
end for
return $min\{D(r(P),h) | h \in V(H)\}$

mappings of the children p_1 and p_2 of p which is equivalent in terms of cost (and is optimal). The circle vertices thus correspond to other cells of the matrix D which contain a similar local tree. Notice that more than one sub-solution may refer to a same mapping as indicated in Figure 2.2, thus leading to a DAG structure when the set of all solutions is considered and representing a compact structure for containing them all. Once built during the first pass over D, this DAG is then visited in pre-order to iteratively extract each such solution in turn.

A pseudo-code for enumerating all solutions is given in Algorithm 2. We use an additional stack M in order to select which sub-solutions (local sub-trees) to add to the reconciliation that is currently being built. This stack is filled with couples of the form $\langle cell, index \rangle$). The function $M(cell)$ returns, in constant time, the couple $\langle cell, index \rangle$ at the top of M, if M is not empty.

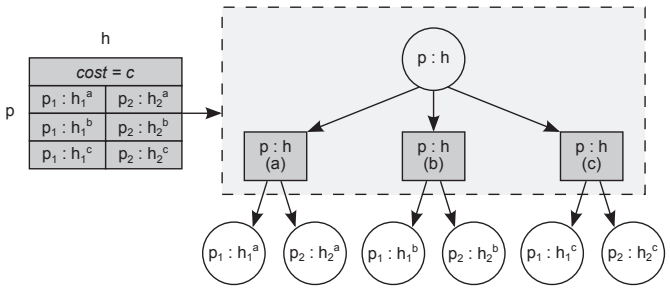


Figure 2.1: Schematic representation of the content of a cell in the dynamic programming matrix: Suppose the cell is related to the association $p : h$ and let $p1$,$p2$ be the two children of p. One single cell-root vertex is created to represent the association p:h (the circular vertex in the picture). This association has a local minimum cost that can be obtained in different ways, that is choosing different associations for $p1$ and $p2$. Each equivalent alternative is represented by a vertex (squared in the picture). The number of alternatives is variable. Each squared vertex has exactly two children corresponding to the associations of $p1$ and $p2$ respectively.

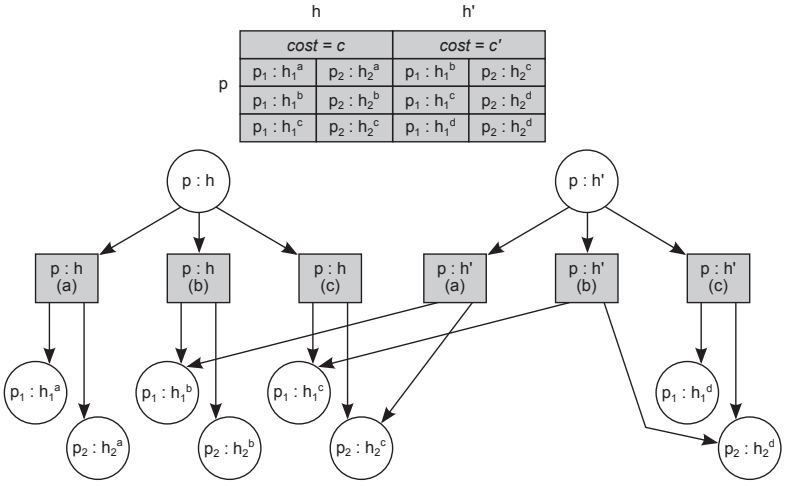


Figure 2.2: The tree structure allows us to save the information in an efficient way. Each sub-solution corresponds to a subtree and there is no need to double it each time it appears in a solution. In particular only one vertex is created for each association and if two different alternatives share this association the respective (square) vertices will point exactly at the same (circular) vertex.

Algorithm 2 Enumerating all optimal solutions

Input: The dynamic programming matrix D
Output: All optimal solutions

for All cells *root* in D containing an optimal mapping of $r(P)$ (or the unique cell mapping $r(P)$ to $r(H)$) **do**
  $currentCell \leftarrow root$
  A stack $M \leftarrow [\emptyset]$ (to be filled with couples of the form $\langle cell, index \rangle$)
  do
    while $currentCell$!= null **do**
      if $|List(current)| \geq 1$ **then**
        //There are different sub-solutions for this mapping
        if $M(currentCell)$ is not in M **then**
          Push($\langle currentCell, 0 \rangle$) in M
          $currentSubsolution \leftarrow 0^{th}$-element of $M(currentCell)$;
        else if $M(currentCell)$ is the last element of M **then**
          //In the final part of the solution I pass to consider the next option
          Pop($\langle currentCell, i \rangle$) from M
          Push($\langle currentCell, i+1 \rangle$;) in M
          currentSubsolution $\leftarrow (i+1)th$-element of $M(currentCell)$
        else
          //In the first part of the current solution, the mappings are the same as for the previous one
          $\langle cell, index \rangle \leftarrow M(currentCell)$
          $currentSubsolution \leftarrow index^{th}$ element of $M(currentCell)$
        end if
      else
        //There is a unique possible sub-solution
        Add to the solution the mapping relative to $currentCell$
        $currentSubsolution \leftarrow 0th$-element of $List(currentCell)$
        //currentSubsolution is unique (or null if the vertex is a leaf.)
        $currentCell$ = the next vertex following the pointers of $currentSubsolution$ (in post-order)
      end if
      Output the solution
      Pop from M until the first couple $\langle s, i \rangle$ is found for which $i < |M(s)| - 1$ and the stack is not empty
    end while
  while M is not empty
end for

Complexity analysis.

The space complexity of EUCALYPT is $O(n^3m)$. For each of the mn steps of the dynamic programming procedure, we create at most n^2 objects. All the additional structures used to iterate over this matrix have size $O(n)$. To evaluate the time complexity of the whole enumeration process, we separate the time needed for filling matrix D, and the time for traversing it in order to produce a single solution, or to enumerate all of them. The number of steps needed for filling the matrix the first time is $O(n^3m)$ because each cell may contain, in the worst case, a list of n^2 pairs of pointers. Since the height of the DAG is bounded by $2m$, it can be traversed using only a linear size support structure. Moreover, at each iteration that leads to one solution, a subtree of size $2m$ of the DAG is visited. In particular, each time we are visiting a parent vertex, we need to add its mapping to the current solution and then follow the DAG looking for the mappings of its two children. An entire solution (which is composed by m mappings) is complete when at most 2 vertices (one parent and one sub-solution) have been visited for each vertex of P. This guarantees that once we produce

the first solution, obtaining each one of the others in turn requires only linear time and linear space.

Finally, it is possible to enumerate only time-feasible solutions. To this purpose we have implemented a time-feasibility test defined in Stolzer et al., 2012 which has a time complexity of $O(n^2)$.

2.2 Experimental results.

We applied EUCALYPT to a number of host-parasite trees available in the literature, and to our own set of interest. We show that in general many optimal solutions exist. Indeed, as already noticed in other studies (see *e.g.* (Bansal, Alm, and Kellis, 2013)), the number can sometimes be huge. We give an example where, for two pairs of host-parasite trees having each less than 41 leaves, the number of solutions is 5120, even when only time-feasible solutions are kept. Depending on the cost vector, this may increase considerably (for the same example, to 4080384). EUCALYPT indeed comes with a procedure (in $O(n^3)$ time) for testing the time feasibility of a solution. The possibility to calculate the number of solutions without explicitly listing them all was also integrated in EUCALYPT. This has the same complexity as to enumerate a single solution. Finally, to facilitate interpreting the results even when a huge number of solutions is observed, the latter are classified in terms of the number of each event (co-speciation, duplication, loss, or host-switch) that they contain. This often reduces considerably the number of different classes of solutions that must be examined further, but the number may remain high (for instance 275) depending on the cost vector.

Datasets

To test EUCALYPT , we selected 12 datasets from the literature. As we are mostly interested in host-parasite systems, the first 10 datasets concern such relations: EC - Encyrtidae & Coccidae (Deng et al., 2013), GL - Gopher & Lice (Hafner and Nadler, 1988), SC - Seabirds & Chewing Lice (Paterson, Palma, and Gray, 2003), RP - Rodents & Pinworms Hugot, 2003, SCF - Smut Fungi & Caryophillaceus plants (Refrégier et al., 2008), PLML - Pelican Lice ML (Hughes et al., 2007) (the trees are generated through a maximum likelihood approach), PLMP - Pelican Lice MP (Hughes et al., 2007) (the trees are generated through a maximum

parsimony approach), RH - Rodents & Hantaviruses (Ramsden, Holmes, and Charleston, 2009), PP - Primates & Pinworm (Hugot, 1999), and FD - Fishs and Dactylogyrus (Balbuena, Míguez-Lozano, and Blasco-Costa, 2013).

In addition, we used a dataset of our own which corresponds to arthropod hosts and a bacterium genus, *Wolbachia*, living inside the cells of their hosts (Simões et al., 2011; Simões, 2012). The datasets were chosen to provide a variety in terms of size of the host and parasite trees: those from the literature are relatively small (from 7 to 100 leaves), while our own data provide an example of much bigger host and parasite trees, each having 387 leaves. Moreover, we were careful that the selected datasets cover, as much as possible, a range of situations in terms of co-evolution and of the expected frequencies of each event. Finally, since Eucalypt can be applied to any type of datasets compatible with the model, we also tested it on a genes-species dataset from (David and Alm, 2011) that had previously been used by (Bansal, Alm, and Kellis, 2013). The dataset has 3983 unrooted gene trees.

To be able to run our algorithm on this dataset, we rooted the trees using an approach similar to (Bansal, Alm, and Kellis, 2013): for each possible rooted version of the unrooted gene tree, we consider the optimal cost of the reconciliation and choose the rooting that has minimum cost among all. We only show the results concerning 4 datasets: COG2085, COG3715, COG4964, COG4965. The choice of these datasets is motivated by the fact that they show different behaviour, in particular as concerns the *k-bounded-All-MPR* problem.

The datasets from the literature are given in Table 2.1, together with the number of leaves in the host and parasite trees.

Comparison with CoRe-Pa and Notung

In Table 2.1, we compare Eucalypt to CoRe-Pa and Notung for some cost vectors that commonly appear in the literature.

As mentioned in the introduction, CoRe-Pa does not always enumerate all solutions, as shown in Table 2.1. In the Additional File 1, we give an explicit example from one of the datasets used ("Smut Fungi & Caryophillaceus plants" (Refrégier et al., 2008) with cost vector $\langle 0, 1, 1, 1 \rangle$) where indeed Eucalypt finds more (correct) solutions, some of which are acyclic, that are time-feasible. The same result is observed for other datasets (examples not shown).

Notice that sometimes when enumerating optimal reconciliations, CoRe-Pa outputs the same one more than once. In all our tests, we were never able to obtain more than 1000 different reconciliations with CoRe-Pa. This explains the presence of the value 1000, more than once, in Table 2.1 and Table 2.2 (each time indicated by a *) while Eucalypt for the same datasets and cost vectors finds many more solutions.

Notung generates only time-feasible reconciliations and their number coincides with the result of Eucalypt for all the datasets used. However, Notung imposes some restrictions on the cost values. Indeed, the cost of a co-speciation is always assumed to be equal to 0 and the cost of a loss positive.

Non positive cost vectors

No assumption needs to be made by Eucalypt concerning the cost values: it thus allows negative ones while co-speciation and loss may have any arbitrary cost. As already mentioned, in this case we can compare it only with CoRe-Pa. The results of these experiments are presented in Table 2.2. In almost all of the cases, CoRe-Pa does correctly determine the total number of (un)feasible optimal solutions.

Results of Eucalypt and discussion

First, we observe that when the size of the tree increases, in most cases the total number of optimal solutions also increases. However, this does not hold for the number of time-feasible optimal solutions. For instance, according to the results given in Table 2.1, for the dataset EC having 7 and 10 leaves, the number of time-feasible optimal solutions is much higher than for the case of dataset COG4964 (having 100 and 27 leaves). Even for the same dataset, this number can be reduced significantly depending on the cost vector. In particular when the cost of the losses is 0, the number of optimal solutions can be huge even for relatively small datasets, such as for example FD (20-51 leaves). This makes it practically impossible to check the time-feasibility of each of them (this explains the presence of a * in some cells of Tables 2.1 and 2.2). Thus, it seems that the cost vector and the topology of the trees together with the mapping of the leaves play a more important role in the total number of time-feasible solutions.

We also tested Eucalypt on the much bigger trees of *Wolbachia* and the

arthropods. Due to limitations in space and time, we could not enumerate all optimal solutions because their number is huge: 1.01×10^{47} for cost vector $\langle -1, 1, 1, 1 \rangle$, 3.87×10^{136} for cost vector $\langle 0, 1, 1, 0 \rangle$, 3.19×10^{48} for cost vector $\langle 0, 1, 1, 1 \rangle$, and finally 1.01×10^{47} for cost vector $\langle 0, 1, 2, 1 \rangle$. We did however enumerate optimal solutions until one was produced that was found acyclic.

For the cost vector $\langle 0, 1, 1, 1 \rangle$, the first produced optimal solution was already acyclic as were those that were enumerated next, hinting to the possibility that the proportion of acyclic solutions is high among all optimal ones. For the remaining cost vectors, the initial optimal solutions enumerated by Eucalypt were indeed all cyclic, and given their number and the time required by each acyclicity test, we stopped the process of checking after one week. Two cases are then possible: either there are no acyclic solutions meaning optimal ones have a higher cost; or the proportion of acyclic solutions is low among all optimal ones.

The results confirm once again that the number of optimal reconciliations can be huge. Moreover, the problem remains even if we restrict the results to only time-feasible solutions. In order to deal with this huge set of solutions, we propose to group them in classes depending on the number of events observed. As shown in Tables 2.1 and 2.2, the number of classes in a set of time-feasible optimal solutions is significantly smaller compared to the size of the set itself.

For instance, for dataset EC and vector $\langle 0, 1, 1, 1 \rangle$, the 10 optimal time-feasible solutions are split in 5 classes $\langle \#_c, \#_d, \#_s, \#_l \rangle$ as follows: 1 for $\langle 4, 1, 4, 1 \rangle$, 3 for $\langle 4, 0, 5, 1 \rangle$, 2 for $\langle 5, 0, 4, 2 \rangle$, 2 for $\langle 3, 1, 5, 0 \rangle$ and, 2 for $\langle 3, 0, 6, 0 \rangle$. For dataset RP and vector $\langle 0, 1, 1, 1 \rangle$, the 16 optimal time-feasible solutions are split in 3 classes: 4 for $\langle 7, 0, 5, 3 \rangle$, 2 for $\langle 4, 0, 5, 1 \rangle$ and, 10 for $\langle 6, 0, 6, 2 \rangle$. Even more interestingly, for dataset SFC and vector $\langle 0, 1, 1, 1 \rangle$, the 144 optimal time-feasible solutions belong to a unique class: $\langle 4, 0, 11, 0 \rangle$. For more details on this type of analysis, we refer to Additional File 1.

Finally, we want to call attention to the fact that, in many cases of datasets and cost vectors, there is no optimal solution that is time-feasible. Indeed, in the case of the datasets from (David and Alm, 2011), from 3983 trees we choose 429 with between 20 and 50 leaves for which the total number of optimal reconciliations is less than 10000. Among these 429 trees (rooted according to the one that leads to a minimum total cost of the reconciliation) and using the vector $\langle 0, 2, 3, 1 \rangle$, 233 (*i.e.* more than half) have no time-feasible solutions. To deal with this problem, we consider the restriction when the host-switches are

constrained to have bounded distance.

k-bounded-All-MPR problem

The main concern of this section is to discuss the effect of bounding the distance of the host-switch events. The variables that will be defined here must be considered relative to a fixed dataset and a fixed cost vector. We denote by $S(k)$ the set of optimal solutions obtained when the maximum distance allowed for a host-switch is k, and denote by opt_k their cost. We also denote by opt^* the optimum cost of an acyclic reconciliation (without any bound on the host-switch distance): observe that this value is in general NP-hard to determine. Clearly, if $S(\infty)$ contains at least one time-feasible solution then $opt_\infty = opt^*$. However, this is not always the case (see Table 2.1 for some examples): let us then consider the case where $opt_\infty < opt^*$. We are now interested in finding an upper bound on the value of opt^* by making use of the possibility given by EUCALYPT to limit the distance of switches. To this purpose, we define k_A as the biggest value of k for which $S(k)$ contains at least one acyclic solution (in general, $opt_\infty < opt^* \leq opt_{k_A}$): observe that in the case of integer cost values, when $opt_\infty - opt_{k_A} = 1$, we have that opt_{k_A} coincides with opt^*. We can determine k_A as follows: for every optimal reconciliation in $S(\infty)$, we keep track of the longest distance observed for a switch and denote by k_{start} the smallest value observed among all optimal solutions; then starting from k_{start}, we decrement this value until at least one time-feasible solution is found. It is interesting to note that k_A is always close to the starting value k_{start}, and that it frequently happens that $opt_\infty - opt_{k_A} = 1$. This shows that the method is efficient in practice as we do not have to check too many values before finding time-feasible optimal solutions. In particular, we applied this idea to some cases where no time-feasible solutions were found: the results are shown in Table 2.4.

Even in the case where $S(\infty)$ contains already some time-feasible solutions, bounding the distance of the switches remains interesting because the number of such solutions can be too large to handle. The basic idea is to choose a value k', with $\infty > k' \geq k_{start}$, and to focus our attention only on $S(k')$: in this way, the optimal cost of the solutions is preserved and $|S(k')| \leq |S(\infty)|$. In real situations, choosing k' must be driven by some *ad hoc* biological consideration: however, in our case, we decided to fix k' to the value that is nearest to k_{start} for which the optimal cost does not change and the number of time-feasible solutions is

strictly positive. Some results are shown in Table 2.3: it is worth observing that k' always either coincides or is only a few steps away from k_{start}, which again shows the efficiency of the method in practice.

2.3 Conclusions and prespectives

Our polynomial delay algorithm, together with its implementation, provides a usable and reliable software for co-phylogeny reconstruction. The comparison to other similar software in terms of running time and, more importantly, of completeness of the solution, shows that Eucalypt should be preferred for the analysis of real datasets. Its properties indeed allows to better explore the behaviour of the event-based model. All the results of this study point to the necessity of introducing new criteria besides parsimony in the model for an optimal reconciliation. The idea of imposing an evolutionary distance to the host-switches is one possible criterion when it can be justified from a biological point of view.

As a future work we would like to enrich the model, considering other types of information, such as for instance geographic, which might enable to indicate that certain mappings of internal vertices are impossible. One consequent improvement of Eucalypt will therefore be to allow the user to indicate that certain associations of internal vertices should be forbidden.

A few mathematical questions remain open in this area. The approximability of finding an optimum acyclic solution to the co-phylogeny reconstruction problem is still unknown. Moreover, the k-bounded-All-MPR problem has not yet been analysed from a complexity point of view. Observe that, since for $k = 2$ the problem becomes polynomial, it would be interesting to investigate the complexity for a fixed k.

Dataset	Leaves		Cost Vector	Reconciliations						
				CoRe-Pa			Eucalypt			
	H	P		#T	#C	#A	#T	#C	#A	#CA
EC	7	10	$\langle 0,1,1,1\rangle$	16	6	10	16	6	10	5
			$\langle 0,1,2,1\rangle$	**14**	0	**14**	18	0	18	6
			$\langle 0,2,3,1\rangle$	**12**	0	**12**	16	0	16	4
GL	8	10	$\langle 0,1,1,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,1,2,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,2,3,1\rangle$	2	0	2	2	0	2	1
SC	11	14	$\langle 0,1,1,1\rangle$	1	0	1	1	0	1	1
			$\langle 0,1,2,1\rangle$	1	0	1	1	0	1	1
			$\langle 0,2,3,1\rangle$	1	0	1	1	0	1	1
RP	13	13	$\langle 0,1,1,1\rangle$	18	2	16	18	2	16	3
			$\langle 0,1,2,1\rangle$	3	1	2	3	1	2	1
			$\langle 0,2,3,1\rangle$	3	1	2	3	1	2	1
SFC	15	16	$\langle 0,1,1,1\rangle$	184	40	144	184	40	144	1
			$\langle 0,1,2,1\rangle$	40	40	0	40	40	0	0
			$\langle 0,2,3,1\rangle$	40	40	0	40	40	0	0
PLML	18	18	$\langle 0,1,1,1\rangle$	**158**	0	**158**	180	0	180	4
			$\langle 0,1,2,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,2,3,1\rangle$	11	0	11	11	0	11	2
PLMP	18	18	$\langle 0,1,1,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,1,2,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,2,3,1\rangle$	**17**	0	**17**	18	0	18	2
RH	34	42	$\langle 0,1,1,1\rangle$	**32**	0	**32**	42	0	42	4
			$\langle 0,1,2,1\rangle$	**158**	**158**	0	2208	2208	0	0
			$\langle 0,2,3,1\rangle$	**22**	**22**	0	288	288	0	0
PP	36	41	$\langle 0,1,1,1\rangle$	**1000***	0	**1000**	5120	0	5120	4
			$\langle 0,1,2,1\rangle$	**11**	0	**11**	72	0	72	2
			$\langle 0,2,3,1\rangle$	**11**	0	**11**	72	0	72	2
FD	20	51	$\langle 0,1,1,1\rangle$	**1000***	**282**	**718**	25184	1792	23392	11
			$\langle 0,1,2,1\rangle$	**108**	**44**	**64**	408	132	276	5
			$\langle 0,2,3,1\rangle$	**22**	**22**	0	80	80	0	0
COG2085	100	44	$\langle 0,1,1,1\rangle$	**1000***	**0**	**1000**	44544	2304	42240	3
			$\langle 0,1,2,1\rangle$	**1000***	**0**	**1000**	37568	480	37088	7
			$\langle 0,2,3,1\rangle$	**888**	0	**888**	46656	0	46656	4
COG3715	100	40	$\langle 0,1,1,1\rangle$	**1000***	**1000**	**0**	1172598	1155958	16640	6
			$\langle 0,1,2,1\rangle$	9	9	0	9	9	0	0
			$\langle 0,2,3,1\rangle$	**13**	**13**	0	33	33	0	0
COG4964	100	27	$\langle 0,1,1,1\rangle$	**85**	**85**	0	224	224	0	0
			$\langle 0,1,2,1\rangle$	**13**	**13**	0	36	36	0	0
			$\langle 0,2,3,1\rangle$	**17**	**17**	0	54	54	0	0
COG4965	100	30	$\langle 0,1,1,1\rangle$	**1000***	**408**	**592**	17408	5632	11776	2
			$\langle 0,1,2,1\rangle$	**141**	0	**141**	640	0	640	2
			$\langle 0,2,3,1\rangle$	**1000***	**276**	**724**	6528	1408	5120	2

Table 2.1: Number of solutions found by each one of the programs CoRe-Pa, Notung and Eucalypt for each dataset and each cost vector $\langle c_c, c_d, c_s, c_l\rangle$. For Eucalypt and CoRe-Pa the columns represent: #T = total number of optimal solutions, #C = total number of cyclic solutions and #A = total number of acyclic solutions. In all cases #A is always equal for both Notung and Eucalypt. For Eucalypt the column #CA denotes the number of event classes in the set of acyclic solutions. CoRe-Pa limits to 1000 the total number of enumerated solutions and these cases are denoted by the symbol $*$

Dataset	Leaves		Cost Vector	Reconciliations						
				CoRe-Pa			Eucalypt			
	H	P		#T	#C	#A	#T	#C	#A	#CA
EC	7	10	$\langle -1,1,1,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,1,1,0\rangle$	**18**	0	**18**	24	0	24	8
GL	8	10	$\langle -1,1,1,1\rangle$	2	0	2	2	0	2	1
			$\langle 0,1,1,0\rangle$	12	0	12	12	0	12	5
SC	11	14	$\langle -1,1,1,1\rangle$	1	0	0	1	0	1	1
			$\langle 0,1,1,0\rangle$	**82**	2	**80**	113	3	110	18
RP	13	13	$\langle -1,1,1,1\rangle$	3	1	2	3	1	2	1
			$\langle 0,1,1,0\rangle$	**69**	**25**	**44**	117	45	72	29
SFC	15	16	$\langle -1,1,1,1\rangle$	40	40	0	40	40	0	0
			$\langle 0,1,1,0\rangle$	**1000***	**741**	**259**	6332	5069	1263	81
PLML	18	18	$\langle -1,1,1,1\rangle$	2	0	0	2	0	2	1
			$\langle 0,1,1,0\rangle$	**45**	**2**	**43**	448	28	420	16
PLMP	18	18	$\langle -1,1,1,1\rangle$	2	0	0	2	0	2	1
			$\langle 0,1,1,0\rangle$	**147**	0	**147**	262	0	262	34
RH	34	42	$\langle -1,1,1,1\rangle$	**197**	**197**	0	1056	1056	0	0
			$\langle 0,1,1,0\rangle$	**1000***	**0**	**1000**	4080384	310284	3770100	275
PP	36	41	$\langle -1,1,1,1\rangle$	**17**	0	**17**	144	0	144	2
			$\langle 0,1,1,0\rangle$	**182**	**8**	**174**	498960	55440	443520	129
FD	20	51	$\langle -1,1,1,1\rangle$	**196**	**86**	**110**	944	368	576	7
			$\langle 0,1,1,0\rangle$	**1000***	**1000**	**0**	1.5×10^{15}	*	*	*
COG2085	100	44	$\langle -1,1,1,1\rangle$	**1000***	**0**	**1000**	109056	26496	82560	3
			$\langle 0,1,1,0\rangle$	**1000***	**0**	**1000**	3.5×10^{11}	*	*	*
COG3715	100	40	$\langle -1,1,1,1\rangle$	**869**	**869**	0	63360	63360	0	0
			$\langle 0,1,1,0\rangle$	**1000***	**0**	**1000**	1.2×10^{12}	*	*	*
COG4964	100	27	$\langle -1,1,1,1\rangle$	**13**	**13**	0	36	36	0	0
			$\langle 0,1,1,0\rangle$	**1000***	**0**	**1000**	8586842	2603598	5983244	300
COG4965	100	30	$\langle -1,1,1,1\rangle$	**1000***	**335**	**665**	44800	13312	31488	5
			$\langle 0,1,1,0\rangle$	**1000***	**0**	**1000**	907176	387192	519984	208

Table 2.2: Number of solutions found by the programs CoRe-Pa and Eucalypt for each dataset and each cost vector $\langle c_c, c_d, c_s, c_l\rangle$. The columns represent: #T = total number of optimal solutions, #C = total number of cyclic solutions and #A = total number of acyclic solutions. For Eucalypt the column #CA denotes the number of event classes in the set of acyclic solutions. CoRe-Pa limits to 1000 the total number of enumerated solutions and these cases are denoted by the symbol $*$.

Dataset	Costvector								
	$\langle -1,1,1,1\rangle$			$\langle 0,1,2,1\rangle$			$\langle 0,2,3,1\rangle$		
	k/k'	$\#AC/\#AC'$	$\#T/\#T'$	k/k'	$\#AC/\#AC'$	$\#T/\#T'$	k/k'	$\#AC/\#AC'$	$\#T/\#T'$
EC	3/3	2/2	2/2	3/3	18/16	18/16	3/3	16/16	16/16
GL	4/4	2/2	2/2	4/4	2/2	2/2	4/4	2/2	2/2
SC	6/6	1/1	1/1	6/6	1/1	1/1	6/6	1/1	1/1
RP	9/9	2/2	3/3	8/8	2/2	8/8	2/2	2/2	3/3
PMP	6/6	2/2	2/2	6/6	2/2	2/2	5/5	11/4	11/4
PML	5/5	2/2	2/2	5/5	2/2	2/2	3/3	18/6	18/6
PP	4/4	144/96	144/96	4/4	72/48	72/48	4/4	72/48	72/48
FD	9/10	576/240	944/512	9/9	276/4	408/8	–	–	–
COG2085	14/14	82560/9408	109056/9408	14/14	37088/4032	37568/4032	14/14	46656/5184	46656/5184
COG4965	16/16	31448/15744	44800/22400	16/16	640/320	640/320	13/16	5120/2560	6528/3328

Table 2.3: For some datasets, the number of optimal time-feasible solutions may be huge when k is unbounded. In some cases, however, by introducing a bound in k we can greatly reduce the number of time-feasible solutions while keeping the optimality of the solutions. For all datasets whose number of acyclic solutions is positive for unbounded k, we identified k_{start} (minimum k whose optimal cost is equal to the optimal cost obtained for unbounded k) and we searched for the minimum $k' \geq k_{start}$ whose number of acyclic solutions is non zero. We executed this procedure for every pair (dataset, cost vector) for which the number of optimal acyclic solutions is positive. In the first column the values for $k = k_{start}$ and k' are given. $\#AC/\#AC'$ denotes the number of optimal acyclic solutions for the case when the switches are unbounded and the case when they are bounded by k', respectively. The same relation is shown for the total number of optimal solutions in the column $\#T/\#T'$.

Dataset	Costvector								
	$\langle -1,1,1,1\rangle$			$\langle 0,1,2,1\rangle$			$\langle 0,2,3,1\rangle$		
	$k_{start} \to k_A$	$o \to o_{k_A}$	$\#A$	$k_{start} \to k_A$	$o \to o_{k_A}$	$\#A$	$k_{start} \to k_A$	$o \to o_{k_A}$	$\#A$
SFC	$7 \to 6$	$6 \to 7$	16	$7 \to 6$	$21 \to 22$	16	$7 \to 5$	$31 \to 35$	12
RH	$6 \to 5$	$8 \to 12$	16	$6 \to 5$	$43 \to 48$	192	$6 \to 5$	$62 \to 68$	48
COG3715	$13 \to 12$	$10 \to 11$	288	$22 \to 6$	$51 \to 176$	6	$22 \to 6$	$80 \to 206$	2
COG4964	$22 \to 4$	$20 \to 208$	30	$13 \to 12$	$33 \to 34$	288	$13 \to 12$	$49 \to 50$	288

Table 2.4: For some datasets (SFC, RH, COG3715 and, COG4964), the number of optimal time-feasible solutions is zero when reconciliations are obtained by using a given cost vector and unbounded k. After identifying k_{start} (minimum k whose optimal cost o is equal to the optimal cost obtained for unbounded k), we decremented k until k_A (maximum k which generates acyclic solutions) is found. For each pair (dataset, cost vector), the following values are given: the decrement of the bound (from k_{start} to k_A), the new optimum found (from o to o_A) and the new number of acyclic solutions ($\#A$).

Chapter 3

Coala

The studies conducted with Eucalypt show that the *most parsimonious scenario* approach presents some limitations that cannot be ignored. A first one is related to the costs that all event-based methods assign to the events in order to apply a parsimony filter. Such costs strongly influence the reconciliations that are inferred. A second issue is that when the phylogenetic trees are too big, the number optimal of solutions is so huge that the reconciliation method may not be useful. To deal with both problems, we developed an algorithm, called Coala , based on an approximate Bayesian computation approach for estimating the frequency of the events. The benefits of this method are twofold: it provides more confidence in the set of costs to be used in a reconciliation, and it allows to estimate the frequency of the events in the cases where a reconciliation method cannot be applied. We present the method in Section3.1, and we then discuss the results obtained on both artificially created and biological datasets in Section 5.2 and 3.3. We provide also an implementation of Coala , available at `http://coala.gforge.inria.fr/`.

A parsimonious solution for reconciling the phylogenetic trees for hosts on one side, and parasites on the other, simply assigns a cost to each of the four types of events and then seeks to minimise the total cost of the mapping. Clearly, the

choice for the cost values is crucial in the solution(s) found. Even if we follow the intuition that the cost of an event should be inversely related to the probability of this event to happen, reasonable cost values for an event-based reconciliation are not easily chosen. It is in fact correct to think that the frequency of the events is not constant across all the situations, and that an event may occur more or less frequently depending on the context. Thus, different pairs of host/parasite phylogenies might be associated to different cost events.

As indicated in Ronquist (2003), if each event is associated to a cost that is inversely related to its likelihood (the more likely is the event, the smaller its cost), then the most parsimonious reconstruction will also, in some sense, be the most likely explanation of the observed data. Likelihood-based approaches should in general be preferred to parsimony-based methods as they remove the subjective step of cost parameter choice and rely instead on a simultaneous inference of parameter values and events. Some attempts in this direction have been done, for instance in testing for co-evolution (Huelsenbeck, Rannala, and Yang, 1997; Huelsenbeck, Rannala, and Larget, 2000). However, the space of possible reconciliations is huge: the reconstruction of histories based on a likelihood approach is thus computationally intensive. A first attempt in this direction appeared in the method in Huelsenbeck, Rannala, and Yang (1997); Huelsenbeck, Rannala, and Larget (2000) which excludes duplications and tends to over-estimate the number of host-switches. In the context of gene/species systems, a Bayesian framework for the reconciliation problem is proposed in Arvestad et al. (2003) but, again the authors' model is not complete as it does not allow for transfers. More recently, (Szöllősi et al., 2013) presented a likelihood method which considers the four types of events but is applied with the objective of reconstructing a species tree starting from multiple gene trees.

The huge space of possible solutions is also an issue, for instance, in population genetics for reconstructing the evolutionary history of a set of individuals. Since the early work of Pritchard et al. (1999), the literature from this domain has seen classical Monte Carlo methods and their variants being replaced by Approximate Bayesian Computation (ABC), a set of more efficient statistical techniques (Beaumont, Zhang, and Balding, 2002). In complex models, likelihood calculation is often unfeasible or computationally prohibitive. ABC methods, also called likelihood-free inference methods, bypass this issue while remaining statistically well-founded (see, for instance, the review of Marin et al. (2012) as

well as the convergence results in Fearnhead and Prangle (2012)).

Following these ideas, we developed an algorithm, called Coala (the name stands for "CO-evolution Assessment by a Likelihood-free Approach", and is also the Portuguese spelling for Koala, the arboreal herbivorous marsupial native to Australia), for estimating the frequency of the events based on a likelihood-free approach. Starting with a prior probability distribution associated to the events (and thus, from a co-evolution model), Coala simulates accordingly the temporal evolution of a set of species (the parasites) following the evolution of another set (the hosts) for which we already have a phylogenetic tree. During such a simulation, Coala generates multi-labelled parasite trees which are then compared to the "known" parasite tree. The ABC principle is to keep the parameter values (event probabilities) giving rise to parasite trees that are "close" to the known one. The output of the algorithm is then a distribution on such parameter values that is a surrogate of the posterior probability for the events which would best explain the observed data.

To the best of our knowledge, the only other method that might be compared to ours is the parameter adaptive approach CoRe-Pa (Merkle, Middendorf, and Wieseke, 2010). In this case, the space of cost vectors is explored either by sampling such vectors at random assuming a uniform distribution model or by using a more sophisticated approach, the so-called Nelder-Head simplex method (Nelder and Mead, 1965). The first appears to be the option by default in CoRe-Pa. In both cases, the function to minimise is the difference between the probabilities directly computed from the cost vector chosen and the actual relative frequencies observed during the reconstruction using such vector. This choice may appear somewhat circular as one would expect that, since reconstruction is driven by the cost vector, the frequency of the events thus reconstructed not only would, but indeed should agree with it.

3.1 Method

General framework

The method we propose relies on an approximate Bayesian computation (ABC). This belongs to a family of likelihood-free Bayesian inference algorithms that attempt to estimate posterior densities for problems where the likelihood is a priori unknown. Given a set of observed data D_0 and starting with a prior

distribution π on the parameter space Θ of the model, the objective is to estimate the parameter values $\theta \in \Theta$ that could lead to the observed data using a Bayesian framework. More precisely, the Bayesian paradigm consists in finding the posterior given D_0 defined as:

$$p(\theta|D_0) = \frac{p(D_0|\theta)\pi(\theta)}{p(D_0)}.$$

If the likelihood function $p(D_0|\theta)$ cannot be derived, then a likelihood-free approximation can be used to estimate this posterior distribution and thus the parameter values. In general, a likelihood-free computation involves a chain of parameter proposals and only accepts a set of parameter values on condition that the model with these values generates data that satisfy a performance criterion with respect to the observed (Sisson, Fan, and M.Tanaka, 2007; Sisson, Fan, and M.Tanaka, 2009). Strict acceptance (or inversely rejection) is based on whether the generated data D_S perfectly matches the observed data D_0. In cases where the probability of perfectly matching the data is very small, a tolerance $d(D_s, D_0) \leq \epsilon$ is adopted to relax the rejection policy, where d is a distance measure. In either case, this is called the *fitting criterion*. Note that this fitting criterion often relies only on a summary statistics instead of the full datasets D_S and D_0. Moreover, for complex models where the prior and posterior densities are believed to be sufficiently different, the acceptance rate is very low and then the use of a likelihood-free Sequential Monte Carlo (SMC) search that involves many iterations leads to a more appropriate strategy. SMC is also preferred among other possible methods as it is flexible, easy to implement, parallelisable and applicable to general settings (Del Moral, Doucet, and Jasra, 2012).

The ABC-SMC algorithms approximate the posterior distribution by using a large set of randomly chosen parameter values. Over sufficiently many iterations and under suitable conditions, the stationary distribution of the Markov chain will approach the distribution of $p(\theta|d(D_S, D_0) \leq \epsilon)$, which will converge to the distribution of the posterior density $p(\theta|D_0)$ if the statistics used to compare the generated data with the real one are sufficient and ϵ is small enough. In our case, the input is a pair of host and parasite phylogenetic trees, denoted by H and P respectively. The observed data is the parasite tree P. The parameter set of the model is composed by the probabilities of each one of four events corresponding to respectively: speciation of the parasite together with a speciation of its host (called *co-speciation*); speciation of the parasite without concomittant speciation

of the host (called *duplication*); switch (also known as jump) of the parasite to another host (called *host-switch*, which is further assumed to be without loss for the original host); and speciation of the host without concomitant speciation of the parasite, and thus loss of the parasite for one of the new host species (called *loss*). We thus have that θ stands for a vector of four probabilities $\langle p_c, p_d, p_s, p_l \rangle$ where $p_c + p_d + p_s + p_l = 1$.

Starting from the host and respecting the probabilities of the events specified in a given parameter set θ_i, we generate M parasite trees, where $M \geq 1$.

Once a parasite tree $\tilde{P}$ is thus simulated, it can be compared to the real parasite tree P by computing a distance between the two. For a given parameter set θ_i, we can then produce a distance summary of the generated trees, and use this as a criterion in the ABC rejection method. The latter selects the parameter set(s) that approximate the observed data within a given tolerance threshold.

The ABC-SMC procedure allows to refine the list of accepted probability vectors by sampling a vector θ_i, introducing a small perturbation to it to produce a vector θ_i', and then collecting a new distance summary for θ_i'.

The list of vectors output in the final step of the algorithm defines the posterior distribution of the co-evolutionary event probabilities for the given pair H and P.

Parasite tree generation algorithm

To simulate the co-evolutionary history of the two input phylogenies and corresponding association of the leaves, we rely on the same event-based model used with Eucalypt .

The evolution of the parasites is simulated by following the evolution of the hosts traversing the phylogenetic tree H from the root to the leaves, and progressively constructing the phylogenetic tree for the parasite. During this process, a single parasite vertex can be in two different states: mapped or unmapped. At the moment of its creation, a new vertex v is unmapped and is assigned a temporary position on an arc a of the host tree H. We denote this position by $\langle v, a \rangle$. From this position, we can decide to map v to a vertex w of H (all co-evolutionary events except for loss), or, in the case of a loss, to move v to another position. In the first case, v is always mapped to the vertex $h(a)$ that is the head of the arc a. We denote this mapping by $[v : w]$ with $w = h(a)$.

Since in all three cases (co-speciation, duplication, and host-switch), the parasite is supposed to speciate, two children are created for v, denoted by v_1 and v_2. Their positioning along arcs of the host then depends on which of the three events took place. In the case of a loss, no child for v is created (at this step) since there is no parasite speciation, and v is just moved to one of the two arcs outgoing from $h(a)$ randomly chosen.

Notice however that, in order to avoid confusing a loss with another event (for instance, a co-speciation), some precaution must be taken as explained more specifically in the next paragraph concerning the simulation of a loss event.

These choices together with the general framework for our parasite tree generation method are provided next.

Starting the generation The generation of the simulated parasite tree $\tilde{P}$ starts with the creation of its root vertex $\tilde{P}_{root}$. This vertex is positioned before the root of H on the arc $a = (\rho, H_{root})$. More than a simple start position, this allows the simulation of events that happened in the parasite tree before the most recent common ancestor of all host species in H. Figure 3.1 a) depicts this initial configuration.

The evolutionary events For any vertex v of $\tilde{P}$ which is not yet mapped and whose position is $\langle v, a \rangle$ (Fig. 3.1 b)), we choose to apply one among the four allowed operations (depending on the probability of each event). In what follows, we denote by a_1, a_2 the arcs outgoing from the head $h(a)$ of the arc a.

- Co-speciation (Fig. 3.1 c)): We apply the mapping $[v : h(a)]$ and we create the vertices v_1 and v_2 as children of v. We position them as follows: $\langle v_1, a_1 \rangle$ and $\langle v_2, a_2 \rangle$. This operation is executed with probability p_c.

- Duplication (Fig. 3.1 d)): We apply the mapping $[v : h(a)]$ and we create the vertices v_1 and v_2 as children of v. Both v_1 and v_2 are positioned on a. This operation is executed with probability p_d.

- Host-switch (Fig. 3.1 e)): We apply the mapping $[v : h(a)]$ and we create the vertices v_1 and v_2 as children of v. We then randomly choose one of the two children and position it on a. Finally, we randomly choose an arc a' that does not violate the time feasibility of the reconstruction so far (in Chapter

6.3). If such an arc does not exist, it is not possible for a host-switch to take place anymore. In this case, we choose between the three remaining events with probability $p_i/(p_c + p_d + p_l)$ with $i \in \{c, d, l\}$. Otherwise, we position v_2 on a'. This operation is executed with probability p_s.

- Loss (Fig. 3.1 f)): This operation is executed with probability p_l and consists in randomly choosing an arc outgoing from the head $h(a)$ of a and positioning v on it. However, if v was created by a duplication event and is being processed for the first time, we have to verify if its sibling vertex v' was already processed and also suffered a loss. In this case, v must be positioned on the same arc a' where v' was positioned. This procedure is adopted to avoid later mappings where a duplication followed by two losses would be confused with a co-speciation.

We also assume that no evolutionary event takes place whenever a leaf of H is reached. This means that, if v is positioned on an arc incoming to a leaf, then v is mapped to the leaf and no further operation is executed. Hence, the generation of $\tilde{P}$ terminates when all the created vertices are mapped (i.e. have reached a leaf of the host tree). Finally, the leaves of the parasite tree $\tilde{P}$ are labelled according to their mapping to the leaves of the host tree. Observe that as more than one parasite can be mapped to the same host, $\tilde{P}$ is a multi-labelled tree. Finally, some combinations of host switches can introduce an incompatibility due to the temporal constraints imposed by the host and parasite trees, as well as by the reconciliation itself. During the generation of the parasite tree, we always allow only for host-switches that do not violate the time-feasibility constraints. For the criteria enabling to asses time-feasibility, we refer to (Stolzer et al., 2012).

Co-phylogeny parameter estimation algorithm

Prior distribution π. The parameter $\theta = \langle p_c, p_d, p_s, p_l \rangle$ lives in the simplex $\mathcal{S}_3$ (the p's are positive and sum up to one). It is then standard to sample θ from a Dirichlet distribution which is a family of continuous multivariate probability distributions parameterised by a vector α of positive real numbers that determine the shape of the distribution (Gelman et al., 2003).

In our simulations, we adopt a uniform Dirichlet distribution (namely $\alpha = (1, 1, 1, 1)$) that corresponds to sampling uniformly from the simplex $\mathcal{S}_3$. This is

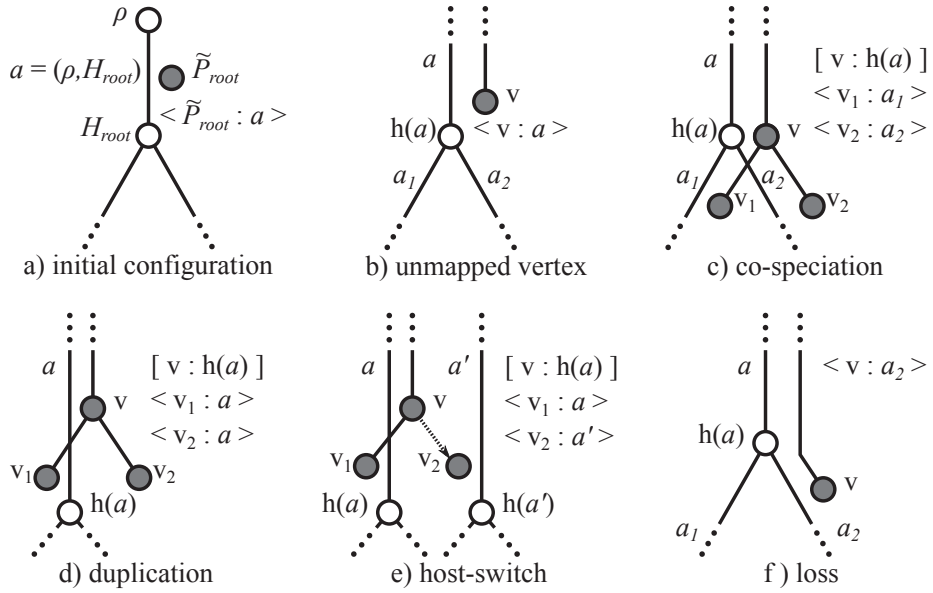


Figure 3.1: Events during the generation of the parasite tree $\tilde{P}$. The host tree has white vertices and the parasite tree grey vertices. The association $\langle v : a \rangle$ indicates that an unmapped parasite vertex v is positioned on the arc a of the host tree. The association $[v : w]$ indicates that the parasite vertex v is mapped to the host vertex w.

often used when there is no previous knowledge favouring one component (e.g. co-evolutionary event) of θ over another. However, the method we implemented allows the user to specify other prior distributions when such knowledge is available.

Choice of summary statistics and fitting criterion The ABC inference method is based on the choice of a summary statistics that describes the data while performing a dimension reduction task. The latter is used to evaluate the quality of agreement (similarity) between the simulated datasets (the generated parasite trees) and the observed (the real parasite tree). In our case, the summary statistics will be based on the measured distances between the generated parasite trees and the real one.

Thus, each simulated tree contributes with its distance from the real parasite to the quality of the vector that generated it. Hence, the distance that will be used must take into account both of the followings: (i) how representative is the simulated tree of the vector that generated it, and (ii) how topologically similar is the simulated tree to the real parasite tree.

Concerning the first point, the intuition is as follows. In our model, when generating a parasite tree, the expected frequency of an event should be close to the corresponding probability value of the parameter vector used to generate the tree. To this purpose, for a given vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$ and for each simulated tree P_θ that was generated according to this vector, we kept track of the number of events $obs = \langle o_c, o_d, o_s, o_l \rangle$ associated to this simulation. We compared the observed number of events to the expected $exp = \langle e_c, e_d, e_s, e_l \rangle$. Observe that the expected number of events can be easily calculated using the size of the parasite tree P and the vector θ. A tree is a good representative if the observed number of events is near to the expected. More formally, for a real parasite tree P, a vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, and a simulated parasite tree P_θ for which the observed number of events are $obs = \langle o_c, o_d, o_s, o_l \rangle$, we define a measure $D_1(P, P_\theta)$ as follows:

$$D_1(P, P_\theta) = \frac{1}{4} \times \sum_{i \in \{c,d,s,l\}} \frac{|e_i - o_i|}{\max\{e_i, o_i\}}$$

As concerns point (ii), we use a metric for comparing phylogenetic trees.

As already mentioned in Section 1.1, there is no easy choice for the metric to use. In our method the leaves of the parasite tree $\tilde{P}$ are labelled according to their mapping to the leaves of the host tree and that more than one parasite can be mapped to the same host. Hence, we are interested in distances between multi-labelled trees. We chose the *maacN* distance define in Section 1.1.

Finally, we propose a distance that is based on two components. The first one takes into account the difference between the expected and observed number of events and the second the difference between the topologies. For a real parasite P, a vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, and a simulated parasite tree P_θ, we define the distance $d(P, P_\theta)$ as follows:

$$d(P, P_\theta) = \alpha_1 D_1(P, P_\theta) + \alpha_2 D_2(P, P_\theta)$$

According to our experiments, the most appropriate values are $\alpha_1 = 0.7$ and $\alpha_2 = 0.3$ but this can be tuned differently by the user. The main drawback of this distance is that it is not a metric; however it achieves good results with respect to discriminating the trees as observed in our experiments. A complete discussion about the choice of α_1 and α_2 is present in the supplementary material (Chapter 6.3).

In COALA , we implemented two other distances both of which are variations of the MAAC. A user can choose the most appropriate one depending on the case. Here we show only the results for the two-components distance as this had the most discriminating power.

Given a parameter vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, we generate M trees and for each of them we consider the distance from the real parasite tree P. From this set of distances, D, we produce a summary, denoted by $S(\theta)$, that characterises the set of trees generated with the parameter vector. In our experiments, we choose $S(\theta)$ as the average of all the produced distances.

The summary $S(\theta)$ is the value that is used in the rejection/acceptance step of the ABC method.

Finally, it is worth noticing that while the choice of a summary statistics (or equivalently here a summary tree distance) is independent from the generation process (co-evolution model), such a choice may have a deep impact on the performance and the results of the method. This is one of the main issues with ABC-related methods. Some recent works have attempted to improve this step (Fearnhead and Prangle, 2012). From the experiments done however, we can already see that the two-components distance seems to be a good enough discriminator.

Approximate Bayesian Computation - Sequential Monte Carlo procedure The ABC-SMC procedure is composed by a sequence of $R > 1$ rounds. For each of these rounds, we define a tolerance value τ_r ($1 \leq r \leq R$) which determines the percentage of parameter vectors to be accepted. Associated to a tolerance value τ_r, we have a threshold ϵ_r which is the largest value of the summary statistics associated to the accepted parameter vectors.

- Initial round ($r = 1$):

  Draw an initial set of N parameter vectors $\{\theta_1^i\}_{(1 \leq i \leq N)}$ from the prior π. Then, for each θ_1^i generate M trees $\{\tilde{P}_j(\theta_1^i)\}_{(1 \leq j \leq M)}$. Select $Q = \tau_1 \times N$ parameter vectors θ_1 that have the smallest $S(\theta_1)$, thus defining the threshold ϵ_1 and the set A_1 of accepted parameter vectors.

- Following rounds ($2 \leq r \leq R$):

1. Sample a parameter vector $\theta^\star$ from the set $A_{(r-1)}$. Create a parameter vector $\theta^{\star\star}$ by perturbing $\theta^\star$. The perturbation is performed by adding to each coordinate of $\theta^\star$ a randomly chosen value in $[-0.01, +0.01]$ and normalising it.
2. Generate M trees $\{\tilde{P}_j(\theta^{\star\star})\}_{(1\leq j\leq M)}$ and compute $S(\theta^{\star\star})$. If $S(\theta^{\star\star}) \leq \epsilon_{(r-1)}$, add $\theta^{\star\star}$ into the quantile set S_r. If $|S_r| < Q$, return to the step 1.
3. Based on the set S_r, select $\tau_r \times Q$ parameter vectors θ_r that have the smallest $S(\theta_r)$, thus defining the threshold ϵ_r and the set A_r of accepted parameters.

The final set A_R of accepted parameter vectors is the result of the ABC-SMC procedure and characterises the list of parameter vectors that may explain the evolution of the pair of host and parasite trees given as input.

Let us observe that, since in all our experiments we are assuming a uniform prior distribution and also are performing the perturbations in a uniform way, the weights induced by the proposals appear to be uniform (Beaumont et al., 2009). However, in the case of a different prior, weights should be used in the process in order to correct the posterior distribution according to the perturbation made.

Clustering the results

Coala implements a hierarchical clustering procedure to group the final list of accepted parameter vectors. The basic process of a hierarchical clustering is as follows. At the beginning, each parameter vector forms a single cluster. Then at each step, the pair of clusters that have the smallest distance to each other are merged to form a new cluster.

The distance that we use between the vectors $\theta = \langle p_c, p_d, p_s, p_l\rangle$ and $\theta' = \langle q_c, q_d, q_s, q_l\rangle$ is the χ^2 distance which is a weighted Euclidean distance defined as follows:

$$d(\theta, \theta') = \sqrt{\sum_{i\in\{c,d,s,l\}} 2 \times \frac{(p_i - q_i)^2}{(p_i + q_i)}}.$$

At the end of this process, we have a single cluster containing all the items represented as a tree (hierarchical cluster tree or dendrogram) showing the relationship among all the original items. As we make no assumptions concerning

the space of the vectors we are dealing with, we chose to apply a more general but still efficient method, introduced in (Langfelder, Zhang, and Horvath, 2007), to select the branches to be cut in the dendrogram. The method proceeds in two steps. Starting with the complete dendrogram, it first identifies preliminary clusters that satisfy some criteria: for example they contain a certain minimum number of objects (to avoid spurious divisions), any two clusters are at least some distance apart etc. (see (Langfelder, Zhang, and Horvath, 2007) for more details). In a second step, all the items that have not been assigned to any cluster are tested for sufficient proximity to preliminary clusters; if the nearest cluster is close enough, the item is assigned to that cluster, otherwise the item remains clustered according to the complete dendrogram.

Finally, once the vectors are split into clusters, we associate to each one a representative parameter vector. To define each coordinate of the "consensus" parameter vector, we take the median value of the respective coordinate in all the parameter vectors which are inside the cluster. We then normalise the "consensus" coordinates to sum up to 1.

3.2 Experimental Results

We evaluated our method in two different ways. First we designed a self-test to show that the principle underlying it is sound and to test it on simulated datasets.

We then extended the evaluation to four real examples that correspond to biological datasets from the literature. This choice was dictated by: (1) the availability of the trees and of their leaf mapping; and (2) the desire to, again, cover for situations as widely different as possible in terms of the events supposed to have taken place during the host-parasite co-evolution. As a matter of fact, the first point drove the choice more than the second: there are not so many examples available from which it is easy to extract the tree and/or leaf mapping and that are big enough to represent meaningful datasets on which to test Coala . All four examples were also analysed in the original paper from which they were extracted by one or more of the existing algorithms that search for a most parsimonious (possibly cyclic) reconciliation (i.e. for a reconciliation of minimum cost). In all cases but one (which is a heuristic strategy and therefore does not guarantee optimality of the solution), the existing algorithms need to

receive as input the cost of the events, which is thus established a priori and drives the conclusions on the results obtained.

Finally, we applied Coala to a biological dataset of our own, representing the co-evolution of bacteria from the *Wolbachia* genus and the various arthropods that host them. This dataset was selected because of its size: the trees have each 387 leaves.

We start by presenting the different datasets used, then the experiments we conducted, and finally the results obtained.

Datasets

Simulated datasets

We first evaluated our model on simulated data. Clearly, in order do this, we have to generate the phylogenies for the hosts and parasites whose co-evolution is being studied in such a way that the probability of each event is known. The basic idea is that if we are able to select a "typical" (or representative) parasite tree P_θ that is generated starting from a host tree H and a given probability vector θ, Coala should be able to list values close to θ among the vectors accepted in the last round.

It is important to observe that many different probability vectors can explain the same pair of trees. Thus, we will consider acceptable if Coala produces clusters that are relatively close to θ.

Due to the high variability of the parasite trees which can be simulated given a host tree H and a vector θ, the task of choosing the most "typical" tree can be hard. To simplify such task and select a typical tree, we impose two conditions which must be observed by the simulated tree. The first one requires that the candidate tree should have a size close to the median for all the trees which are simulated using H and θ. The second condition requires that the observed number of events of a candidate tree should be very close to the expected number given θ.

In practical terms, we execute the following procedure: in order to get realistic datasets we choose a real host tree H (see in Chapter 6.3). Then, given a probability vector θ and H, we generate 2000 parasite trees using our model, without imposing any limit on the size of the generated trees. We then compute the median size of all generated trees and we filter out those whose size is far

from this value (difference greater than 1 or 2 leaves from the median value). Finally, we select as typical tree P_θ the one that shows the smallest χ^2 distance between the vector θ and the vector of observed frequencies of events.

We generated in this way 9 datasets (H,P) associated to the following 9 probability vectors:
$\theta_1 = \langle 0.70, 0.10, 0.10, 0.10 \rangle$,
$\theta_2 = \langle 0.80, 0.15, 0.01, 0.04 \rangle$,
$\theta_3 = \langle 0.75, 0.01, 0.16, 0.08 \rangle$,
$\theta_4 = \langle 0.70, 0.05, 0.02, 0.23 \rangle$,
$\theta_5 = \langle 0.60, 0.20, 0.00, 0.20 \rangle$,
$\theta_6 = \langle 0.55, 0.00, 0.20, 0.25 \rangle$,
$\theta_7 = \langle 0.45, 0.10, 0.15, 0.30 \rangle$,
$\theta_8 = \langle 0.40, 0.20, 0.10, 0.30 \rangle$ and
$\theta_9 = \langle 0.30, 0.20, 0.40, 0.10 \rangle$ (see in Chapter 6.3).
The choice of vectors was done with the aim to cover different patterns of probability. All datasets were generated with the same host tree H of 36 leaves. This tree corresponds to the host tree of the primates/pinworms co-phylogenetic study described in (Hugot, 1999). This choice was motivated by the wish to have a dataset distinct from the ones used for evaluating the method.

Finally, for a pair of host and parasite trees (H, P_θ), we ran Coala 50 times. In each run j, we computed the quality q_j that corresponded to how well the method was able to recover the target vector θ used for generating the dataset P_θ.

Biological datasets

Dataset 1 – Gopher and Louse: The pocket gopher (host) and louse (parasite) is a well studied case of co-evolution which appears in many co-phylogenetic works (Page, 1990; Page, 1994; Hafner and Page, 1995; Johnson, Drown, and Clayton, 2000). The pair of trees which we adopted as dataset was extracted from the work of Johnson, Drown, and Clayton (2000) (see in Chapter 6.3). The host tree contains 15 species and the parasite tree 17. An analysis that had been done previously by Hafner and Page (1995) using TreeMap (Page, 1994) had produced a reconciliation with 10 co-speciations, 5 duplications, 1 host-switch and 20 losses. Johnson et al. suggested that some of the observed incongruences might be the result of artifactual differences due to incorrectly inferred phylogenies

with weakly supported vertices. They then proposed an original method to first identify such artifacts, revise the two trees accordingly and then apply the existing methods for reconciliation. They thus show that fewer incongruence events are required than those inferred: the reconciliation they propose (using again TreeMap) has 8 incongruence events – 1 duplication, 3 host-switches, and 4 losses – and 12 instead of 10 co-speciations.

Dataset 2 – Flavobacterial endosymbionts and their insect hosts: This dataset was extracted from the work of Rosenblueth et al. (2012) and is composed of a pair of host and parasite trees which have each 17 species (see in Chapter 6.3). The parameter adaptive approach of CoRe-Pa (Merkle, Middendorf, and Wieseke, 2010) was used to infer the more appropriate cost vectors for analysing this dataset. Nine such vectors were produced. However, only one, $\langle c_c = 0.088, c_d = 0.325, c_s = 0.339, c_l = 0.248\rangle$, was associated to a feasible reconciliation in the sense that host-switches happened between contemporary species only (the branch length was used to infer this information). Since CoRe-Pa can produce unfeasible (i.e. cyclic) solutions during the parameter adaptive approach, Rosenblueth et al. decided to complement their study with Jane 3 (Conow et al., 2010), which uses a genetic algorithm approach to produce only acyclic reconciliations. They thus started with the only cost vector obtained by CoRe-Pa associated to a feasible reconciliation, however transforming it into integer numbers (a requirement of the software), and then gradually changed the costs until a feasible reconciliation was produced (again using branch-length information). This procedure resulted in the cost vector $\langle c_c = 1, c_d = 1, c_s = 1, c_l = 2\rangle$ and a reconciliation with 9 co-speciations, 0 duplication, 7 host-switches and 1 loss, the same as obtained by CoRe-Pa.

Dataset 3 – Anther smut fungi and their caryophyllaceous hosts: The anther smut disease is caused by the *Microbotryum* fungi which infect *Caryophyllaceous* plants. This case was studied by Refrégier et al. (2008) who explored the influence of a good species delimitation on the results of a co-phylogenetic reconciliation. By comparing a traditional approach for assigning a fungal strain to each host species to another approach where the fungal species are better defined, they showed that the former can introduce a bias towards co-speciation events due to the existence of many vertices that do not correspond to speciation events, but

rather to a separation into different strains belonging in some cases to a same generalist species. The reconciliations proposed by Refrégier et al. for the pair of trees using the second (more accurate) approach have from 0 to 3 co-speciations, no duplication, 12 to 15 host-switches and 0 to 2 losses. For this dataset, we selected the pair of trees with a better definition of species which are composed of, respectively, 18 host and 16 parasite species. However, for this pair of trees, there are four parasite species that are associated each to two host species, a case not currently considered by our model. We thus decided to consider all the parasite trees that may be obtained by removing one association of each parasite that is linked to 2 host species. By doing this, we obtained 16 pairs of trees with distinct associations among host and parasite leaves (see in Chapter 6.3).

Dataset 4 – Rodents and Hantaviruses: This dataset is taken from the paper Ramsden et al. (Ramsden, Holmes, and Charleston, 2009, Figure 2) and considers the co-evolution of *Hantaviruses* with their insectivore and rodent hosts. The host tree is composed by a total of 34 hosts (28 rodents and 6 insectivores) and the parasite tree by 42 hantaviruses. It was strongly believed that hantaviruses co-speciated with rodents since their phylogenetic trees have topological similarities with three consistently well-defined clades (Hughes and Friedman, 2000; Plyusnin and Morzunov, 2001; Nemirov, Vaheri, and Plyusnin, 2004; Jackson and Charleston, 2004). The authors show that to support this hypothesis, the evolutionary rate of the RNA sequences of the hantaviruses should be several orders of magnitude smaller than the rates which are normally observed in RNA viruses that replicate with RNA-dependent RNA polymerase (Hanada, Suzuki, and Gojobori, 2004). By analysing the co-phylogenetic reconciliations, the authors show that scenarios with more than 20 co-speciations are statistically non-significant. To explain the topological congruences, the authors point to the fact that host-switching followed by pathogen speciation can generate congruence between trees, particularly when pathogens preferentially switch among closely related hosts. Based on this fact and on the observed patterns of amino acid replacement observed in these viruses (compatible with host-specific adaptation), the authors conclude that the co-evolutionary history of these hosts and parasites is the result of a recent history of preferential host-switching and local adaptation.

Dataset 5 – *Wolbachia* and their arthropod hosts: *Wolbachia* is a large monophyletic genus of intracellular bacteria phylogenetically very diverse and currently considered the most abundant endosymbionts in arthropods. In insects alone, it is estimated that over 65% of the species are thus infected by *Wolbachia*. The dataset used in this paper corresponds to *Wolbachia* species that were detected in an extensive set of arthropods collected from 4 young, isolated islands (less than 5 Myr. old) (Simões et al., 2011; Simões, 2012). The trees are a subset of those discussed in (Simões et al., 2011; Simões, 2012) where we retained only those parasites which were associated to a unique host, the latter diverge by at least 2% at the level of the *CO1* genes that were used for reconstructing their phylogenetic tree while the *Wolbachia* sequences (corresponding to the *fbpA* gene) differ by at least one SNP. Each resulting tree is composed of 387 leaves. The initial results presented in (Simões, 2012) seemed to indicate that host-switches might be quite frequent even among hosts that are physiologically and molecularly very distinct and thus phylogenetically distant.

Experiments and results

Experimental parameters

All datasets were processed by Coala configured with the same parameters. For each dataset, we generated $N = 2000$ parameter vectors in the first round. For each of the vectors, we generated $M = 1000$ parasite trees using our method. We required these trees to have a size at most twice the one of the real parasite, otherwise the tree was discarded as being too different from the real parasite. If a given vector did not generate M such trees in 5000 trials, then the vector was immediately associated to a distance equal to 1 which indicated that it represented badly the real data.

We used the average of all the 1000 distances produced as a fitting criterion in the rejection/acceptance step of the ABC method. The tolerance value used in the first round was $\tau_1 = 0.1$. For the remaining rounds $2 \leq i \leq R$, we defined $\tau_i = 0.25$. Notice that $\tau_1 \times N = 200$ defines the size Q of the quantile set which must be produced in each new round. Thus, after the last round, we have $\tau_R \times Q = 50$ accepted vectors. These vectors are grouped into clusters and a representative vector is associated to each cluster as explained in the Subsection "Clustering the results".

We ran the experiments using $R = 3$ and $R = 5$ rounds. The number of rounds is an important parameter which defines the characteristics of the list of accepted parameter vectors.

However, observe that a high number of rounds will tend to overfit the data and thus hide a possible variability in the list of accepted vectors that could provide significant alternatives for explaining the studied pair of trees.

Since we are interested in exploring different alternatives for each dataset, we present only the results which were obtained after running Coala for 3 rounds. The results involving 5 rounds may be found in the in Chapter 6.3.

The experiments were executed at the IN2P3 Computing Center (`http://cc.in2p3.fr/?lang=en`). For the simulated datasets, each pair of trees were processed with 3 threads for speeding up the simulation process. The time necessary to complete 5 rounds for all 50 runs varied from 1 to 2 days depending on the size of the trees. For the biological datasets 1 to 4, we also used 3 threads. The observed execution times for 5 rounds were between a couple of hours for the smallest dataset (Dataset 1) and one day for Dataset 4. Due to its size, the dataset *Wolbachia*-arthropods was processed with 150 threads and it required approximately 8 days to complete 5 rounds.

Results on the simulated datasets

Self-Test As concerns the self-test, we designed the following procedure. Let P_θ denote the simulated parasite tree chosen in correspondence of the probability vector θ, as explained in the paragraph "Simulated Datasets". We recall that the host tree H remains the same during all the self-test experiments. For a pair of host and parasite trees (H, P_θ), we ran Coala 50 times. In each run j, we computed the quality q_j that corresponded to how well the method was able to recover the target vector θ used for generating the dataset P_θ. To do this, for each run j, we considered the representative vectors of the clusters produced as output. We computed the χ^2 distance for each of the representative vectors to the target vector θ and set q_j to the smallest value among them. Figure3.2 shows the distribution of the quality values which were obtained at the end of each round (from 2 to 5) for the simulated datasets θ_1, θ_3, θ_4, and θ_7 (the results for the remaining datasets can be found in Chapter 6.3). Figure 3.3 shows the histograms of the event probabilities observed for the 50 parameter vectors with

smallest χ^2 distance at the end of each round for dataset θ_3 (again, the results for the remaining datasets are available in Chapter 6.3).

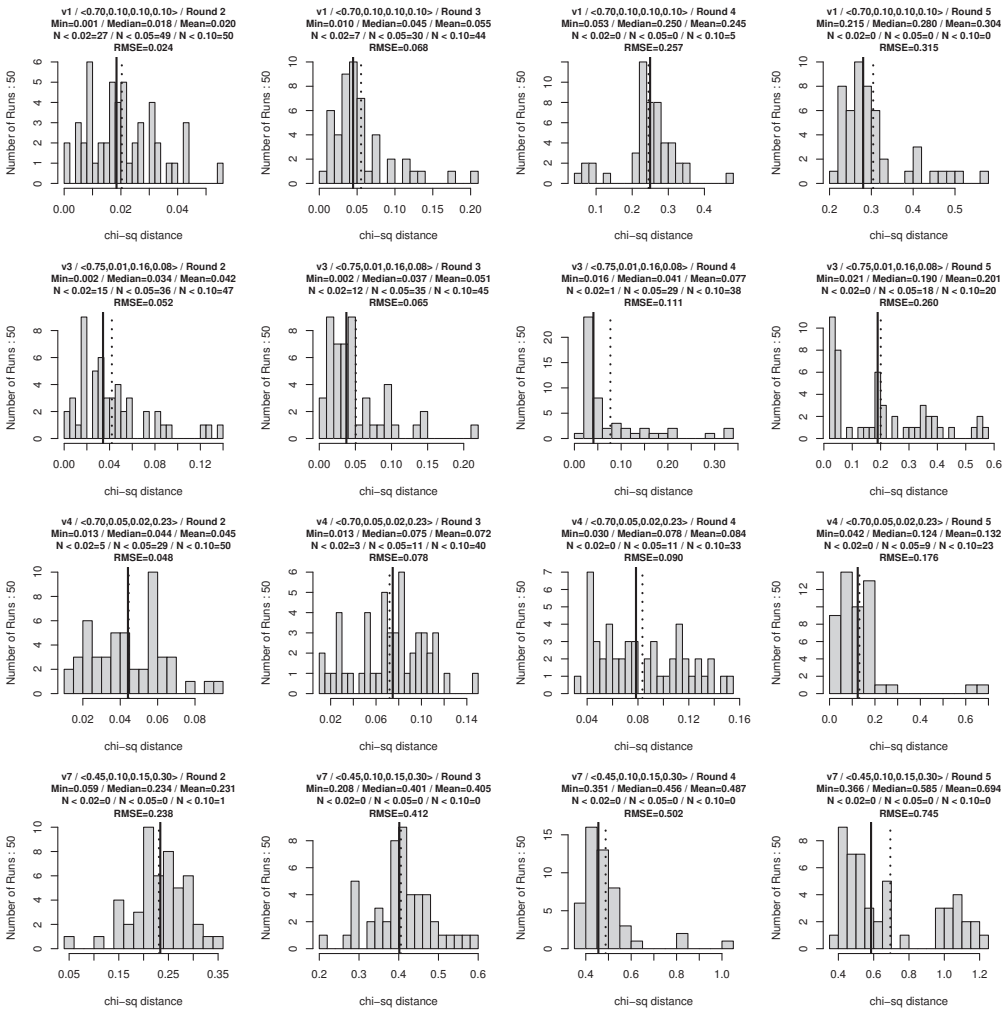


Figure 3.2: For each simulated dataset, we ran Coala 50 times and, at the end of each round (from 2 to 5), we took note of the cluster whose representative parameter vector had the smallest χ^2 distance to the probability vector used to generate the simulated dataset. The histograms show the distribution of the smallest χ^2 distance observed on each one of the 50 runs at the end of each round (for the simulated datasets $v_1 = \theta_1$, $v_3 = \theta_3$, $v_4 = \theta_4$, and $v_7 = \theta_7$.). The solid and dotted vertical lines indicate median and mean values, respectively.

Results on the first four biological datasets

The four biological datasets were processed by COALA as described in the Section "Experimental parameters". Table 3.1 shows the representative parameter vectors and Figure 3.4 the histograms of the event probabilities of the list of accepted vectors obtained at the end of the third round.

Let $\langle n_c, n_d, n_s, n_l \rangle$ be the vector of the events related to each such reconciliation and let $\langle f_c, f_d, f_s, f_l \rangle$ be the vector of frequencies where $f_i = n_i / \sum_{i \in \{c,d,s,l\}} n_j$. We can now compare for each of the trees the values f_i to p_i and see how close to each other they are. We performed this test for each dataset and for each one of its representative parameter vectors (clusters). The detailed results may be found in Chapter 6.3 These show that our simulator is coherent in the sense that the generated parasite trees present reconciliations with an expected value of the frequency of each event almost equal to the corresponding value of the parameter vector used to generate the tree.

In order to compare our results in relation to the existing literature, we transformed each one of the representative parameter vectors $\langle p_c, p_d, p_s, p_l \rangle$ into a vector of costs that was then used to compute optimal reconciliations between the host and parasite trees given as input. The transformation was done by defining $c_i = -\ln p_i$, with $i \in \{c, d, s, l\}$, which is based on a commonly accepted idea that the cost of an event is inversely related to its probability. Indeed, if p_i is equal to 1, then we expect all the events to be of type i, thus the cost of the corresponding event must be 0. Similarly, if p_i is equal to 0, we expect that event i never happens, and thus the cost must be assigned to $+\infty$.

To the best of our knowledge, the only methods that enumerate all optimal reconciliations are CoRe-Pa (Merkle, Middendorf, and Wieseke, 2010), NOTUNG (Stolzer et al., 2012) and EUCALYPT (Donati, Baudet, Sinaimeri, Crescenzi, Sagot, "EUCALYPT: Efficient tree reconciliation enumerator", in press at *Algorithms for Molecular Biology* and `http://eucalypt.gforge.inria.fr/`). However CoRe-Pa in some cases misses solutions, probably because it considers some additional constraint. NOTUNG does not allow co-speciation costs different from zero and the remaining event costs must be described by integer values. We thus present the results of EUCALYPT which allows the configuration of all event costs and accepts real numbers.

Table 3.2 shows, for each dataset, the vector of costs (c_c, c_d, c_s, c_l) produced

by transforming the representative parameter vectors obtained after the third round (Table 3.1). Column *Opt* indicates the cost of the optimal solution and columns #*c*, #*d*, #*s*, #*l* the numbers of each event type which are observed among the enumerated scenarios. Finally, columns #*A* and #*C* indicate, respectively, the total number of acyclic and cyclic scenarios.

Figure 3.4 shows for each dataset the distribution of each event probability observed for the 50 vectors accepted by Coala in the third round.

Dataset	*Cluster*	p_c	p_d	p_s	p_l	*#vectors*
1	1	0.794	0.002	0.101	0.104	18
	2	0.790	0.118	0.085	0.007	15
	3	0.864	0.061	0.000	0.074	12
	4	0.522	0.280	0.001	0.197	5
2	0	0.030	0.000	0.557	0.413	1
	1	0.461	0.258	0.000	0.281	24
	2	0.554	0.000	0.270	0.176	20
	3	0.910	0.016	0.058	0.016	5
3	1	0.744	0.001	0.090	0.165	19
	2	0.397	0.000	0.353	0.250	18
	3	0.512	0.231	0.000	0.257	10
	4	0.036	0.000	0.610	0.354	3
4	1	0.851	0.082	0.000	0.066	25
	2	0.473	0.204	0.000	0.323	10
	3	0.238	0.349	0.000	0.413	8
	4	0.580	0.002	0.282	0.136	7

Table 3.1: Representative probability vectors produced by Coala at Round 3

Results on the *Wolbachia*-arthropods dataset

The *Wolbachia*-arthropods dataset was also processed by Coala as described in the Section "Experimental parameters". Table 3.4 shows the three clusters which were obtained at the end of the third round.

Similarly to the analysis performed for the small biological datasets, we transformed each one of the representative parameter vectors into a vector of costs that was then used to compute optimal reconciliations between the host and parasite trees given as input.

Due to the size of the trees (387 leaves each), the total number of solutions is huge which makes impossible the enumeration of all the results. For this dataset, we therefore only computed the costs of the optimal solutions and the total

Dataset	*Cluster*	c_c	c_d	c_s	c_l	*Opt*	#c	#d	#s	#l	#A	#C
1	1	0.231	6.502	2.293	2.267	16.462	12	0	4	2	3	0
	2	0.235	2.140	2.467	4.934	22.085	10	0	6	1	5	0
	3	0.146	2.792	9.721	2.601	42.366	12	2	2	6	2	0
	4	0.650	1.272	7.601	1.625	35.280	12	2	2	6	2	0
2	0	3.517	13.816	0.584	0.885	14.044	1	0	15	2	2944	0
	1	0.775	1.355	7.824	1.270	48.664	11	2	3	11	2	0
	2	0.591	8.517	1.310	1.736	16.217	9	0	7	1	1	0
	3	0.094	4.160	2.844	4.154	24.892	9	0	7	1	1	0
3	1	0.295	7.264	2.405	1.804	42.056	4	0	11	8	2	34
	2	0.924	9.567	1.042	1.385	24.952	3	0	12	7	128	0
	3	0.670	1.464	8.517	1.360	76.235	7	8	0	44	1	0
	4	3.313	13.816	0.494	1.039	17.502	1	0	14	7	1024	0
4	1	0.161	2.496	9.210	2.717	153.544	22	11	8	18	0	12
	2	0.748	1.592	9.210	1.130	105.393	22	19	0	52	1	0
	3	1.436	1.053	8.112	0.884	97.548	22	19	0	52	1	0
	4	0.545	6.266	1.265	1.996	72.588	17	5	19	4	4	0

Table 3.2: Event vectors obtained by transforming probability vectors (Table 3.1) into cost vectors

number of solutions. Additionally, for each cluster, we sampled 10000 solutions and we checked for the presence of acyclic solutions. Table 3.3 summarises the obtained results.

3.3 Discussion

Comparing the results of the different datasets, we can see a big variety of probability vectors. This stresses the need to use estimated cost vectors instead of those traditionally found in the literature which may in some cases not be adapted to a given biological situation. Moreover, the estimated probabilities of each event provide already important information on the co-evolution dynamics of hosts and parasites.

In general, Coala must be calibrated according to the needs of the study. If we are interested in obtaining a wider set of possible explanations, a small number of rounds is recommended. On the other hand, if this set must be restricted, we must increase the number of rounds. This decision must be taken with care because even if an event probability vector is more likely to explain a pair of trees and their associations, it does not mean that it reflects the real evolutionary scenario (this may happen because the DTL model is only an

approximation of the real evolutionary scenario, and also because we do not have an infinite amount of observations through the knowledge of all taxa). We thus believe that setting Coala with 3 rounds offers the best compromise between variability and specificity.

Simulated datasets

Up to a certain level of co-speciation probability (≥ 0.50), our results (Figure 3.2) show that in the rounds 2 and 3, Coala is able to select parameter vectors that are close to the target probability vector. Looking to the histograms of these two rounds, we can observe that in most of the runs, the closest parameter vector has low χ^2 distance to the target. After the third round, this tendency changes and the closest parameter vectors show high χ^2 distances indicating that Coala is mainly selecting vectors which are far from the target one.

Since Coala is based on an ABC-SMC approach, the accepted vectors in one round have summary statistics (i.e. average distance) smaller than the ϵ defined in the previous round. This means that at each new round, Coala is selecting parameter vectors that have more probability of explaining the pair of trees given as input because their simulated parasite trees are, on average, closer to the real one.

Although we try to choose the best representative parasite tree P for each pair (H, θ), we cannot guarantee that θ is the best explanation for the association between H and P. Even so, Coala was able to select parameter vectors that are close to the target probability vector along the first rounds. Figure 3.3 shows the histograms of the event probabilities observed among the 50 parameter vectors with smallest χ^2 distance at the end of each round for dataset θ_3, and confirms these observations. We can see that at round 2, the median and mean event probabilities (solid and dotted vertical lines) are very close to the target value (dashed vertical line). When we increase the number of rounds, the distance between the median/mean probabilities and the target values increases.

When we decrease the co-speciation probability to values smaller than 0.50, Coala selects very few vectors which are close to the target vector. When the co-speciation probability decreases while the duplication and host-switch probabilities increase, the variability of the tree topologies observed increases exponentially. Due to this, selecting a typical tree becomes an almost impossible task and this may explain the obtained results. Increasing the number of

simulated trees to compute the summary statistics might enable to improve the quality of the results. However, this would require a much longer execution time.

Biological datasets

Dataset1 – Gopher and Louse: For this dataset, we obtained 4 clusters. The first three are characterised by a high co-speciation probability value (> 0.79) while the last one has an intermediate value (0.52). Each one of the three first clusters presents a very low value for one of the remaining events. The last vector has the highest duplication and loss values (0.28 and 0.20, respectively) and no host-switch. The scenario proposed in (Johnson, Drown, and Clayton, 2000) contains 12 co-speciations, 1 duplication, 3 host-switches and 4 losses. Except for cluster 2, the reconciliations obtained by using the corresponding cost vectors also show 12 co-speciation events. However, the number of duplications, host-switches and losses differ from the scenario given in (Johnson, Drown, and Clayton, 2000). The first cluster shows 0 duplication, 4 host-switches and 2 losses configuring a situation which minimises the number of incongruence events. Despite a low host-switch probability, the parsimonious reconciliations obtained for clusters 3 and 4 show the same scenario: 2 duplications, 2 host-switches and 6 losses. The cost vector related to cluster 2 has the characteristic of having a high cost for losses. This results in a reconciliation scenario with a high number of host-switches, a smaller number of co-speciations (10 instead of 12), only 1 loss and no duplication.

Dataset 2 – Flavobacterial endosymbionts and their insect hosts: In this case, we obtain 3 non-singleton clusters which are quite different from each other. Cluster 0 is formed by a single accepted vector which did not cluster with any other because it is too far apart. Cluster 1 shows probabilities of 0.46, 0.26 and 0.28, respectively, for co-speciation, duplication and loss. After transforming these into costs, the obtained reconciliation scenarios have 11 co-speciations, 2 duplications, 3 host-switches and 11 losses. Clusters 2 and 3 show very low duplication probability. While cluster 2 exhibits intermediate values for the remaining probabilities, cluster 3 has a very high co-speciation probability value (0.91) and low host-switch (0.06) and loss (0.02). Due to the low duplication value, these clusters show the same reconciliation scenario: 9 co-speciations, 0

duplications, 7 host-switches and 1 loss (which is identical to the one proposed by Rosenblueth et al. (2012)).

Dataset 3 – Anther smut fungi and their caryophyllaceous hosts: As explained previously, in order to analyse this dataset we considered all the parasite trees that may be obtained by removing one association of each parasite that was linked to 2 host species. This led to the generation of 16 pairs of trees (each identified by a letter from $\mathcal{A}$ to $\mathcal{P}$) with distinct associations. We conducted a study to explore the influence of the different associations on the obtained results (see in Chapter 6.3).

In our experiments, we took all the 50 parameter vectors that were accepted at the end of the third round for each one of the 16 pairs of trees and we executed the same procedure that Coala uses to group vectors into clusters. A total of 7 clusters were enough to group all the 800 accepted parameter vectors. Each pair of trees had accepted parameter vectors spread in at least four distinct clusters, and therefore we could not clearly group such pairs according to the obtained clusters.

We repeated this experiment with the list of accepted parameter vectors which were obtained at the end of the fifth round. In this case, 6 clusters were enough to group all the 800 parameter vectors. However, now most of the trees had accepted parameter vectors spread in only one or two clusters. Moreover, we could easily define 4 groups of pairs of trees that are explained by the same probability vector plus 2 singletons.

To present a result here, we chose the pair of trees $\mathcal{M}$. Looking at the results of the fifth round, this pair is grouped with another 3 other pairs ($\mathcal{A}$, $\mathcal{J}$, and $\mathcal{L}$) in a cluster that contains the highest number of elements (including some parameter vectors of another 5 pairs of trees).

For the pair of trees $\mathcal{M}$, Coala generated 4 clusters at the end of the third round. Except for cluster 3, all the others exhibit zero probability of duplication. Clusters 1, 2 and 4 show an inverse correlation between their co-speciation and host-switch probabilities: when one of them is high, the other is low. Cluster 3 points to a situation where the probability of host-switch is zero, and the probabilities of co-speciation, duplication and loss are, respectively, 0.51, 0.23 and 0.26. After transforming the probabilities into costs, each cluster obtained a different reconciliation scenario, showing that this pair of trees is specially

sensible to the chosen cost vector. The scenario proposed by Refrégier et al. (2008) was recovered by Cluster 2 (3 co-speciations, 0 duplications, 12 host-switches and 7 losses). Moreover, Cluster 1 presents an alternative scenario which is very close: 4 co-speciations, 0 duplications, 11 host-switches and 8 losses.

Dataset 4 – Rodents and Hantaviruses: Despite the topological congruence observed for this pair of trees, Ramsden, Holmes, and Charleston (2009) show that the co-evolutionary history of hantaviruses and their host is explained by preferential host-switches and local adaptation. Looking at Table 3.1, we can observe that the clusters 1, 2, and 3 have representative vectors with zero probability for host-switch events: cluster 1 has a very high co-speciation probability (0.85), while clusters 2 and 3 have probability values which are almost equally distributed among co-speciation, duplication and loss events.

After transforming these vectors into costs (Table 3.2), we obtain scenarios with a high number of co-speciations which is considered non-significant by Ramsden, Holmes, and Charleston (2009).

Differently from the others, cluster 4 shows a vector with host-switch probability higher than the probabilities of duplication and loss. When converted into costs, this generates time-consistent scenarios with 17 co-speciations, 5 duplications, 19 host-switches and 4 losses, a result much closer to the explanation given by Ramsden, Holmes, and Charleston (2009). These results reinforce the idea that, although Coala is able to identify vectors which can explain a pair of trees, having a prior knowledge of the dynamics of the interactions of the two groups of species is important to identify the clusters that better explain their co-evolution.

Dataset 5 – *Wolbachia* and their arthropod hosts: For the *Wolbachia*-arthropods dataset, we obtained three distinct clusters. All have significantly high co-speciation probabilities (> 0.77). The first cluster has a very low duplication probability and a host-switch probability around 0.5. The two other clusters point to a relatively high duplication probability and low level of host-switches. The difference between them is related to the probability of losses which is around 0.14 for Cluster 2 and zero for Cluster 3.

Cluster 1 goes in the direction of what was presented in (Simões, 2012) where the author suggested that in the last 3 Myr., there were many transfers

of *Wolbachia*, including between different arthropod Orders, that is over large phylogenetic distances. Clusters 2 and 3 point to an opposite scenario.

What is most striking with the results obtained for this dataset is the absolutely huge number of optimal reconciliations that can be derived for all clusters. For the small sampling that we performed, we were able to find feasible (acyclic) solutions only with the cost vector produced with the event probabilities of Cluster 3. However, the results obtained with all the other four datasets used here lead us to suggest that the number of feasible solutions might quite possibly remain large.

Cluster	c_c	c_d	c_s	c_l	*Opt*	*Solutions*	*Acyclic solutions*
1	0.144	5.116	2.899	2.623	917.475	5.4×10^{43}	No
2	0.260	2.551	4.595	1.961	1407.877	9.8×10^{40}	No
3	0.037	3.817	4.269	13.816	1375.725	1.6×10^{51}	Yes

Table 3.3: Total number of solutions obtained by transforming probability vectors (Table 3.4) into cost vectors for *Wolbachia*-arthropods datasets

Cluster	p_c	p_d	p_s	p_l	*#vectors*
1	0.866	0.006	0.055	0.073	26
2	0.771	0.078	0.010	0.141	22
3	0.964	0.022	0.014	0.000	2

Table 3.4: Representative probability vectors produced by Coala , at the end of the third round, while processing *Wolbachia*-arthropods datasets

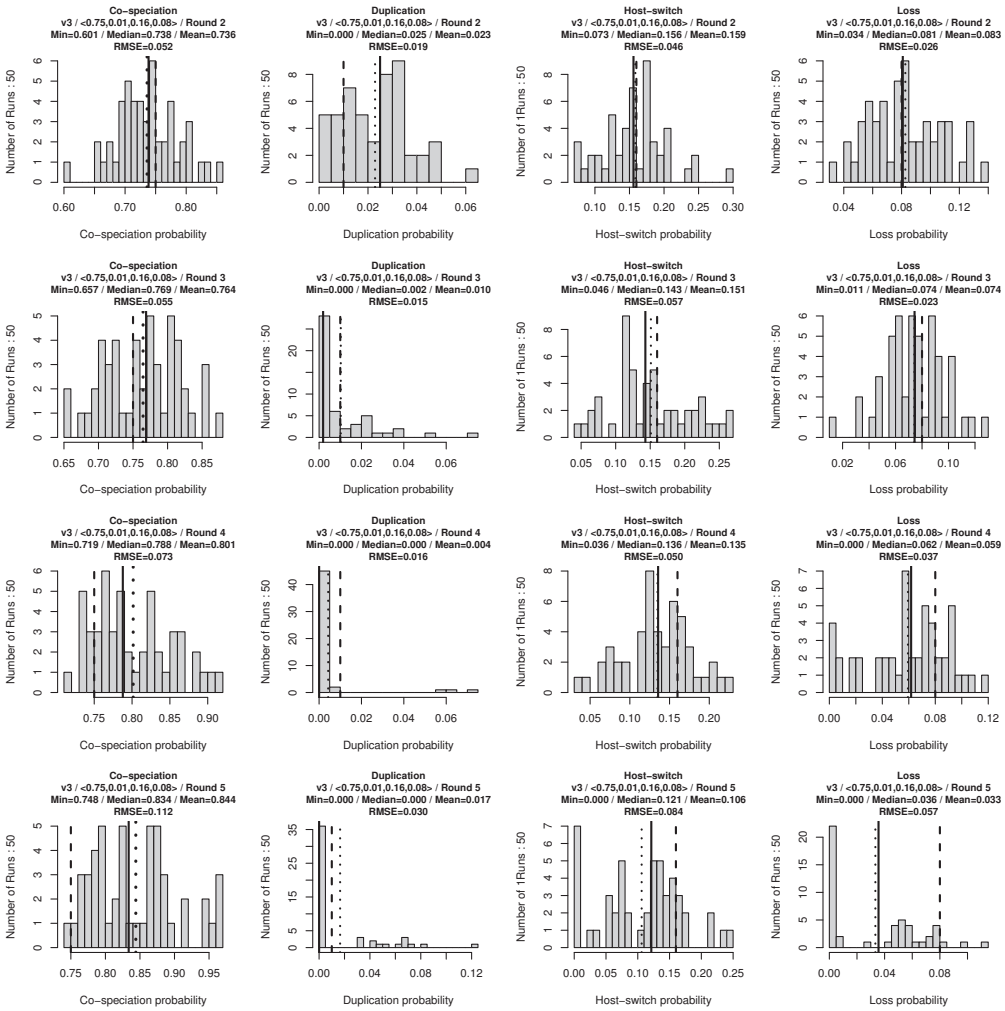


Figure 3.3: For each simulated dataset, we ran COALA 50 times and, at the end of each round (from 2 to 5), we took note of the cluster whose representative parameter vector had the smallest χ^2 distance to the probability vector used to generate the simulated dataset. The histograms show the distribution of the event probabilities observed on the list of parameter vectors which have the smallest χ^2 distance on each run for the dataset $v_3 = \theta_3$. The solid and dotted vertical lines indicate median and mean values, respectively. The dashed vertical line indicates the "target" value.

3.4 Conclusions and prespectives

We observe in the results we obtained on a diverse selection of datasets, that the costs inferred by our simulations may be very different across datasets, thus motivating the use of estimated instead of fixed costs. Such costs may

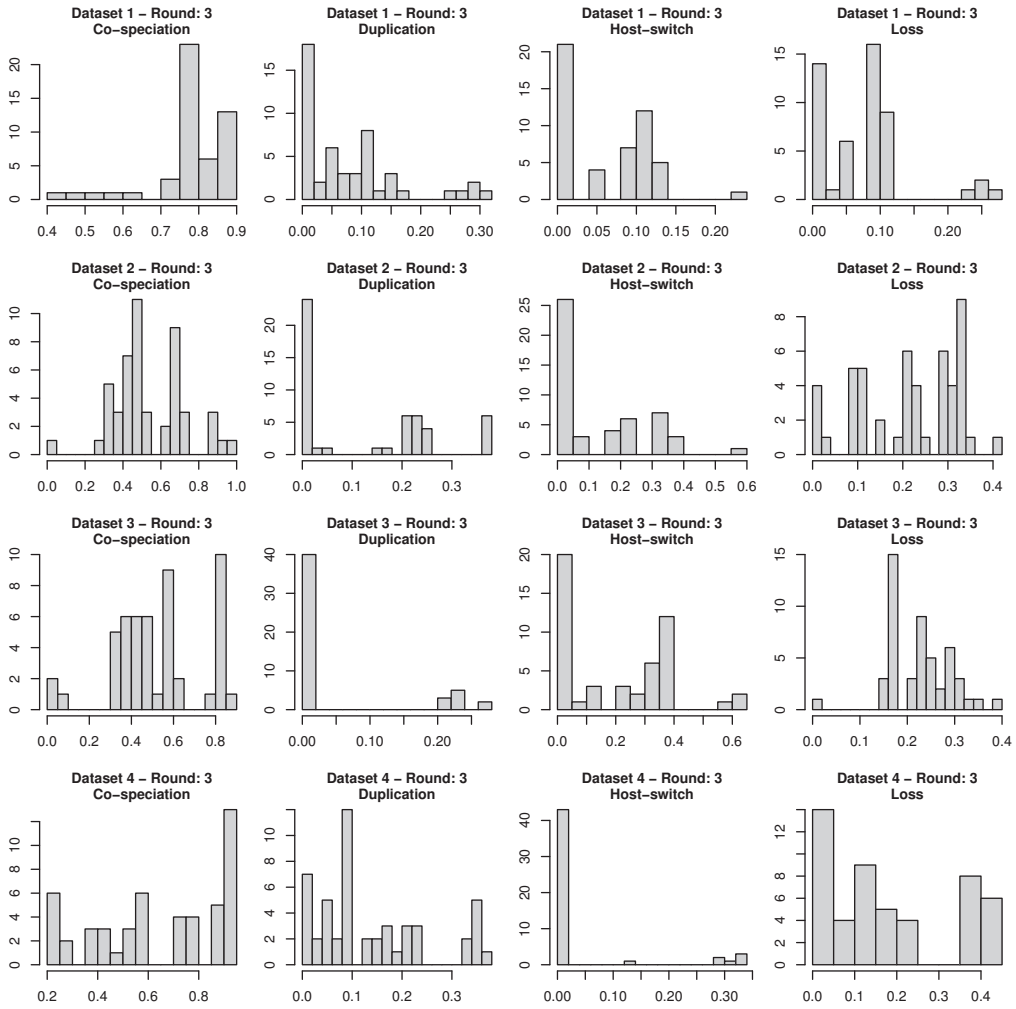


Figure 3.4: Distribution of the probability values for each event type observed on the parameter values accepted on the third round while processing the biological datasets 1 to 4.

even differ widely for a same pair of host-parasite trees, as is observed for the *Wolbachia*-arthropods dataset.

These costs are inversely related to their likelihood, and so to their expected frequency. For this reason, providing information on the frequencies of the events is an important issue, in particular in the cases where the reconciliation methods fail to find a solution. The latter can happen, for instance, if all the optimal solutions that are identified by the existing reconciliation algorithms

are biologically unfeasible due to the presence of cycles since finding an acyclic reconciliation is an NP-hard problem. In addition, if the host and parasite trees are large (for instance, of the order of hundreds of taxa), these cases cannot be handled by the existing reconciliation algorithms in the sense that there are too many solutions to test for acyclicity.

In general, the reconciliation model presented in the previous chapter could be enriched in many aspects, and this would influence also the model used in Coala. For instance, we should consider as a future work the case where the input phylogenies are not fully resolved, meaning that the trees are not binary. A more efficient exploration of the parameter space is another important future issue that would significantly increase the efficiency of our procedure, and also allow to handle larger trees.

It is important to observe that most studies on co-phylogeny assume that the phylogenies of the organisms are correct. Clearly, this may affect the results observed. It would therefore be interesting to be able to infer the co-phylogenetic reconciliation directly from sequence data.

Finally, the accuracy of the results obtained by our method depends on the choice of the metric used for comparing trees. Designing new metrics that can be computed efficiently while still capturing the similarity for multi-labelled, not fully resolved trees is therefore another important future issue which we believe is also interesting per se.

Part II

Network decomposition

Chapter 4

Analysis of contig networks

In this second part of our work, graph theory is applied to a biological subject that belongs to the area of DNA sequencing techniques. In particular, the problem we want to address is to find an order and an orientation to a set of medium size fragments called *contigs*. This process is known as *scaffolding* and will be presented, together with some mathematical preliminaries, in the present chapter. We developed an original model for scaffolding, together with a new software, MeDuSa, that implements our algorithm. The results concerning this model are presented in Chapter 5.

During the development of MeDuSa, we encountered some interesting problems of a pure theoretical nature. In particular, we focused our attention on one, called *Implicit Hitting Set*, that has not yet been studied in terms of its enumeration complexity. This problem is interesting both for its potential applications in biology and from a strictly mathematical point of view. Our results and ideas concerning this topic will be presented in Chapter 6.

4.1 Mathematical background

Vertex-disjoint path cover

Let $G = (V, E)$ be an undirected weighted graph. A *path* in G of length h is a sequence of distinct vertices $(v_1, \ldots, v_h)$, where $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, h-1$.

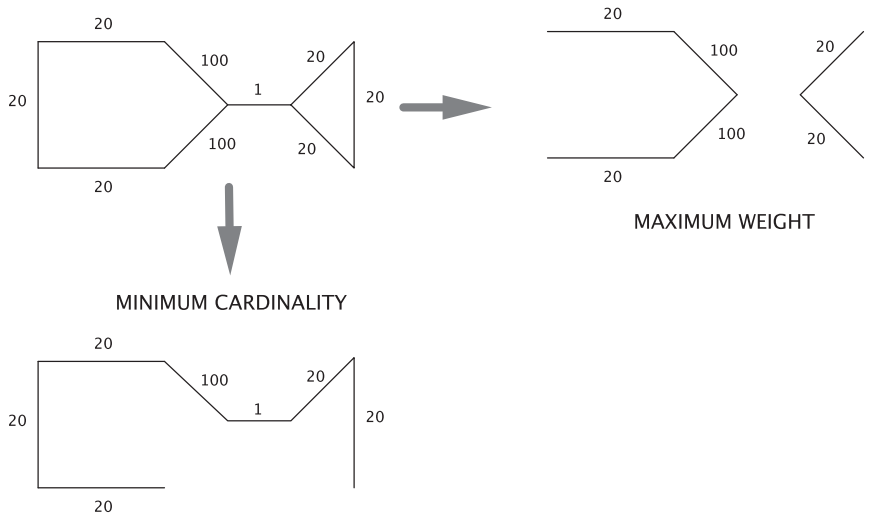


Figure 4.1: Two vertex-disjoint path covers for the same graph, optimizing different values.

A *path cover* C of G is a set of vertex-disjoint paths $P_1, \ldots, P_s$ that cover all the vertices of G. In a weighted graph, a cover C has total weight $w(C) = \sum_{e \in C} w(e)$. Observe that, by an abuse of notation, we say that an edge belongs to a cover ($e \in C$), if it belongs to any of its paths. In this sense, a set of paths can also be seen as a set of edges.

From an optimisation point of view, a cover can be characterised by both values s (its cardinality) and w (its weight). The path cover having minimum cardinality in general does not coincide with the cover having maximum weight (see Figure 4.1). This means that it is not possible to optimise both values at a same time.

We can define then two different optimisation problems:

Problem 2
Minimum path cover

Given an undirected graph, $G = (V, E)$, compute a vertex-disjoint path cover $C \subseteq E$ of minimum cardinality.

Observe that, given two covers C_1, C_2 of G, if the number of edges in C_1 is

greater than the number of edges in C_2, then the cardinality of C_1 is less than the cardinality of C_2.

A second optimisation problem is the following:

Problem 3 *Maximum weight path cover*

Given an undirected weighted graph, $G = (V, E)$, compute a vertex-disjoint path cover $C \subseteq E$, with maximum total weight.

Both previous problems are clearly $NP-$complete since they contain HAMILTONIAN PATH as a sub-problem. Moreover the two problems are equivalent if the weights of the edges are all equal.

There are different ways for addressing hard problems maintaining a provable solution quality and run-time bounds. We want to present two different approaches here.

A first way is to look for an *approximation* algorithm. This algorithm is defined as follows:

Definition 9 *ρ-approximation algorithm.*

Given a maximisation problem with optimum value opt, a $\frac{1}{\rho}$*-approximation algorithm* gives a solution of quality q, with

$$\frac{opt}{\rho} < q < opt.$$

Given a minimisation problem with optimum value opt, a ρ-approximation algorithm gives a solution of quality q with

$$opt < q < \rho \cdot opt.$$

The most efficient approximation algorithm for solving Problem 3 is presented in (Moran, Newman, and Wolfstahl, 1990). It has time complexity in $O(|E| \log |E|)$ and an approximation ratio of $\frac{1}{2}$. In the same paper, two other algorithms are given: the first requires $O(|E|^2 \log\log\log |V|)$ time and has an approximation ratio of $\frac{2}{3}$. The second is again a $\frac{2}{3}-$approximation algorithm and uses directed graphs; its time complexity is $O(|V|(|E| + |V| \log |V|))$.

Another approach to solve $NP-$hard problems is to redefine the optimisation criterion. A solution is said to be *minimal* (*maximal*) if it does not contain any other solution as a proper subset (*superset*). This property is equivalent to the minimality requirement and can be intuitively explained as a property of non

redundancy. Minimal solutions are easier to find than minimum ones and, depending on the context, this requirement is sufficient for obtaining useful solutions.

We then introduce a third problem:

Problem 4 *Minimal path cover*

Given an undirected graph, $G = (V, E)$, compute a vertex-disjoint path cover $C \subseteq E$, that does not contain another cover as a proper subset.

This problem can be efficiently solved with a greedy approach. In the following section, the biological application that first motivated our interest in covering problems is presented.

4.2 Biological application: the Scaffolding Problem

DNA sequencing is the process of determining the precise order of nucleotides within a DNA molecule and is an important task in comparative, functional and structural genomics.

The first DNA sequences were obtained in the early 1970s and a lot of research has been done since then. Nowadays, a number of different sequencing technologies are used (Illumina, Roche 454, SOLID...) and grouped in what is called *Next-Generation Sequencing* (NGS), or high-throughput sequencing. These technologies allow to sequence DNA and RNA much more quickly and cheaply than the previously Sanger method, but the fragments produced (*reads*) are also much shorter than before. In order to assemble the reads into longer sequences, a series of automatic procedures are applied that lead, at the end of the process, to a set of medium-size fragments called *contigs*. At this point, the genome finishing process does not depend on *de novo* assembly anymore and other techniques have to be applied to connect the contigs together in order to obtain the complete (*closed*) genome sequence.

The polymerase chain reaction (PCR) is a biochemical technology used to amplify a DNA fragment of which the initial and the final sequences are known. This technology can help to reconstruct the missing parts of a genome and fill the gap between the contigs. If two contigs are consecutive in the genome, it is possible to link them with a PCR but, since the set of contigs is not ordered, one does not know which fragments will be successfully linked experimentally.

Since each experiment is expensive in terms of both cost and time, the possibility of trying all the combinations between the contigs is not feasible.

It is also worth noticing that the two strands of DNA, initially connected, were separated at the beginning of the process, and the entire assembly procedure is done on single-stranded fragments.[1] This means that there is no guarantee that the contigs obtained all belong to the same strand. For closing the genome, it is, however, important to divide the contigs in two, depending on which strands they belong to; this is done by specifying the *orientation* in which the sequence must be read (as it is or "reverse and complement").

These are the reasons why a mathematical model that identifies the correct order among the fragments and the orientation of each single contig is important. A set of fragments ordered and oriented (but not yet experimentally closed) is called a *scaffold*, and the process of ordering and orientating a set of fragments is called *scaffolding*.

The preferred approach to genome scaffolding is currently based on assembling the sequenced reads into contigs and then using paired-end information to join them into scaffolds. Most of the methods based on such an approach have several preparatory steps in which read and contig libraries are first converted to a specific format, then mapped against each other by means of an external aligner (e.g. BWA, (Li and Durbin, 2009) or Bowtie, (Langmead et al., 2009)) and finally used to possibly join the contigs together.
At the end of this pipeline, a scaffolding graph is usually constructed and a plethora of different methods can be used to analyse the graph and produce the resulting scaffold structure. Currently available methods/software include Sopra (Dayarian, Michael, and Sengupta, 2010), Scarpa (Donmez and Brudno, 2013), Mip (Salmela et al., 2011), Opera (Gao, Sung, and Nagarajan, 2011), Grass (Gritsenko et al., 2012) and Sspace (Boetzer et al., 2011). A recent survey (Hunt et al., 2014) analyses and benchmarks most of these recent and sophisticated scaffolding software. The authors showed that, in general, they are not satisfying either in terms of usability or in terms of the quality of the solution, leading to the conclusion that there is still space for improvements in this area.

An alternative approach for scaffolding genomes relies on the use of a complete (closed) reference genome to guide the ordering and the orientation of

[1] This is not a limit since, given the nucleotide sequence of one strand, the second one can be uniquely reconstructed.

the contigs. Many available methods exist for mapping (and then scaffolding) the generated draft contigs (Galardini et al., 2011), (Darling et al., 2004), (Silva et al., 2013), (Hijum et al., 2005), (Kolmogorov et al., 2014). This approach is also used in some specific contexts, such as for ancient DNA fragments reconstruction (see for example (Husemann and Stoye, 2010) or (Rajaraman, Tannier, and Chauve, 2013)), where reads information is not available or reliable.

These software differ in terms of their overall strategy and implementation but, in general, i) they allow for only a single reference genome (e.g. (Galardini et al., 2011)); ii) when multiple genomes are allowed, these have to be closed (e.g. (Husemann and Stoye, 2010)); and iii) a reference phylogeny accounting for the evolutionary relationships among the selected taxa is to be provided to guide a multi-reference genome-based scaffolding (e.g. (Kolmogorov et al., 2014)). None of the above mentioned approaches is capable of ignoring all of these constraints that, taken together, represent important practical limitations. Indeed, with the exception of model organisms, reliable closed reference genomes are not always available. Moreover, especially in the case of bacteria, genomic rearrangements among closely related organisms may introduce important structural differences, hampering the scaffolding procedure based on a single genome as reference. Finally, the requirement of a reliable phylogenetic reconstruction can pose a significant challenge, since it is not always straightforward for some bacterial taxa for which the large genetic variability in gene content inside a same species can lead to very different phylogenies depending on which molecular marker and/or approach is used.

To overcome the difficulties that characterise currently available methods, we developed MeDuSa (Multi-Draft based Scaffolder), an algorithm for scaffolding draft genomes by ordering and orientating a set of *de novo* obtained contigs and thus speeding up genome finishing. This is presented in the next chapter.

Chapter 5

MeDuSa

In this Chapter, we present MeDuSa (Multi-Draft based Scaffolder), an algorithm for genome scaffolding. MeDuSa exploits the information obtained from a set of (draft or closed) genomes from related organisms to determine the correct order and orientation of the contigs. MeDuSa formalises the scaffolding problem by means of a combinatorial optimisation formulation on graphs and implements an efficient constant factor approximation algorithm to solve it. In contrast to currently used scaffolders, it does not require either prior knowledge on the microrganisms dataset under analysis (e.g. their phylogenetic relationships) or the availability of paired-end read libraries. This makes usability and running time two additional important features of our method. Moreover, benchmarks and tests on real bacterial datasets showed that MeDuSa is highly accurate and, in most cases, outperforms traditional scaffolders. MeDuSa can be found at `http://combo.dbe.unifi.it/medusa`.

In the previous Chapter, we listed the limits and problems of currently available scaffolding software. Before presenting in detail MeDuSa, our solution to the scaffolding problem, we want to give an overview of what characterises this method with respect to the available techniques: i) it formalises the scaffolding problem by means of a combinatorial optimisation formulation on graphs and implements an efficient constant factor approximation algorithm to solve it; ii)

it allows for multiple reference genomes to be used during scaffolding; iii) it does not require prior knowledge on the evolutionary relationships (i.e. a phylogenetic tree) among the reference set of organisms; and iv) it can handle both draft and complete reference genomes. This latter point is of great importance in practice since, in current public databases, the availability of draft genomes greatly exceeds that of completely sequenced ones (`www.genomesonline.org`). Moreover, since retrieving the additional information needed by the above mentioned scaffolders can be a challenging task, an algorithm that does not rely on such prior knowledge is of great interest and allows the inclusion of a larger set of genomes for the scaffolding process.

The strategy of MeDuSa is based on the intuition that a set of genomes related to the target one can be used for assigning a relative position to each contig, and that this kind of information is easily available in practice. Specifically, those contigs mapping on adjacent regions in these other genomes are considered to be neighbours in the resulting scaffold. MeDuSa formalises such scaffolding problem as a path cover problem in a graph and solves it with *ad hoc* optimisation techniques. The underlying algorithm has been implemented both in the form of a web-server and a stand-alone software.

Testing MeDuSa on different microbial datasets revealed that our software performs very well in comparison to others currently available and answers some of the implicit requests pointed out by Hunt et al. (Hunt et al., 2014) in their review, i.e. usability and accuracy of the obtained results.

5.1 Method

Definitions and notation

A *contig* is a fragment of a source DNA sequence. Let T be the *target genome* consisting of a set of n contigs $c_0, ..., c_{n-1}$.

An *ordering* of T corresponds to finding the true relative positions of the contigs c_i in the source sequence. The *orientation* of a contig indicates which strand of the source sequence it belongs to. We denote the reverse and complement of a contig c by $\bar{c}$.

Consecutive contigs in the ordering can be joined into a longer (gapped) supercontig called *scaffold*. The solution of a scaffolding problem consists of one scaffold per chromosome of the target genome. Formally, such goal can be

expressed as finding a partial order of the elements in T whose Hasse diagram is a collection of vertex-disjoint paths: each of these paths is a scaffold. In the ideal solution, a single scaffold for each chromosome is given. Consider in addition a collection $\mathcal{D} = \{D_0, ..., D_{k-1}\}$ of *comparison* genomes, where $D_0, ..., D_{k-1}$ are sets of contigs. Our algorithm is designed to determine a set of scaffolds on T and an orientation of its contigs by making use of the additional information provided by $\mathcal{D}$.

Let T and $\mathcal{D}$ be given. We map the contigs of T on the contigs of D_h, for all D_h in $\mathcal{D}$.

A contig c *hits* a contig d if it or its reverse and complement aligns to d (we call *hit* the subsequence between the first and the last matching positions of c on d). We use the software MUMmer (Kurtz et al., 2004) to align the contigs and recover hits with high similarity.

If c hits more than once the contigs of D_h, we call *best hit* of c on D_h the hit with maximum coverage.

Let us denote the first position of the best hit of c_i on D_h by p_h^i. We define also a boolean variable o_h^i which is $true$ if c_i is mapping straight and $false$ if it maps reverse.

Observe that the values p_h^i and o_h^i are defined if and only if the contig c_i hits D_h.

We are going to use these two kinds of information in different steps: the first one to assign an order to the contigs and the second to define an orientation for each of them.

A Combinatorial Optimisation Formulation

We construct an undirected weighted graph $G = (V, E)$ as follows. Let us associate a vertex to each contig, regardless of its orientation. We list all the best hits for every contig of the target genome on each contig of any comparison genomes in increasing order of their first positions.

If p_h^i and p_h^j are both defined, $p_h^i < p_h^j$, and there is no $l \in \{0, ..., k-1\}$ so that $p_h^i < p_h^l < p_h^j$, we say that c_i and c_j are *h-adjacent*. Let us define $A(c_i, c_j) = \{h : c_i \text{ is } h\text{-adjacent to } c_j\}$. There is an edge between v_i and v_j if $A(c_i, c_j) \neq \emptyset$, i.e. $E = \{(v_i, v_j) : c_i \text{ is } h\text{-adjacent to } c_j \text{ for some } h \in A(c_i, c_j)\}$.

The weight of an edge is given as $w(v_i, v_j) = |A(c_i, c_j)|$; since the cardinality of $\mathcal{D}$ is k, the weights range from 1 to k.

We call *Scaffolding Graph* the undirected weighted graph thus constructed.

Observe that an Hamiltonian path on the *Scaffolding Graph* can be interpreted as a scaffold of the entire set of contigs, in more general terms, a vertex-dispoint path cover of this graph can be interpreted as a set of scaffolds. This is why we decide to model our problem in terms of covering problems, already discussed in Section 4.1. Our first goal is to minimize the number of scaffolds, that is to minimize the cardinality of the cover. It is also true that we want to take in account the confidence of the edges. Since we already see that it is not possible to optimize at the same time the cardinality of the cover and its weight, a priority among the two values has to be chosen;

From a biological point of view, the goal of a scaffolding is usually to reduce the number of fragments of a given set of contigs. In this sense, the number of paths (scaffolds) has to minimised since the graph is weighted according to the confidence in each link between two contigs. Moreover, it is usually better to obtain a partial solution that is highly correct, instead of having a unique scaffold containing a high number of false positive joints. A more complete discussion about this choice will be presented at the end of this Chapter as future perspectives. Given this, we decided to look for a disjoint paths cover of maximum weight. In this formalisation, every path is a scaffold of ordered contigs. The *Contig Scaffolding Problem* can then be formulated as an instance of Problem 3. Unfortunately, this problem is NP-complete and we therefore opted for the most efficient approximation algorithm that is known.

Definition 10
Contig Scaffolding Problem

Given a scaffold graph $G = (V, E)$, determine a maximum paths cover of G and assign a direction to each path.

We implemented for this the most efficient algorithm presented in (Moran, Newman, and Wolfstahl, 1990), that gives a $\frac{1}{2}-$approximated solution.

By applying this algorithmic approach, we transform the graph G into a set of vertex-disjoint paths. The traversal of any path establishes a total order of the contigs of T in the path. Without loss of generality, let us consider an arbitrary but fixed order of the vertices. We start the traversal of any path from the vertex of degree one with lower index.

We obtain a set of directed paths that represent the scaffolds. We now need

to assign an orientation to each contig.

Orientation assignment

In this section, we take the orientation of the contigs into consideration. Suppose that v_i and v_j are adjacent in the graph, and $v_i < v_j$ in the order. As a consequence, $p_h^i < p_h^j$ for all $h \in A(c_i, c_j)$. For each $h \in A(c_i, c_j)$, one among the four possible relative orientations for the contigs is verified.

1. $o_h^i = true \wedge o_h^j = true$;
2. $o_h^i = false \wedge o_h^j = true$;
3. $o_h^i = false \wedge o_h^j = false$;
4. $o_h^i = true \wedge o_h^j = false$;

The *cover* constructed in the previous section can be seen, after the order assignment, as a set of directed paths. We assign to each arc $\langle v_i, v_j \rangle$ a label $l(v_i, v_j) = l$, with $l \in \{1, \ldots, 4\}$, that corresponds to the case most frequently observed for $h \in A(c_i, c_j)$. Observe that $l(v_i, v_j)$ gives a unique orientation for its tail (v_i) and its head (v_j). We denote by $tail(v_i, v_j)$ the orientation for c_i contained in $l(v_i, v_j)$ and by $head(v_i, v_j)$ the orientation for c_j.

Consider now two consecutive arcs $\langle v_1, v_2 \rangle$ and $\langle v_2, v_3 \rangle$ in a path. We say that the label assignment of the consecutive arcs is *consistent* if and only if $head(v_1, v_2) = tail(v_2, v_3)$, that is, if the two arcs propose a coherent orientation for v_2.

The orientation assignment for the contigs of T in a same scaffold is given by consistent label assignments for consecutive arcs.

Without loss of generality, let us suppose that we start the traversal of any path from the vertex with lower index. We initialise an empty scaffold. Then, if the label assignment of any two consecutive arcs is consistent, we add the contigs corresponding to the arcs in the scaffold with the orientation suggested by the labels; otherwise, if it is not consistent, we add the vertices of the first arc to the scaffold, then we cut the second arc, and start to traverse a new path.

The complexity of the entire procedure is linear in the number of vertices.

We will now discuss some examples on the orientation assignment. Consider the following path $v_1 < v_2 < v_3 < v_4$ and suppose that the most frequent orientations for the arcs are:

- for $\langle v_1, v_2 \rangle : (o_h^i = true \wedge o_h^j = false)$
- for $\langle v_2, v_3 \rangle : (o_h^i = false \wedge o_h^j = true)$
- for $\langle v_3, v_4 \rangle : (o_h^i = true \wedge o_h^j = false)$

The orientation assignment will be: $c_1, \bar{c_2}, c_3, \bar{c_4}$.

Observe that for the opposite direction for the path ($v_4 < v_3 < v_2 < v_1$), we have a symmetrical information (since the hits are always symmetric):

- for $\langle v_4, v_3 \rangle : (o_h^i = true \wedge o_h^j = false)$
- for $\langle v_3, v_2 \rangle : (o_h^i = false \wedge o_h^j = true)$
- for $\langle v_2, v_1 \rangle : (o_h^i = true \wedge o_h^j = false)$

We will thus obtain the following consistent orientation: $c_4, \bar{c_3}, c_2, \bar{c_1}$

Consider now the same path $v_1 < v_2 < v_3 < v_4$ but suppose that the most frequent orientations for the arcs are:

- for $\langle v_1, v_2 \rangle : (o_h^i = true \wedge o_h^j = false)$
- for $\langle v_2, v_3 \rangle : (o_h^i = false \wedge o_h^j = true)$
- for $\langle v_3, v_4 \rangle : (o_h^i = false \wedge o_h^j = false)$

There is no way to give a consistent orientation. The arc $\langle v_3, v_4 \rangle$ has to be discarded (the opposite direction is symmetric).

Observe that, for simplicity, we assumed for now that the most frequent orientation is unique but in some (rare) cases, this may not be true. If more than one relative orientation is detected with same frequency, we try all of them in case of inconsistency.

Multiple solutions

The solution to the path covering problem, even in the case of weighted edges, is not unique in general. This is due to the fact that the order in which the edges with same weight are processed influences the solution. In MeDuSa, the default method uses a stable sorting for the edges of the graph to avoid random behaviour. As an option, a second method is provided in which an arbitrary number of solutions is generated, keeping the one that induces the minimum number of scaffolds. We generate these solutions by doing a random

Dataset name	Organism	# replicons	# contigs	Reads	# drafts
BCEN	*B. cenocepacia* j2315	4	1223	In-house performed Illumina HiSeq	4
ECOL	*E. coli* K12	1	451	SRR001665 + SRR001666	25
RSPH	*R. sphaeroides* 2.4.1	7	564	SRR522246	2
MTUB	*M. tuberculosis*	1	116	In-house performed Illumina HiSeq	13

Table 5.1: Microbial datasets used for benchmarking.

sampling from the uniform distribution of the ordered permutations of the edges. Observe that, since many permutations can lead to the same solution, there is no guarantee about the distribution of the solutions themselves.

5.2 Experimental Results

In this section, we present the results we obtained when applying our software to benchmarks (see Table 5.1). In particular, we first analysed how MeDuSa performs on real genome scale datasets in terms of mistakes, completeness and number of reconstructed scaffolds, and how the choice of the draft genomes used for scaffolding influences the results. Then, we compared the performance of MeDuSa to those of five other scaffolders. In order to evaluate the reliability of the solutions generated by our algorithm, we chose real bacterial datasets for which (at least) one whole genome had already been completed, that is "closed", and used this as a positive reference. From now on we will refer to the following metrics to evaluate the results of our tests: i) number of correct joins, i.e. the number of true positives; ii) accuracy, the number of true positives divided by the number of proposed joins; iii) recovered information, the number of true positives divided by the expected number of joins; iv) overall number of reconstructed scaffolds; v) N50 and NG50 metrics; and vi) total length of joined fragments. Observe that the expected joins correspond to the number of contigs minus the number of chromosomes. Moreover a join between two contigs is considered correct if and only if: 1) the contigs are directly consecutive in the genome (no other contig appears in between and they belong to the same replicon); and 2) the orientation of the two fragments is correct.

Genome scale datasets

MeDuSa was tested on datasets of genomes from four microbial species (Table 5.1), each of which composed as follows:

Dataset	# c	# s	# proposed	# correct	# wrong	Recovered info	Lenght	N50
BCEN	1223	25 (19)	1198	1148 (96%)	50(11)	94%	7022736	1319133
RSPH	564	78 (46)	486	397(82%)	99(12)	66%	4224838	143091
ECOL	451	9 (6)	442	321(73%)	121(38)	71%	44425341	2386641
MTUB	116	1	115	105 (91%)	10(3)	91%	4338452	4338452

Table 5.2: Accuracy and completeness statistics. The columns of the table show: **# c**: number of contigs in the dataset. **# s**: number of scaffolds output by Medusa. **# proposed**: the total number of joints found by Medusa. **# correct**: number of correct joints, in parentheses the accuracy value is given. **# wrong**: number of correct joints, in parentheses is specified how many on them are inversions. **Recovered info**: the percentage of true informations recovered.**Lenght** in pb.**N50** stiatistic.

- a target genome (the draft genome to be scaffolded).
- a set of draft genomes from (more or less) closely related strains (named comparison genomes) to be used in the scaffolding pipeline of MeDuSa.

For each of the tested datasets, the target genome was obtained from the sequencing reads using Abyss (Simpson et al., 2009). Several *k*-mer values were tried for each dataset. The one leading to the best assembly (as described in (Fondi et al., 2014)]) was chosen and used as input for MeDuSa afterwards. Although genome assembly information is not necessary to use our method, we preferred building the target genome from reads in order to use exactly the same instance as input for the other programs during benchmarking (see section *Benchmarking*).

The results of these tests are summarised in Table 5.2. The general goal of reducing the fragmentation of the set of contigs is achieved surprisingly well. The number of fragments obtained after MeDuSa is applied is significantly smaller than the initial number of contigs. Also, in most cases, the majority of the scaffolds is composed of more than one original contig, that is, is multi-contig. Remarkably, in the case of the MTUB dataset (for which a complete genome was available among the comparison ones), the result is a single scaffold with an overall length close to the one of the input draft.

Influence of the taxonomical distance

The choice of the set of comparison genomes is left to the user and depends mostly on the organism under study. Nevertheless, some guidelines can be extracted from experimental analyses on the present datasets. The results displayed in Table 5.3 clarify how the phylogenetic distance between target and

Target: *E.coli*			
Genus of comparison	# Scaffolds	# wrong joins	recovered info
E. coli	9 (6)	122	71%
Escherichia	46 (20)	93	70%
Shigella	32 (14)	112	68%
Vibrio	439 (7)	10	0,4%
Pseudomonas	441 (7)	9	0.2%
Acinetobacter	451	0	0%

Table 5.3: Influence of phylogenetic distance between target and comparison drafts. The number of multi-contig scaffolds is reported in parentheses. The number of comparison drafts is always 15.

comparison genomes influences the scaffolding procedure. In all the 6 tests, the same draft genome (*Escherichia coli* K12) is used as a target. Each time a different set of comparison draft genomes, in increasing order of phylogenetic distance from the target but with a fixed size (15 genomes), is created and used in the pipeline. In particular, in (1) the comparison drafts belong to different strains of *E. coli*; in (2) they belong to the *Escherichia* genus (excluding *E.coli* species); in (3) organisms belonging to the *Shigella* genus are used; in tests (4) and (5) representatives of the *Pseudomonas* and *Vibrio* genera are used, respectively; finally, in (6), genomes from representatives of the genus *Acinetobacter* are used.

The results obtained from these tests indicate that completeness is quite affected by the phylogenetic distance between the comparison and the target genomes (Table 5.3). After a certain taxonomical distance, the information provided by the comparison draft is insufficient and the solution becomes very poor (the number of scaffolds is close to the initial number of contigs). This means that the comparison drafts should be chosen as close as possible to the target. In microbial genomics, this is usually not a problem because some more or less closely related draft genomes are likely to be present for (virtually) each newly sequenced genome. It is worth noticing that, when the information extracted from comparison genomes is not sufficient, a very few number of joins are proposed (false positives are very rare) and this avoids misleading suggestions in joining the fragments.

Varying the number of comparison genomes

The second parameter in the choice of the comparison dataset is the number of genomes to use. This aspect has been investigated using, again, *E. coli* K12

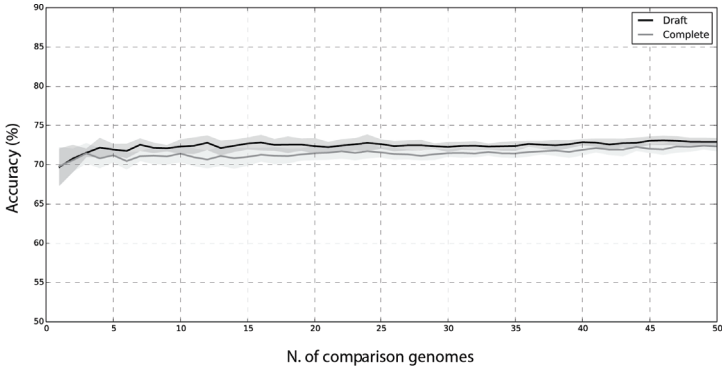


Figure 5.1: Variation of accuracy in respect to the number of comparison genomes. The gray shade along the lines represents the 95% confidence interval across all the performed permutations.

as target: 50 draft genomes belonging to this species where selected, and 50 different instances were built, with an increasing number of comparison drafts genomes (from 1 to 50) used during each test. This increase was performed consistently only adding new drafts to the previous set. Since the choice of the order in which the drafts are added could influence the solution, all the tests were repeated 10 times, each time varying the relative order of the comparison genomes. Moreover, since MeDuSa allows mixing closed and draft genomes in the comparison set, we tested how the presence of closed genomes affected the behaviour of the algorithm. To do this, another set of tests was performed using 50 closed genomes instead of drafts in the comparison set. For each dataset, the following values are presented: accuracy (Figure 6.1), recovered information (Figure 5.2) and number of scaffolds (Figure 5.3).

Interestingly, the symmetry of these results suggests that our method is sufficiently robust to noise created by redundant information. The small number of false positives is confirmed by the extreme stability of the accuracy level, shown in Figure 6.1. These considerations are true whether either closed or draft genomes are used as the comparison set. The use of closed genomes, as expected, gives more information and the completeness of the solution is higher. On the other hand, the accuracy in this case is slightly lower.

This can be explained by at least two lines of evidence. From a biological viewpoint, complete genomes may embed structural variations (e.g. duplicated

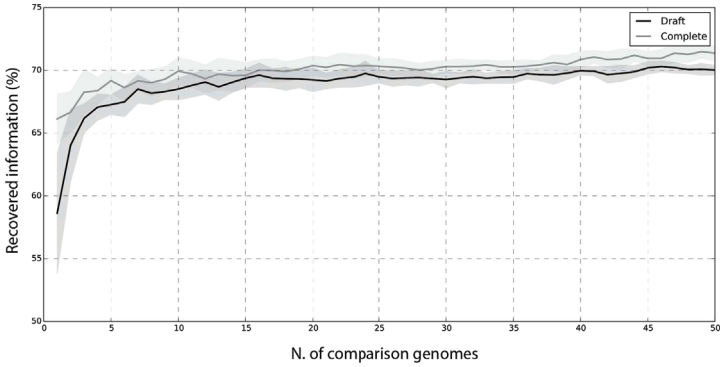


Figure 5.2: Recovered information in respect to the number of comparison genomes. The gray shade along the lines represents the 95% confidence interval across all the performed permutations.

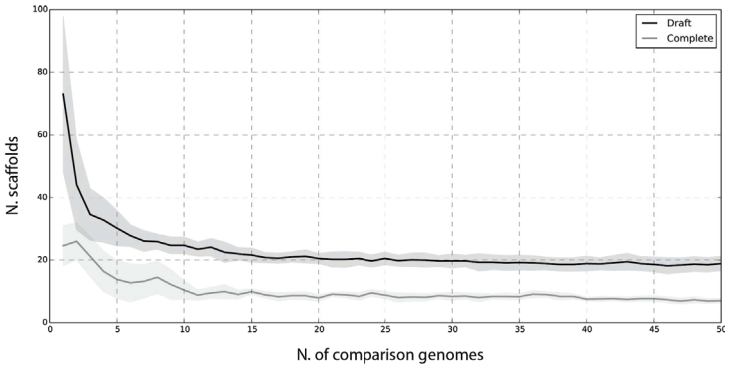


Figure 5.3: Number of scaffolds in relation to the number of (draft or complete) comparison genomes used. The gray shade along the lines represents the 95% confidence interval across all the performed permutations.

and/or inverted regions) that, due to *de novo* assembly issues, might not be observed in their fragmented draft counterparts. These biological features, in turn, may hinder the scaffolds reconstruction and possibly lead to wrong joins.

Moreover, from an informational viewpoint, including complete genomes in the comparison dataset may lead to an increased number of predicted joins and, consequently, to a higher false positive rate.

Benchmarking

The performance of MeDuSa was compared to those of five other programs, namely Sopra (Dayarian, Michael, and Sengupta, 2010), Scarpa (Donmez and Brudno, 2013), Opera (Gao, Sung, and Nagarajan, 2011), Sspace (Boetzer et al.,

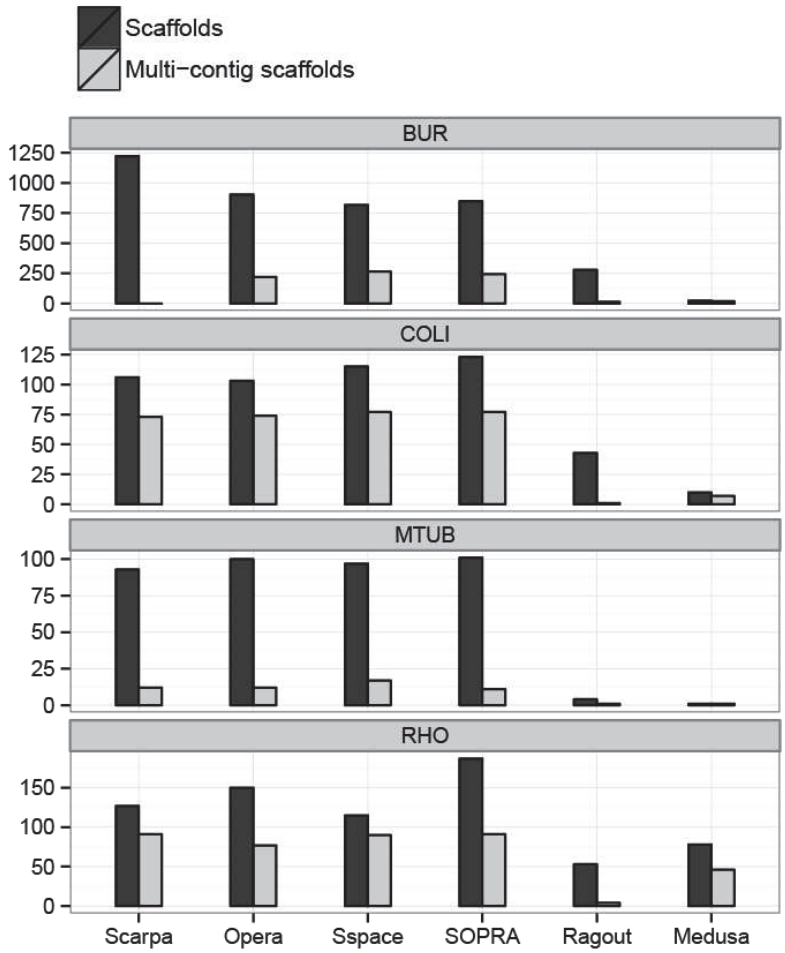


Figure 5.4: Comparison between the performances of MeDuSa and those from other selected scaffolders in terms of number of generated scaffolds and multi-contig scaffolds.

2011), ragout (Kolmogorov et al., 2014). The first four of these scaffolders are paired ends-based and the choice to use these specific ones was based on both their performances and their usability as assessed by (Hunt et al., 2014). The choice to use the recently developed ragout relies on the fact that it implements an overall strategy that resembles that of MeDuSa, although requiring more input information (phylogenetic tree of the analysed genomes). Options and parameters (e.g. the choice of reads mapper) for each of the paired ends-based methods were selected among those leading to the best performances on genome-scale data as reported in (Hunt et al., 2014). Each paired ends-based software was used both on trimmed (using DynamicTrimming from the SolexaQA package

(Cox, Peterson, and Biggs, 2010) and PHRED 30 as the quality threshold) and untrimmed reads datasets. Indeed reads trimming is usually performed after a sequencing run in order to remove poor quality bases although, in some cases, it may lead to a loss of information during scaffolding. We here report the values for the option – trimmed or untrimmed – leading to the best results in terms of the scaffolds assembled by MEDUSA. With the exception of insert length (that was set to its appropriate value for each dataset), all the other parameters were set to their default value. As for RAGOUT, the reconstruction of the reference phylogenetic tree was performed using OMA (Roth, Gonnet, and Dessimoz, 2008) with default parameters. As indicated by the results of these tests (reported in Figure 5.4), the number of scaffolds produced by our algorithm is lower than that produced by all the other four paired end-based scaffolders in all the performed tests. Notably, RAGOUT and MEDUSA produce similar results on each dataset, with the latter leading to a lower number of scaffold in the BCEN and ECOL datasets and both of them leading to a single scaffold with the MTUB dataset. What is particularly interesting is also the high percentage of multi-contig scaffolds over the total number of scaffolds reconstructed by MEDUSA (75%, on average), a crucial aspect since minimisation of the number of scaffolds is clearly the final goal of any scaffolding method. As expected, the analysis of the N50 metrics revealed that MEDUSA outperforms all the other paired ends-based scaffolders and produces results that are, in most cases, similar to RAGOUT (see Additional File 1). Additionally, in Figure 5.5, we report accuracy and recovered information for the software tested herein and for MEDUSA. This comparison revealed that our algorithm is capable of results that overlap (and, in some cases, outperform) those from other currently available programs, even in terms of reliability of the proposed solution.

In conclusion, both the very high percentage of true joins recovered and the low percentage of errors observed make MEDUSA very competitive with the other scaffolders in general, including those exploiting a similar strategy (i.e. RAGOUT). It is to be noticed, however, that MEDUSA requires far less information in respect to the aforementioned methods and this greatly increases its usability. Also, MEDUSA performs very well in respect to all the other benchmarked software in terms of required running time. Indeed, all the paired end-based tools generally have long running time due to their re-processing and read mapping stages and, on our datasets, none of them was able to complete the

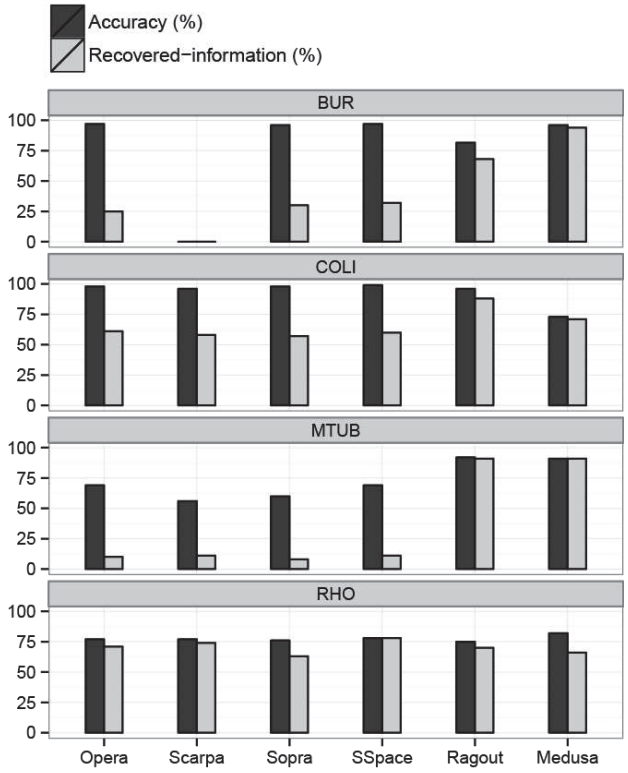


Figure 5.5: Accuracy and recovered info comparison among the benchmarked tools and MeDuSa. Observe that no scaffold was generated on the BCEN dataset by Scarpa

.

scaffolding in less than about 2 ours. Despite the fact that ragout processes input files far more quickly (23, 4, 16 and 90 minutes for the MTUB, RSPH, BCEN and ECOL datasets, respectively), it requires two operations that can be quite time consuming when dealing with a high number of genomes, i.e. computation of orthologous groups of sequences and phylogenetic tree reconstruction. The same datasets were scaffolded by MeDuSa in less than ten minutes, on average.

5.3 Conclusion and perspectives

Draft genome scaffolding is a key step in the finishing stages of microbial genomic pipelines, and automatic procedures in this field have a relevant impact on laboratory activity. We developed a method that, unlike traditional software, relies neither on paired-end information of sequencing reads nor on a phylogenetic distance between the micro-organisms used in the analysis. This

drastically increases the usability of the software while simultaneously reducing the computational time.

Using real microbial datasets, we show that the algorithm implemented in MeDuSa is in most cases capable of producing less and longer scaffolds in comparison to commonly used scaffolders, while maintaining high accuracy and correctness of the predicted joins. Nevertheless, many aspects of the method could be improved. Currently, the function that assigns a weight to every hit is boolean. This aspect could be improved by assigning different a weight to an hit that is proportional to its quality instead of using a fixed threshold. Additional information might also be used, such as gene synteny.

From an algorithmic point of view, one major question remains open. As mentioned in the Method section, besides the total weight of the cover, a second value should be optimised that corresponds to the number of components. The *Contig Scaffolding Problem* could thus be re-formulated as follows:

Definition 11
Contig Scaffolding Problem

Given a scaffold graph $G = (V, E)$, determine a minimum paths cover of G; if more than one exists, determine the one of maximum weight.

Unfortunately this problem is NP-complete and no approximation algorithm is known for it.

It is possible to reduce Problem 11 to Problem 10 in the following way.

Let $G = (V, E)$ be the scaffolding graph, w the function that assigns a weight to each edge and $S = w(G)$ the total weight of G. Starting from the original weights, we define a new function $\bar{w}(e) = S + w(e)$. With this change, the paths cover of G with maximum weight, $\bar{w}^*$, has also minimum size. We recall that maximising the number of edges of a cover is equivalent to minimising its cardinality.

Theorem 1
Let $G = (V, E)$ be a weighted graph with weight function w, and for each $e \in E$, let $\bar{w}(e) = S + w(e)$ with $S = w(G) = \sum_{e \in E} w(e)$. If P is a paths cover of G of maximum weight $\bar{w}^* = \bar{w}(P)$, then P has minimum size.

Proof. Let P be a cover with maximum weight $\bar{w}^* = \bar{w}(P)$ and let m be its number of edges. Consider any paths cover P' of G such that $\bar{w}(P) \geq \bar{w}(P')$,

and let m' be its number of edges. We can prove that $m \geq m'$. We have that $\bar{w}(P) = Sm + w(P)$ and $\bar{w}(P') = Sm' + w(P')$, and thus that:

$$\bar{w}(P) = Sm + w(P) > Sm' + w(P'),$$

that is:

$$w(P) > S(m' - m) + w(P').$$

By absurd, if $m' > m$, we obtain $w(P) > S(m' - m) + w(P') > S$ which contradicts the assumption. Therefore $m' \leq m$. As a consequence, the size of P' is greater or equal to the size of P. □

By Theorem 1, the *Contig Scaffolding Problem* can be addressed by solving the maximum weight path cover problem for G with respect to the new weight function $\bar{w}$. However, the previous reduction clearly does not preserve approximation, which means that the optimum weight gives the optimum number of components but the value of the weight in an approximate solution that does not guarantee anything about the ratio between the cardinality of the approximate solution and the one of the optimum.

We believe that finding an efficient approach to this last problem could be very interesting in practice and not only for this specific application.

Chapter 6

Hitting set problem and its implicit formulation

During the development of the model presented in Chapter 5, we considered various possible formulations of the scaffolding problem. Apart from the vertex-disjoint paths problem, we considered other candidates, such as spanning tree and independent set.

More interestingly, the collection of such problems can be translated into a unique general framework, called *hitting set approach*. This led us to further investigate this mathematical problem and dedicate part of our research to some open problems related to it. A final goal is to study the whole family of hitting set problems, first the more general ones, and then constructing a hierarchy in terms of time complexity. We focused our attention on the recent *implicit* version of the problem, and in particular on the enumeration of minimal solutions. In this Chapter, we formalise the problem in different ways and prove that, in its most general formulation, the enumeration is NP-hard. In Section 6.3, a series of open questions and lines of research are presented. In particular, we would like to investigate the problem in two directions: *top-bottom*, by identifying under which conditions the problem is no longer NP-complete in terms of enumerating solutions; and *bottom-top*, by defining some sufficient conditions under which the problem is polynomial

time solvable. We know that some sub-cases of the implicit hitting set problem are efficiently solvable and this guarantees that such a condition must exist.

6.1 Definitions and relation to the classical problem.

Hitting set problem

Let us first define the classical hitting set problem.

Problem 5 *Minimum Hitting Set Problem*

Given a universe $U = \{e_0, ..., e_{n-1}\}$ and a collection $T = \{S_0, ..., S_m\}$ of subsets of U, find a set of minimum cardinality $H \subseteq U$ so that $\forall i(H \cap S_i \neq \emptyset)$.

Let us consider now an example of how the vertex-disjoint paths cover problem can be translated in terms of a hitting set. Given a graph $G = (V, E)$, with $U = E$ and T the set of triplets of edges incident to a vertex, determining a hitting set for the couple $\langle U, T \rangle$ can be used to find a set of edges to be removed in order to obtain a vertex-disjoint paths cover. This is because the composition of T guarantees that at least one edge has been removed from all the triplets incident to a same vertex, meaning that all the vertices have degree < 3.

In this way, we obtain a sub-graph where all the degrees are bounded. This is a first step towards finding a vertex-disjoint paths cover, but it is not sufficient. Indeed, observe that it is possible that a component of the sub-graph is a simple cycle. (all the degrees are equal to 2). If we want to guarantee that also this special kind of cycles are avoided, the instance has to be enriched as follows.

We define two sets: T_1 is the set of triplets of edges incident to a vertex and T_2 is the set of simple cycles of G. Let $U = E$ and $T = T_1 \cup T_2$. In this way, not only the degree of each vertex is reduced, but at least one edge for each simple cycle will be removed. Observe that a minimal hitting set built in this way, when removed, leaves a paths cover of minimal cardinality of G. See Figure 6.1 for an example.

We have a hitting set formulation of our problem, however this instance cannot be built in polynomial time since the number of cycles can be exponential. Observe also that, on the contrary, a polynomial test exists to check if a given set of edges is a hitting set: given a set of edges H, consider the graph $G' =$

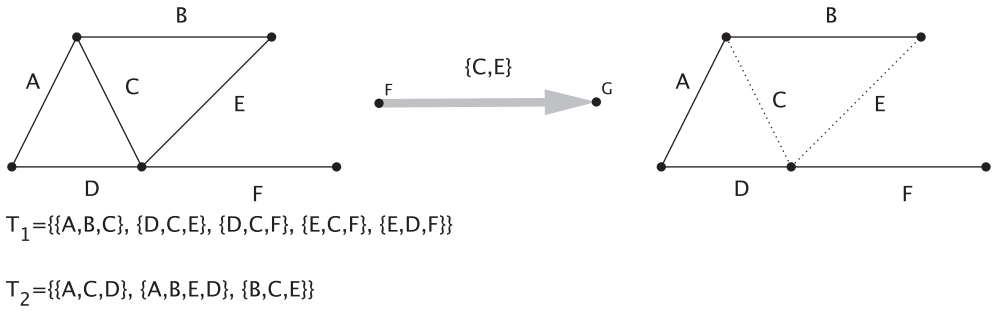


Figure 6.1: An example of a covering problem translated in terms of a hitting set. The universe is $U = E$, the family to hit is $T = T_1 \cup T_2$, where T_1 is the set of triplets of edges incident to each vertex and T_2 the set of simple cycles of the graph. A subset of E that hits all the elements of T, when removed, leaves a path. In this example, the set $\{C, E\}$ is a hitting set and its complement is indeed a single path.

$(V, E - H)$. H is a hitting set if and only if G' is acyclic and its maximum degree is 2. This is an example of what leads to the definition of an implicit version of this problem. In the next Section, various formal definitions of the *Implicit Hitting Set* problem are given and the relationship between the different existing formulations is discussed.

The Implicit Hitting Set problem was defined for the first time in Chandrasekaran et al., 2011. It differs from the classical formulation by the fact that the family of sets to be hit is not explicitly given. Instead, an oracle is provided which, given a candidate H, determines if H is a hitting set. This is a generalisation of the classical problem motivated by the fact that, in practice, the collection of sets is typically too large to be listed explicitly, but in many cases one way to decide whether a candidate is a hitting set or not can be designed efficiently.

The idea of applying Implicit Hitting Set to biological problems is not new, actually its first application was presented in Moreno-Centeno and Karp, 2013, and corresponds to a version of the multi-sequence alignment problem. In the above paper, the *Minimum Implicit Hitting Set* problem (see Section 6.1) is solved with a heuristic. Up until now, no approximation algorithm is known for this problem.

The implicit formulation

The implicit version of HITTING SET PROBLEM can be defined as follows:

Problem 6 *Minimum Implicit Hitting Set Problem*

Given a universe U and a collection T of subsets of U, find a hitting set of minimum cardinality when T is not given explicitly. In its place, a polynomial oracle is provided that, for any $H \subseteq U$, decides whether H hits T.

Notice that an implicit version is theoretically more difficult than an explicit one, since it is always possible to build a polynomial oracle from the explicit family. This means that, given the hardness of finding a minimum hitting set, also the MINIMUM IMPLICIT HITTING SET problem is NP-hard. This is not true if we look for a *minimal* solution instead of a *minimum* one.

A hitting set is said to be minimal if there is no element that can be removed without loosing the property of being a hitting set. Indeed, the following simple greedy approach finds a minimal hitting set in polynomial time: consider the candidate set $H \subseteq U$. At the beginning $H = U$ (the entire universe is always a hitting set). For each element $e \in U$, evaluate (with the oracle) whether $H - \{e\}$ is a hitting set. If yes remove the element, and start again with $H = H - \{e\}$. At the end, H will clearly be a minimal hitting set.

Enumerating all the solutions of an NP-hard problem is *a fortiori* NP-hard also. This means that we have no hope to find a polynomial total time algorithm to list all minimum hitting sets, but it is still interesting to investigate the possibility to list all minimal hitting sets.

In order to define Problem 6, we use a very general definition of oracle as a boolean function. The oracle formalisation can be strengthened, potentially influencing the complexity of the problem. We decided to consider two versions: the first one is characterised by a weak oracle and the second by a strong oracle.[1]

We then define the following two problems, that are actually our subject of interest:

Problem 7 *eak) Implicit Hitting Set Problem*

Given a finite universe U and an oracle that, for any $H \subseteq U$, decides whether H is a hitting set, list all the minimal hitting sets.

[1] It is important to notice that the NP-completeness of finding a minimum implicit hitting set still holds if Problem 6 is defined with a strong oracle.

Problem 8
(Strong) Implicit Hitting Set Problem

Given a finite universe U and an oracle that, for any $H \subseteq U$ decides whether H is a hitting set and if not, returns a set that is not hit as a certificate, list all the minimal hitting sets.

We now have three problems:

A) List all the minimal solutions with a weak oracle for T.

B) List all the minimal solutions with a strong oracle for T.

C) List all the minimal solutions with the explicit definition of T.

The relationship among these three problems is the following:

$$A \geq B \geq C$$

6.2 NP-hardness results

As a first result, we are going to prove an hardness result for Problem 7 Since we are dealing with enumeration problems, we will slightly abuse of terminology by saying that, for example, Problem 7 is NP-hard. More precisely, this means that no polynomial total time algorithm exists for Problem 7, unless $P = NP$.

Our strategy is to define a new concept, that we called *Implicit Hitting System*, which establishes a duality between *hitting set problems* and *independent set problems*. We then prove that enumerating all minimal hitting sets is equivalent to enumerating all minimal elements of a hitting system, which itself is equivalent to enumerating all minimal elements of an independent system. This last problem has been proven to be NP-complete.

Hitting System

Definition 12
Hitting System

We call $(U, \mathcal{F})$ a *hitting system* if and only if:

1. U is finite.
2. $\mathcal{F}$ is not empty.
3. $(h \in \mathcal{F} \wedge h \subseteq h') \Rightarrow h' \in \mathcal{F}$ (**Property 1**).

Problem 9
Implicit hitting system probl

Given a hitting system where $\mathcal{F}$ is not given explicitly and an oracle that, given $H \subseteq U$, decides in polynomial time whether $H \in \mathcal{F}$, list all the minimal elements of $\mathcal{F}$.

Equivalence between implicit hitting set and implicit hitting system.

In order to reduce Problem 9 to our original one, we have to show that, given an implicit hitting system $(U, \mathcal{F})$, we can always define a collection T of subsets of U for which the minimal elements of $\mathcal{F}$ are the family of minimal hitting sets for (U, T). In this way, solving the minimal hitting set enumeration problem would be equivalent to enumerating the minimal elements of $\mathcal{F}$.

Theorem 2

Given a finite universe U and a non-empty family of subsets $\mathcal{F}_1$ for which Property 1 holds, there exists another family $\mathcal{F}_2$ of subsets of U such that the set of minimal elements of $\mathcal{F}_1$ is the set of minimal hitting sets of $\mathcal{F}_2$.

Proof. It is known that a family of subsets of a given universe can be represented by a hypergraph. A *simple* hypergraph is a pair (V, E) where the elements of E are subsets of U and for any couple of edges E_1, E_2, we have that $(E_1 \not\subset E_2)$.

Given $\mathcal{F}_1^m$ the set of minimal elements of $\mathcal{F}_1$, the pair $(U, \mathcal{F}_1^m)$ is a simple hypergraph.

The *transversal* of a simple hypergraph $tr(G)$ is another simple hypergraph where the edges are exactly all the minimal hitting sets of G (Eiter, 1994). The function tr, that transforms a hypergraph in its transversal, was proved to be idempotent, so that $tr(tr(G)) = G$.

This guarantees that the edges of $tr((U, \mathcal{F}_1^m))$ define a family for which $\mathcal{F}_1^m$ is exactly the set of minimal hitting sets; this is the $\mathcal{F}_2$ family we are looking for. In other words, listing all the minimal elements of $(U, \mathcal{F})$ is equivalent to listing the minimal hitting sets of (U, E) where E are the edges of $tr((U, \mathcal{F}_1^m))$. □

Theorem 3

Enumerating the minimal sets of an implicit hitting system is equivalent to listing all the minimal implicit hitting sets.

Proof. From Theorem 2. □

Theorem 3 gives us the relationship between the Hitting Set and the Hitting System enumeration problems. In this way, proving that the latter is NP-hard allows us to state that enumerating minimal hitting sets with a boolean oracle is NP-hard. In the following section, we are going to reduce the Hitting System enumeration problem to the Independent System enumeration problem, that has been proven to be NP-complete. (Lawler, Lenstra, and Kan, 1980)

Independent and Inclusion systems are dual in terms of enumeration

Definition 13
Independent System

We call $(U, \mathcal{F})$ an *Independent System* if and only if:

1. U is finite.
2. $\mathcal{F}$ is not empty.
3. $(h \in \mathcal{F} \wedge h' \subseteq h) \Rightarrow h' \in \mathcal{F}$ (**Property 2**).

Problem 10
Independent System Problem

Given an independent system $(\mathcal{U}, \mathcal{F})$, enumerate all the maximal elements of $\mathcal{F}$.

Given a Hitting System (U, F_1), we want to construct an Independent System such that a bijection exists between the minimal elements of the first and the maximal elements of the last. More formally, given a Hitting System (U, F_1) (Property 1 holds), let $F_2 = \{\overline{H} : H \in F_1\}$, where $\overline{H} = U - H$. We prove the following proposition:

Proposition 3
Duality between hitting set and independent sets.

P1) F_2 *is finite and not empty.*

P2) $(S \in F_2 \wedge S' \subseteq S) \Rightarrow S' \in F_2$.

P3) A bijective function f *exists such that* H *is a minimal element of* F_1 *if and only if* $f(S)$ *is a maximal element of* F_2.

We can prove that all the three points above are true:

Proof. $P1$ is trivially true. We have to prove $P2$. Remember that, by definition, for each element $H \in F_1$, the following holds: $H \subseteq H' \Rightarrow H' \in F_1$ (Property 1). Consider an element $S \in F_2$ and $S' \subseteq S$. We know, by construction, that $S = \overline{H}$

for some $H \in F_1$. Moreover $\overline{S} \subseteq \overline{S'}$, that is $H \subseteq \overline{S'}$. By Property 1, $\overline{S'} \in F_1$, and by construction, we have that $S' \in F_2$.

It remains to prove $P3$. If S is a maximal element in F_2, we can show that $\overline{S}$ is a minimal element of F_1. Assume, by absurd, that $\overline{S}$ is not minimal: this would mean that an element $e \in \overline{S}$ ($e \notin S$) exists such that $\overline{S} - \{e\} \in F_1$. But this would mean that $\overline{\overline{S} - \{e\}} = S \cup \{e\} \in F_2$ which violates the maximality of S. In the other direction, if H is a minimal element of F_1, we can show that $\overline{H}$ is a maximal element of F_2. Assume, by absurd, $\overline{H}$ is not maximal: this would mean that an element $e \in H$ exists such that $\overline{H} \cup \{e\} \in F_2$. But this would mean that $\overline{\overline{H} \cup \{e\}} = H - \{e\} \in F_1$ which violates the minimality of H. □

We have now all the elements that prove our main result. We first reduced the NP-hard Problem 10 to a new one (Problem 9) and reduced the latter to our original Problem (Problem 7). This chain of results prove the following theorem:

Theorem 4

There is no polynomial total time algorithm for Implicit Hitting Set *problem with a weak oracle, unless $P = NP$.*

An alternative proof of Theorem 4, where satisfiability is reduced to 9, is given in Chapter 6.3.

6.3 Conclusions and perspectives

The Implicit Hitting Set is a powerful and interesting framework and the NP-completeness of its more general formulation is, by itself, not satisfying as a clarification of the computational nature of this problem. The characterisation of the enumeration of implicit hitting sets remains open. Our lines of research concerning this topic follow two directions. First we would like to answer the question "what makes the problem hard?". In this sense, we need to clarify the relationship between Problem 7 and Problem 8 as in principle a stronger oracle can make the problem tractable. The proof cannot be immediately adapted for the strong oracle definition. On the other hand, it is not easy to use the additional information (the certificate) to design a polynomial delay algorithm. Our feeling is that the NP-hardness still holds and that the oracle with certificate would not provide any substantial contribution to the complexity of the problem.

The second question we want to ask is "What makes the problem easy?". It is known that some enumeration problems, belonging to the family of hitting set problems, can be solved with a polynomial total time algorithm, or even with a polynomial delay one.

The following provides some examples:

Problem 11
Maximum spanning trees enumeration.

Given a weighted graph G, enumerate all the acyclic subgraphs of G of maximum weight.

Problem 12
Minimum feedback arc set enumeration.

Given a weighted graph G, enumerate all the sets of arcs of minimum cardinality that make the graph acyclic when removed.

A third example for which a polynomial delay algorithm is known is the minimal feedback vertex set problem. [2]

Problem 13
Minimal feedback vertex set enumeration.

Given a directed graph G, enumerate all the minimal subsets of its vertices that contain at least one vertex of any directed cycle.

For all the three previous problems, a polynomial delay algorithm is known (Schwikowski and Speckenmeyer, 2002), and all of them are sub-problems of the hitting set enumeration problem.

Indeed the following problems are equivalent to minimal feedback arc set and minimal feedback vertex set respectively:

Problem 14
Hitting set formulation of mFAS

Given a directed graph $G = (V, A)$, find a minimum hitting set for $(V, \mathcal{C})$ where C is composed by the set of vertices of any simple cycle of G.

Problem 15
Hitting set formulation of mFVS

Given a directed graph $G = (V, A)$, find a minimum hitting set for $(V, \mathcal{C})$ where C is composed by the set of edges of any simple cycle of G.

Given that an efficient enumeration of hitting sets is possible in some cases, a first question is then: it it possible to identify a property that makes the

[2] Observe that the *minimum* feedback vertex set problem is instead NP-complete.

enumeration of implicit hitting sets easy? The second is: can we prove that other problems belong to the subset of efficiently solvable ones?

The duality between Hitting Systems and Independent Systems (Proposition 1) helps us to identify a first sufficient condition under which the problem of enumerating minimal hitting sets becomes easy. We considered a special class of independent systems called *matroids*.

Definition 14 *Matroid*

$M = \langle U, F \rangle$ is a matroid if and only if:

1. M is an Independent System.
2. $S, S' \in F \wedge |S| > |S'|$ then an element $e \in S$ exists so that $S' \cup \{e\} \in F$. (Augmentation Property)

Given a matroid $M = \langle U, F \rangle$, we call a maximal element of F a *basis* of M. The basis of a given matroid can be enumerated in polynomial delay. From this fact and from the duality given by Proposition 1, we proved that:

Proposition 2

Given a hitting system $M = \langle U, F \rangle$, it is possible to efficiently enumerate all the minimal elements of F if for $\langle U, \bar{F} \rangle$ the augmentation property holds.

We believe that this duality could lead to some interesting clarification of the general problem we are investigating.

Another direction to follow is to adapt known enumeration techniques to new cases. These techniques (Schwikowski and Speckenmeyer, 2002; Avis and Fukuda, 1996) can be applied to a series of problems and provide a polynomial enumeration algorithm for the set of solutions. Usually a meta-structure is built such that the vertices represent the solutions and the edges represent a successor relation between solutions. Then a polynomial total time algorithm for traversing this structure is given.

These methods suggested to us another way to characterise a subset of easy problems. Intuitively, this property consists in the fact that, given an optimal solution, it is possible to do local changes in order to find another set of optimal solutions. This can then be formalized in the following notion of polynomial *successor* function.

Definition 15
Enumeration meta-structure.

Given a problem P and its set of optimal solutions $\mathcal{S}$, the *successor* function $s : \mathcal{S} \to \mathcal{P}(\mathcal{S})$, is a function which associates to each optimal solution a set of optimal solutions. This set has size polynomial in the size of a solution and has to be found in polynomial time with respect to the size of the solutions. We build a directed meta-graph Ψ, where the vertices are the elements of $\mathcal{S}$ and the arcs are induced by s. Moreover an additional property is required for s: given two optimal solutions (vertices) s_1 and s_2, there must be a path in Ψ between them.

If such a meta-structure can be defined, a polynomial delay algorithm exists for P since the algorithm presented in Schwikowski and Speckenmeyer, 2002 can be used. These techniques have already been applied to sub-problems of the hitting set enumeration and could be used to find original enumeration algorithms. In particular, we focused our attention on the following problem:

Problem 16
Maximum cut

Given a graph $G = (V, E)$, find a set of nodes S for which $C = \{(u, v) \in E : u \in S \wedge v \in \bar{S}\}$ has maximum size.

This problem is equivalent to the following ones:

Problem 17
Odd cycle transversal

Given a graph $G = (V, E)$, find a minimum hitting set for $(V, \mathcal{C})$ where C is composed by the set of edges of any odd cycle of G.

Problem 18
Maximum bipartition

Given a graph $G = (V, E)$, find the maximum set of edges that induces a bipartite sub-graph.

We are currently working on defining a *successor function* for this problem in order to apply the algorithm described in Schwikowski and Speckenmeyer, 2002.

Conclusions

We can try now to summarise what has already been done and what has yet to be done concerning the topics I touched in my PhD studies, starting from the *co-phylogeny reconstruction problem*. This problem is well known in computational biology and an extensive literature was already presented when we started working on it. At the very beginning, the interest for this topic was driven by experimental reasons but we soon discovered that many theoretical questions were still not solved and, even more surprisingly, that the currently available software is not satisfying in terms of efficiency and correctness for the enumeration of optimal solutions. We then decided to focus on this topic on purely theoretical terms and we proposed some original algorithms for doing a co-phylogenetic analysis.

The first approach we used is the event-based parsimony one. In this field, many algorithms had already been proposed but a clear analysis in terms of complexity and an efficient and correct implementation were missing. We then designed a *polynomial delay* enumeration algorithm for finding all the minimum-cost solutions without repetitions.

This algorithm is implemented in a publicly available software, *Eucalypt*. The reduced running time of our algorithm is not only an interesting feature for the experimental biologist, but allowed us also to perform some extensive tests on some big datasets, in order to investigate the behaviour of the event-based model itself. We found out that the multiplicity of optimal solutions can be an issue in practice, something that had never been discussed extensively. Moreover the choice of the cost for the different evolutionary events has a big influence on the

results and this also must be taken in consideration. We found out that, on real datasets, the number of different optimal solutions can be indeed so high that the reliability of a choice of a single one can be very weak. Many papers in the current literature use parsimonious reconstructions without underlying enough the fact that they have chosen it among a lot of solutions, that are equivalent in mathematical terms, and without explaining the rationale for the choice.

A first response in this direction has been to define a new version of the problem in which the host switches are bounded by a certain distance. This distance can be adjusted by the user in order to restrict the choice to more realistic solutions. As discussed in Section 2.3, many other enrichments could be done to make Eucalypt more customisable and allow to discriminate more among the solutions. We hope to integrate some of these features in the next version of the tool.

A second response to the weaknesses of the event-based approach has been to develop a statistical model, based on a likelihood-free method, that allows to infer reasonable costs for different events and, at the same time, to have a first description of the host-parasite relationship in terms of the more probable events. The algorithm is implemented in the software Coala. An important role in this approach is played by an original algorithm that, given a phylogenetic tree for the host, simulates the evolution of the parasites. The simulated trees are then evaluated with respect to a novel distance over phylogenetic trees. We defined this distance trying to find a compromise between efficiency and sensibility, but we believe that other ideas could be investigated in this direction. Since different situations are characterised by different needs, we find it interesting to present new phylogenetic distances and to analyse them also in terms of distribution (for random trees) and practical performances, and not only in terms of theoretical complexity bounds. A deeper analysis of the distance defined for Coala and the proposal of new phylogenetic distances seem to us interesting topics for the future.

From a more theoretical point of view, some important questions about the *co-phylogeny reconstruction problem* have to be considered in the future. In particular we would like to investigate the time complexity of the k-bounded problem and to look for an approximation algorithm for the NP-complete problem of finding an optimal *acyclic* solution. Observe that, as we have shown in 2.2, Eucalypt can be considered already a good heuristic because it finds almost always the

optimal acyclic solutions for real datasets, but from a theoretical point of view the approximability of the problem is still unknown.

We are convinced that these questions have no easy answers, and this is one of the reasons that made us consider other biological applications in parallel to co-evolution. We decided to focus our attention on the *contig scaffolding problem*. This problem, as all the aspects of DNA sequencing that can be formalised and automatised, has received a lot of attention recently, with the advent of next generation sequencing techniques. We proposed a new method that tries to answer to some known issues of this field. In particular, our software MeDuSa requires little information and uses an efficient algorithmic approach. The simplicity of our mathematical model allows the user to order and orientate very rapidly (locally or via a web interface), a great number of DNA fragments. The accuracy and the completeness of our results are very good in comparison with other techniques. We are currently working on a second version of the code that embeds an automatic search for the comparison drafts and explores some other possibilities for the algorithmic approach.

During the development of this last project, we encountered some interesting open questions about the computational complexity of the *implicit hitting sets enumeration* problem. We distinguished two different formulations of this problem and proved that, in its general definition, the enumeration problem is hard. We are trying to adapt the proof to a stronger definition.

We are also trying to define sufficient conditions that make the problem tractable. We identified two properties that make the enumeration of minimal hitting sets feasible. The first condition is satisfied if the hitting system is the dual of a matroid. The second condition requires that a polynomial successor function between solutions can be defined. These two properties are quite strong, and our hope is to unify them in a more general condition. On the other hand, these conditions can already help us to design some new polynomial enumeration algorithms. We are currently focusing on the enumeration of *maximal cuts*, that is equivalent to enumerating *minimal bipartitions* in a graph.

Additional Material

Additional Material for Chapter 2

Example of time-feasible optimal reconciliation not found by CoRe-Pa.

Here we provide a time-feasible reconciliation for the dataset "Smut Fungi & Caryophillaceus plants" Refrégier et al., 2008 with cost vector $\langle 0, 1, 1, 1 \rangle$, that is not found by CoRe-Pa.

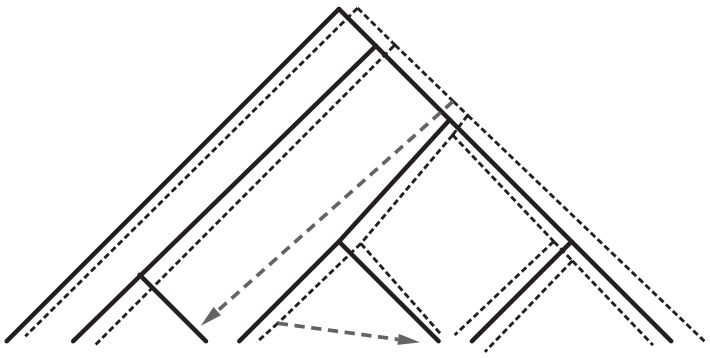


Figure 1: Example of a time-feasible reconciliation not found by CoRe-Pa

Statistics on the optimal reconciliations

Here we show for each dataset and for each cost vector $\langle c_c, c_d, c_s, c_l \rangle$ some statistics concerning the optimal reconciliations.

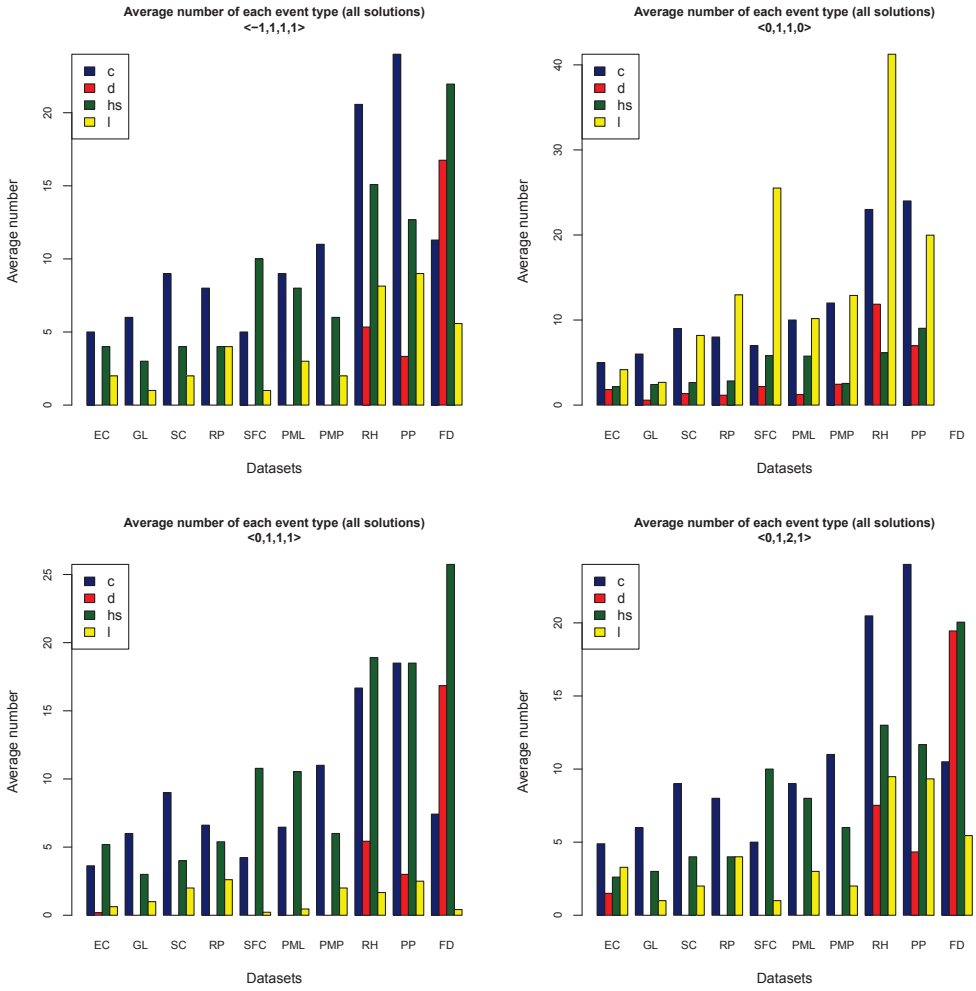


Figure 2: For each dataset we show the average number of each event in all optimal reconciliations.

Additional Material for Chapter 3

Choosing the parameters for the distance measure

Due to the high number of simulated trees which are compared to the "real" parasite tree, Coala needs a distance that is fast to compute. Moreover, the distance must take into account both of the following: (i) how representative is the simulated tree of the vector that generated it, and (ii) how topologically similar is the simulated tree to the real parasite tree.

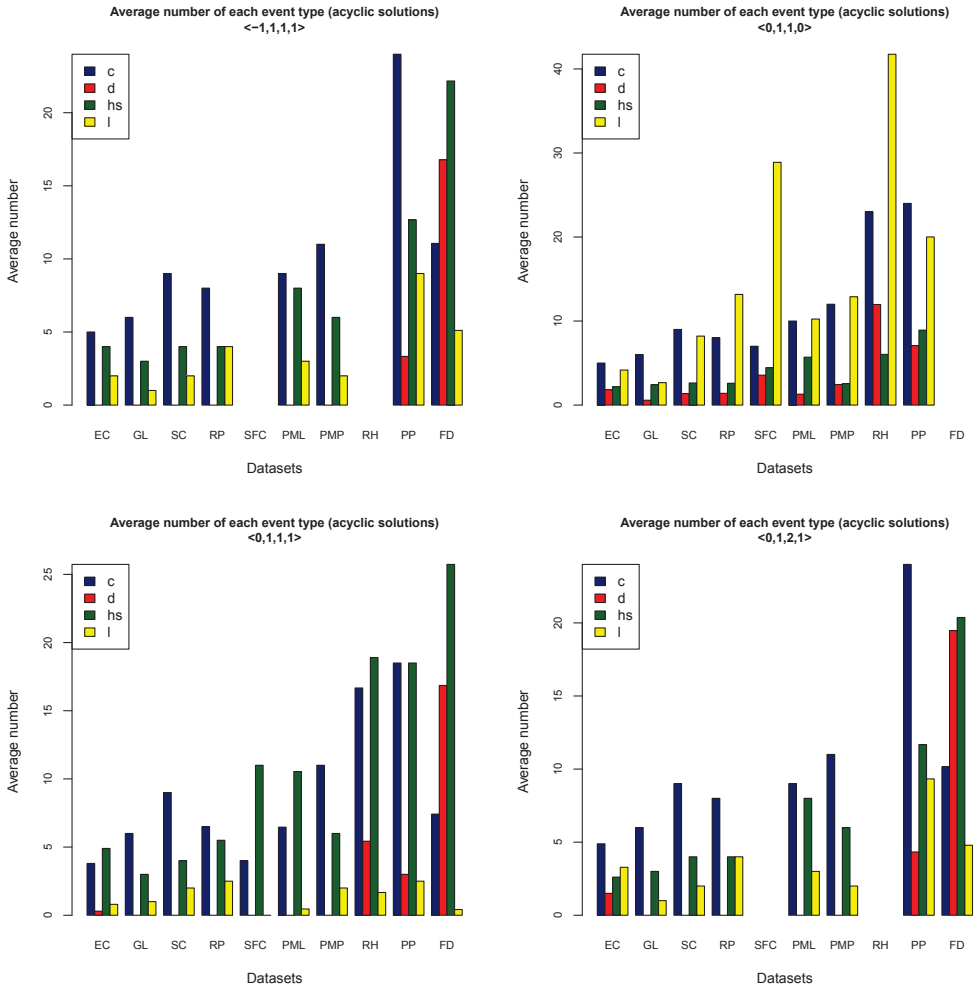


Figure 3: For each dataset we show the average number of each event in all acyclic optimal reconciliations.

Concerning the first point, the intuition is as follows. In our model, when generating a parasite tree, the expected frequency of an event should be close to the corresponding probability value of the parameter vector used to generate the tree. To this purpose, for a given vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$ and for each simulated tree P_θ that was generated according to this vector, we kept track of the number of events $obs = \langle o_c, o_d, o_s, o_l \rangle$ associated to this simulation. We compared the observed number of events to the expected $exp = \langle e_c, e_d, e_s, e_l \rangle$. Observe that the expected number of events can be easily calculated using the

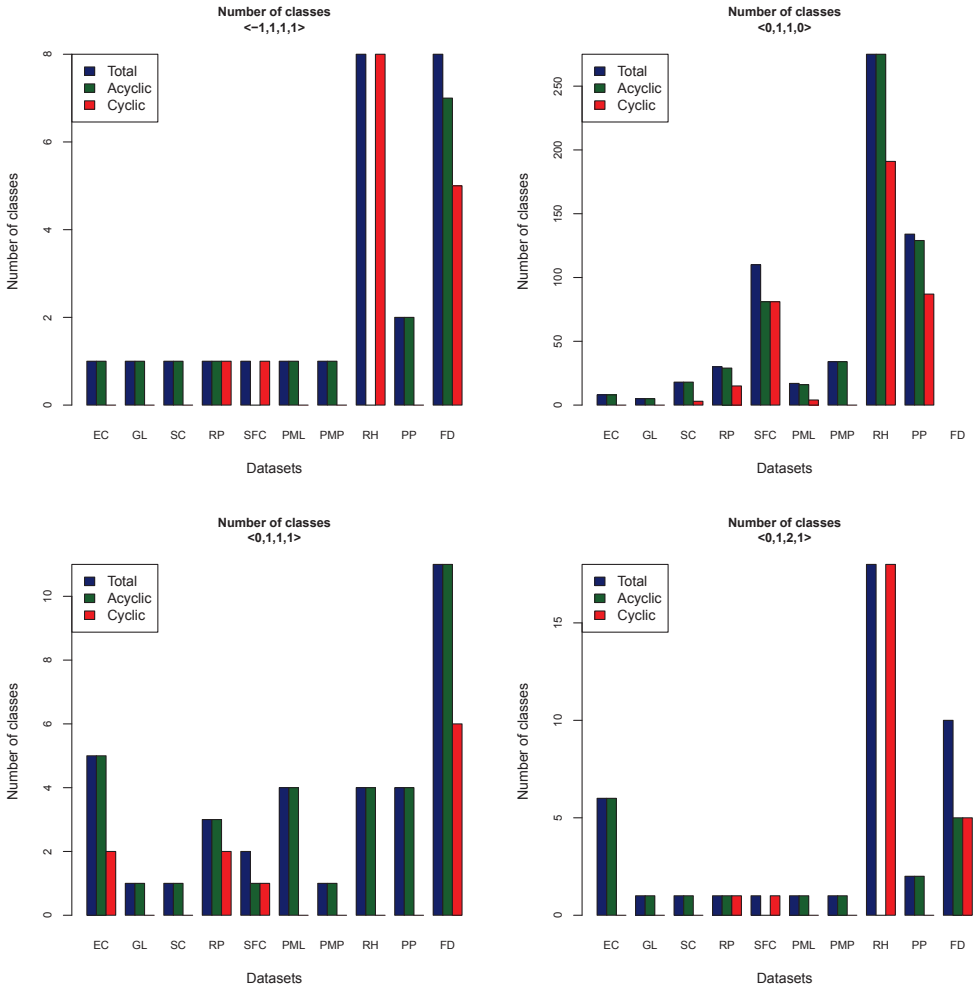


Figure 4: For each dataset we show the number of different solution classes in all optimal reconciliations, in the ones that are cyclic and also in the acyclic ones.

size of the host tree and the vector θ. A tree is a good representative if the observed number of events is near to the expected. More formally, for a real parasite tree P, a vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, and a simulated parasite tree P_θ for which the observed number of events are $obs = \langle o_c, o_d, o_s, o_l \rangle$, we define a measure $D_1(P, P_\theta)$ as follows:

$$D_1(P, P_\theta) = \frac{1}{4} \times \sum_{i \in \{c,d,s,l\}} \frac{|e_i - o_i|}{\max\{e_i, o_i\}}$$

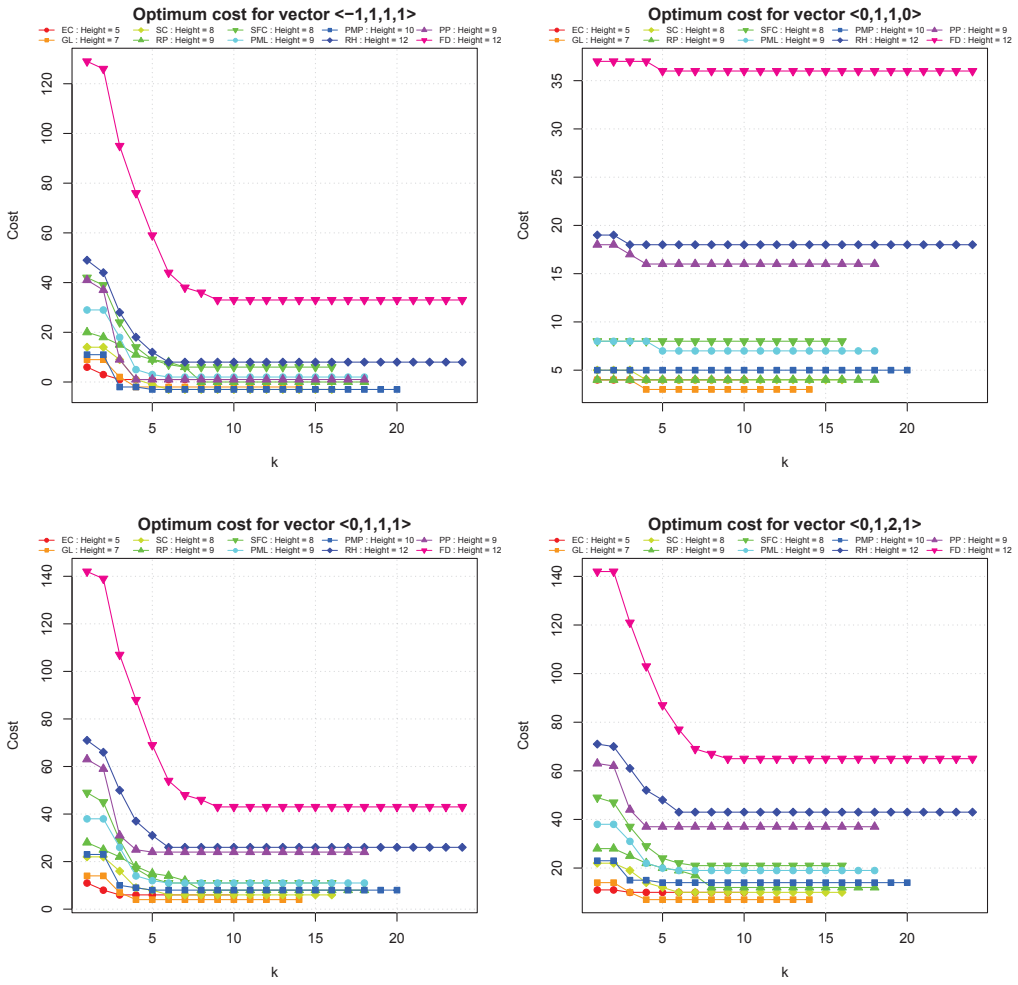


Figure 5: For all datasets we show the relation between the number of optimal solutions and the value k, of the maximum allowable distance of a switch.

As concerns point (ii), we use a metric for comparing phylogenetic trees. In our context, the distance that best meets the requirement of efficiency and accuracy appears for now to be the *maximum agreement area cladogram* (MAAC) (Ganapathy et al., 2006). This is a generalisation for multi-labelled trees of the well-known *maximum agreement subtree* (Finden and Gordon, 1985; Farach-Colton, Przytycka, and Thorup, 1995). It corresponds to the number of leaves in the largest isomorphic subtree that is common to two (multi-labelled) trees. Clearly this isomorphism takes in account the labels of the trees. The MAAC distance

can be calculated in $O(n^2)$ time where n is the size of the largest input tree (Ganapathy et al., 2006).

We use a normalised version of MAAC that takes into account also the number of leaves that are common between the two trees. More formally, given two trees P and P' with leaf sets $L(P)$ and $L(P')$ respectively, we define the measure $D_2(P, P')$ as follows:

$$D_2(P, P') = \begin{cases} 1 - \dfrac{MAAC(P, P')}{|L(P) \cap L(P)'|} & \text{if } L(P) \cap L(P)' \neq \emptyset \\ 1 & \textit{otherwise}. \end{cases}$$

Observe that the intersection operation involves multi-sets. We recall that a multi-set is a generalisation of a set where the elements are allowed to appear more than once, hence the operations take into account their multiplicity.

Finally, we propose a distance that is based on two components. The first one takes into account the difference between the expected and observed number of events. The second takes into account the difference between the topologies. For a real parasite P, a vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, and a simulated parasite tree P_θ, we define the distance $d(P, P_\theta)$ as follows:

$$d(P, P_\theta) = \alpha_1 D_1(P, P_\theta) + \alpha_2 D_2(P, P_\theta)$$

Here, we have to determine the values for α_1 and α_2 which give weight to each component. A high value for α_1 and a low value for α_2 tend to select trees which show size and number of events close to the expected values taking into account the real parasite tree. Inversely, a low value for α_1 and a high value for α_2 favour trees which have high topological similarity with the real parasite but that do not necessarily reflect the expected number of events.

One important factor that we have to take into account while doing this experiment is that the simulated trees are generated according to a model which follows the host tree H. As a result of this characteristic, given two trees P_1 and P_2 that are generated by the model using the same parameter vector, we have a higher chance that the distances $d(H, P_1)$ and $d(H, P_2)$ are going to be smaller than the distance $d(P_1, P_2)$. This effect is increased when the parameter vector allows the generation of parasite trees which are less constrained by the topology of the host tree (for instance, when we have high host-switch probability).

We scaled the weight such that $\alpha_1 + \alpha_2 = 1$ and conducted experiments to evaluate different combinations of values. For each one of the pairs of trees

which were created for the self-test, we adopted values for α_1 from 0 to 1 (in steps of 0.1) and we generated 5000 trees using two probability vectors θ_A and θ_B. Vector θ_A was the one originally used to create the pairs of trees and vector $\theta_B = \langle 0.97, 0.01, 0.01, 0.01 \rangle$ was chosen to simulate cases where most of the trees are identical to the host tree (high co-speciation probability).

Table 1 shows the average distance, which is computed according to a given value for α_1, for the 5000 trees which were produced with each one of the probability vectors θ_A and θ_B. Line $\theta_A - \theta_B$ shows the difference between the averages of each vector. We observe that for small values of α_1, the probability vector θ_B shows smaller average distances suggesting that topological information only is not enough to identify vectors that are close to the original one.

If we make $\alpha_1 \geq 0.6$, the difference between the averages $(\theta_A - \theta_B)$ becomes negative for all pairs of trees. Starting from $\alpha_1 = 0.6$, the distance thus starts to favour trees which are generated by parameter vectors which are close to the original one.

We opted to fix $\alpha_1 = 0.7$ and consequently $\alpha_2 = 0.3$ because we believe that this combination is able to select trees that respect the expected frequencies of events without completely ignoring the topological information.

Generating simulated datasets for the self-test of Coala

The procedure for generating simulated datasets can be found in the main text. Table 2 lists the vectors which were used to create the simulated datasets based on a host tree H of 36 leaves. This tree corresponds to the host tree of the primates/pinworms co-phylogenetic study described in (Hugot, 1999). This choice was motivated by the wish to have a dataset distinct from the ones used for evaluating the method.

In the following pages, we can find the resulting phylogenetic trees.

$\theta_1 = \langle 0.70, 0.10, 0.10, 0.10 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.337	0.334	0.330	0.326	0.322	0.318	0.314	0.310	0.306	0.302	0.298
θ_B	0.078	0.126	0.175	0.223	0.271	0.319	0.368	0.416	0.464	0.512	0.561
$\theta_A - \theta_B$	0.259	0.207	0.155	0.103	0.051	-0.002	-0.054	-0.106	-0.158	-0.210	-0.263
$\theta_2 = \langle 0.80, 0.15, 0.01, 0.04 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.176	0.193	0.209	0.226	0.243	0.260	0.277	0.294	0.311	0.328	0.345
θ_B	0.036	0.090	0.145	0.199	0.254	0.308	0.363	0.417	0.472	0.526	0.581
$\theta_A - \theta_B$	0.140	0.102	0.065	0.027	-0.010	-0.048	-0.086	-0.123	-0.161	-0.198	-0.236
$\theta_3 = \langle 0.75, 0.01, 0.16, 0.08 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.218	0.227	0.235	0.244	0.252	0.261	0.269	0.278	0.286	0.295	0.303
θ_B	0.034	0.082	0.129	0.177	0.224	0.272	0.320	0.367	0.415	0.462	0.510
$\theta_A - \theta_B$	0.184	0.145	0.106	0.067	0.028	-0.011	-0.050	-0.089	-0.129	-0.168	-0.207
$\theta_4 = \langle 0.70, 0.05, 0.02, 0.23 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.050	0.078	0.106	0.134	0.162	0,190	0.218	0.246	0.274	0.302	0.330
θ_B	0.006	0.046	0.087	0.127	0.167	0,207	0.247	0.288	0.328	0.368	0.408
$\theta_A - \theta_B$	0.043	0.031	0.019	0.007	-0.005	-0.017	-0.030	-0.042	-0.054	-0.066	-0.078
$\theta_5 = \langle 0.60, 0.20, 0.00, 0.20 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.216	0.221	0.226	0.231	0.236	0.241	0.246	0.251	0.256	0.261	0.266
θ_B	0.046	0.098	0.151	0.203	0.255	0.308	0.360	0.413	0.465	0.517	0.570
$\theta_A - \theta_B$	0.170	0.123	0.075	0.028	-0.019	-0.067	-0.114	-0.161	-0.209	-0.256	-0.303
$\theta_6 = \langle 0.55, 0.00, 0.20, 0.25 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.380	0.363	0.346	0.329	0.312	0.295	0.278	0.261	0.244	0.227	0.210
θ_B	0.255	0.266	0.277	0.288	0.299	0.310	0.321	0.332	0.342	0.353	0.364
$\theta_A - \theta_B$	0.125	0.097	0.069	0.041	0.013	-0.015	-0.043	-0.071	-0.099	-0.126	-0.154
$\theta_7 = \langle 0.45, 0.10, 0.15, 0.30 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.475	0.463	0.450	0.438	0.425	0.413	0.400	0.388	0.375	0.363	0.350
θ_B	0.415	0.414	0.412	0.410	0.409	0.407	0.405	0.404	0.402	0.400	0.399
$\theta_A - \theta_B$	0.060	0.049	0.038	0.027	0.017	0.006	-0.005	-0.016	-0.027	-0.038	-0.048
$\theta_8 = \langle 0.40, 0.20, 0.10, 0.30 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.497	0.486	0.476	0.465	0.455	0.444	0.434	0.423	0.413	0.402	0.391
θ_B	0.434	0.451	0.468	0.485	0.501	0.518	0.535	0.551	0.568	0.585	0.602
$\theta_A - \theta_B$	0.062	0.035	0.008	-0.019	-0.047	-0.074	-0.101	-0.128	-0.156	-0.183	-0.210
$\theta_9 = \langle 0.30, 0.20, 0.40, 0.10 \rangle$											
α_1	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
θ_A	0.597	0.571	0.546	0.521	0.495	0.470	0.444	0.419	0.394	0.368	0.343
θ_B	0.267	0.301	0.335	0.369	0.403	0.437	0.471	0.505	0.538	0.572	0.606
$\theta_A - \theta_B$	0.329	0.270	0.211	0.152	0.092	0.033	-0.026	-0.086	-0.145	-0.204	-0.263

Table 1: Evaluation of α values using the pairs of trees which were produced for the self-test.

Simulated dataset	p_c	p_d	p_s	p_l	Number of leaves simulated tree
$v1 = \theta_1$	0.70	0.10	0.10	0.10	75
$v2 = \theta_2$	0.80	0.15	0.01	0.04	87
$v3 = \theta_3$	0.75	0.01	0.16	0.08	52
$v4 = \theta_4$	0.70	0.05	0.02	0.23	27
$v5 = \theta_5$	0.60	0.20	0.00	0.20	79
$v6 = \theta_6$	0.55	0.00	0.20	0.25	34
$v7 = \theta_7$	0.45	0.10	0.15	0.30	50
$v8 = \theta_8$	0.40	0.20	0.10	0.30	107
$v9 = \theta_9$	0.30	0.20	0.40	0.10	111

Table 2: Probability vectors which were used to create the simulated datasets. All datasets use the same host tree H of 36 leaves.

Self-test of Coala

Each one of the pairs of host and simulated parasite tree were processed by Coala which was configured to simulate 2000 vectors (parameter `-N`) at the first round and generate 1000 trees per simulated vector (parameter `-M`). We ran Coala from 2 to 5 rounds, always clustering the accepted vectors of each round (parameter `-cluster`). The tolerance values were set to 0.10 for the first round and 0.25 for the remaining rounds.

Each pair of trees were processed 50 times (per round). For each one of these runs, we identified the cluster whose representative parameter vector has the smallest χ^2 distance to the "target" probability vector, which was used to create the pair of trees. In the following two pages, we can find the histograms of the χ^2 distances between the closest parameter vector of each run and the target probability vector. Each row represents a dataset and each column represents one round (from 2 to 5). The blue and magenta lines indicate median and mean distance values respectively.

After the distance histograms, we have nine pages showing, for each dataset, the distribution of the event probabilities which were observed in the list of accepted parameter vectors that are closer to the target one in each run. Each column represents an event type and each row represents one round (from 2 to 5). The blue and magenta lines indicate median and mean distance values and the dashed orange line represents the target probability value.

Up to a certain level of co-speciation probability (≥ 0.50), our results show that in the rounds 2 and 3 Coala frequently selects parameter vectors that are close to the target probability vector. After the third round, Coala tends to select vectors that, on average, show lower summary statistics. Such behaviour may indicate that running Coala for many rounds may overfit the data.

When we decrease the co-speciation probability to values smaller than 0.50, Coala selects very few vectors which are close to the target vector. When the co-speciation probability decreases and the duplication and host-switch probabilities increase, a huge number of trees would have the same significant probability to be generated. Thus, the variability on the topologies of the trees that will be practically generated increases notably. Due to this, selecting a typical tree becomes an almost impossible task and this may explain the obtained results. It may be that increasing the number of simulated trees to compute

the summary statistics would improve the quality of the results, however, this would require a much longer execution time.

Phylogenetic Trees of the Biological Datasets

In the following pages, we present the phylogenetic trees of the four biological datasets which were extracted from the literature and used in our experiments:

Dataset 1 Gopher and Louse (Johnson, Drown, and Clayton, 2000).

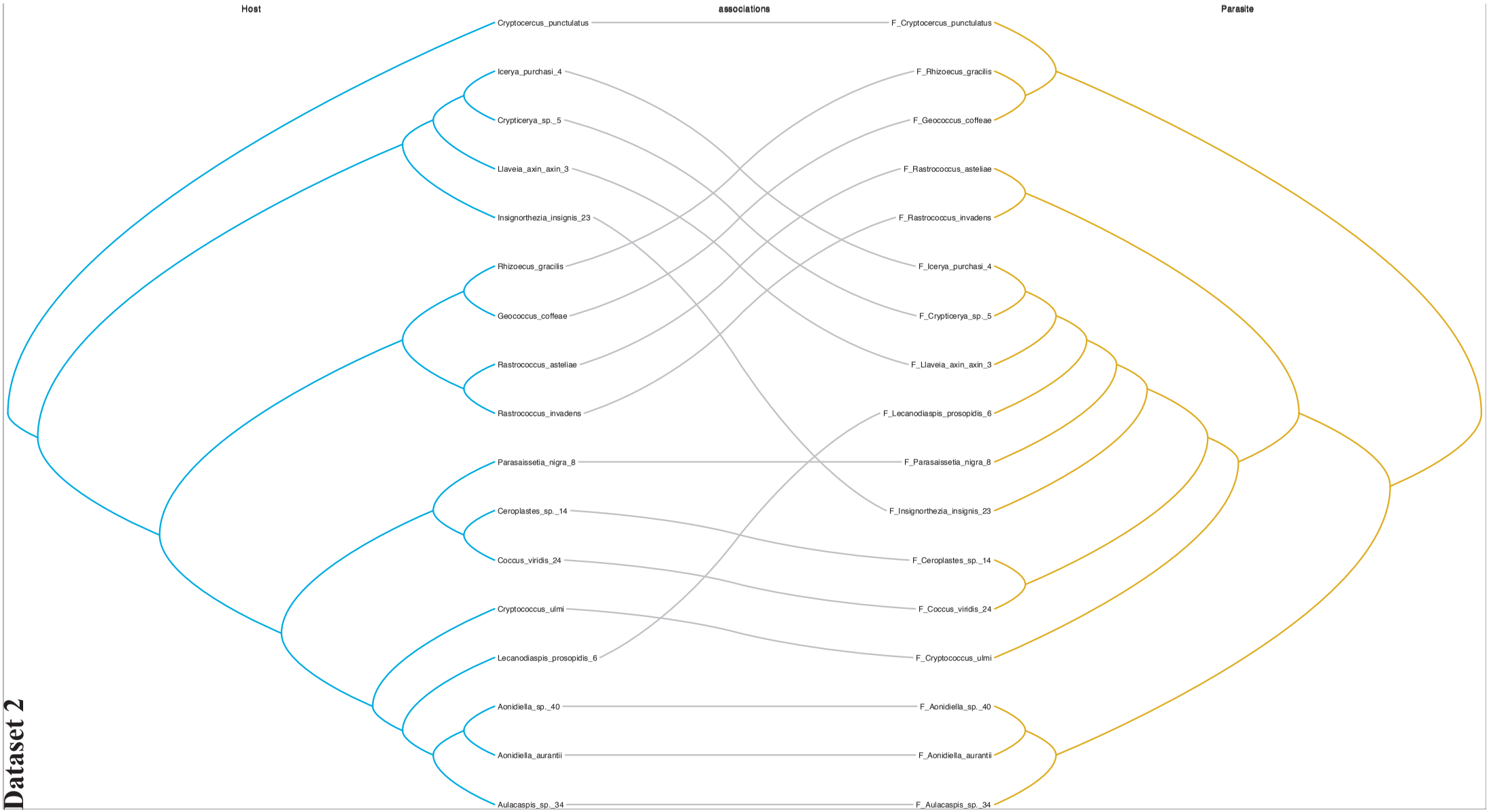
Dataset 2 Flavobacterial endosymbionts and their insect hosts (Rosenblueth et al., 2012).

Dataset 3 Anther smut fungi and their caryophyllaceous hosts (Refrégier et al., 2008).

Dataset 4 Rodents and Hantaviruses (Ramsden, Holmes, and Charleston, 2009).


Host
associations
Parasite
O._heterodus
O._cherriei
O._cavator
O._underwoodi
O._hispidus
Z._trichopus
P._bulleri
C._castanops
C._merriami
G._bursarius_b
G._bursarius_a
G._breviceps
G._personatus
T._bottae
T._talpoides
G._costaricensis
G._cherriei
G._panamensis
G._setzeri
G._chapini
G._trichopi
G._nadleri
G._expansus
G._geomydis
G._oklahomensis
G._ewingi
G._texanus
G._actuosi
G._perotensis
G._thomomyus
T._minor
T._barbarae

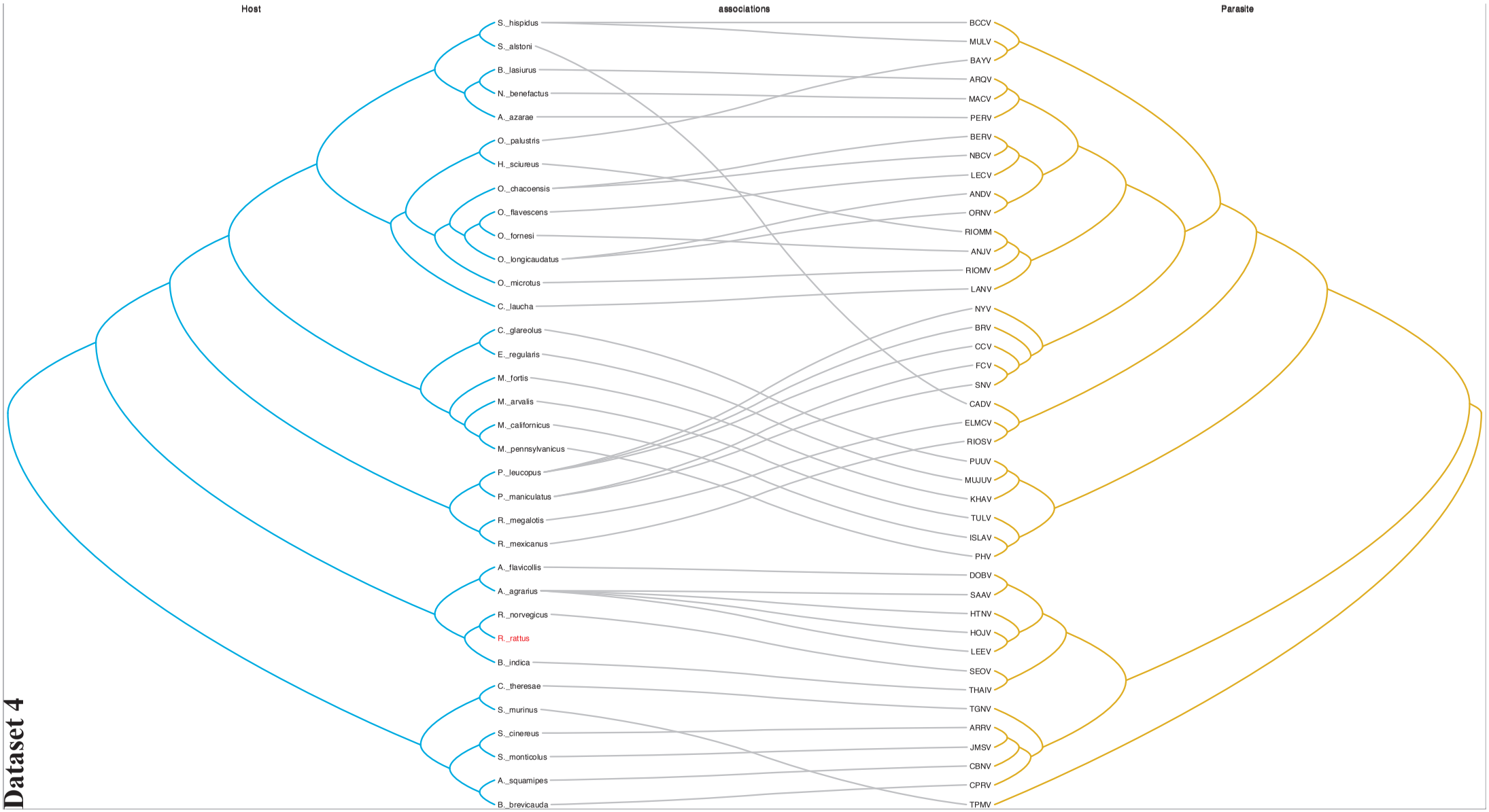
Dataset 1


Host
associations
Parasite
Cryptocercus_punctulatus
Icerya_purchasi_4
Crypticerya_sp._5
Llaveia_axin_axin_3
Insignorthezia_insignis_23
Rhizoecus_gracilis
Geococcus_coffeae
Rastrococcus_asteliae
Rastrococcus_invadens
Parasaissetia_nigra_8
Ceroplastes_sp._14
Coccus_viridis_24
Cryptococcus_ulmi
Lecanodiaspis_prosopidis_6
Aonidiella_sp._40
Aonidiella_aurantii
Aulacaspis_sp._34
F_Cryptocercus_punctulatus
F_Rhizoecus_gracilis
F_Geococcus_coffeae
F_Rastrococcus_asteliae
F_Rastrococcus_invadens
F_Icerya_purchasi_4
F_Crypticerya_sp._5
F_Llaveia_axin_axin_3
F_Lecanodiaspis_prosopidis_6
F_Parasaissetia_nigra_8
F_Insignorthezia_insignis_23
F_Ceroplastes_sp._14
F_Coccus_viridis_24
F_Cryptococcus_ulmi
F_Aonidiella_sp._40
F_Aonidiella_aurantii
F_Aulacaspis_sp._34

Dataset 2


Host
associations
Parasite
Stellaria
D._gratianopolitanus
D._sylvestris
D._carthusianorum
D._monspessulanus
G._repens
Sa._officinialis
S._aucalis
S._otites
S._nutans
S._lemmonii
S._caroliniana
S._virginica
S._vulgaris
S._latifolia
S._dioica
L._flos-cuculi
L._flos-jovis
MvSA
MvStel
MvDsp2
MvDsp1
MvDc
MvSaoff
MvSlem
MvSot
MvSn
MvSv2
MvLfc
MvSl
MvSd
MvSv1
MvLfj
MvSspA

Dataset 3


Host
associations
Parasite
S._hispidus
S._alstoni
B._lasiurus
N._benefactus
A._azarae
O._palustris
H._sciureus
O._chacoensis
O._flavescens
O._fornesi
O._longicaudatus
O._microtus
C._laucha
C._glareolus
E._regularis
M._fortis
M._arvalis
M._californicus
M._pennsylvanicus
P._leucopus
P._maniculatus
R._megalotis
R._mexicanus
A._flavicollis
A._agrarius
R._norvegicus
R._rattus
B._indica
C._theresae
S._murinus
S._cinereus
S._monticolus
A._squamipes
B._brevicauda
BCCV
MULV
BAYV
ARQV
MACV
PERV
BERV
NBCV
LECV
ANDV
ORNV
RIOMM
ANJV
RIOMV
LANV
NYV
BRV
CCV
FCV
SNV
CADV
ELMCV
RIOSV
PUUV
MUJUV
KHAV
TULV
ISLAV
PHV
DOBV
SAAV
HTNV
HOJV
LEEV
SEOV
THAIV
TGNV
ARRV
JMSV
CBNV
CPRV
TPMV

Dataset 4

Results obtained by Coala on biological datasets

In the following pages, we present some of the results produced by Coala while processing the biological datasets 1 to 4.

Coala was executed with the following parameters:

- -N 2000: Number of parameter vectors which are simulated for the first round;
- -M 1000: Number of trees which are considered in the computation of the summary statistics for a given parameter vector;
- -R 3 or 5: Number of rounds;
- -t 0.1,0.25,0.25 (3 rounds) or -t 0.1,0.25,0.25,0.25,0.25 (3 rounds): Vector of tolerances for each round;
- -plot: Create plots for each round;
- -cluster: Cluster the list of accepted vectors of the last round.

We configured Coala to cluster the list of accepted parameter vectors at the end of the rounds 3 and 5. For each dataset, Table 3 shows the list of representative parameter vectors observed in the clusters of each round.

Round	Dataset	Cluster	p_c	p_d	p_s	p_l	# vectors
3	1	1	0.794	0.002	0.101	0.104	18
		2	0.790	0.118	0.085	0.007	15
		3	0.864	0.061	0.000	0.074	12
		4	0.522	0.280	0.001	0.197	5
	2	0	0.030	0.000	0.557	0.413	1
		1	0.461	0.258	0.000	0.281	24
		2	0.554	0.000	0.270	0.176	20
		3	0.910	0.016	0.058	0.016	5
	3	1	0.744	0.001	0.090	0.165	19
		2	0.397	0.000	0.353	0.250	18
		3	0.512	0.231	0.000	0.257	10
		4	0.036	0.000	0.610	0.354	3
	4	1	0.851	0.082	0.000	0.066	25
		2	0.473	0.204	0.000	0.323	10
		3	0.238	0.349	0.000	0.413	8
		4	0.580	0.002	0.282	0.136	7
5	1	1	0.829	0.157	0.001	0.013	25
		2	0.811	0.101	0.087	0.000	19
		3	0.795	0.000	0.112	0.093	6
	2	1	0.954	0.042	0.000	0.004	26
		2	0.892	0.001	0.100	0.007	17
		3	0.470	0.253	0.000	0.277	7
	3	1	0.832	0.000	0.000	0.168	14
		2	0.817	0.000	0.000	0.183	13
		3	0.825	0.000	0.000	0.175	12
		4	0.839	0.000	0.000	0.161	8
		5	0.826	0.000	0.001	0.173	3
	4	1	0.911	0.088	0.000	0.000	18
		2	0.925	0.075	0.000	0.000	11
		3	0.902	0.098	0.000	0.000	9
		4	0.907	0.092	0.000	0.002	8
		5	0.925	0.073	0.000	0.002	4

Table 3: Representative probability vectors produced by COALA at rounds 3 and 5 while processing the biological datasets 1 to 4.

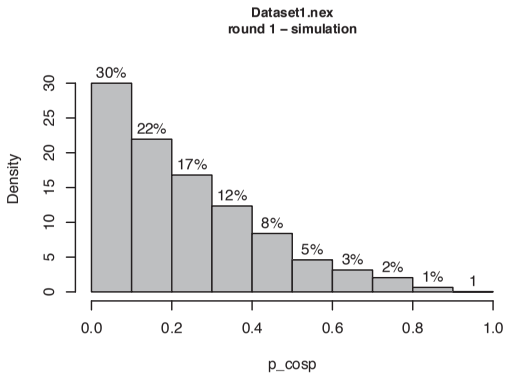
Here, we list the histograms of distances and event probabilities obtained at the end of each one of the 5 rounds.

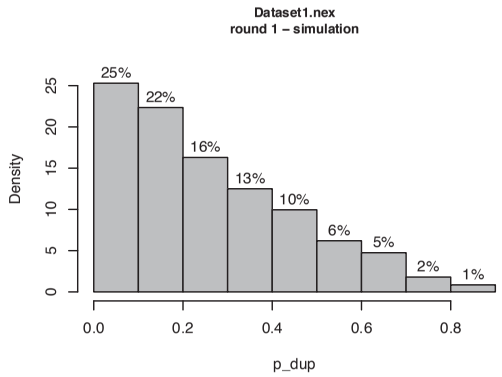
On each page, we have the results for a Dataset D at round R:

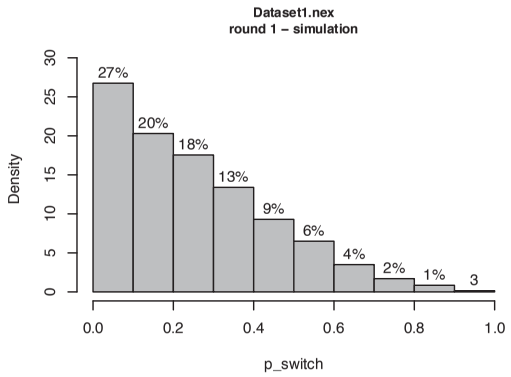
First row Event probability histograms (co-speciation, duplication, host-switch and loss) of all simulated parameter vectors at round R.

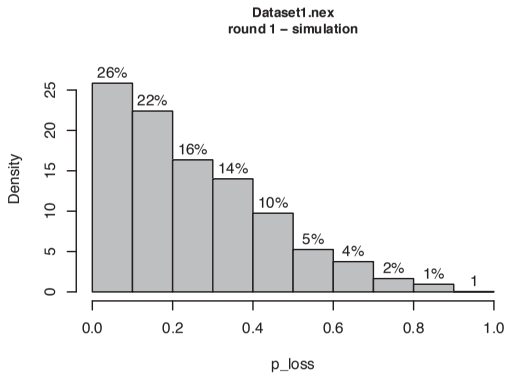
Second row Event probability histograms (co-speciation, duplication, host-switch and loss) of all accepted parameter vectors at round R.

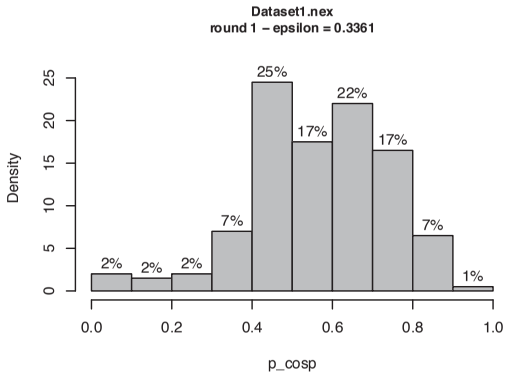
Third row Representative distance histograms for all simulated parameter vectors (first column) and accepted parameter vectors (second column).

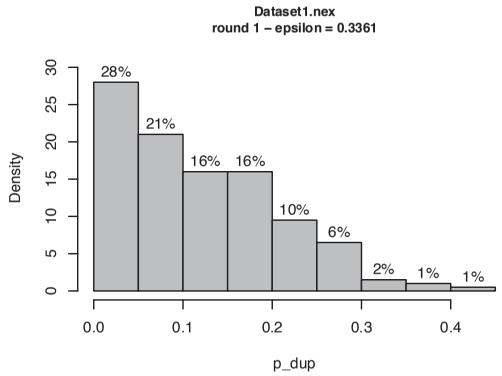


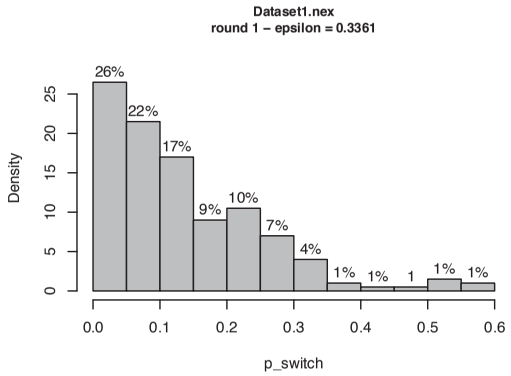


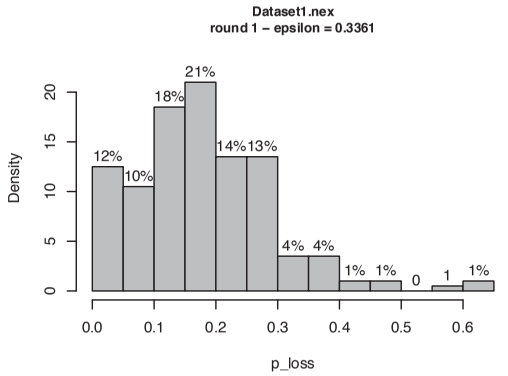


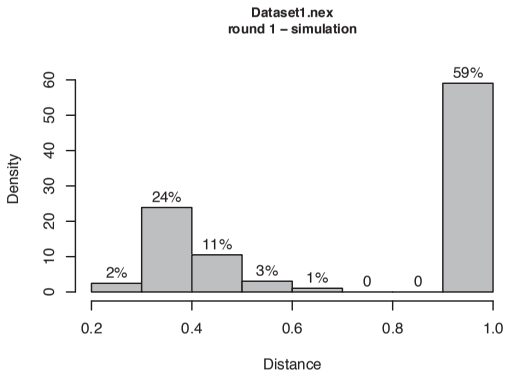


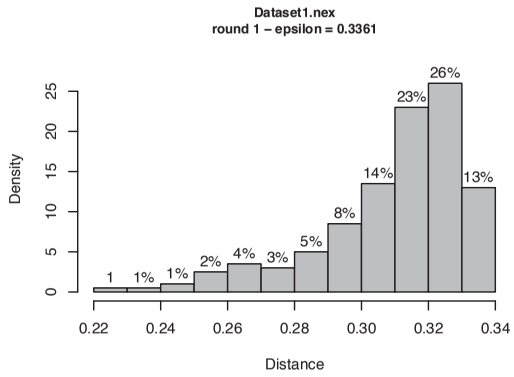




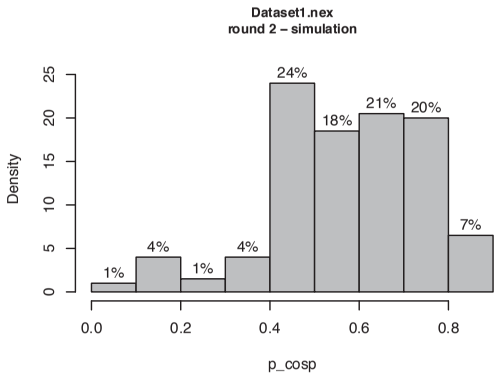


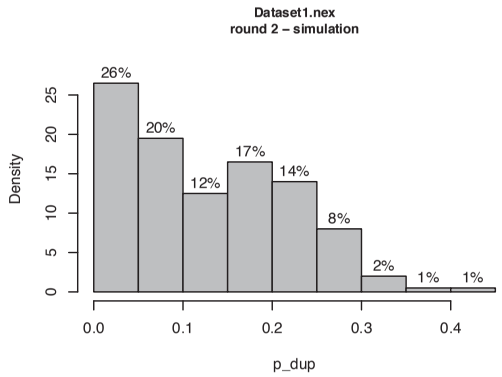


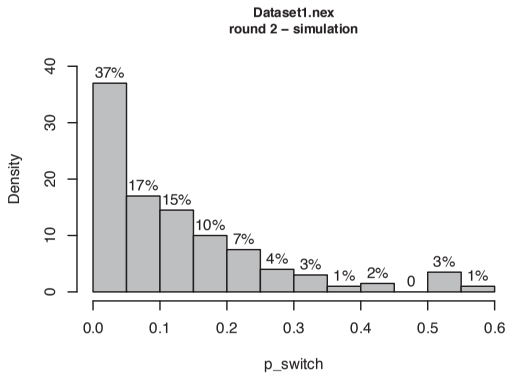


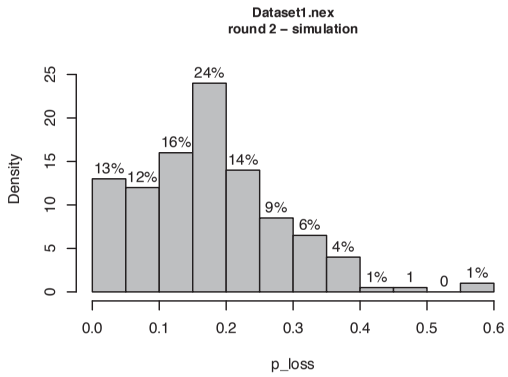


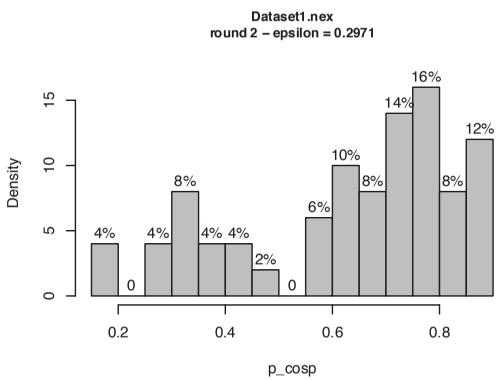
Nexus file: Dataset1.nex
Host/Parasite tree: 15/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 2 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

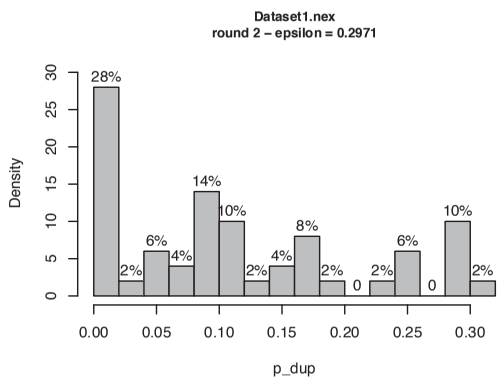


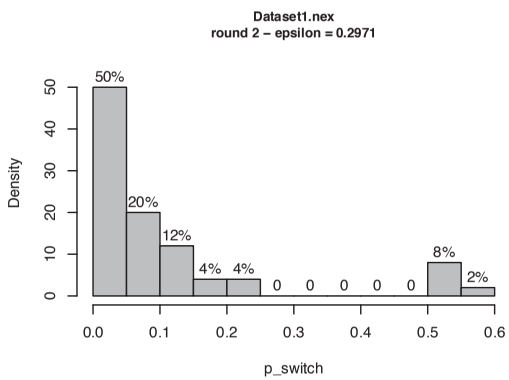


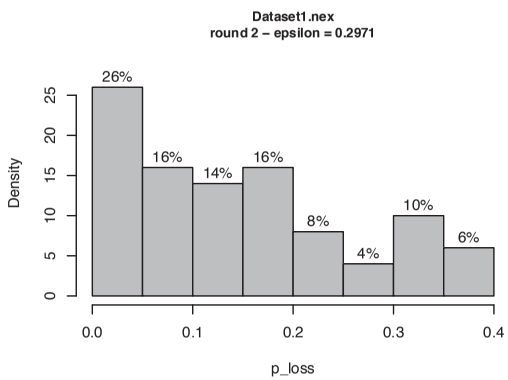


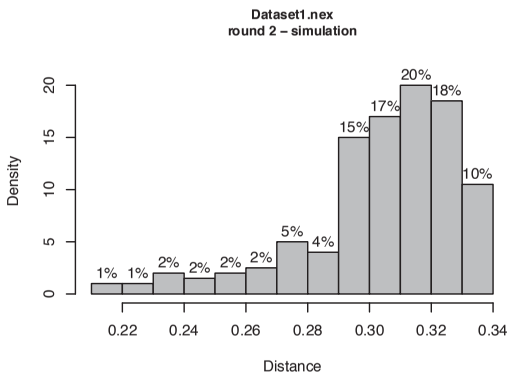


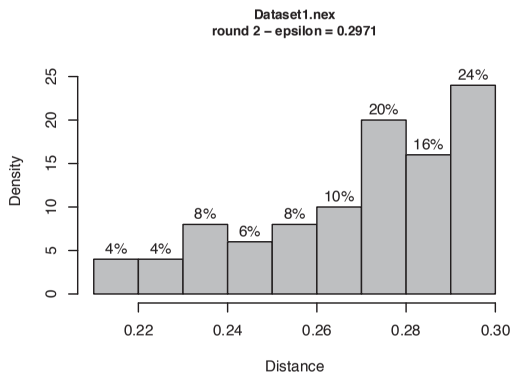




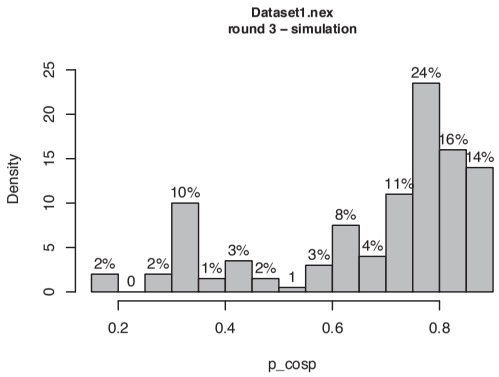


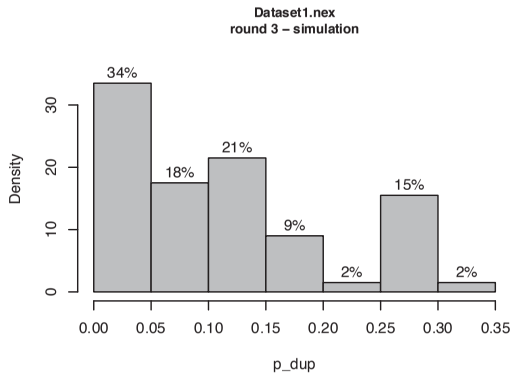


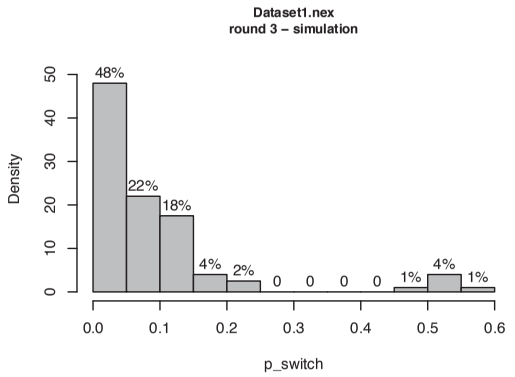


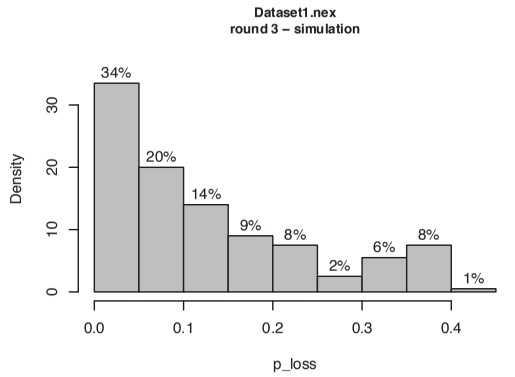


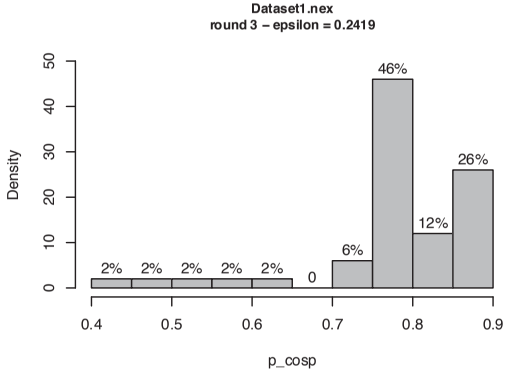
Nexus file: Dataset1.nex
Host/Parasite tree: 15/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 3 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

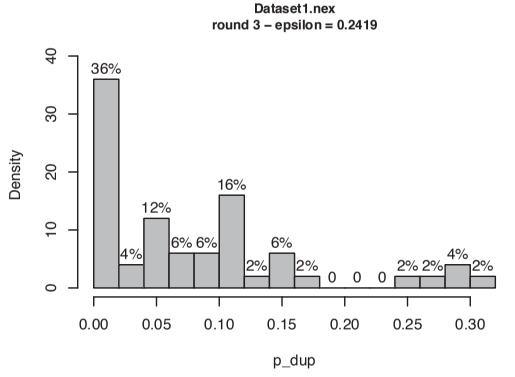


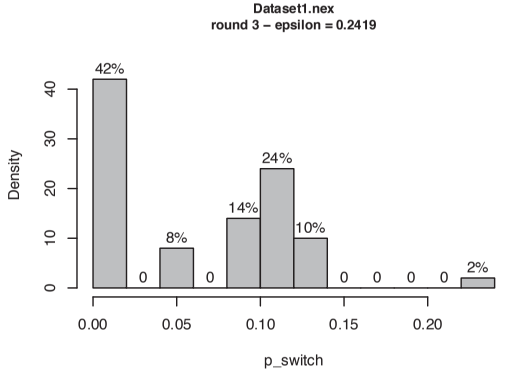


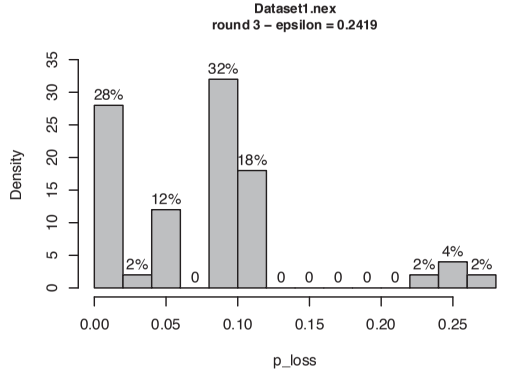


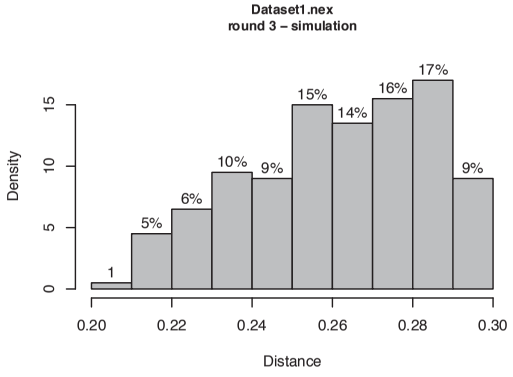


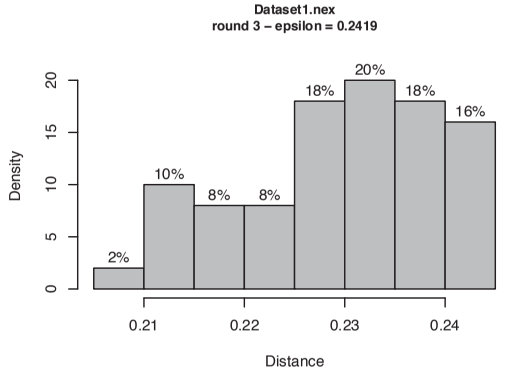




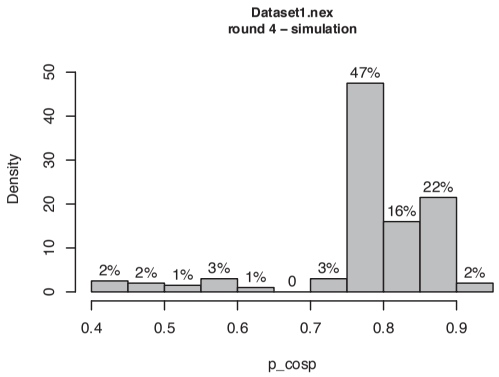


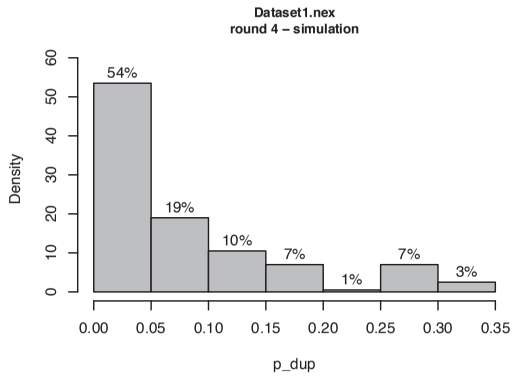


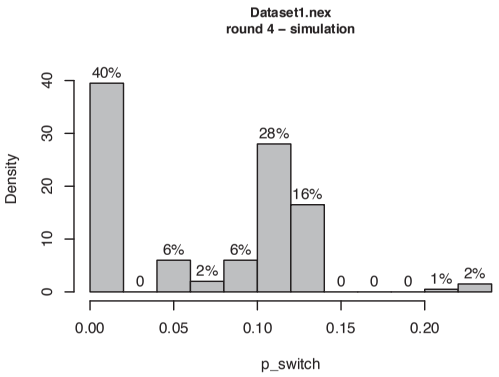


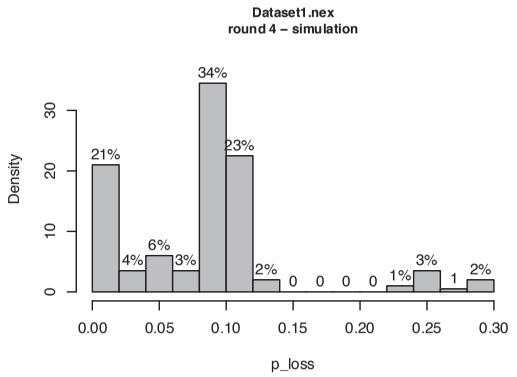


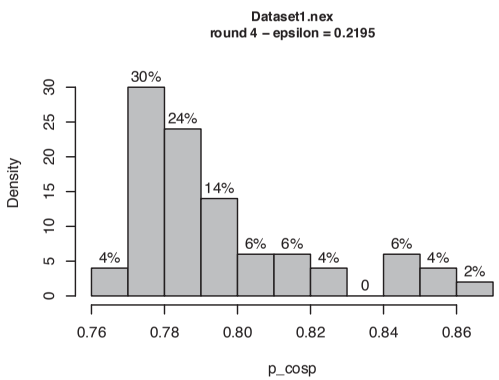
Nexus file: Dataset1.nex
Host/Parasite tree: 15/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 4 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

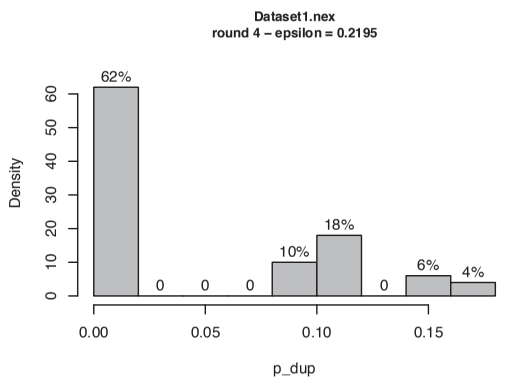


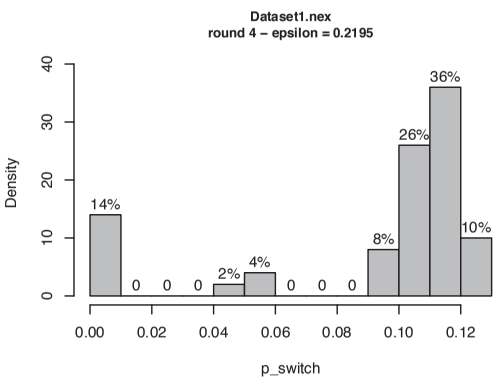


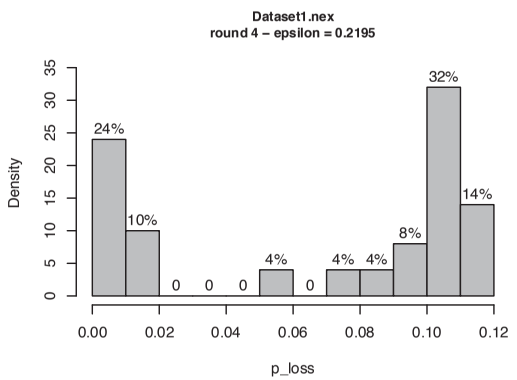


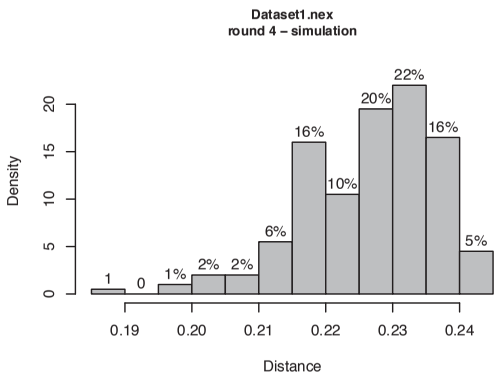


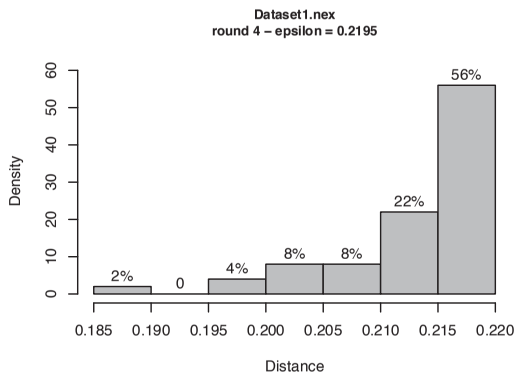




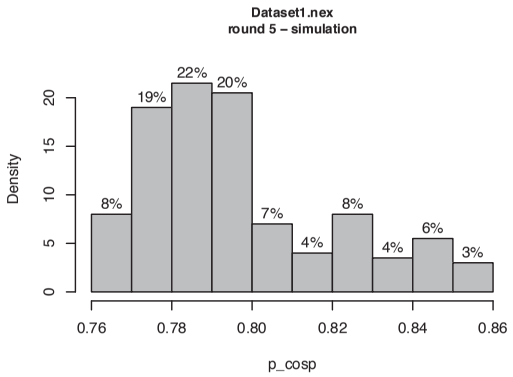


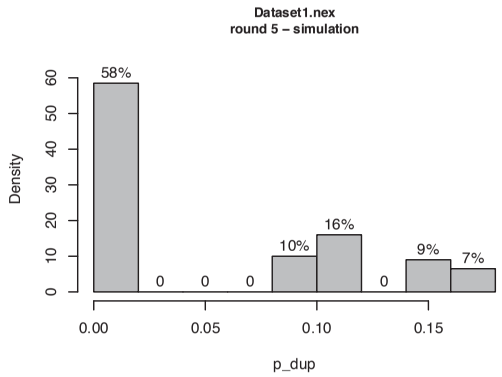


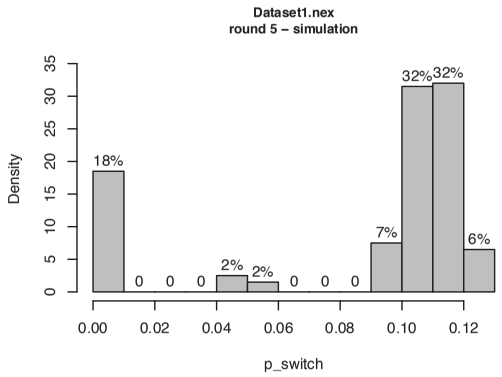


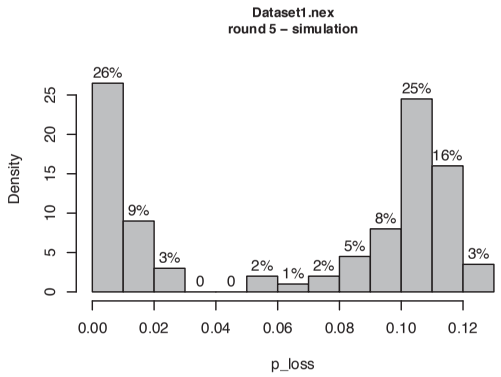


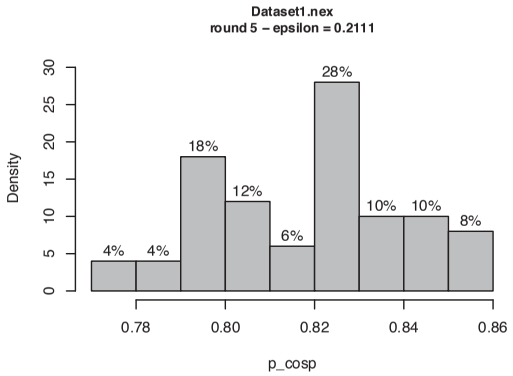
Nexus file: Dataset1.nex
Host/Parasite tree: 15/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

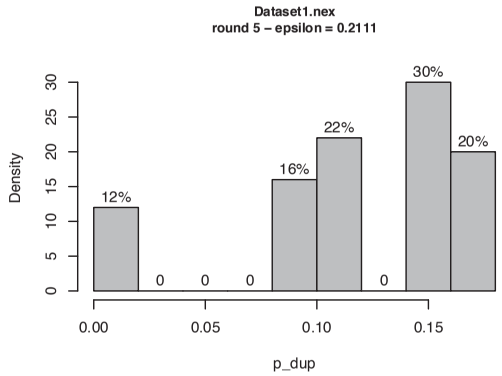


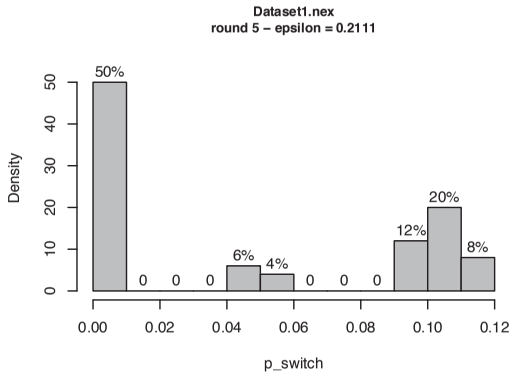


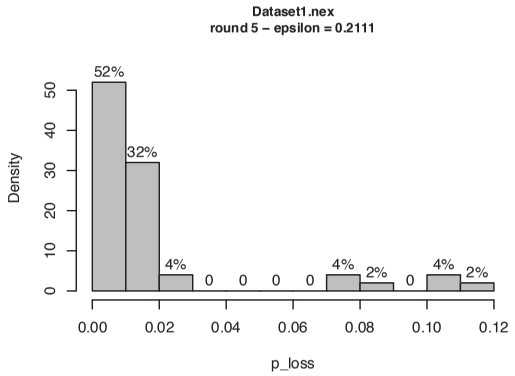


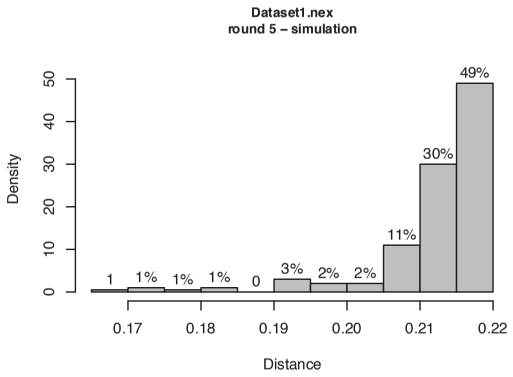


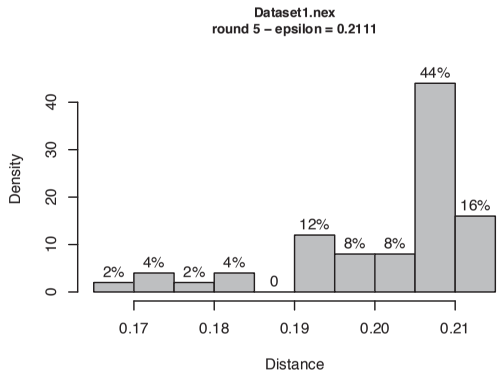




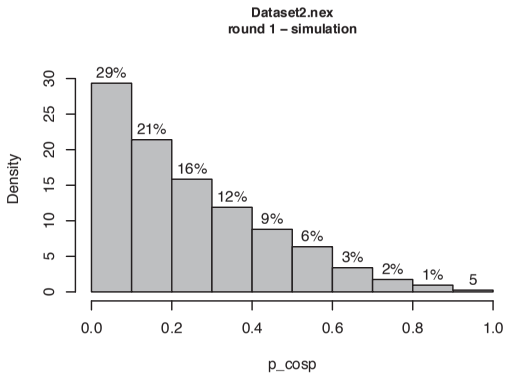


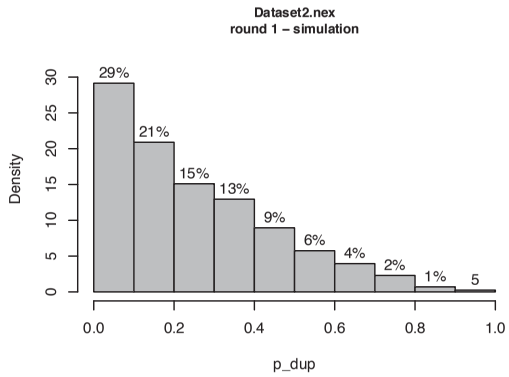


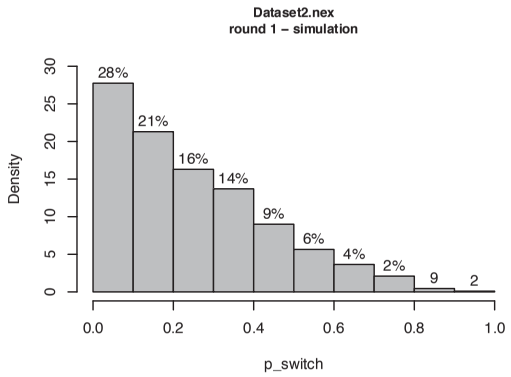


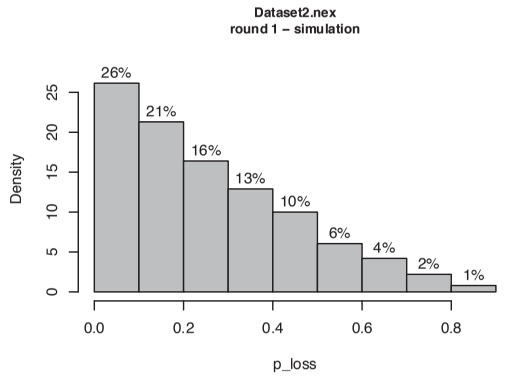


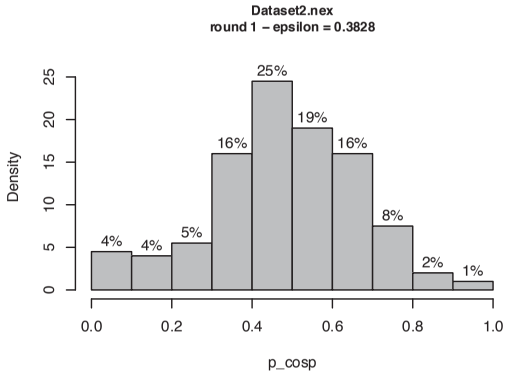
Nexus file: Dataset1.nex
Host/Parasite tree: 15/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

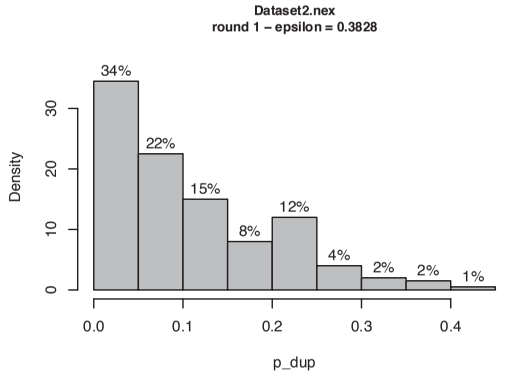


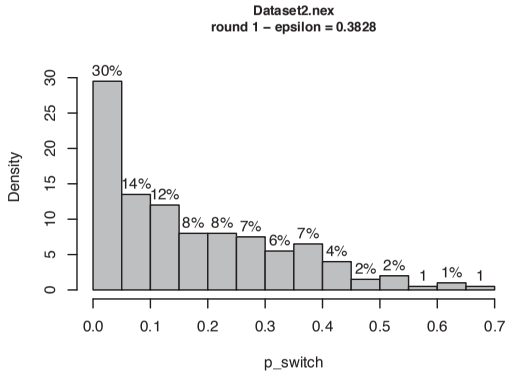


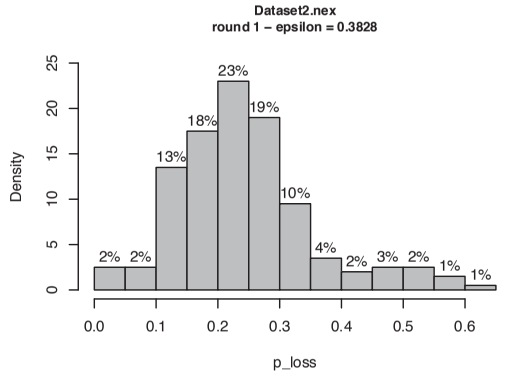


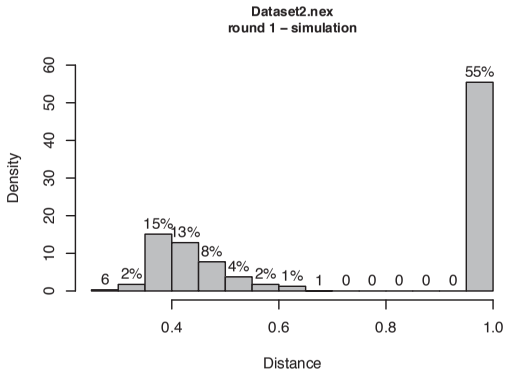


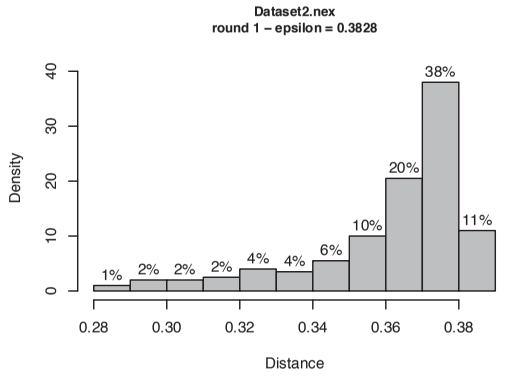




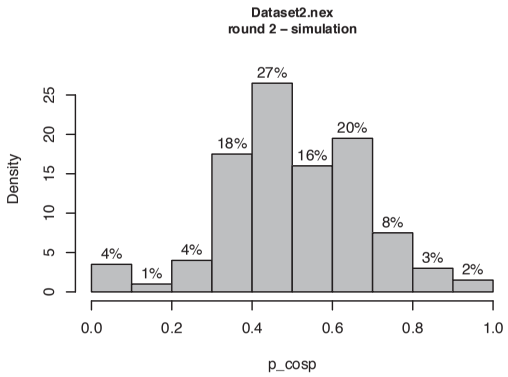


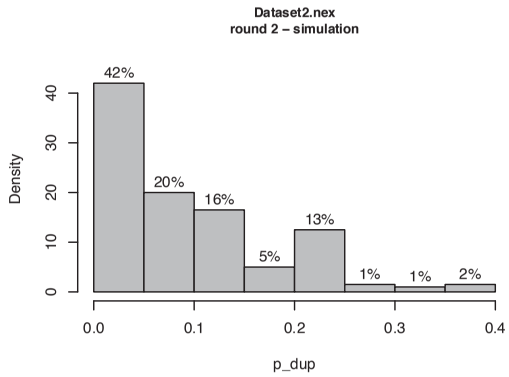


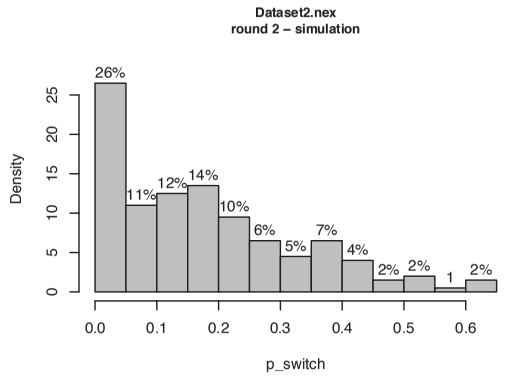


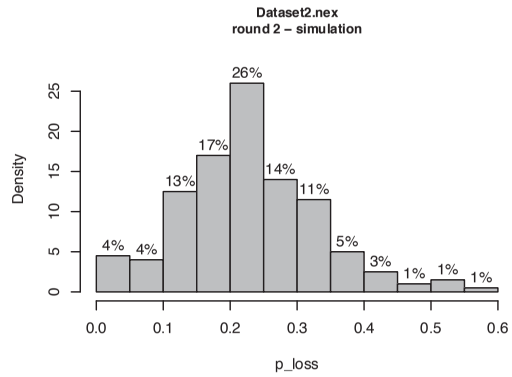


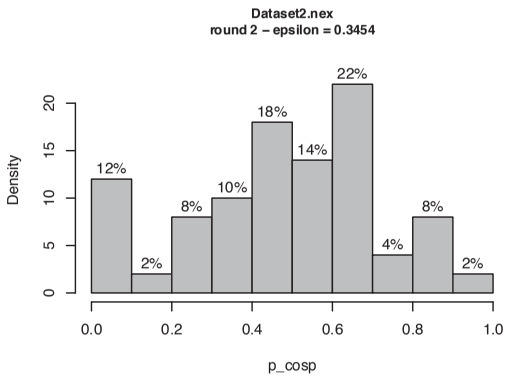
Nexus file: Dataset2.nex
Host/Parasite tree: 17/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 2 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

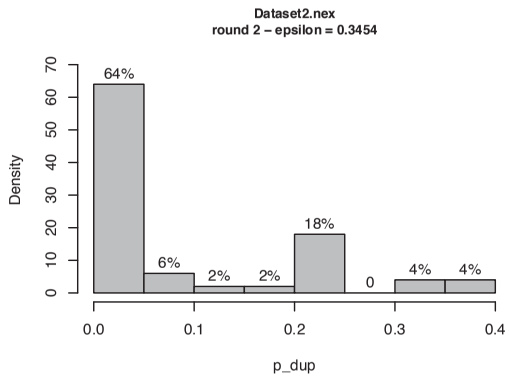


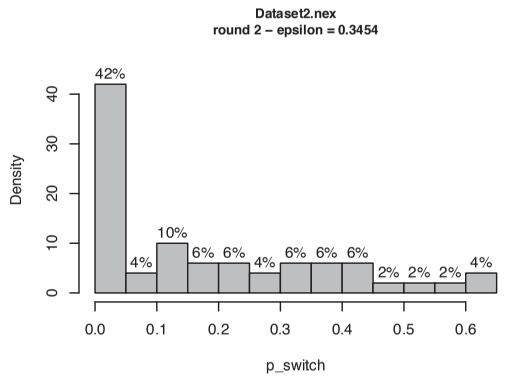


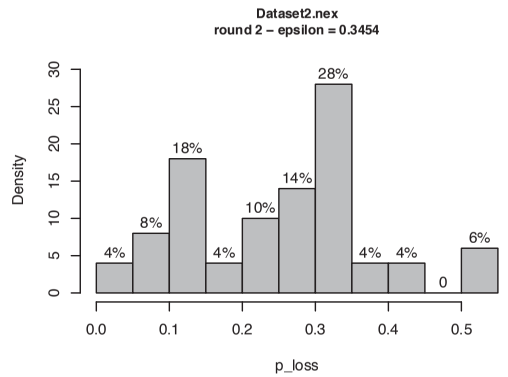


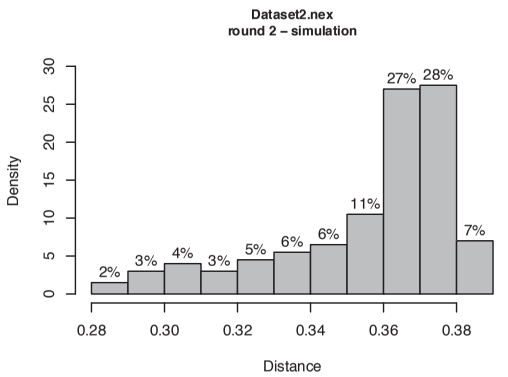


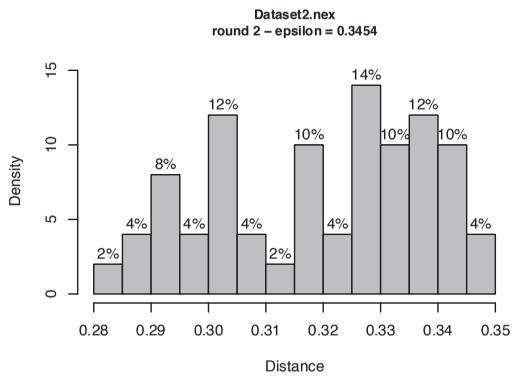




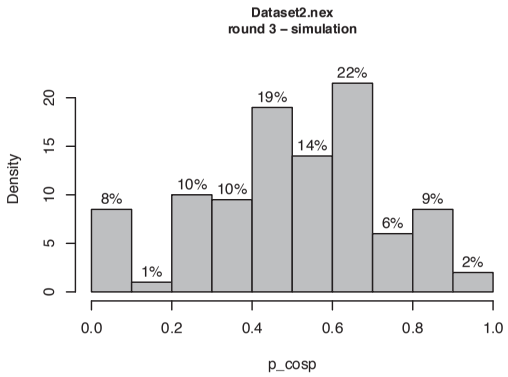


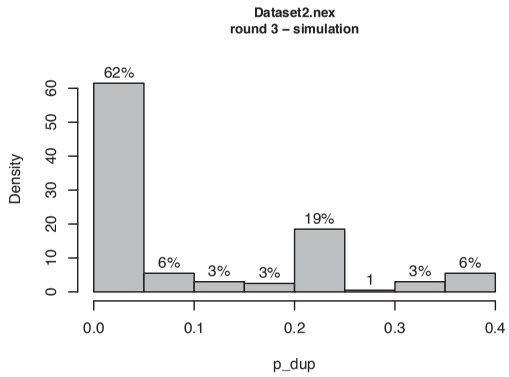


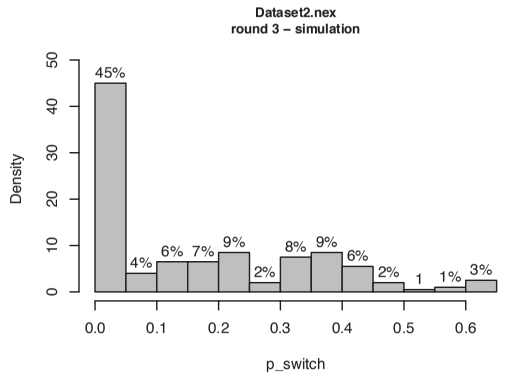


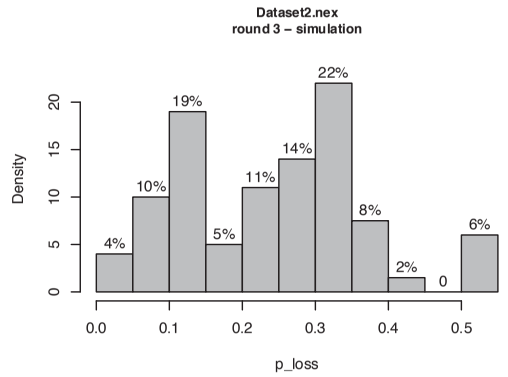


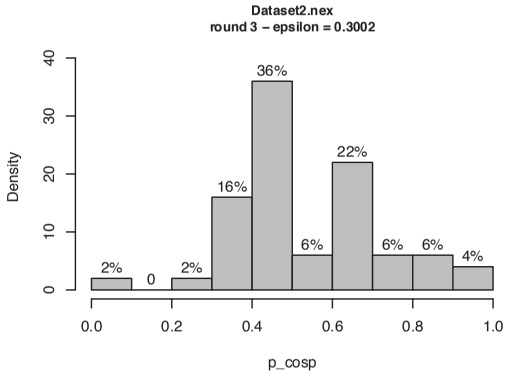
Nexus file: Dataset2.nex
Host/Parasite tree: 17/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 3 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

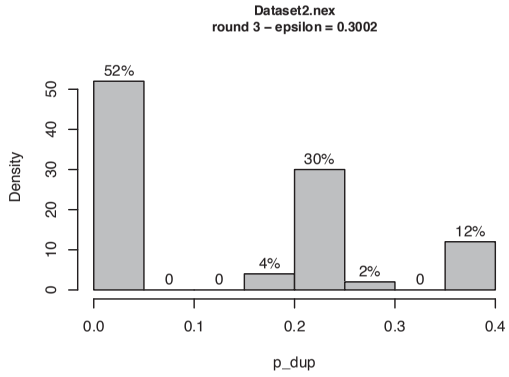


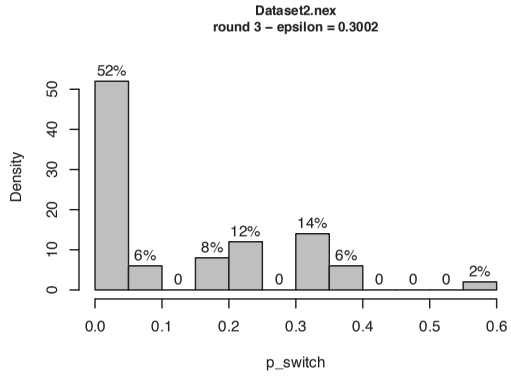


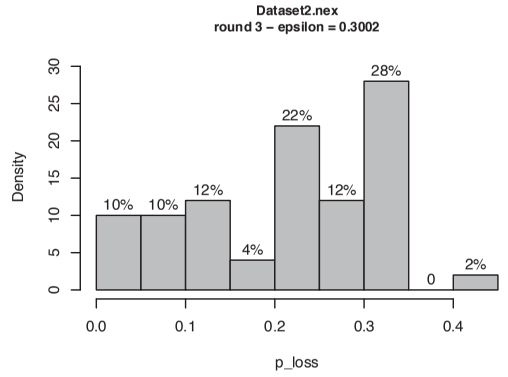


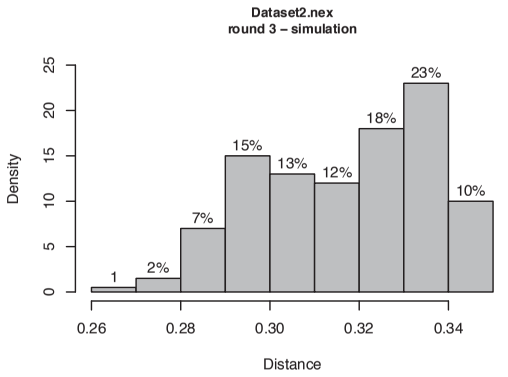


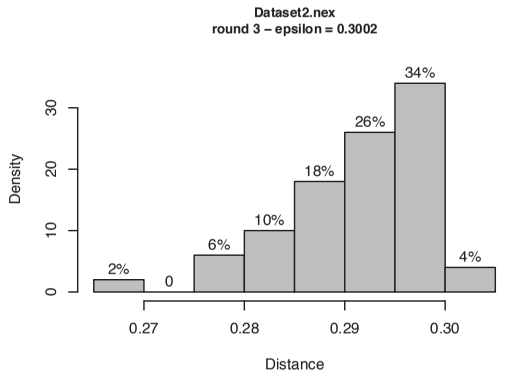




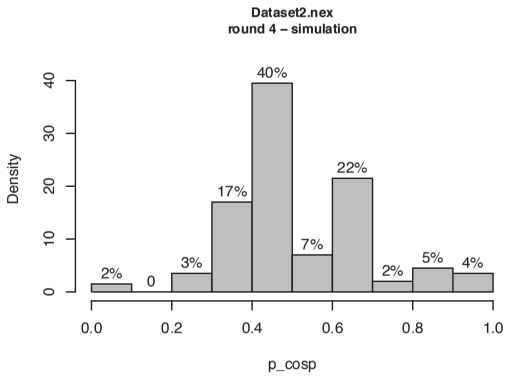


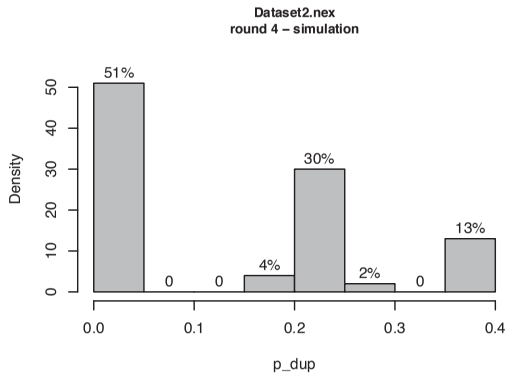


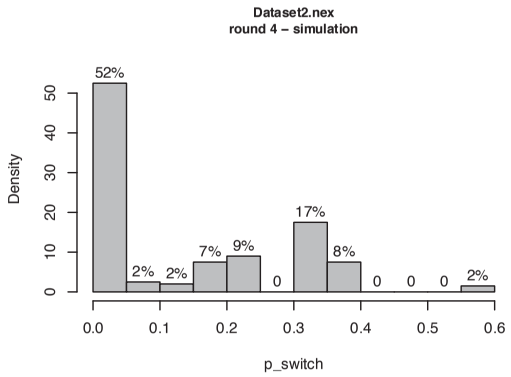


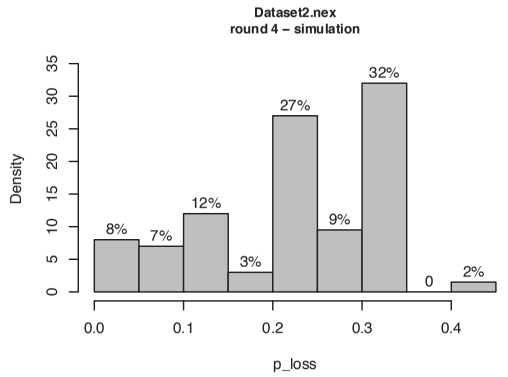


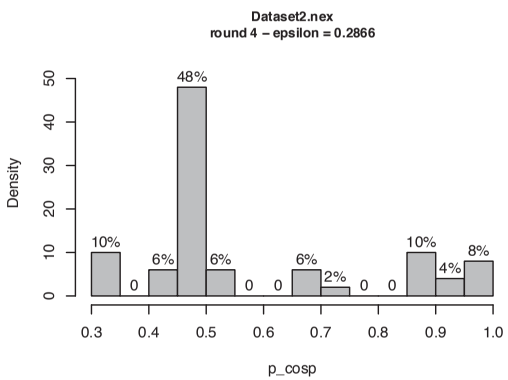
Nexus file: Dataset2.nex
Host/Parasite tree: 17/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 4 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

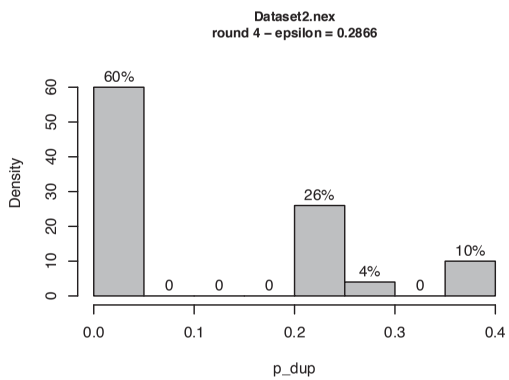


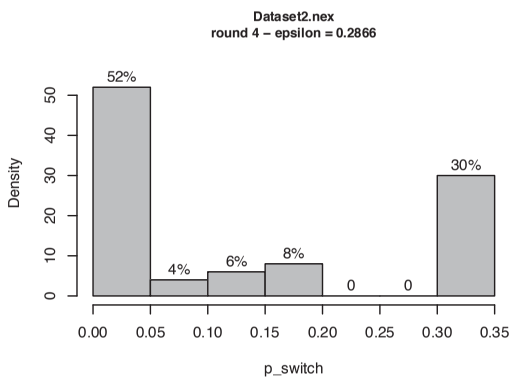


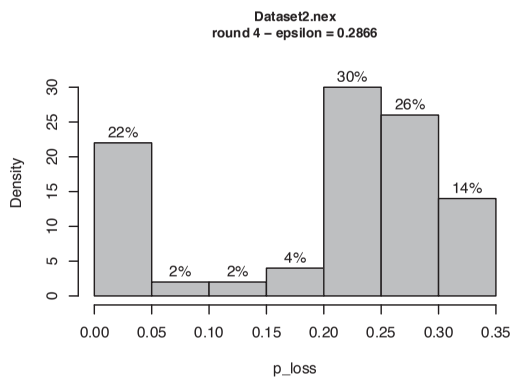


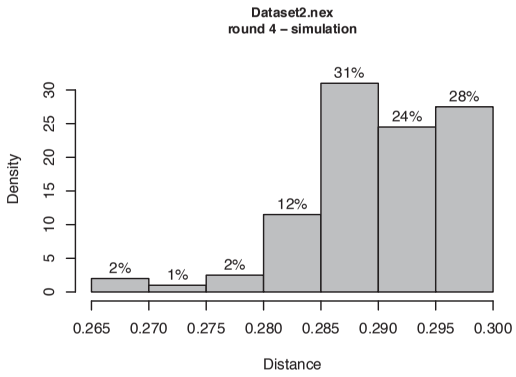


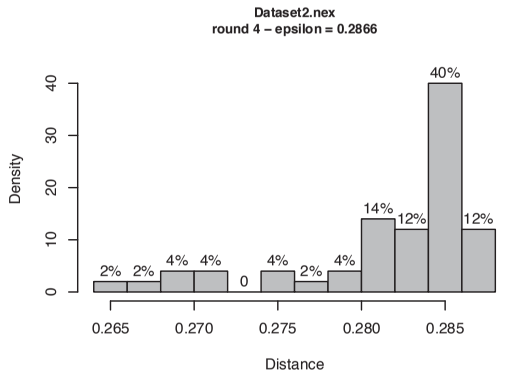




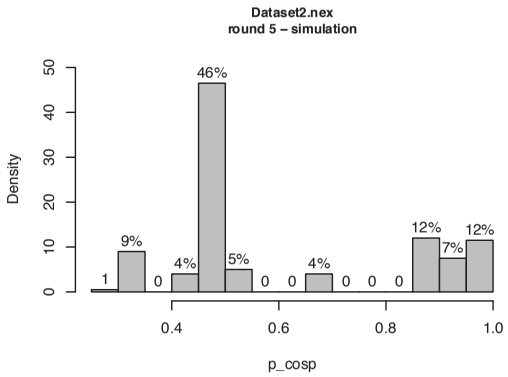


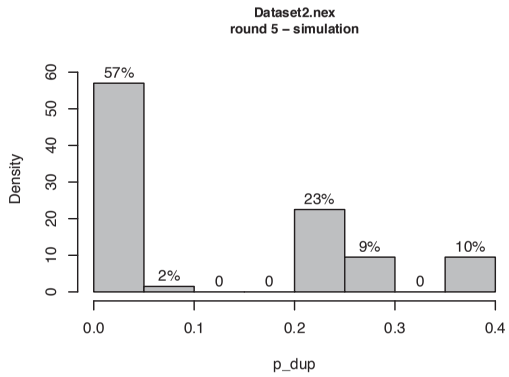


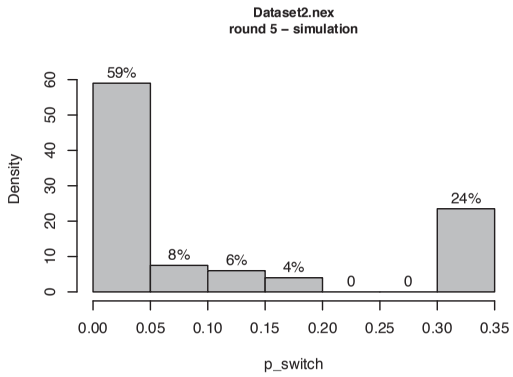


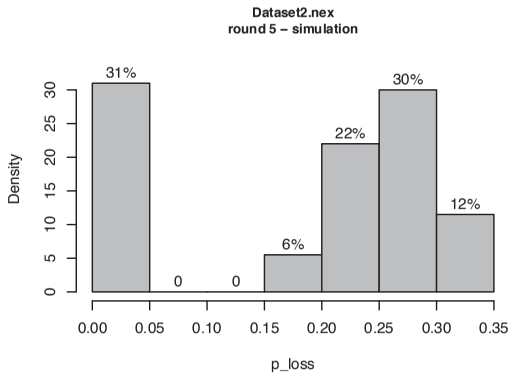


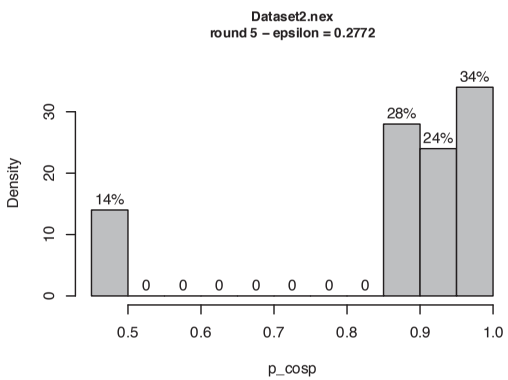
Nexus file: Dataset2.nex
Host/Parasite tree: 17/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

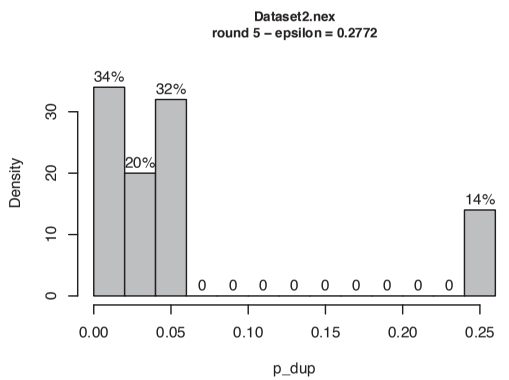


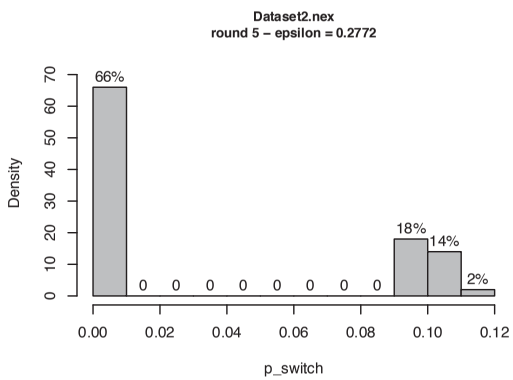


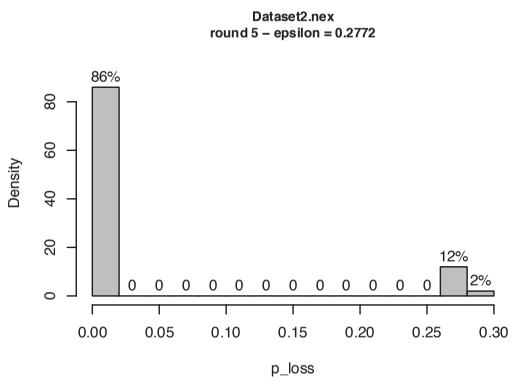


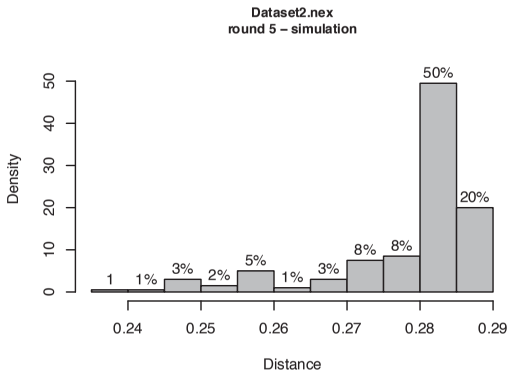


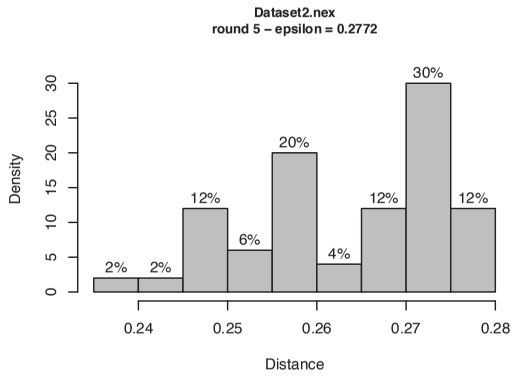




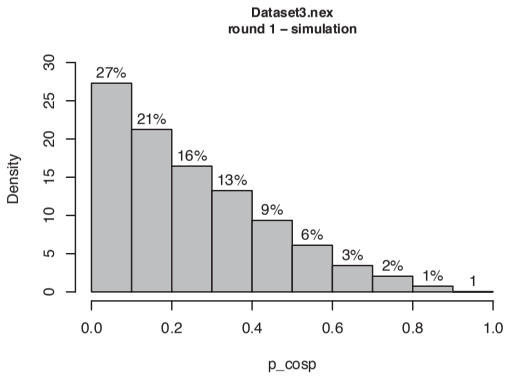


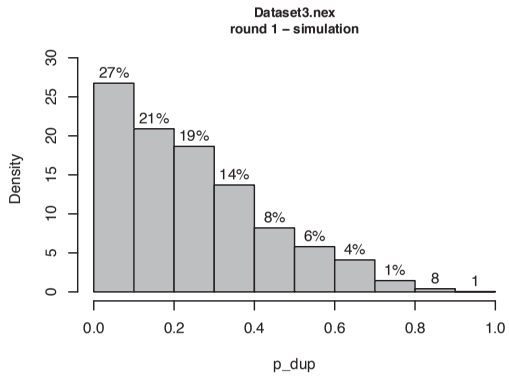


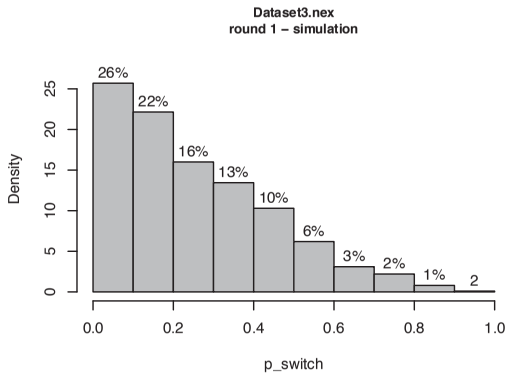


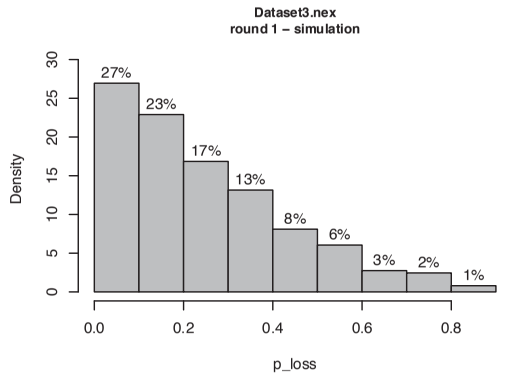


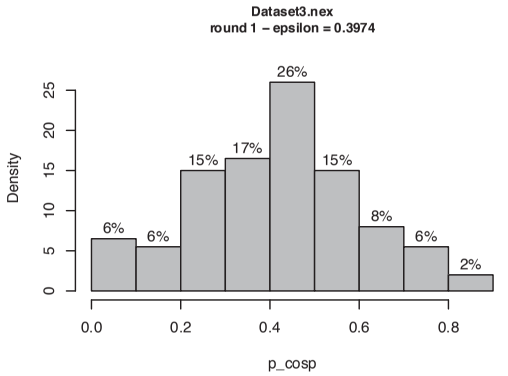
Nexus file: Dataset2.nex
Host/Parasite tree: 17/17 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

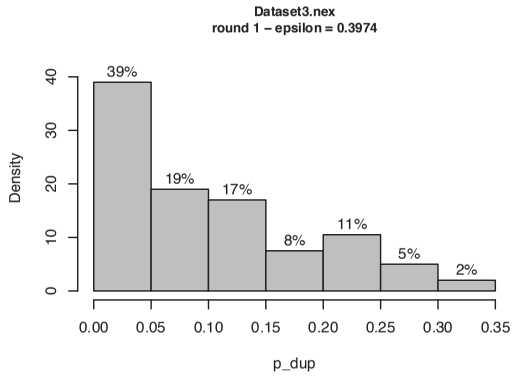


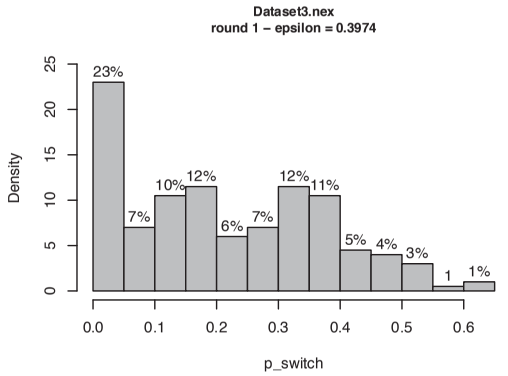


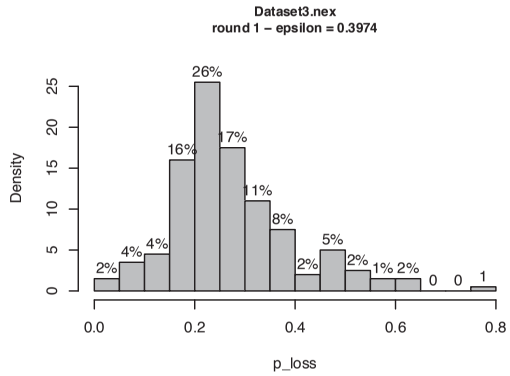


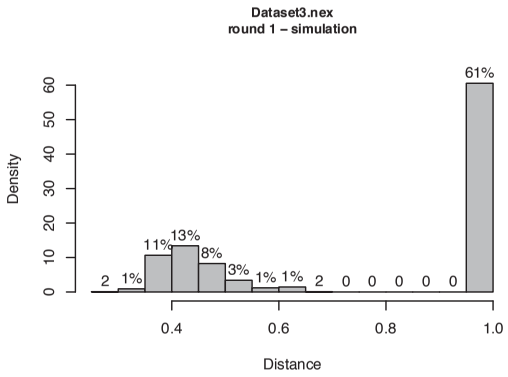


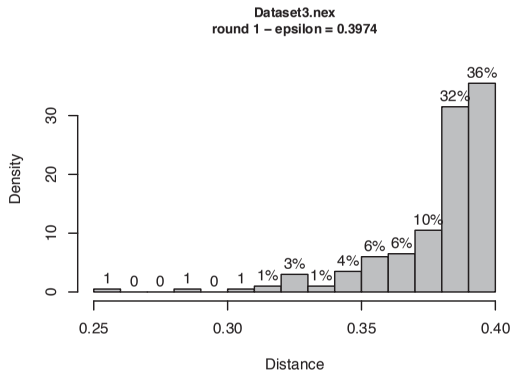




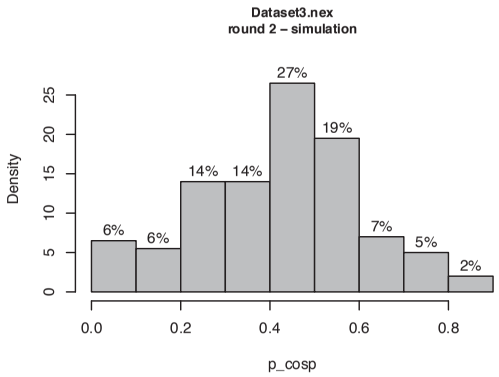


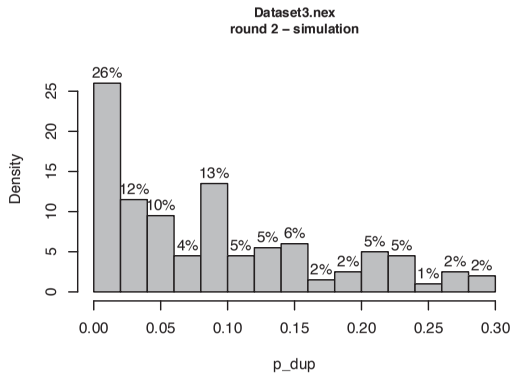


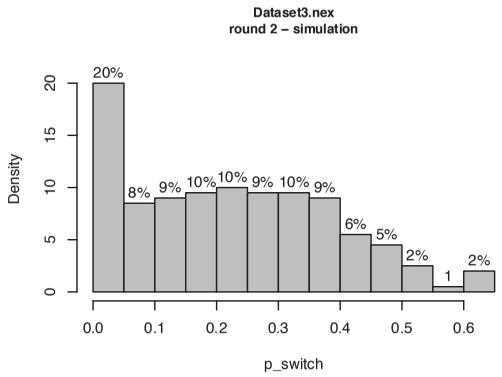


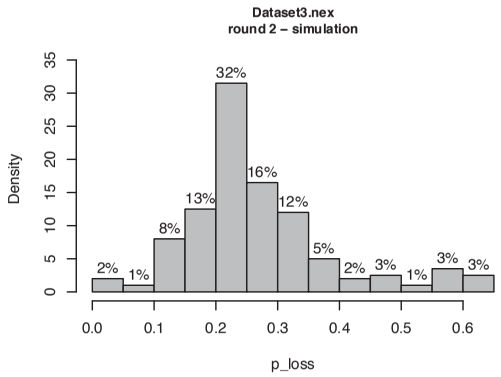


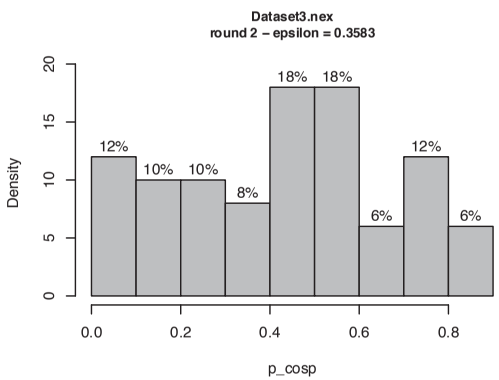
Nexus file: Dataset3.nex
Host/Parasite tree: 18/16 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 2 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

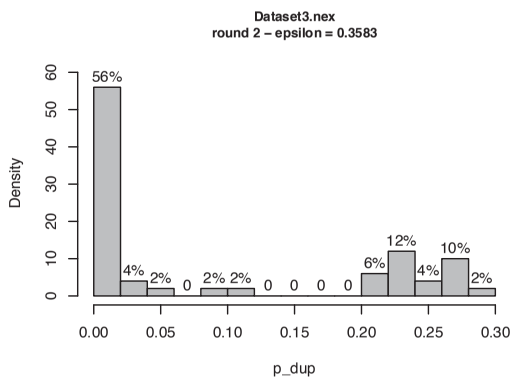


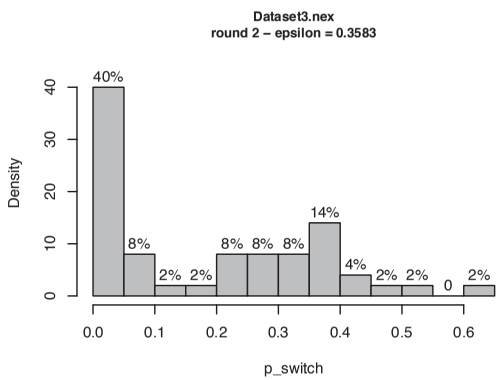


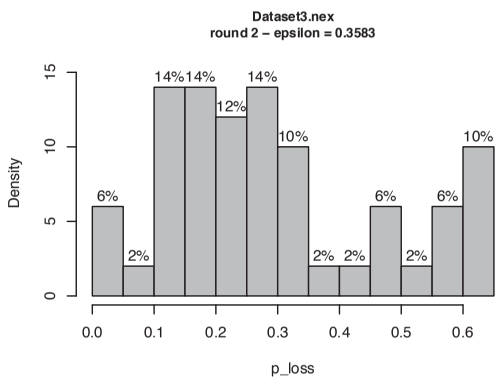


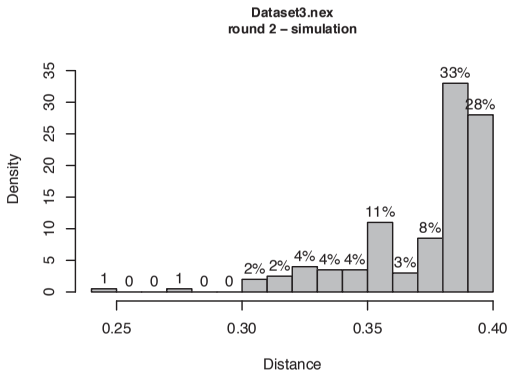


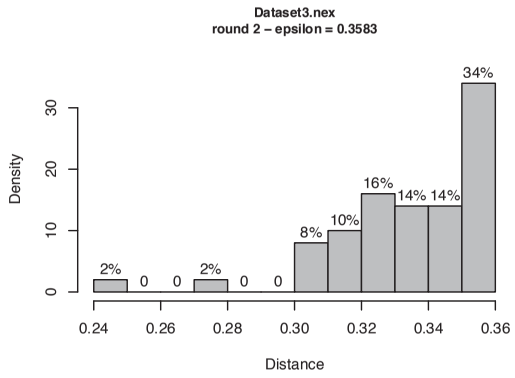




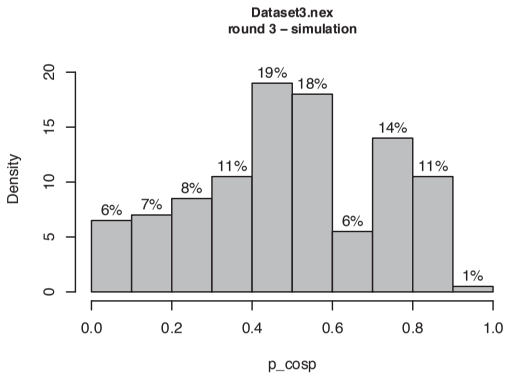


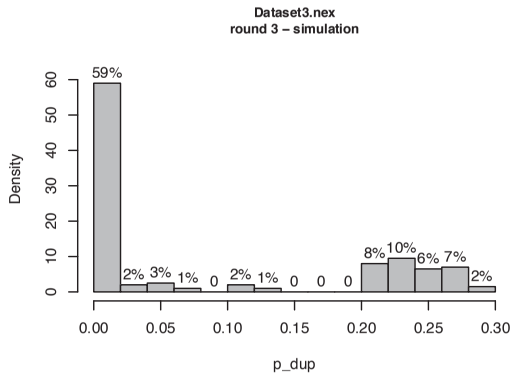


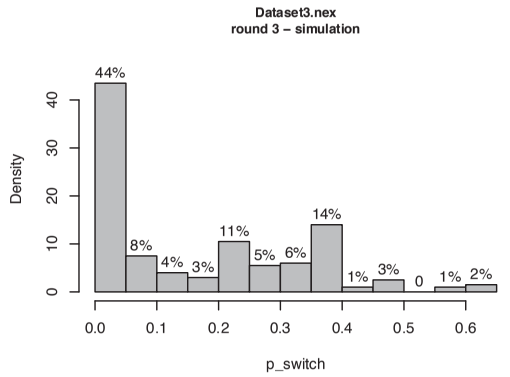


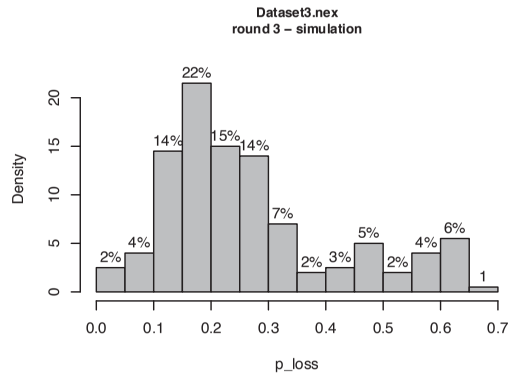


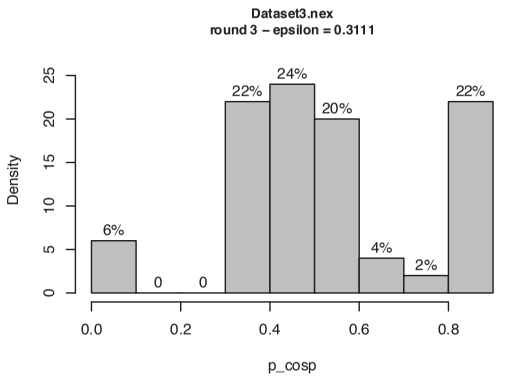
Nexus file: Dataset3.nex
Host/Parasite tree: 18/16 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 3 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

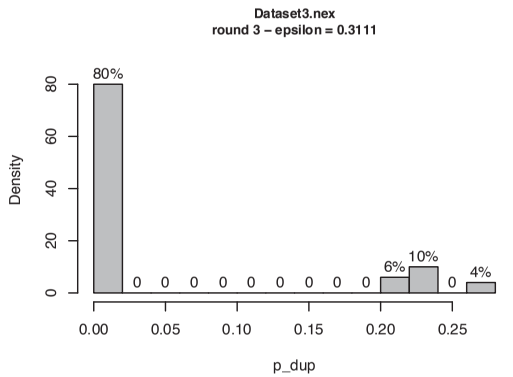


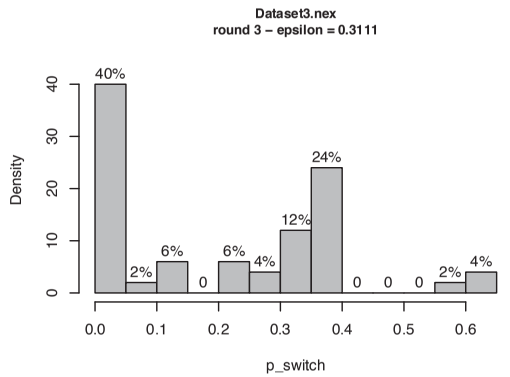


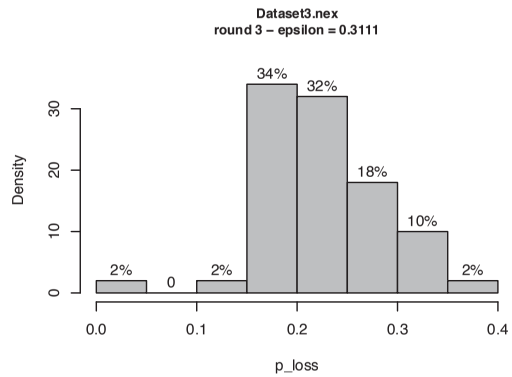


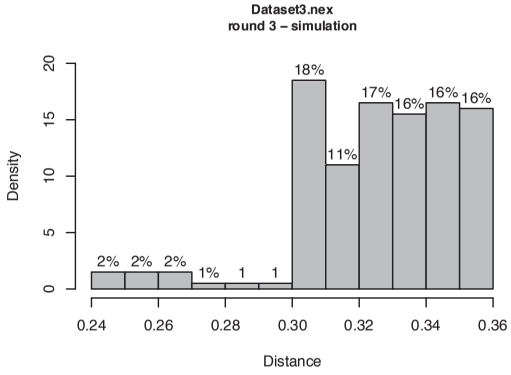


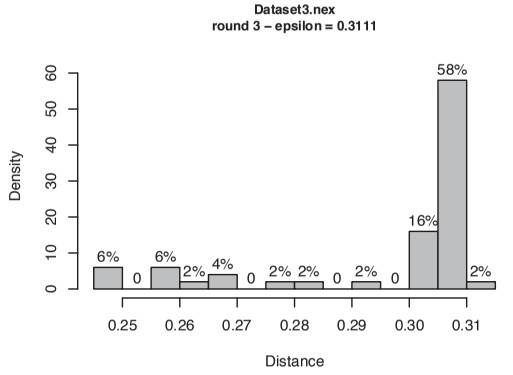




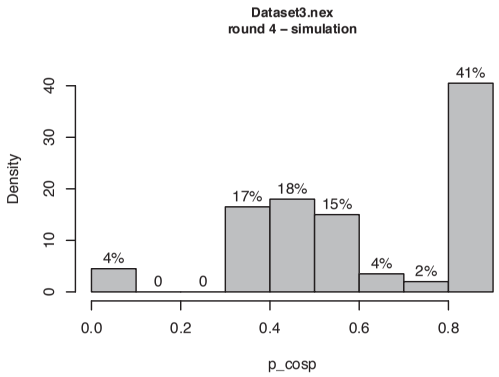


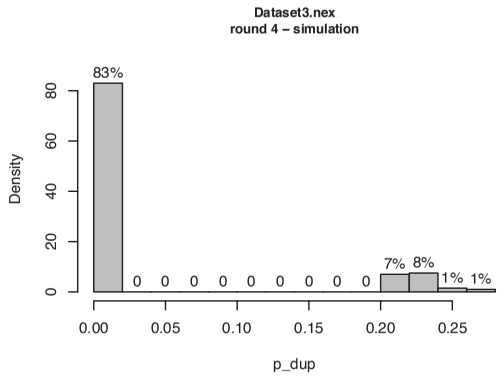


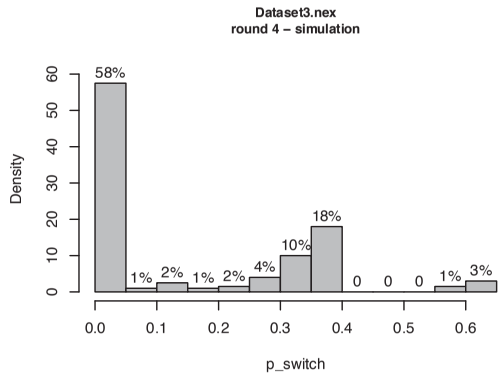


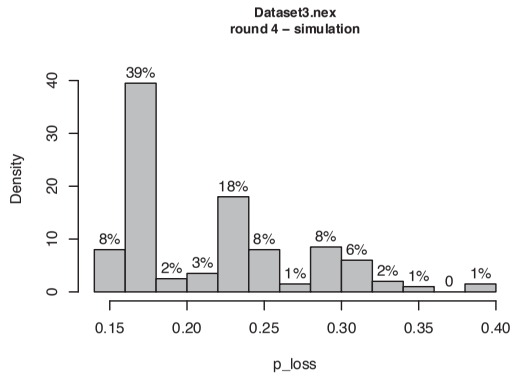


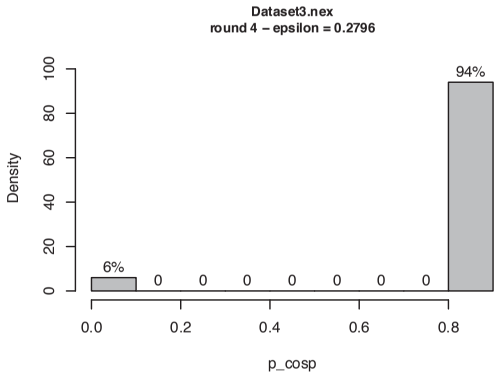
Nexus file: Dataset3.nex
Host/Parasite tree: 18/16 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 4 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

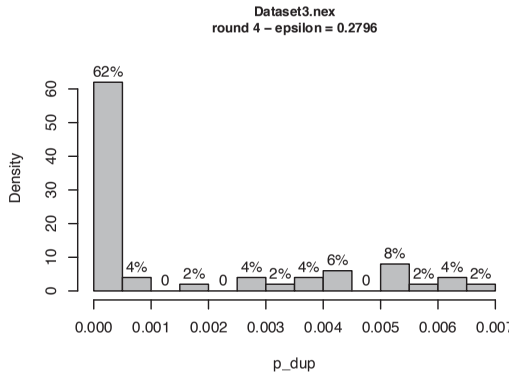


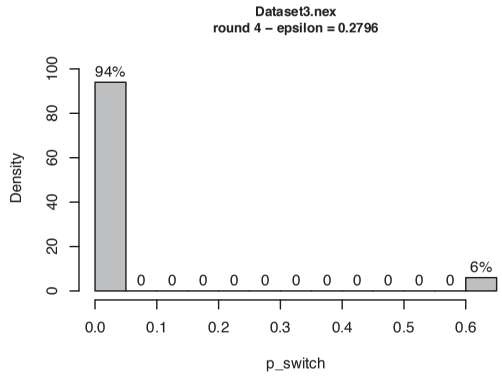


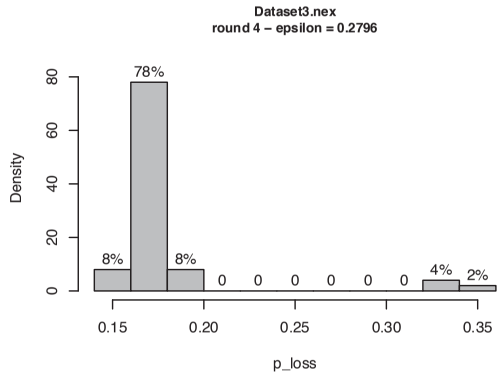


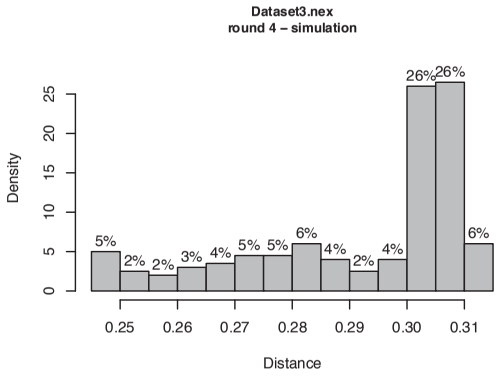


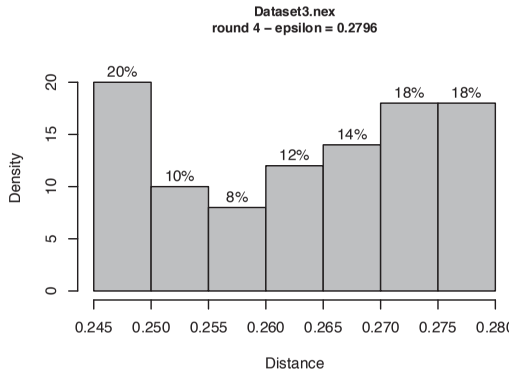




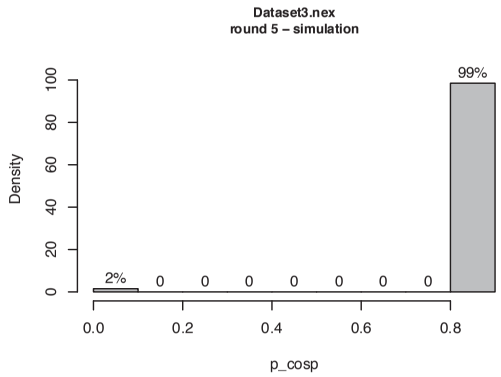


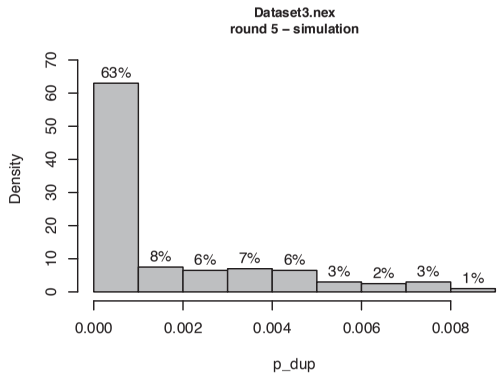


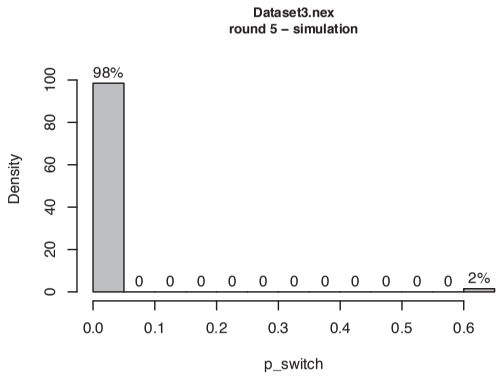


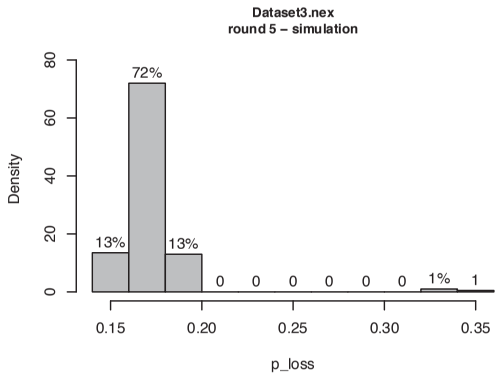


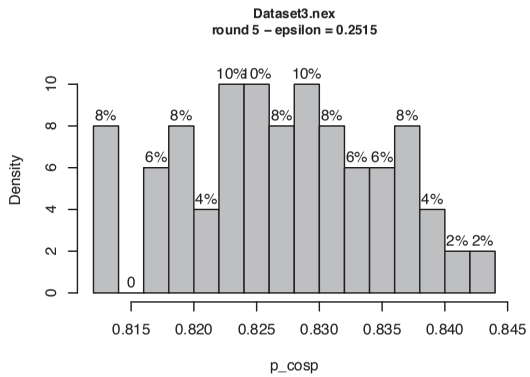
Nexus file: Dataset3.nex
Host/Parasite tree: 18/16 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

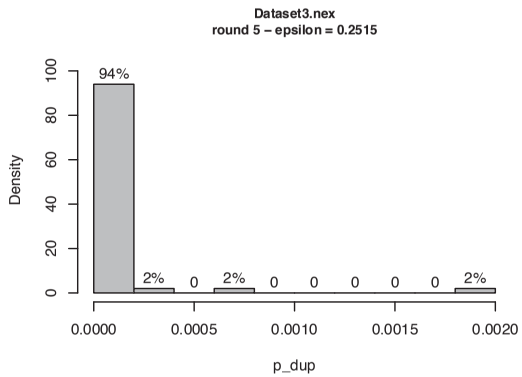


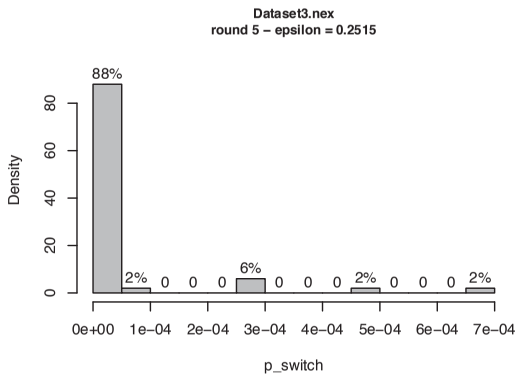


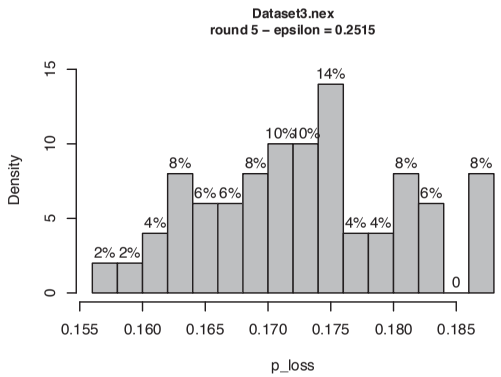


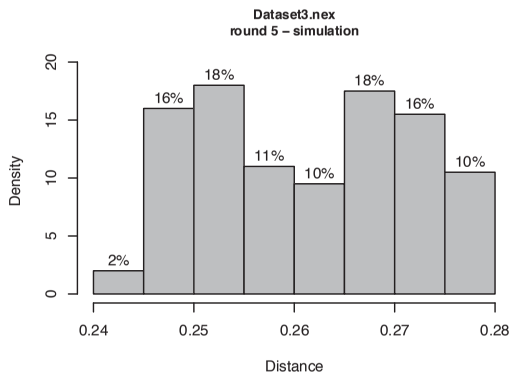


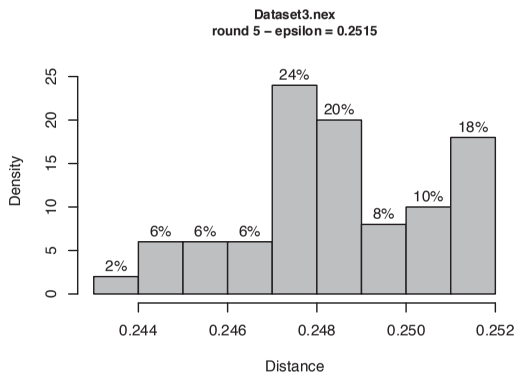




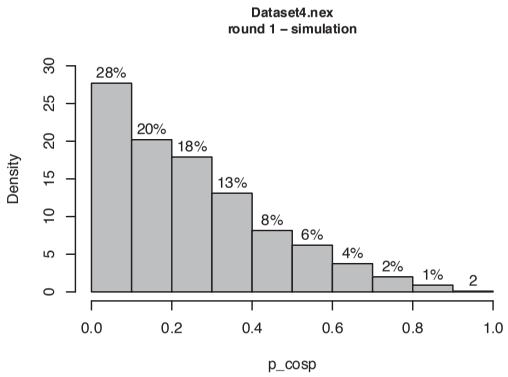


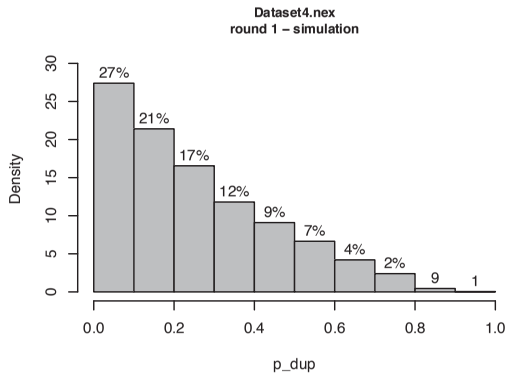


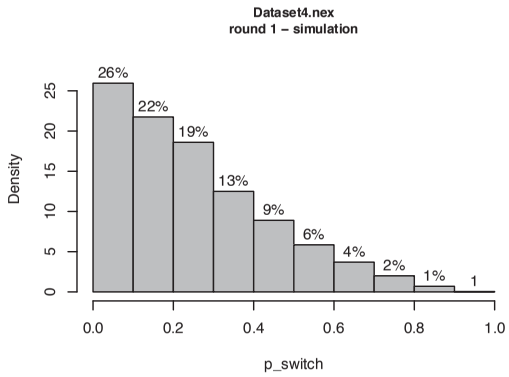


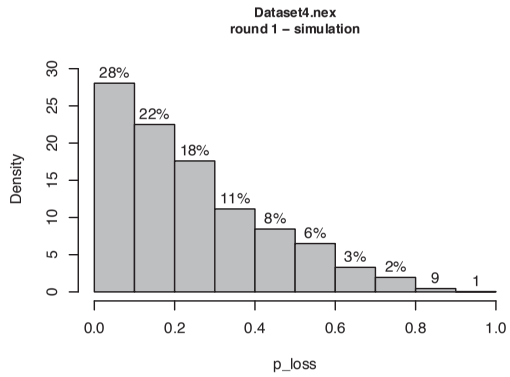


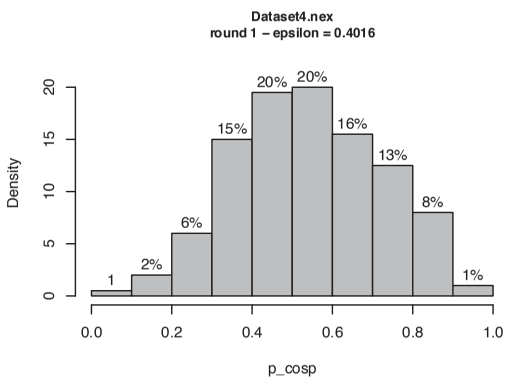
Nexus file: Dataset3.nex
Host/Parasite tree: 18/16 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

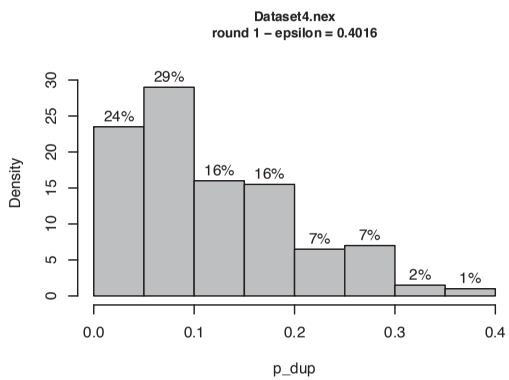


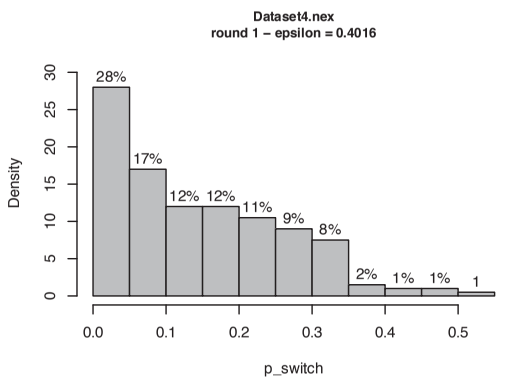


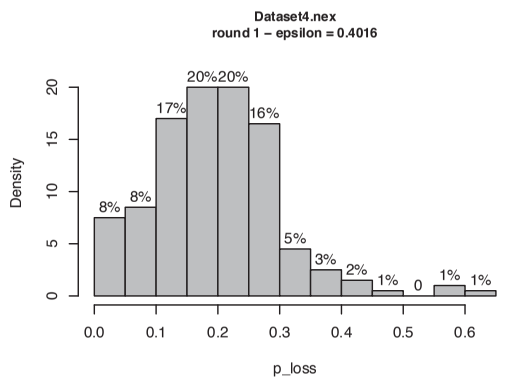


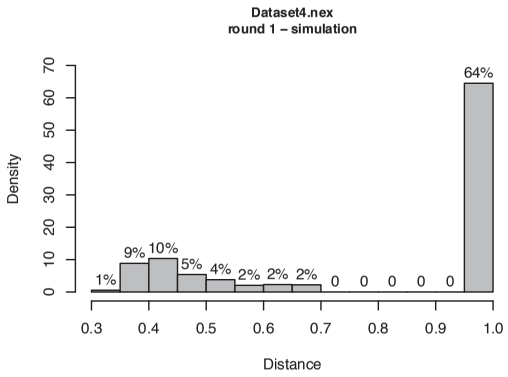


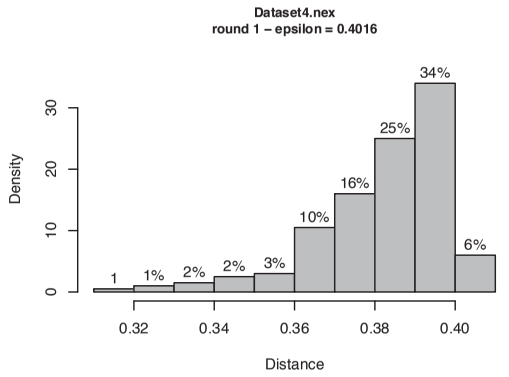




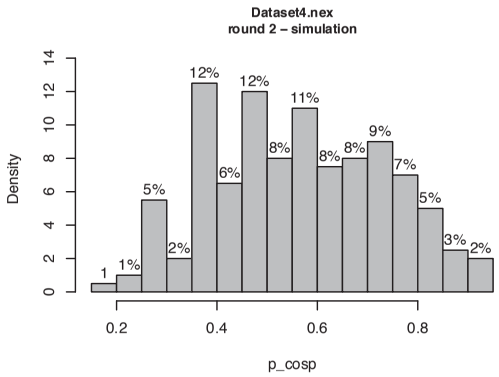


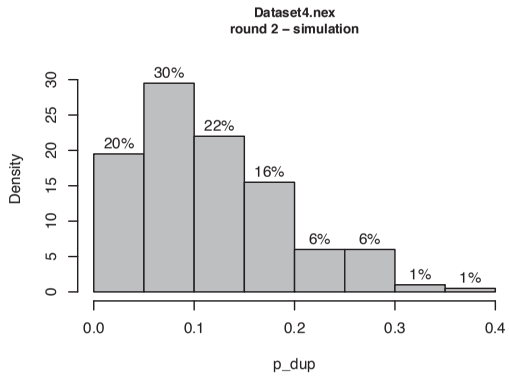


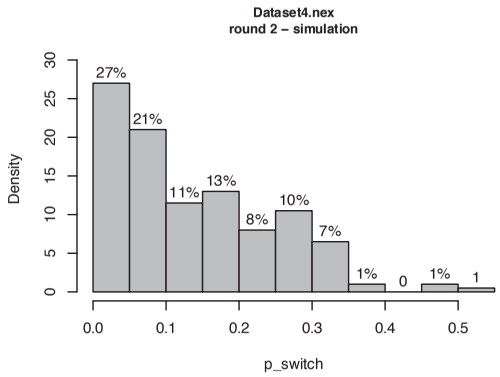


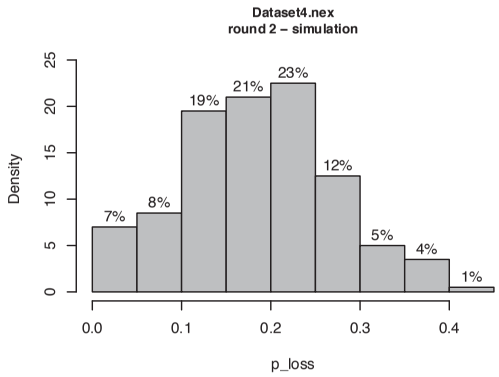


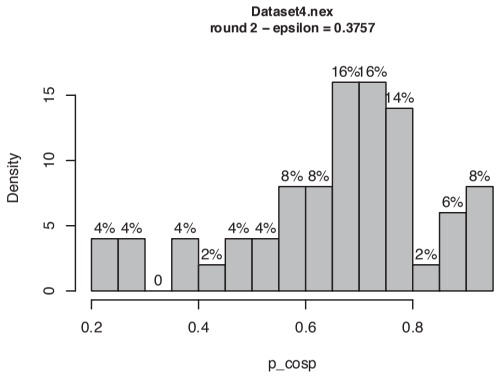
Nexus file: Dataset4.nex
Host/Parasite tree: 34/42 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 2 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

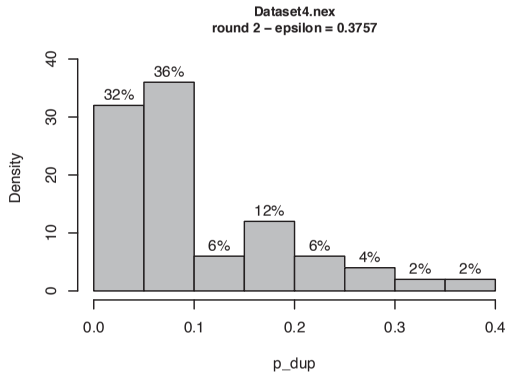


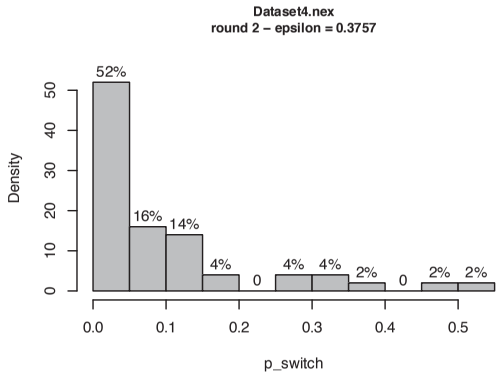


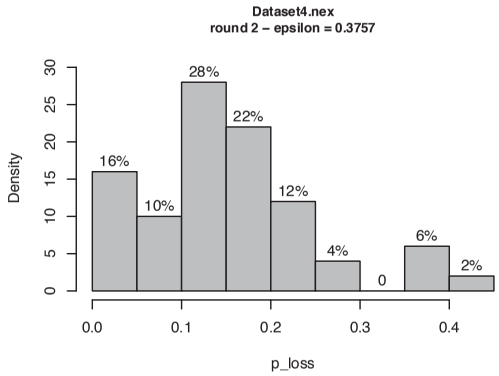


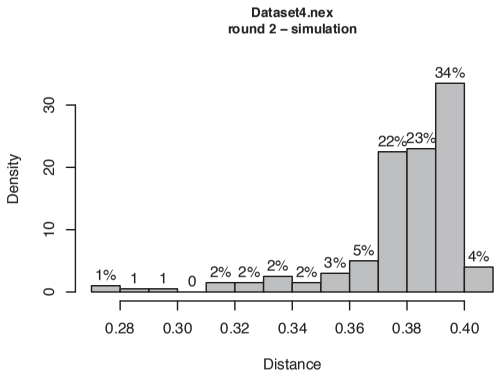


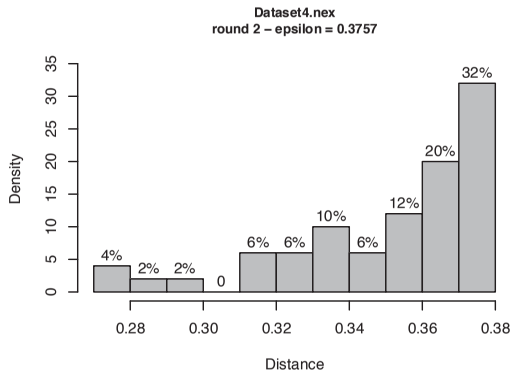




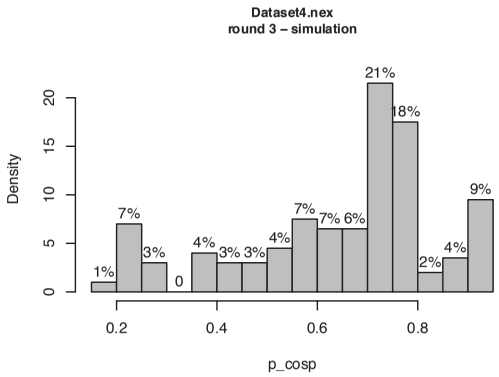


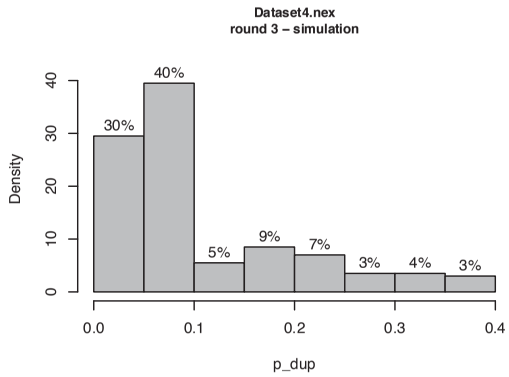


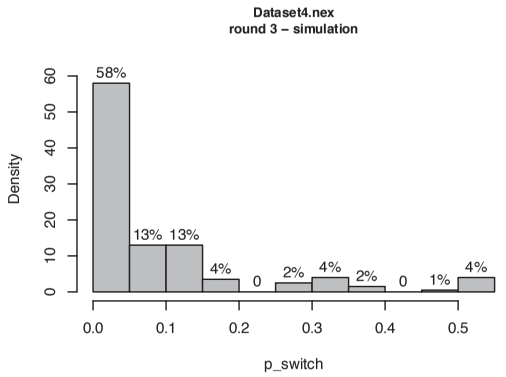


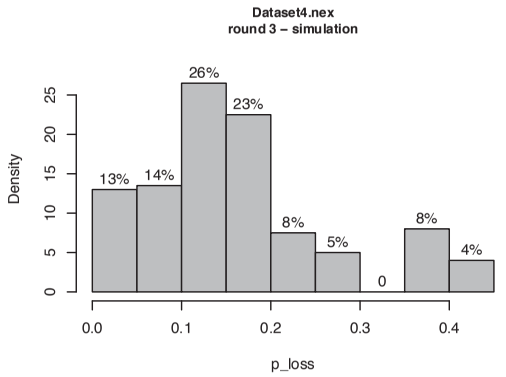


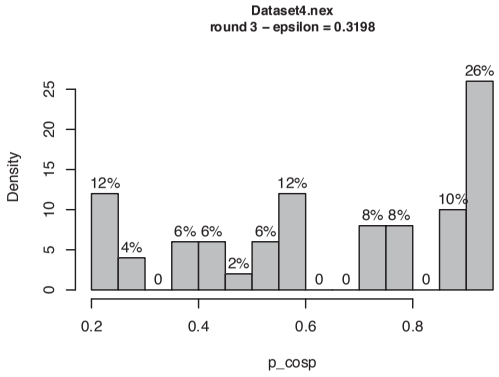
Nexus file: Dataset4.nex
Host/Parasite tree: 34/42 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 3 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

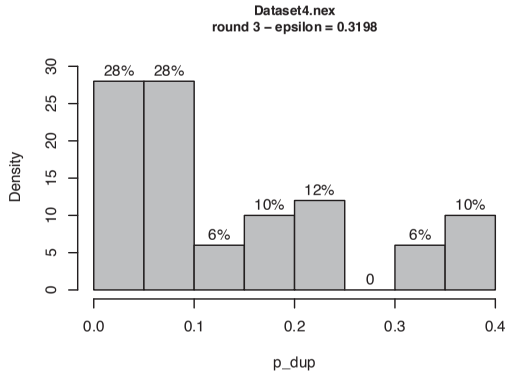


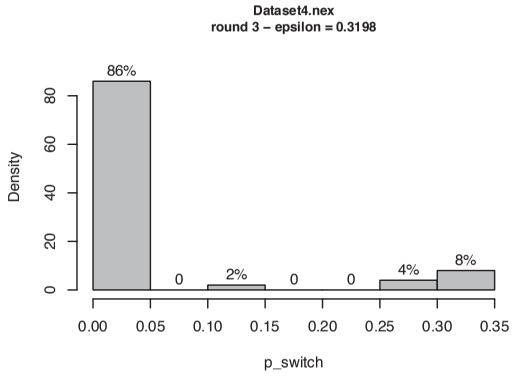


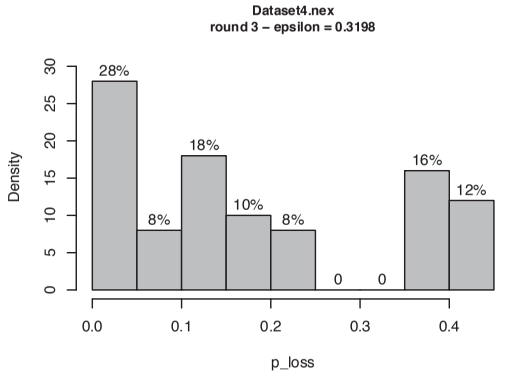


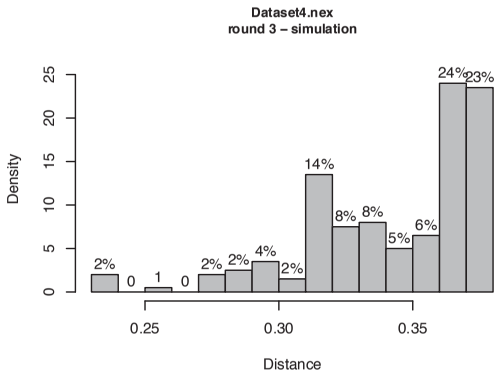


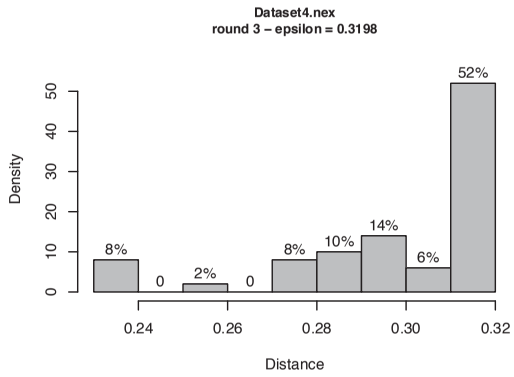




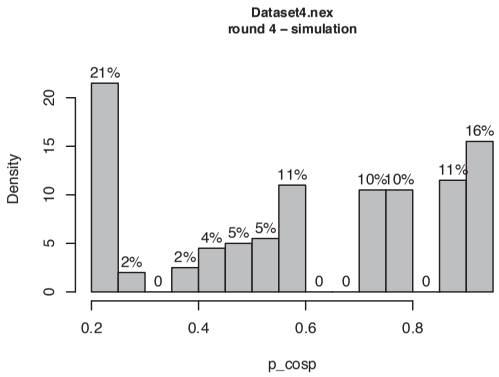


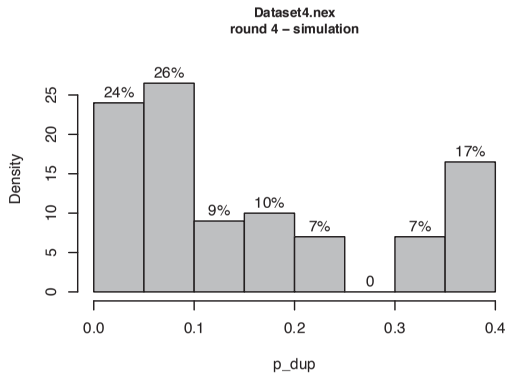


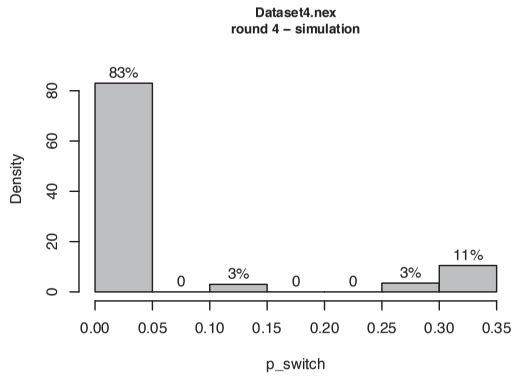


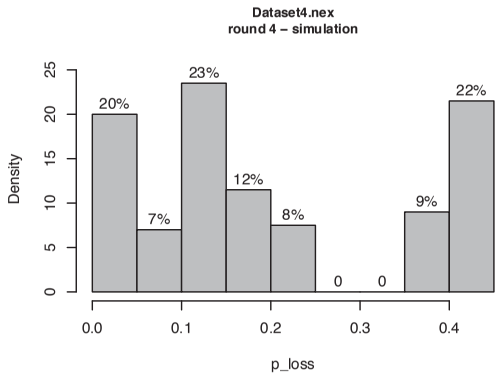


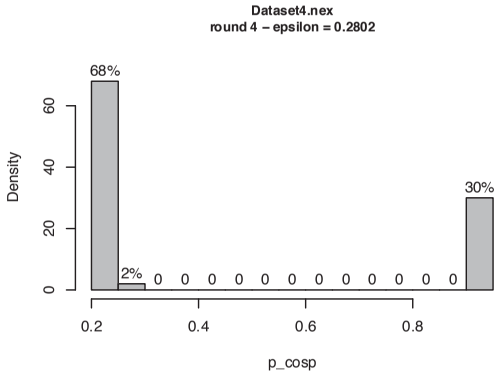
Nexus file: Dataset4.nex
Host/Parasite tree: 34/42 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 4 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

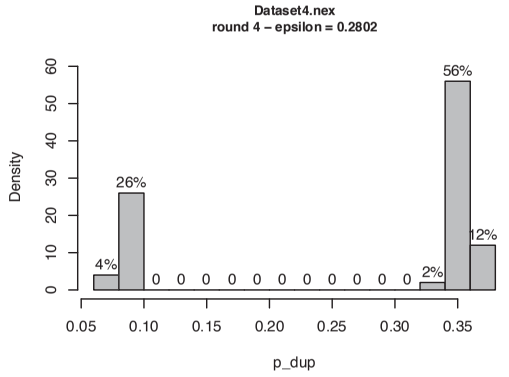


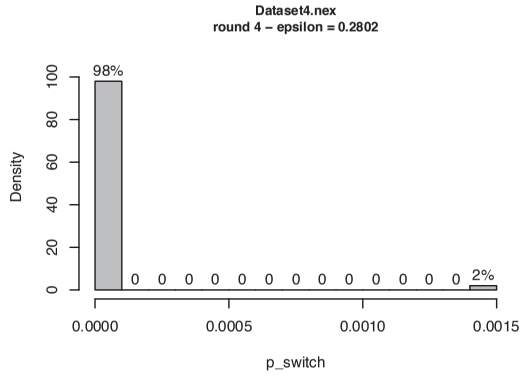


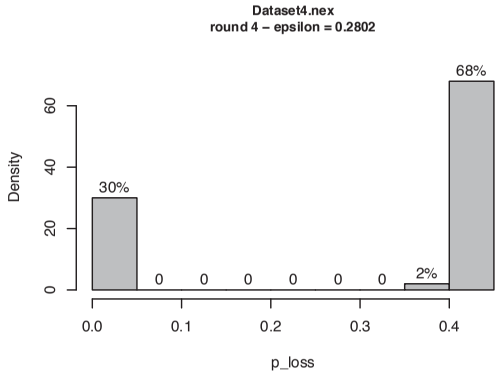


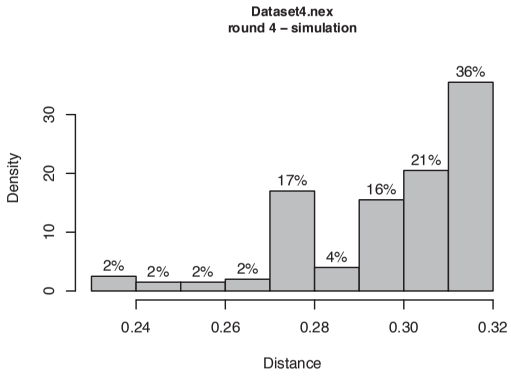


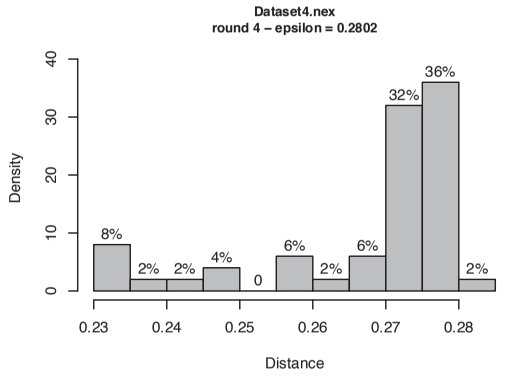




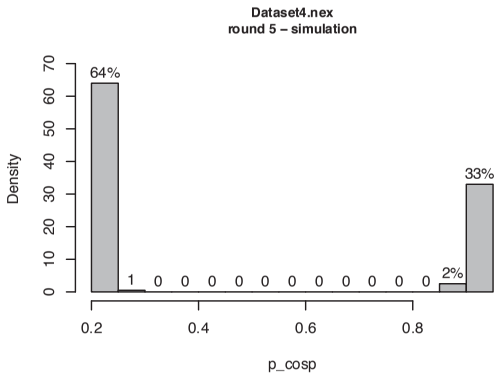


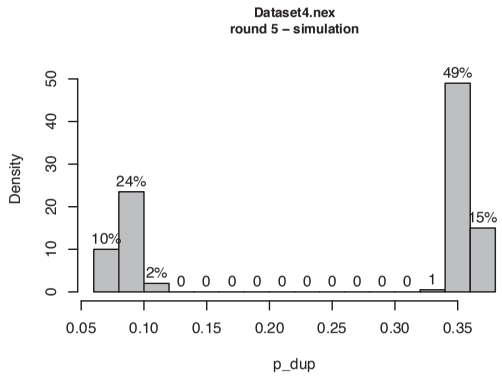


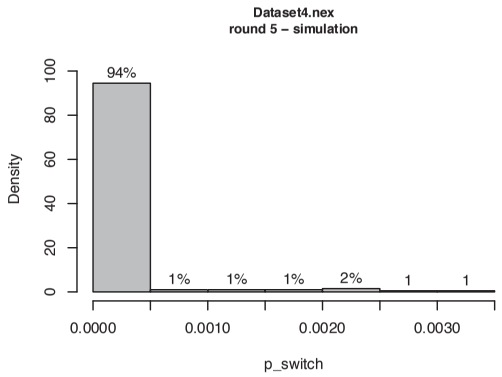


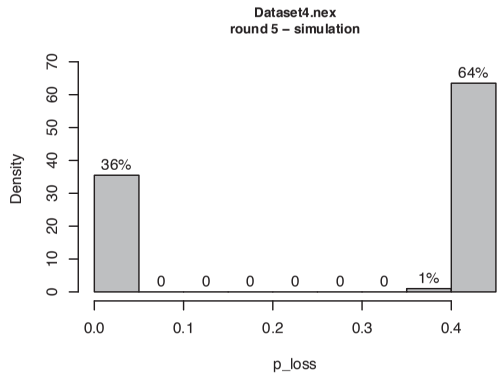


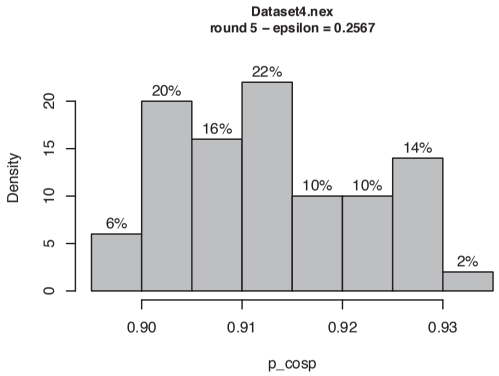
Nexus file: Dataset4.nex
Host/Parasite tree: 34/42 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

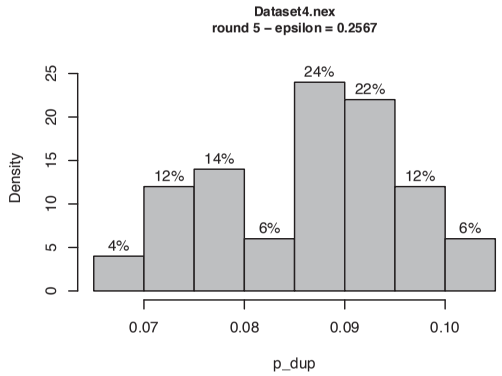


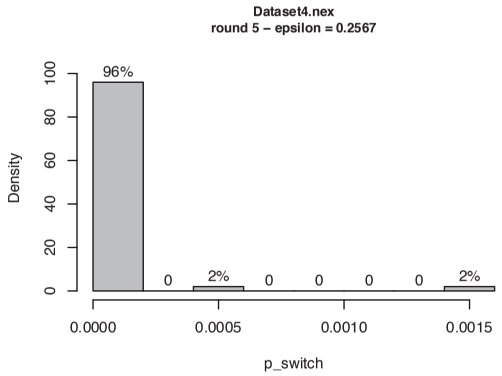


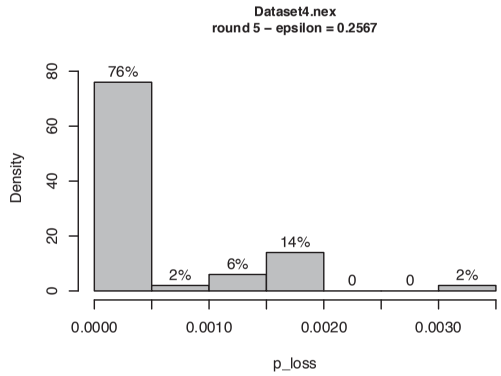


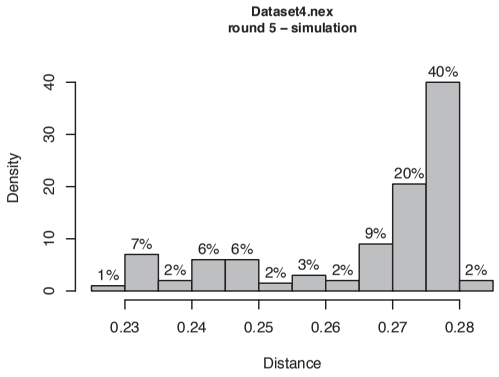


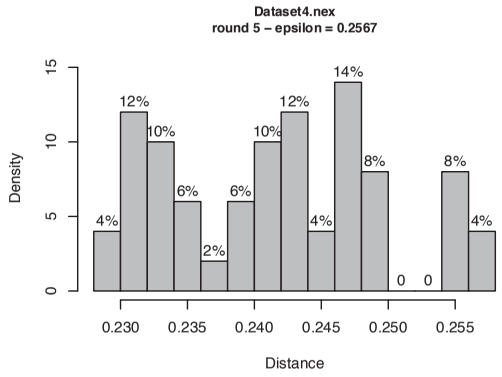












Nexus file: Dataset4.nex
Host/Parasite tree: 34/42 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

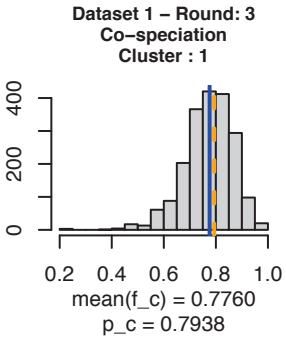
Comparison between probabilities and frequency of events

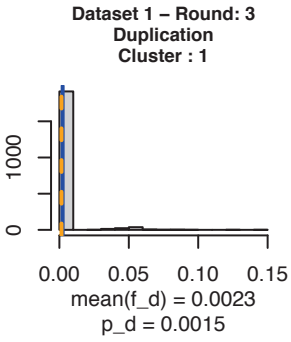
Let $\langle n_c, n_d, n_s, n_l \rangle$ be the vector of the events related to a given reconciliation. We can compute its corresponding vector of frequencies $\langle f_c, f_d, f_s, f_l \rangle$ in the following way: $f_i = n_i / \sum_{i \in \{c,d,s,l\}} n_j$.

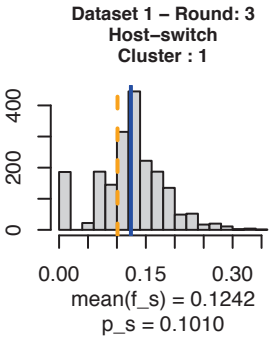
If we generate a parasite tree by fixing a host tree H and a probability vector $\theta = \langle p_c, p_d, p_s, p_l \rangle$, we can thus compare if the vectors of frequencies of the generated trees are close to θ.

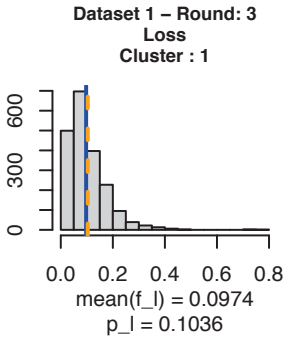
To perform this test, we looked for the results of the biological datasets (Table 3). For each dataset D, we took the set C_D^R of representative probability vectors of the clusters produced at round R ($R \in 3, 5$). Then, for each vector $\theta_i \in C_D^R$, we used our simulation model to generate 2000 trees using the host tree H_D (of the dataset D) and θ_i.

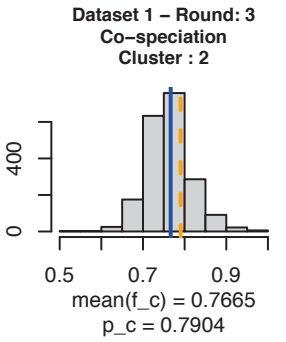
In the next pages, we can find for each pair dataset and round, the histograms of the event frequencies observed on the 2000 simulated trees generated for each cluster. The blue vertical line indicates the mean frequency value and the dashed orange line represents the probability value used in the simulation. We can observe that both values (mean frequency value and probability value) are, in general, very close independently of the dataset and probability vector.

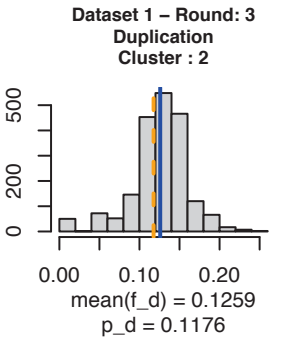

Dataset 1 – Round: 3
Co–speciation
Cluster : 1
mean(f_c) = 0.7760
p_c = 0.7938

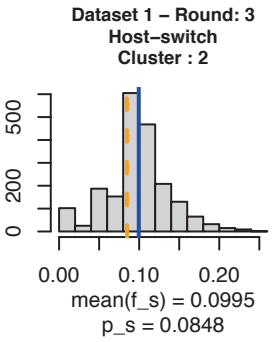

Dataset 1 – Round: 3
Duplication
Cluster : 1
mean(f_d) = 0.0023
p_d = 0.0015

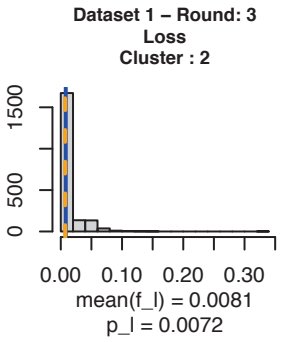

Dataset 1 – Round: 3
Host–switch
Cluster : 1
mean(f_s) = 0.1242
p_s = 0.1010

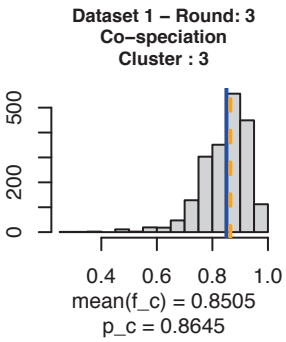

Dataset 1 – Round: 3
Loss
Cluster : 1
mean(f_l) = 0.0974
p_l = 0.1036

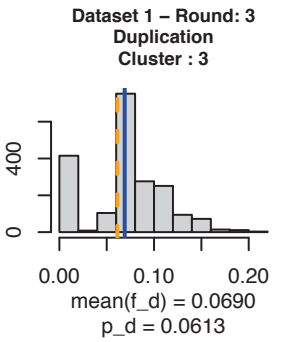

Dataset 1 – Round: 3
Co–speciation
Cluster : 2
mean(f_c) = 0.7665
p_c = 0.7904

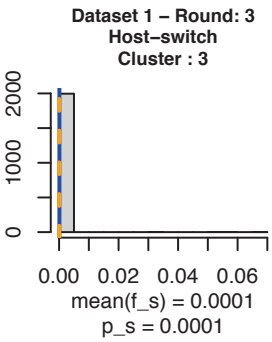

Dataset 1 – Round: 3
Duplication
Cluster : 2
mean(f_d) = 0.1259
p_d = 0.1176

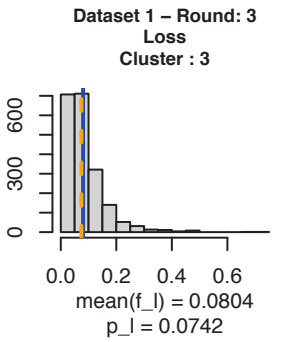

Dataset 1 – Round: 3
Host–switch
Cluster : 2
mean(f_s) = 0.0995
p_s = 0.0848

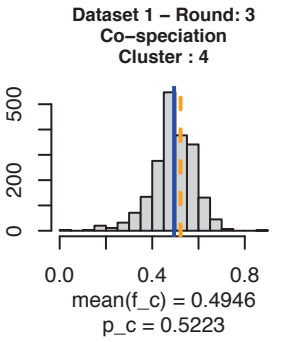

Dataset 1 – Round: 3
Loss
Cluster : 2
mean(f_l) = 0.0081
p_l = 0.0072

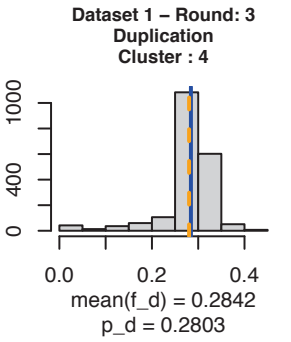

Dataset 1 – Round: 3
Co–speciation
Cluster : 3
mean(f_c) = 0.8505
p_c = 0.8645

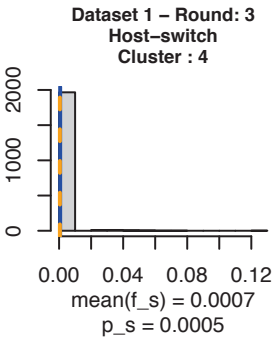

Dataset 1 – Round: 3
Duplication
Cluster : 3
mean(f_d) = 0.0690
p_d = 0.0613

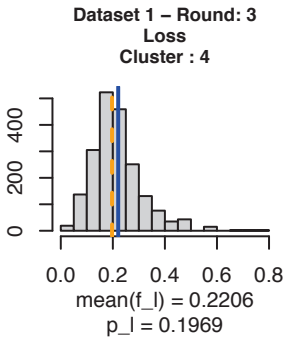

Dataset 1 – Round: 3
Host–switch
Cluster : 3
mean(f_s) = 0.0001
p_s = 0.0001

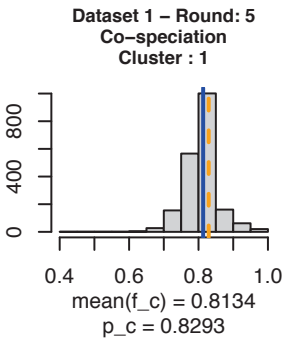

Dataset 1 – Round: 3
Loss
Cluster : 3
mean(f_l) = 0.0804
p_l = 0.0742

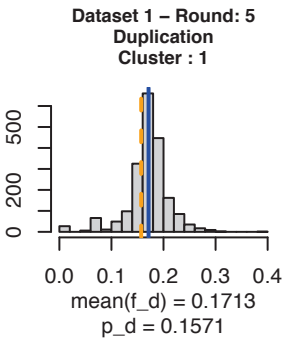

Dataset 1 – Round: 3
Co–speciation
Cluster : 4
mean(f_c) = 0.4946
p_c = 0.5223

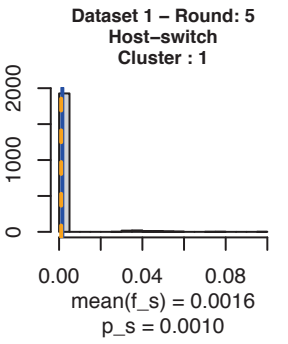

Dataset 1 – Round: 3
Duplication
Cluster : 4
mean(f_d) = 0.2842
p_d = 0.2803

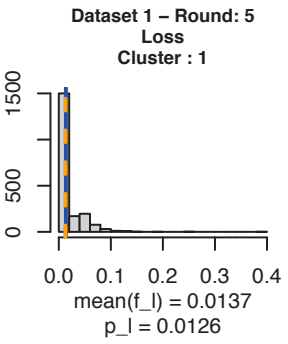

Dataset 1 – Round: 3
Host–switch
Cluster : 4
mean(f_s) = 0.0007
p_s = 0.0005

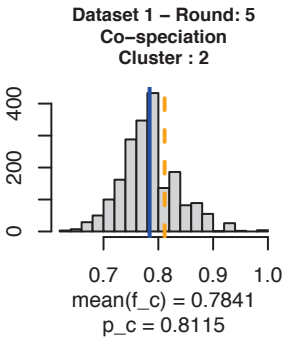

Dataset 1 – Round: 3
Loss
Cluster : 4
mean(f_l) = 0.2206
p_l = 0.1969

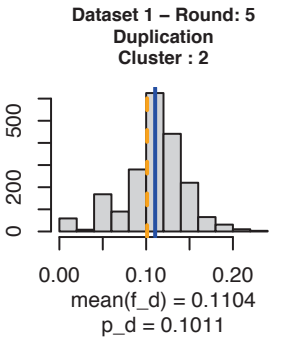

Dataset 1 – Round: 5
Co–speciation
Cluster : 1
mean(f_c) = 0.8134
p_c = 0.8293

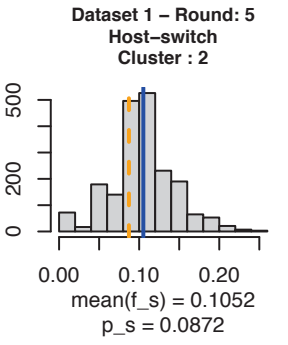

Dataset 1 – Round: 5
Duplication
Cluster : 1
mean(f_d) = 0.1713
p_d = 0.1571

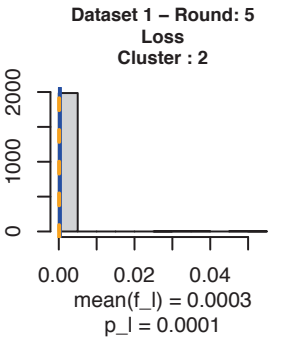

Dataset 1 – Round: 5
Host–switch
Cluster : 1
mean(f_s) = 0.0016
p_s = 0.0010

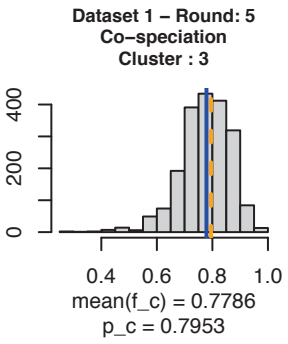

Dataset 1 – Round: 5
Loss
Cluster : 1
mean(f_l) = 0.0137
p_l = 0.0126

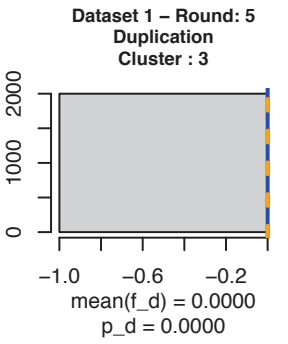

Dataset 1 – Round: 5
Co–speciation
Cluster : 2
mean(f_c) = 0.7841
p_c = 0.8115

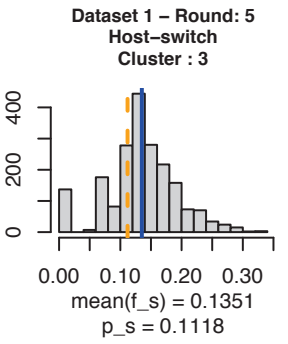

Dataset 1 – Round: 5
Duplication
Cluster : 2
mean(f_d) = 0.1104
p_d = 0.1011

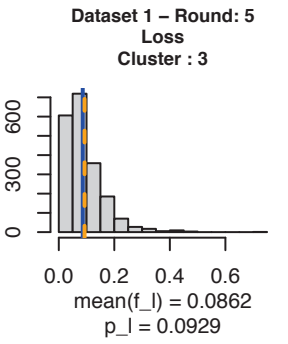

Dataset 1 – Round: 5
Host–switch
Cluster : 2
mean(f_s) = 0.1052
p_s = 0.0872

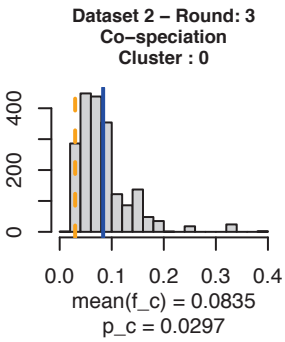

Dataset 1 – Round: 5
Loss
Cluster : 2
mean(f_l) = 0.0003
p_l = 0.0001

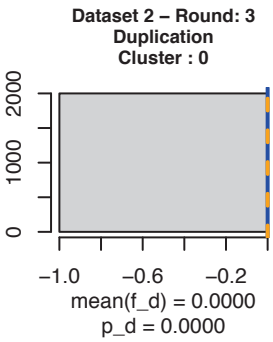

Dataset 1 – Round: 5
Co–speciation
Cluster : 3
mean(f_c) = 0.7786
p_c = 0.7953

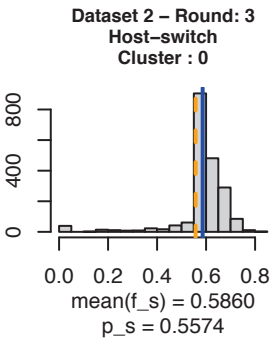

Dataset 1 – Round: 5
Duplication
Cluster : 3
mean(f_d) = 0.0000
p_d = 0.0000

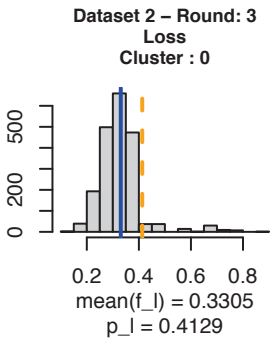

Dataset 1 – Round: 5
Host–switch
Cluster : 3
mean(f_s) = 0.1351
p_s = 0.1118

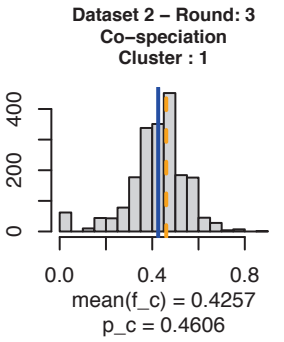

Dataset 1 – Round: 5
Loss
Cluster : 3
mean(f_l) = 0.0862
p_l = 0.0929

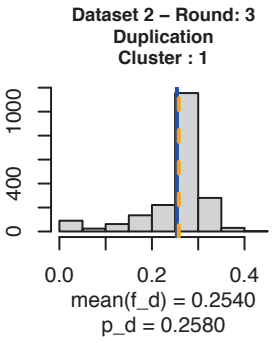

Dataset 2 – Round: 3
Co–speciation
Cluster : 0
mean(f_c) = 0.0835
p_c = 0.0297

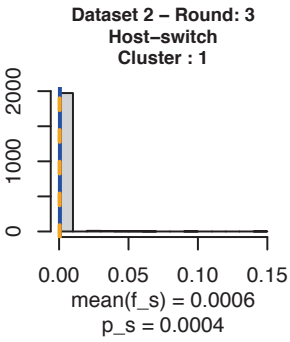

Dataset 2 – Round: 3
Duplication
Cluster : 0
mean(f_d) = 0.0000
p_d = 0.0000

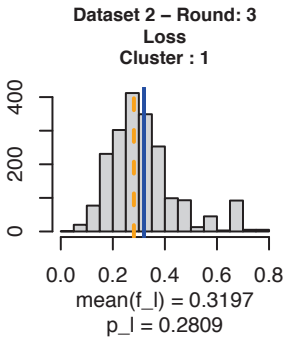

Dataset 2 – Round: 3
Host–switch
Cluster : 0
mean(f_s) = 0.5860
p_s = 0.5574

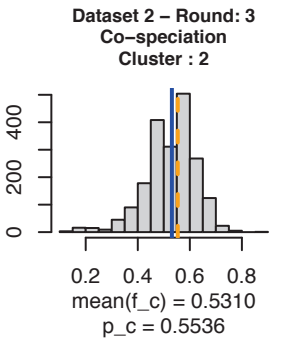

Dataset 2 – Round: 3
Loss
Cluster : 0
mean(f_l) = 0.3305
p_l = 0.4129

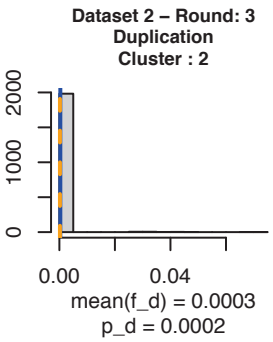

Dataset 2 – Round: 3
Co–speciation
Cluster : 1
mean(f_c) = 0.4257
p_c = 0.4606

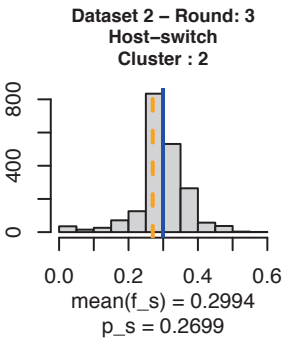

Dataset 2 – Round: 3
Duplication
Cluster : 1
mean(f_d) = 0.2540
p_d = 0.2580

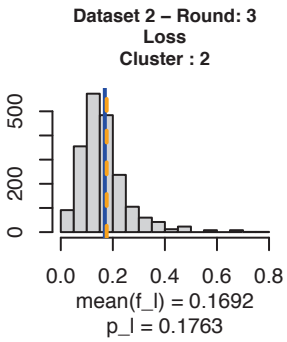

Dataset 2 – Round: 3
Host–switch
Cluster : 1
mean(f_s) = 0.0006
p_s = 0.0004

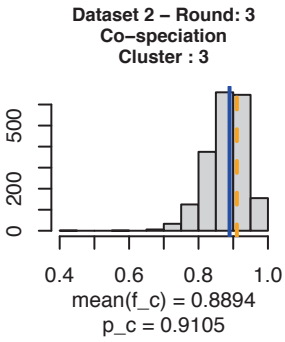

Dataset 2 – Round: 3
Loss
Cluster : 1
mean(f_l) = 0.3197
p_l = 0.2809

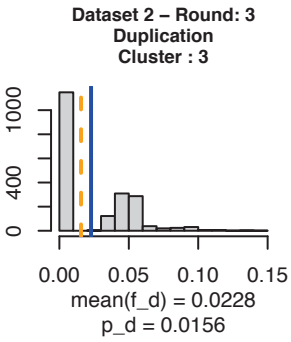

Dataset 2 – Round: 3
Co–speciation
Cluster : 2
mean(f_c) = 0.5310
p_c = 0.5536

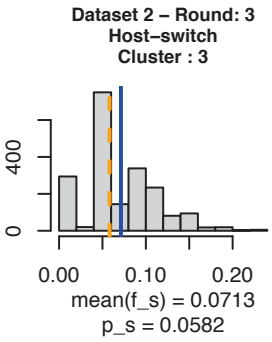

Dataset 2 – Round: 3
Duplication
Cluster : 2
mean(f_d) = 0.0003
p_d = 0.0002

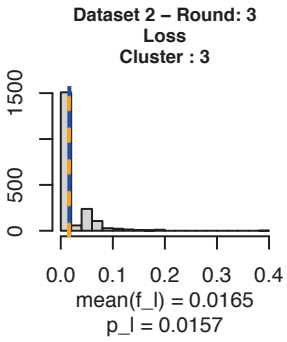

Dataset 2 – Round: 3
Host–switch
Cluster : 2
mean(f_s) = 0.2994
p_s = 0.2699

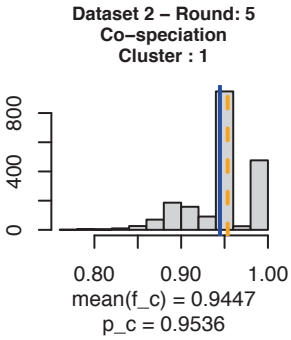

Dataset 2 – Round: 3
Loss
Cluster : 2
mean(f_l) = 0.1692
p_l = 0.1763

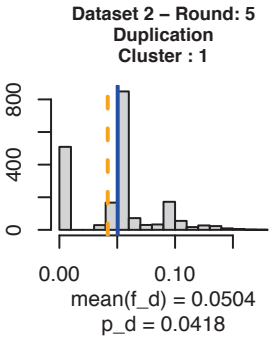

Dataset 2 – Round: 3
Co–speciation
Cluster : 3
mean(f_c) = 0.8894
p_c = 0.9105

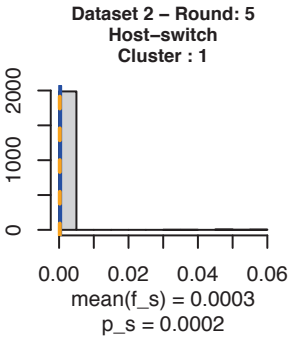

Dataset 2 – Round: 3
Duplication
Cluster : 3
mean(f_d) = 0.0228
p_d = 0.0156

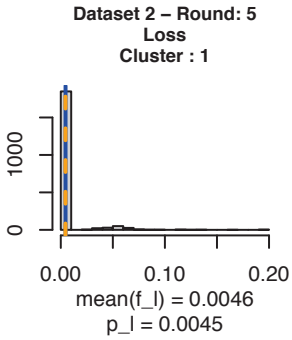

Dataset 2 – Round: 3
Host–switch
Cluster : 3
mean(f_s) = 0.0713
p_s = 0.0582

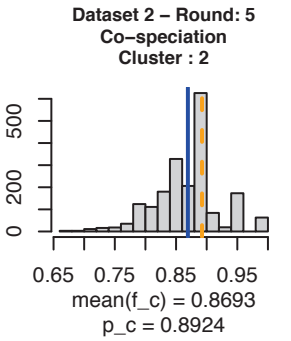

Dataset 2 – Round: 3
Loss
Cluster : 3
mean(f_l) = 0.0165
p_l = 0.0157

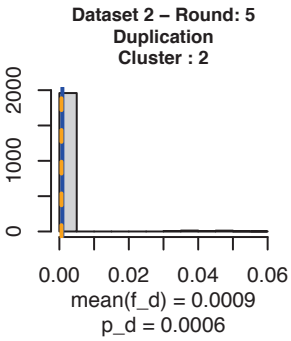

Dataset 2 – Round: 5
Co–speciation
Cluster : 1
0
400
800
0.80
0.90
1.00
mean(f_c) = 0.9447
p_c = 0.9536

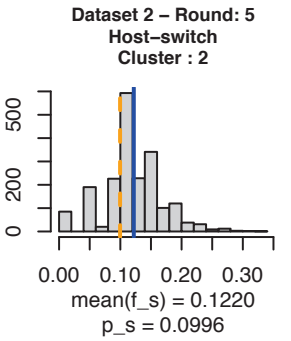

Dataset 2 – Round: 5
Duplication
Cluster : 1
0
400
800
0.00
0.10
mean(f_d) = 0.0504
p_d = 0.0418

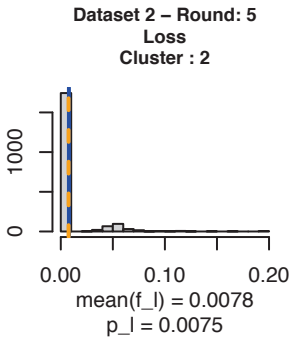

Dataset 2 – Round: 5
Host–switch
Cluster : 1
0
1000
2000
0.00
0.02
0.04
0.06
mean(f_s) = 0.0003
p_s = 0.0002

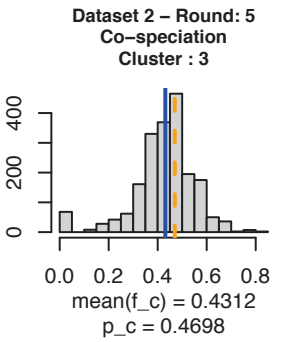

Dataset 2 – Round: 5
Loss
Cluster : 1
0
1000
0.00
0.10
0.20
mean(f_l) = 0.0046
p_l = 0.0045

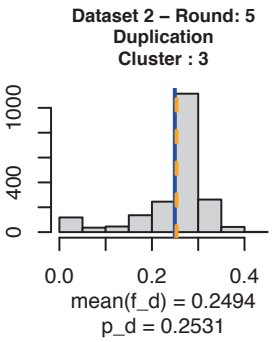

Dataset 2 – Round: 5
Co–speciation
Cluster : 2
0
200
500
0.65
0.75
0.85
0.95
mean(f_c) = 0.8693
p_c = 0.8924

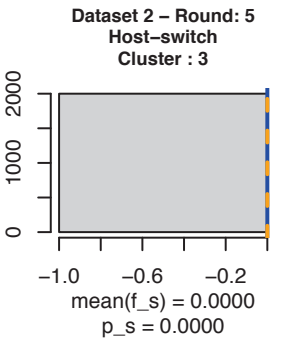

Dataset 2 – Round: 5
Duplication
Cluster : 2
0
1000
2000
0.00
0.02
0.04
0.06
mean(f_d) = 0.0009
p_d = 0.0006

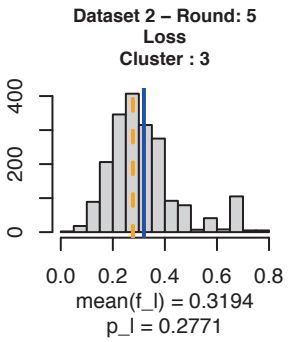

Dataset 2 – Round: 5
Host–switch
Cluster : 2
0
200
500
0.00
0.10
0.20
0.30
mean(f_s) = 0.1220
p_s = 0.0996

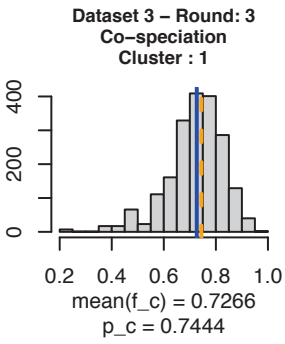

Dataset 2 – Round: 5
Loss
Cluster : 2
0
1000
0.00
0.10
0.20
mean(f_l) = 0.0078
p_l = 0.0075

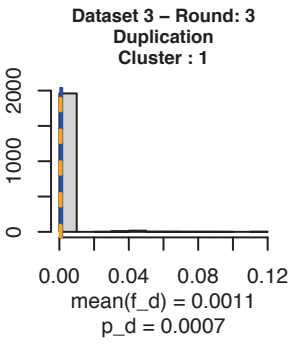

Dataset 2 – Round: 5
Co–speciation
Cluster : 3
0
200
400
0.0
0.2
0.4
0.6
0.8
mean(f_c) = 0.4312
p_c = 0.4698

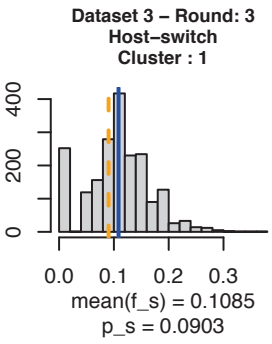

Dataset 2 – Round: 5
Duplication
Cluster : 3
0
400
1000
0.0
0.2
0.4
mean(f_d) = 0.2494
p_d = 0.2531

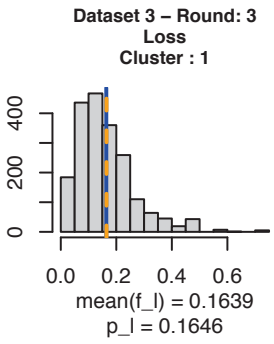

Dataset 2 – Round: 5
Host–switch
Cluster : 3
0
1000
2000
−1.0
−0.6
−0.2
mean(f_s) = 0.0000
p_s = 0.0000

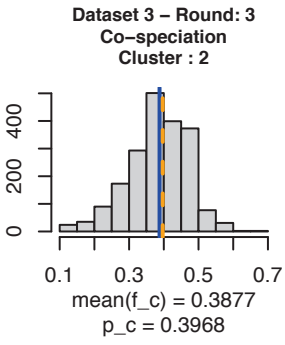

Dataset 2 – Round: 5
Loss
Cluster : 3
0
200
400
0.0
0.2
0.4
0.6
0.8
mean(f_l) = 0.3194
p_l = 0.2771

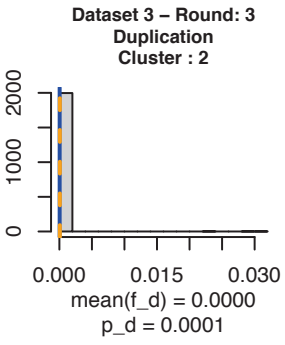

Dataset 3 – Round: 3
Co–speciation
Cluster : 1
mean(f_c) = 0.7266
p_c = 0.7444

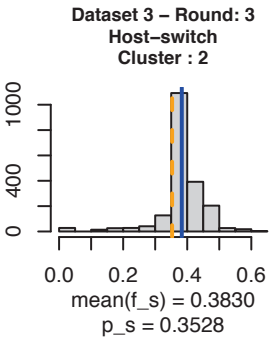

Dataset 3 – Round: 3
Duplication
Cluster : 1
mean(f_d) = 0.0011
p_d = 0.0007

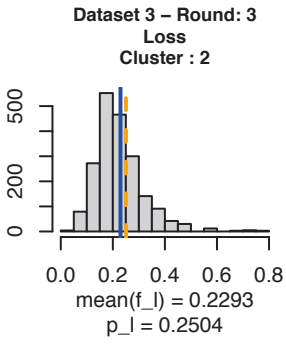

Dataset 3 – Round: 3
Host–switch
Cluster : 1
mean(f_s) = 0.1085
p_s = 0.0903

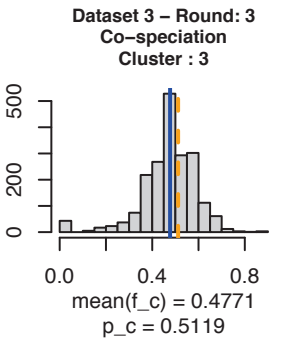

Dataset 3 – Round: 3
Loss
Cluster : 1
mean(f_l) = 0.1639
p_l = 0.1646

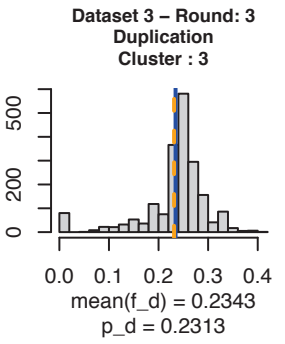

Dataset 3 – Round: 3
Co–speciation
Cluster : 2
mean(f_c) = 0.3877
p_c = 0.3968

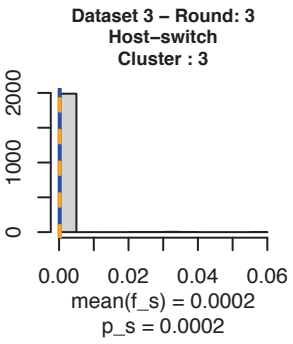

Dataset 3 – Round: 3
Duplication
Cluster : 2
mean(f_d) = 0.0000
p_d = 0.0001

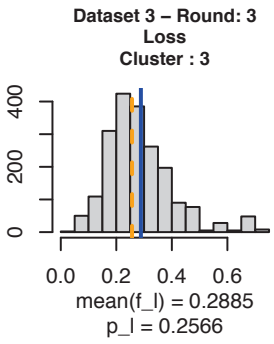

Dataset 3 – Round: 3
Host–switch
Cluster : 2
mean(f_s) = 0.3830
p_s = 0.3528

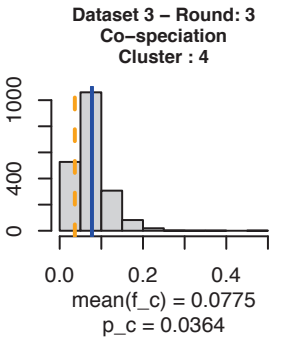

Dataset 3 – Round: 3
Loss
Cluster : 2
mean(f_l) = 0.2293
p_l = 0.2504

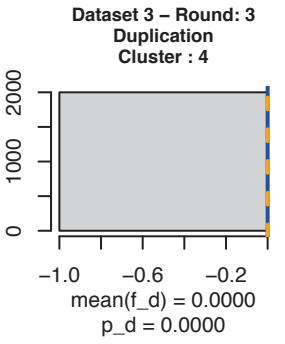

Dataset 3 – Round: 3
Co–speciation
Cluster : 3
mean(f_c) = 0.4771
p_c = 0.5119

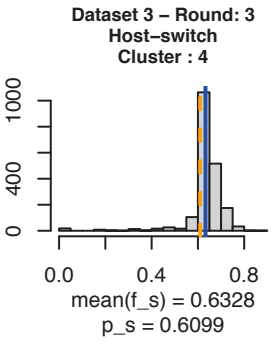

Dataset 3 – Round: 3
Duplication
Cluster : 3
mean(f_d) = 0.2343
p_d = 0.2313

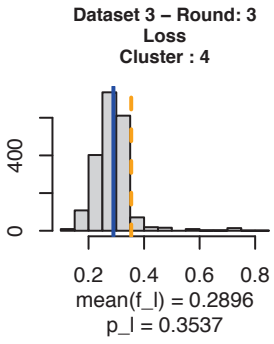

Dataset 3 – Round: 3
Host–switch
Cluster : 3
mean(f_s) = 0.0002
p_s = 0.0002

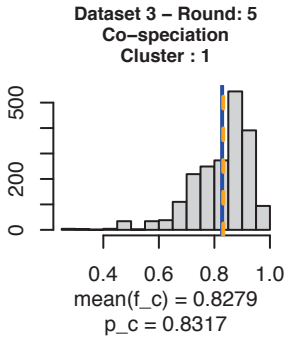

Dataset 3 – Round: 3
Loss
Cluster : 3
mean(f_l) = 0.2885
p_l = 0.2566

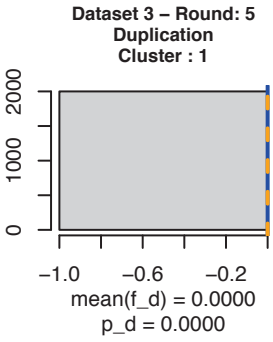

Dataset 3 – Round: 3
Co–speciation
Cluster : 4
mean(f_c) = 0.0775
p_c = 0.0364

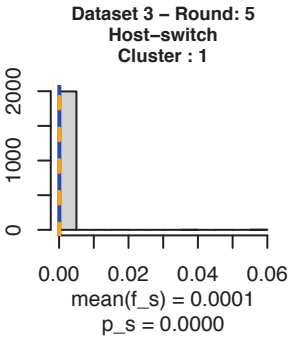

Dataset 3 – Round: 3
Duplication
Cluster : 4
mean(f_d) = 0.0000
p_d = 0.0000

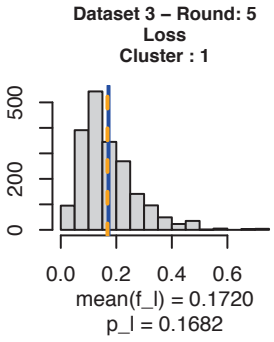

Dataset 3 – Round: 3
Host–switch
Cluster : 4
mean(f_s) = 0.6328
p_s = 0.6099

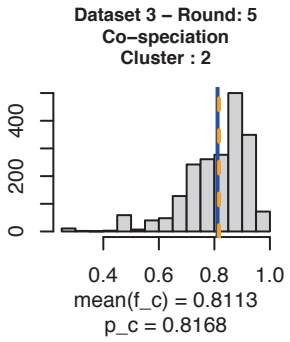

Dataset 3 – Round: 3
Loss
Cluster : 4
mean(f_l) = 0.2896
p_l = 0.3537

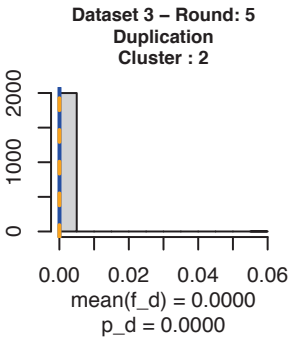

Dataset 3 – Round: 5
Co–speciation
Cluster : 1
mean(f_c) = 0.8279
p_c = 0.8317

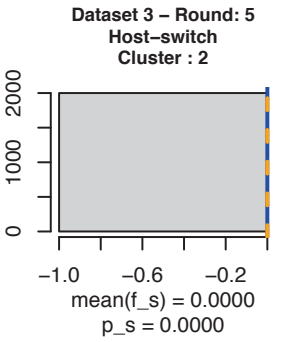

Dataset 3 – Round: 5
Duplication
Cluster : 1
mean(f_d) = 0.0000
p_d = 0.0000

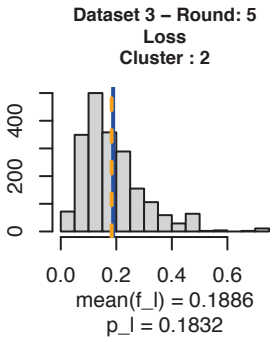

Dataset 3 – Round: 5
Host–switch
Cluster : 1
mean(f_s) = 0.0001
p_s = 0.0000

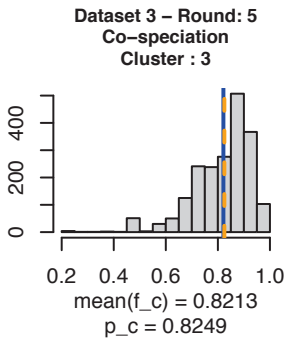

Dataset 3 – Round: 5
Loss
Cluster : 1
mean(f_l) = 0.1720
p_l = 0.1682

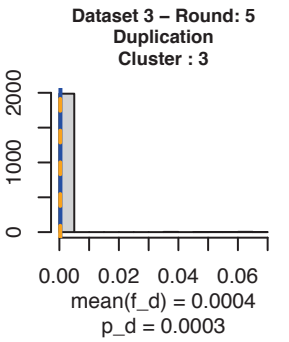

Dataset 3 – Round: 5
Co–speciation
Cluster : 2
mean(f_c) = 0.8113
p_c = 0.8168

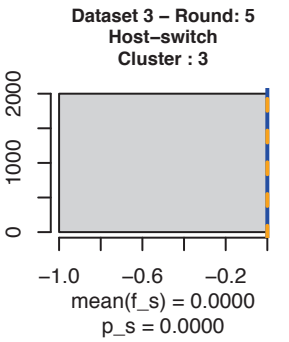

Dataset 3 – Round: 5
Duplication
Cluster : 2
mean(f_d) = 0.0000
p_d = 0.0000

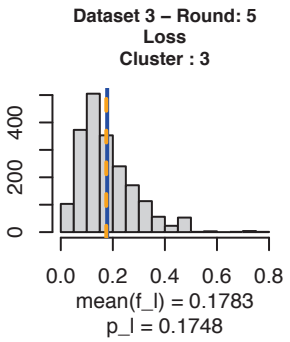

Dataset 3 – Round: 5
Host–switch
Cluster : 2
mean(f_s) = 0.0000
p_s = 0.0000

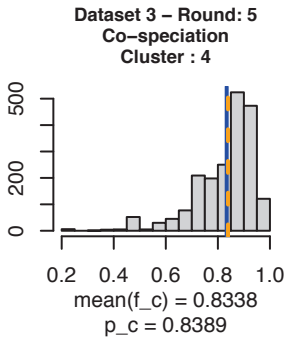

Dataset 3 – Round: 5
Loss
Cluster : 2
mean(f_l) = 0.1886
p_l = 0.1832

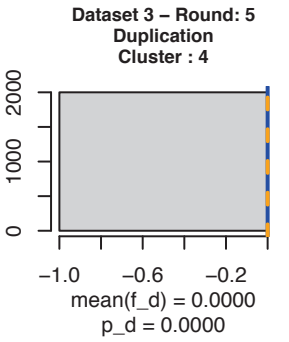

Dataset 3 – Round: 5
Co–speciation
Cluster : 3
mean(f_c) = 0.8213
p_c = 0.8249

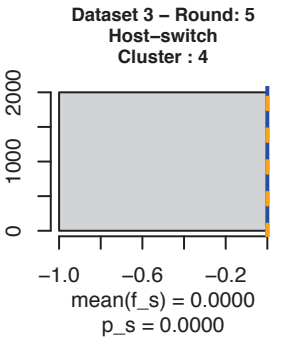

Dataset 3 – Round: 5
Duplication
Cluster : 3
mean(f_d) = 0.0004
p_d = 0.0003

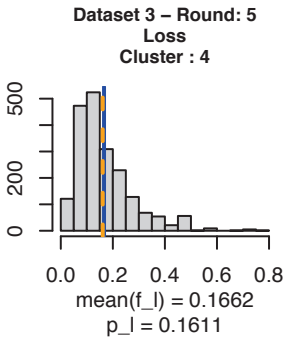

Dataset 3 – Round: 5
Host–switch
Cluster : 3
mean(f_s) = 0.0000
p_s = 0.0000

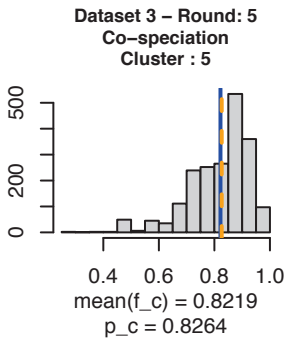

Dataset 3 – Round: 5
Loss
Cluster : 3
mean(f_l) = 0.1783
p_l = 0.1748

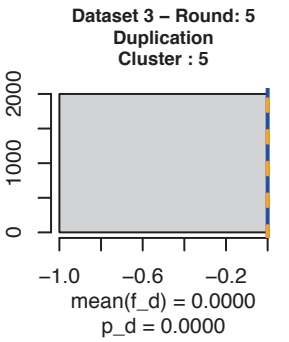

Dataset 3 – Round: 5
Co–speciation
Cluster : 4
mean(f_c) = 0.8338
p_c = 0.8389

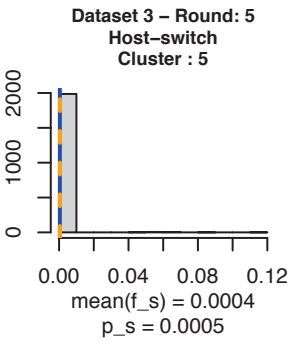

Dataset 3 – Round: 5
Duplication
Cluster : 4
mean(f_d) = 0.0000
p_d = 0.0000

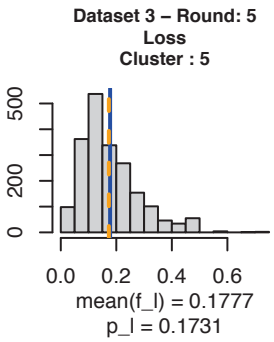

Dataset 3 – Round: 5
Host–switch
Cluster : 4
mean(f_s) = 0.0000
p_s = 0.0000

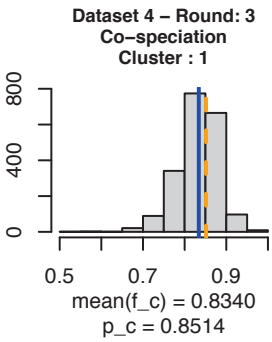

Dataset 3 – Round: 5
Loss
Cluster : 4
mean(f_l) = 0.1662
p_l = 0.1611

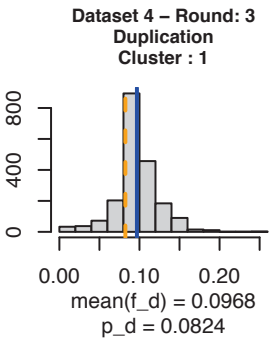

Dataset 3 – Round: 5
Co–speciation
Cluster : 5
mean(f_c) = 0.8219
p_c = 0.8264

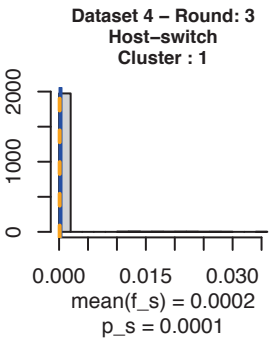

Dataset 3 – Round: 5
Duplication
Cluster : 5
mean(f_d) = 0.0000
p_d = 0.0000

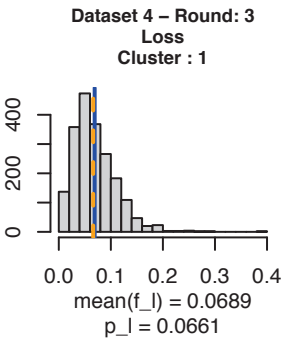

Dataset 3 – Round: 5
Host–switch
Cluster : 5
mean(f_s) = 0.0004
p_s = 0.0005

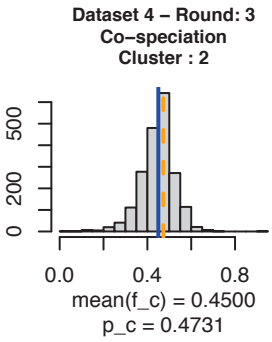

Dataset 3 – Round: 5
Loss
Cluster : 5
mean(f_l) = 0.1777
p_l = 0.1731

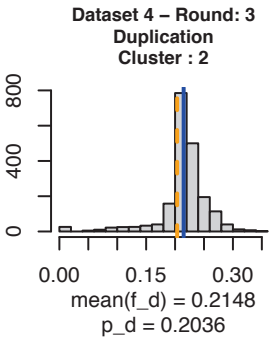

Dataset 4 – Round: 3
Co–speciation
Cluster : 1
mean(f_c) = 0.8340
p_c = 0.8514

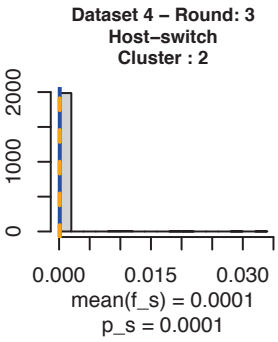

Dataset 4 – Round: 3
Duplication
Cluster : 1
mean(f_d) = 0.0968
p_d = 0.0824

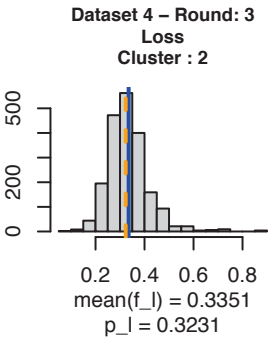

Dataset 4 – Round: 3
Host–switch
Cluster : 1
mean(f_s) = 0.0002
p_s = 0.0001

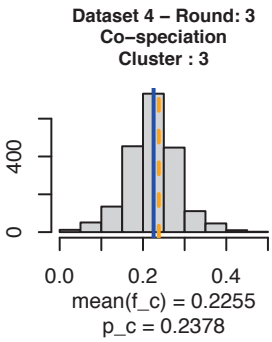

Dataset 4 – Round: 3
Loss
Cluster : 1
mean(f_l) = 0.0689
p_l = 0.0661

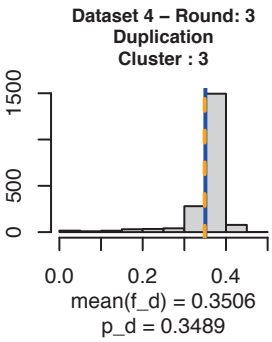

Dataset 4 – Round: 3
Co–speciation
Cluster : 2
mean(f_c) = 0.4500
p_c = 0.4731

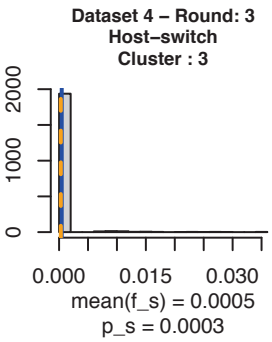

Dataset 4 – Round: 3
Duplication
Cluster : 2
mean(f_d) = 0.2148
p_d = 0.2036

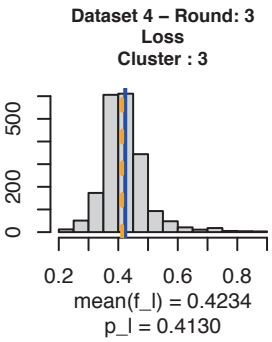

Dataset 4 – Round: 3
Host–switch
Cluster : 2
mean(f_s) = 0.0001
p_s = 0.0001

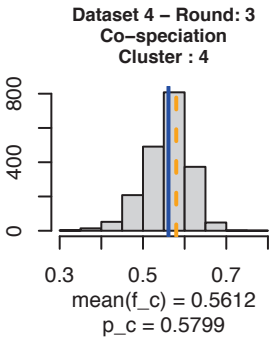

Dataset 4 – Round: 3
Loss
Cluster : 2
mean(f_l) = 0.3351
p_l = 0.3231

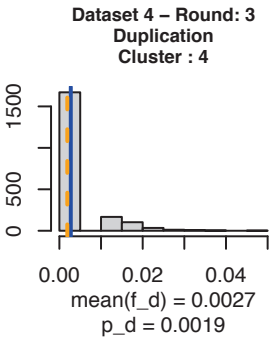

Dataset 4 – Round: 3
Co–speciation
Cluster : 3
mean(f_c) = 0.2255
p_c = 0.2378

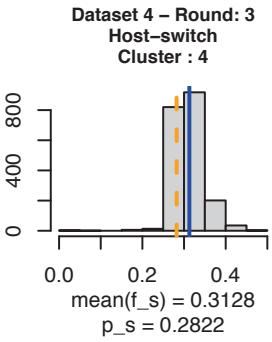

Dataset 4 – Round: 3
Duplication
Cluster : 3
mean(f_d) = 0.3506
p_d = 0.3489

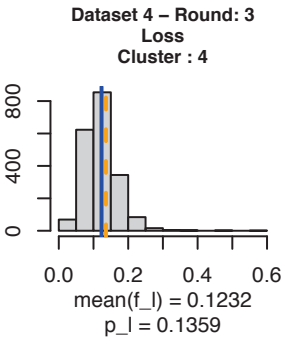

Dataset 4 – Round: 3
Host–switch
Cluster : 3
mean(f_s) = 0.0005
p_s = 0.0003

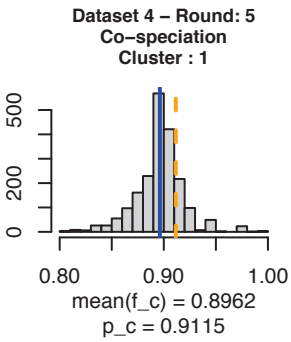

Dataset 4 – Round: 3
Loss
Cluster : 3
mean(f_l) = 0.4234
p_l = 0.4130

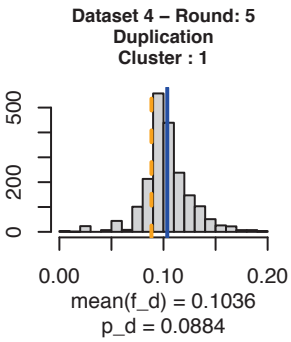

Dataset 4 – Round: 3
Co–speciation
Cluster : 4
mean(f_c) = 0.5612
p_c = 0.5799

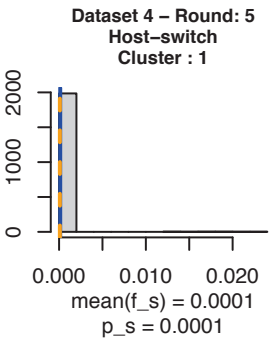

Dataset 4 – Round: 3
Duplication
Cluster : 4
mean(f_d) = 0.0027
p_d = 0.0019

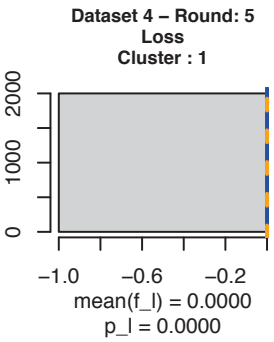

Dataset 4 – Round: 3
Host–switch
Cluster : 4
mean(f_s) = 0.3128
p_s = 0.2822

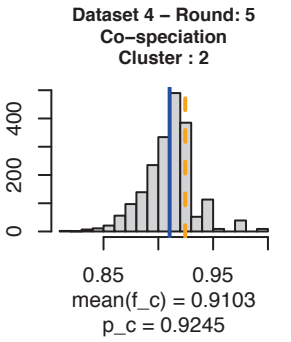

Dataset 4 – Round: 3
Loss
Cluster : 4
mean(f_l) = 0.1232
p_l = 0.1359

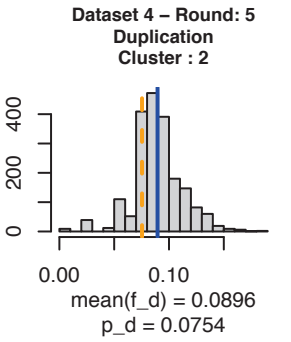

Dataset 4 – Round: 5
Co–speciation
Cluster : 1
0
200
500
0.80
0.90
1.00
mean(f_c) = 0.8962
p_c = 0.9115

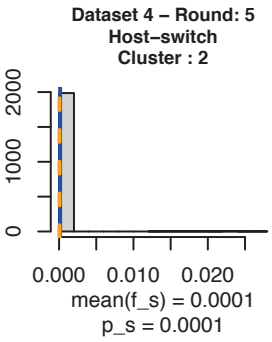

Dataset 4 – Round: 5
Duplication
Cluster : 1
0
200
500
0.00
0.10
0.20
mean(f_d) = 0.1036
p_d = 0.0884

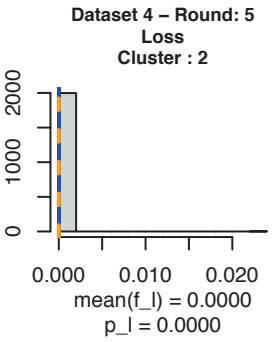

Dataset 4 – Round: 5
Host–switch
Cluster : 1
0
1000
2000
0.000
0.010
0.020
mean(f_s) = 0.0001
p_s = 0.0001

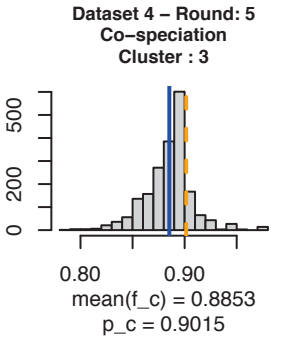

Dataset 4 – Round: 5
Loss
Cluster : 1
0
1000
2000
−1.0
−0.6
−0.2
mean(f_l) = 0.0000
p_l = 0.0000

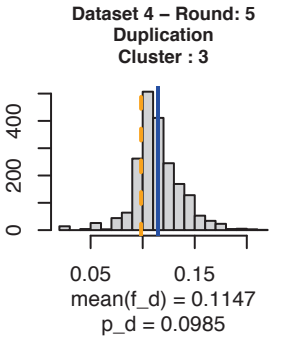

Dataset 4 – Round: 5
Co–speciation
Cluster : 2
0
200
400
0.85
0.95
mean(f_c) = 0.9103
p_c = 0.9245

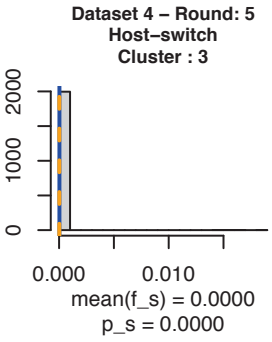

Dataset 4 – Round: 5
Duplication
Cluster : 2
0
200
400
0.00
0.10
mean(f_d) = 0.0896
p_d = 0.0754

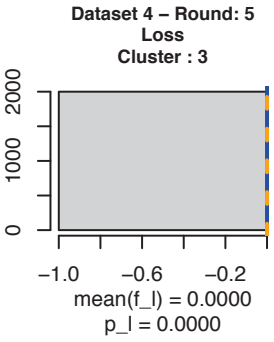

Dataset 4 – Round: 5
Host–switch
Cluster : 2
0
1000
2000
0.000
0.010
0.020
mean(f_s) = 0.0001
p_s = 0.0001

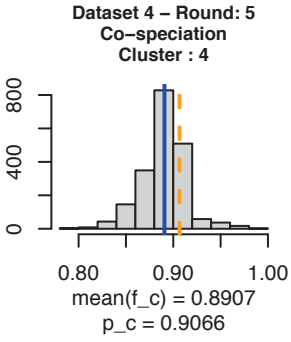

Dataset 4 – Round: 5
Loss
Cluster : 2
0
1000
2000
0.000
0.010
0.020
mean(f_l) = 0.0000
p_l = 0.0000

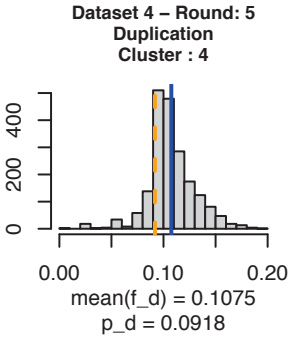

Dataset 4 – Round: 5
Co–speciation
Cluster : 3
0
200
500
0.80
0.90
mean(f_c) = 0.8853
p_c = 0.9015

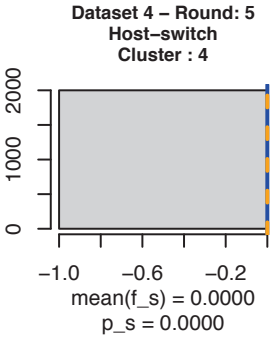

Dataset 4 – Round: 5
Duplication
Cluster : 3
0
200
400
0.05
0.15
mean(f_d) = 0.1147
p_d = 0.0985

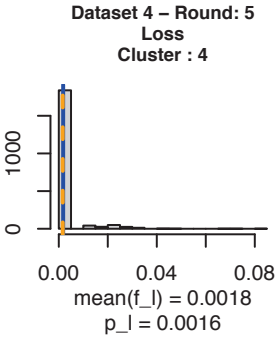

Dataset 4 – Round: 5
Host–switch
Cluster : 3
0
1000
2000
0.000
0.010
mean(f_s) = 0.0000
p_s = 0.0000

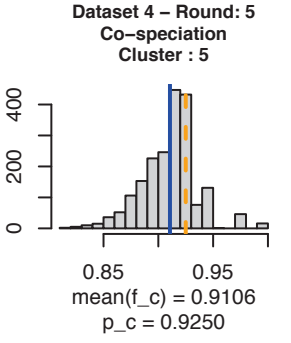

Dataset 4 – Round: 5
Loss
Cluster : 3
0
1000
2000
−1.0
−0.6
−0.2
mean(f_l) = 0.0000
p_l = 0.0000

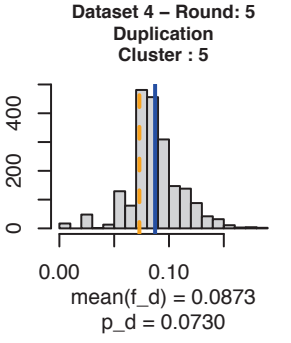

Dataset 4 – Round: 5
Co–speciation
Cluster : 4
0
400
800
0.80
0.90
1.00
mean(f_c) = 0.8907
p_c = 0.9066

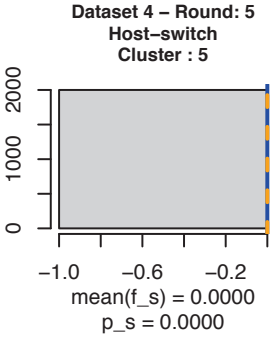

Dataset 4 – Round: 5
Duplication
Cluster : 4
0
200
400
0.00
0.10
0.20
mean(f_d) = 0.1075
p_d = 0.0918

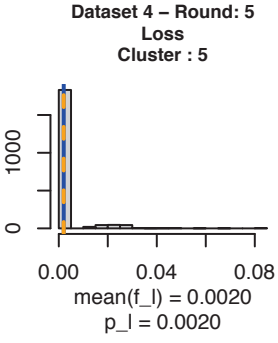

Dataset 4 – Round: 5
Host–switch
Cluster : 4
0
1000
2000
−1.0
−0.6
−0.2
mean(f_s) = 0.0000
p_s = 0.0000


Dataset 4 – Round: 5
Loss
Cluster : 4
0
1000
0.00
0.04
0.08
mean(f_l) = 0.0018
p_l = 0.0016


Dataset 4 – Round: 5
Co–speciation
Cluster : 5
0
200
400
0.85
0.95
mean(f_c) = 0.9106
p_c = 0.9250

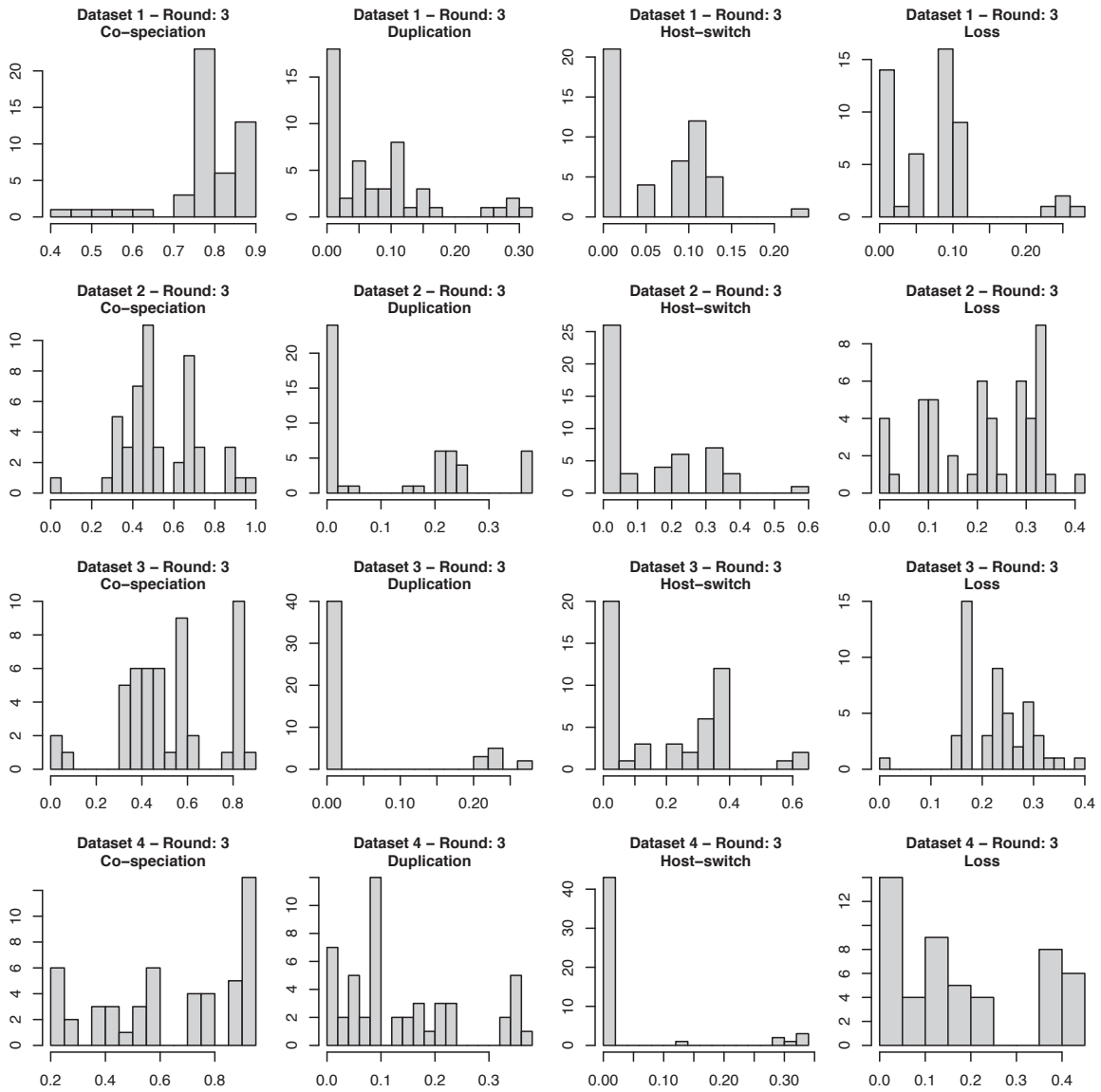

Dataset 4 – Round: 5
Duplication
Cluster : 5
0
200
400
0.00
0.10
mean(f_d) = 0.0873
p_d = 0.0730

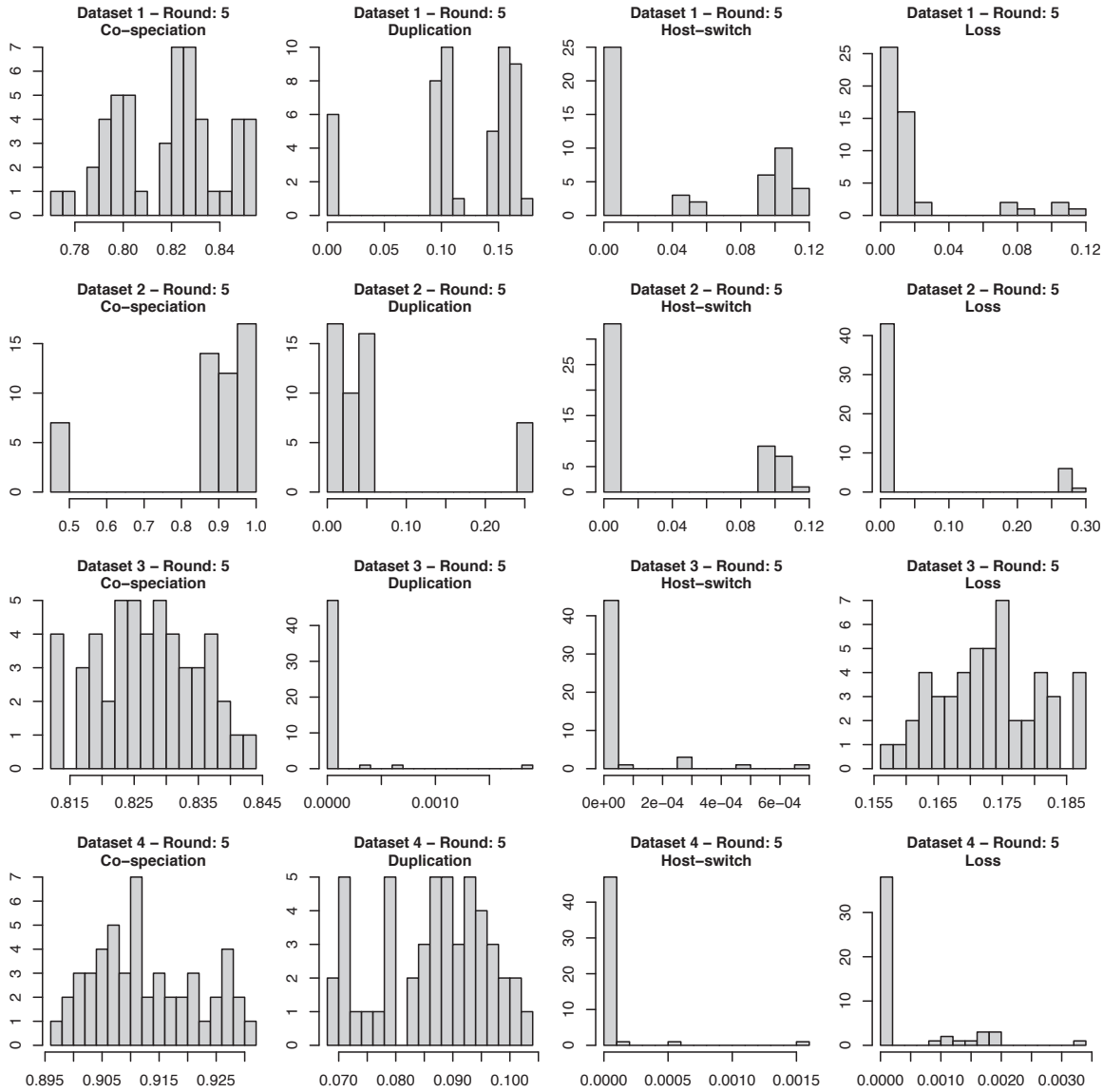

Dataset 4 – Round: 5
Host–switch
Cluster : 5
0
1000
2000
−1.0
−0.6
−0.2
mean(f_s) = 0.0000
p_s = 0.0000


Dataset 4 – Round: 5
Loss
Cluster : 5
0
1000
0.00
0.04
0.08
mean(f_l) = 0.0020
p_l = 0.0020

Distribution of probability values of accepted vectors

For better visualisation, the following plots show the distribution of the event probability values observed for the 50 accepted vectors at the end of rounds 3 and 5 for each one of the four biological datasets. On the plots of the third round, we can observe more variability in the list of accepted vectors. In the fifth round, COALA produces a list of accepted vectors which show a stronger characteristic of uni-modality and indicate that, with a higher number of rounds, the model converges in the direction of a local maximum and eventually may overfit.


Dataset 1 – Round: 3
Co–speciation
Dataset 1 – Round: 3
Duplication
Dataset 1 – Round: 3
Host–switch
Dataset 1 – Round: 3
Loss
Dataset 2 – Round: 3
Co–speciation
Dataset 2 – Round: 3
Duplication
Dataset 2 – Round: 3
Host–switch
Dataset 2 – Round: 3
Loss
Dataset 3 – Round: 3
Co–speciation
Dataset 3 – Round: 3
Duplication
Dataset 3 – Round: 3
Host–switch
Dataset 3 – Round: 3
Loss
Dataset 4 – Round: 3
Co–speciation
Dataset 4 – Round: 3
Duplication
Dataset 4 – Round: 3
Host–switch
Dataset 4 – Round: 3
Loss


Dataset 1 – Round: 5
Co–speciation
Dataset 1 – Round: 5
Duplication
Dataset 1 – Round: 5
Host–switch
Dataset 1 – Round: 5
Loss
Dataset 2 – Round: 5
Co–speciation
Dataset 2 – Round: 5
Duplication
Dataset 2 – Round: 5
Host–switch
Dataset 2 – Round: 5
Loss
Dataset 3 – Round: 5
Co–speciation
Dataset 3 – Round: 5
Duplication
Dataset 3 – Round: 5
Host–switch
Dataset 3 – Round: 5
Loss
Dataset 4 – Round: 5
Co–speciation
Dataset 4 – Round: 5
Duplication
Dataset 4 – Round: 5
Host–switch
Dataset 4 – Round: 5
Loss

Transforming probability vectors into cost vectors

In order to compare the results obtained for the biological datasets in relation to the existing literature, we transformed each one of the representative parameter vectors $\langle p_c, p_d, p_s, p_l \rangle$ (Table 3) into a vector of costs that was then used to compute optimal reconciliations between the host and parasite trees given as input.

The transformation was done by defining $c_i = -\ln p_i$, with $i \in \{c, d, s, l\}$, which is based on a commonly accepted idea that the cost of an event is inversely related to its probability. Indeed, if p_i is equal to 1, then we expect all the events to be of type i, thus the cost of the corresponding event must be 0. Similarly, if p_i is equal to 0, we expect that event i never happens, and thus the cost must be assigned to $+\infty$.

After transforming probability vectors into cost vectors, we ran EUCALYPT to compute the cost and to enumerate all optimal reconciliations. EUCALYPT also outputs the number of events observed in the set of optimal reconciliations and offers a feature to filter off the ones that are cyclic (time-unfeasible solutions). For each dataset and each cluster, Table 4 shows the obtained cost vector (c_c, c_d, c_s and, c_l), the cost of the optimal solution (Opt), the number of events observed in the optimal reconciliations (#c, #d, #s and, #l) and the number of acyclic (#A) and cyclic (#C) solutions.

Round	Dataset	Cluster	c_c	c_d	c_s	c_l	Opt	$\#c$	$\#d$	$\#s$	$\#l$	$\#A$	$\#C$
3	1	1	0.231	6.502	2.293	2.267	16.462	12	0	4	2	3	0
		2	0.235	2.140	2.467	4.934	22.085	10	0	6	1	5	0
		3	0.146	2.792	9.721	2.601	42.366	12	2	2	6	2	0
		4	0.650	1.272	7.601	1.625	35.280	12	2	2	6	2	0
	2	0	3.517	13.816	0.584	0.885	14.044	1	0	15	2	2944	0
		1	0.775	1.355	7.824	1.270	48.664	11	2	3	11	2	0
		2	0.591	8.517	1.310	1.736	16.217	9	0	7	1	1	0
		3	0.094	4.160	2.844	4.154	24.892	9	0	7	1	1	0
	3	1	0.295	7.264	2.405	1.804	42.056	4	0	11	8	2	34
		2	0.924	9.567	1.042	1.385	24.952	3	0	12	7	128	0
		3	0.670	1.464	8.517	1.360	76.235	7	8	0	44	1	0
		4	3.313	13.816	0.494	1.039	17.502	1	0	14	7	1024	0
	4	1	0.161	2.496	9.210	2.717	153.544	22	11	8	18	0	12
		2	0.748	1.592	9.210	1.130	105.393	22	19	0	52	1	0
		3	1.436	1.053	8.112	0.884	97.548	22	19	0	52	1	0
		4	0.545	6.266	1.265	1.996	72.588	17	5	19	4	4	0
5	1	1	0.187	1.851	6.908	4.374	38.620	12	0	4	2	3	0
		2	0.209	2.292	2.440	9.210	25.924	10	0	6	1	5	0
		3	0.229	13.816	2.191	2.376	16.264	12	0	4	2	3	0
	2	1	0.048	3.175	8.517	5.404	65.445	9	0	7	1	1	0
		2	0.114	7.419	2.307	4.893	22.051	9	0	7	1	1	0
		3	0.755	1.374	13.816	1.283	56.179	9	7	0	31	1	0
	3	1	0.184	13.816	10.127	1.783	112.590	7	0	8	17	0	1
		2	0.202	11.736	13.816	1.697	140.783	7	0	8	17	0	1
		3	0.192	8.112	13.816	1.744	139.152	7	5	3	32	3	0
		4	0.176	13.816	11.513	1.826	124.346	7	0	8	17	0	1
		5	0.191	10.414	7.601	1.754	91.931	7	0	8	17	0	1
	4	1	0.093	2.426	9.316	13.816	239.044	17	6	18	4	2	0
		2	0.078	2.585	9.903	13.816	250.344	17	6	18	4	2	0
		3	0.104	2.318	10.820	13.816	265.661	17	6	18	4	2	0
		4	0.098	2.388	13.816	6.438	251.916	22	9	10	14	0	48
		5	0.078	2.617	13.816	6.215	250.393	22	9	10	14	0	48

Table 4: The probability vectors shown in Table 3 were transformed into costs and used as input in EUCALYPT for the enumeration of optimal reconciliations. For each dataset and each cluster, we have the obtained cost vector (c_c, c_d, c_s and, c_l), the cost of the optimal solution (Opt), the number of events observed in the optimal reconciliations ($\#c$, $\#d$, $\#s$ and, $\#l$) and the number of acyclic ($\#A$) and cyclic ($\#C$) solutions.

Results obtained by Coala on the dataset of Anther smut fungi and their caryophyllaceous hosts

Dataset 3 (Anther smut fungi and their caryophyllaceous hosts) is composed by a host tree which contains 18 leaves and a parasite tree which contains 16 leaves. The parasite tree has 4 taxa which are associated to two distinct host species. The current evolutionary model used by Coala cannot handle such cases. We thus opted to remove at random one association for each of these 4 taxa.

To evaluate the effect of this decision on the results, we ran Coala for each one of the sixteen possible combinations of associations (each identified by a letter from $\mathcal{A}$ to $\mathcal{P}$) and looked at the results which were obtained on the third and fifth rounds.

For this experiment, we ran Coala with the following parameters:

- -N 2000: Number of parameter vectors which are simulated for the first round;
- -M 1000: Number of trees which are considered in the computation of the summary statistics for a given parameter vector;
- -R 5: Number of rounds;
- -t 0.1,0.25,0.25,0.25,0.25: Vector of tolerances for each round.

After running Coala , we considered the list of accepted parameter vectors at the end of the third and fifth rounds of all 16 pairs of trees ($16 \times 50 = 800$ parameter vectors per round). We then executed the clustering procedure to group the list of accepted parameter vectors of each round with the objective of understanding how related are the distinct pairs of trees. Table 5 shows the list of obtained clusters for each round. For each cluster, the table indicates the total number of accepted vectors that it contains. The columns $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{P}$ indicate the amount of accepted vectors that are in the cluster and belong to the list of accepted vectors of each pair of trees.

The results of the third round show a variability higher than the one in the results of the fifth round. For the third round, there are only two pair of trees ($\mathcal{H}$ and $\mathcal{J}$) for which a majority of the accepted parameter vectors are concentrated in a single cluster, indicating that each pair of trees may have multiple explanations. Moreover, by just looking at the distribution of the

accepted vectors into the different clusters, it is difficult to define which pairs of trees are explained by similar probability vectors.

As observed in the other experiments, when we increase the number of rounds, Coala produces a low number of clusters suggesting that the selected parameter vectors are very close to a local maximum. This effect can also be observed in the clusters that we obtained for the fifth round. Differently from the third round, most of the pairs of trees have a majority of their accepted vectors concentrated in a single cluster (the only exception is the pair $\mathcal{I}$). Moreover, by looking at the results of the fifth round, we can easily group the pairs of trees according to the cluster which concentrates most of their accepted parameter vectors:

1. $\mathcal{A}$, $\mathcal{J}$, $\mathcal{L}$, $\mathcal{M}$
2. $\mathcal{D}$, $\mathcal{G}$, $\mathcal{K}$, $\mathcal{N}$
3. $\mathcal{F}$, $\mathcal{H}$, $\mathcal{O}$, $\mathcal{P}$
4. $\mathcal{B}$, $\mathcal{E}$
5. $\mathcal{C}$
6. $\mathcal{I}$

Round	*Cluster*	p_c	p_d	p_s	p_l	#	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$	$\mathcal{P}$
3	1	0.3782	0.0003	0.3823	0.2392	257	21	23	16	15	15	22	17	7	6	27	15	19	18	19	8	9
	2	0.6033	0.1831	0.0005	0.2131	139	3	1	7	6	14	2	10	27	19		10	7	5	9	5	14
	3	0.3892	0.2874	0.0009	0.3225	136	14	12	4	14		9	6	1	12		18	14	5	8	11	8
	4	0.6890	0.0009	0.1129	0.1972	128	7	4	6	5	4	14	10	9	5	11		6	18	11	16	2
	5	0.0959	0.0041	0.5659	0.3341	96		3	17	10	15		7	1	7	4	7	4	3	3	3	12
	6	0.9100	0.0241	0.0100	0.0559	24	5					3		5	1				1		4	5
	7	0.0026	0.4280	0.0051	0.5643	20		7			2					8					3	
5	1	0.8007	0.0010	0.0027	0.1956	221	42	3	2			17			21	33		50	50			3
	2	0.4373	0.2967	0.0000	0.2660	206	8			46		1	39		17		45			50		
	3	0.9318	0.0183	0.0009	0.0489	162						26		50	12						27	47
	4	0.0012	0.4078	0.0013	0.5897	120		47			33					17					23	
	5	0.0468	0.0000	0.5964	0.3567	80			48	4	17		6				5					
	6	0.3040	0.0000	0.4726	0.2234	11						6	5									

Table 5: Clustering of the accepted vectors obtained for all 16 possible pairs of trees of the dataset "Anther smut fungi and their caryophyllaceous hosts" (after removing double associations)

Another interesting result can be observed when we analyse the evolution of the clusters through the rounds. Cluster 1 of the third round ($C1 - R3$) concentrates 257 of the 800 parameter vectors and has representatives of all pairs of trees indicating that the associated probability vector $\langle 0.3782, 0.0003, 0.3823, 0.2392 \rangle$ is able to explain most of these pairs of trees. However, after executing two more rounds, cluster $C1 - R3$ evolves in the direction of cluster 6 of the fifth round which is represented by the probability vector $\langle 0.3040, 0.0000, 0.4726, 0.2234 \rangle$ and has only 11 accepted parameter vectors of only two pairs of trees ($\mathcal{F}$ and $\mathcal{G}$). This result indicates that Coala tends to select vectors which are more specific to each pair of trees when we increase the number of rounds.

This is crucial to determine the number of rounds that should be executed by Coala . If we desire more variability, reflecting in a higher number of alternatives for evolutionary scenarios, Coala should be set to execute around 3 rounds. If more specificity is needed, a higher number of rounds can be adopted but always having in mind that this may restrict too much the possible explanations for a given pair of trees.

Results obtained by Coala on the *Wolbachia*-arthropods dataset

In the following pages, we present some of the results produced by Coala when processing the *Wolbachia*-arthropods dataset. This dataset is composed by a pair of host and parasite trees containing each 387 taxa.

Coala was executed with the same parameters which we adopted for processing the other four datasets:

- -N 2000: Number of parameter vectors which are simulated for the first round;
- -M 1000: Number of trees which are considered in the computation of the summary statistics for a given parameter vector;
- -R 3 or 5: Number of rounds;
- -t 0.1,0.25,0.25 (3 rounds) or -t 0.1,0.25,0.25,0.25,0.25 (3 rounds): Vector of tolerances for each round;
- -plot: Create plots for each round;
- -cluster: Cluster the list of accepted vectors of the last round.

We configured COALA to cluster the list of accepted parameter vectors at the end of rounds 3 and 5. Table 6 shows the list of representative parameter vectors observed in the clusters of each round.

Round	Cluster	p_c	p_d	p_s	p_l	# vectors
3	1	0.866	0.006	0.055	0.073	26
	2	0.771	0.078	0.010	0.141	22
	3	0.964	0.022	0.014	0.000	2
5	1	0.883	0.000	0.057	0.060	48
	2	0.977	0.015	0.000	0.008	2

Table 6: Representative probability vectors produced by COALA while processing the *Wolbachia*-arthropods dataset

Here, we list the histograms of distances and event probabilities obtained at the end of each one of the 5 rounds.

In each page, we have the results of a round R for the *Wolbachia*-arthropods dataset:

First row Event probability histograms (co-speciation, duplication, host-switch and loss) of all simulated parameter vectors at round R.

Second row Event probability histograms (co-speciation, duplication, host-switch and loss) of all accepted parameter vectors at round R.

Third row Representative distance histograms for all simulated parameter vectors (first columns) and accepted parameter vectors (second column).

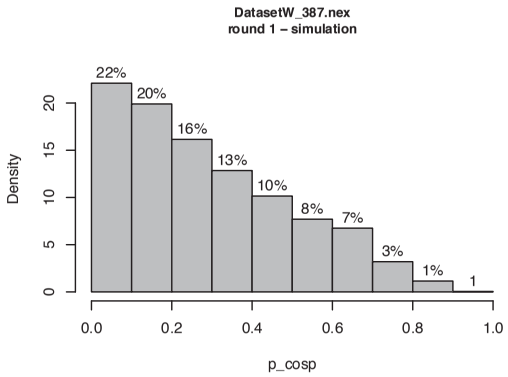
Round	Cluster	c_c	c_d	c_s	c_l	*Opt*	*Solutions*	*Acyclic solutions*
3	1	0.144	5.116	2.899	2.623	917.475	5.4×10^{43}	No
	2	0.260	2.551	4.595	1.961	1407.877	9.8×10^{40}	No
	3	0.037	3.817	4.269	13.816	1375.725	1.6×10^{51}	Yes
5	1	0.124	13.816	2.872	2.810	910.674	3.1×10^{50}	Yes
	2	0.023	4.227	13.816	4.816	4113.489	9.8×10^{40}	No

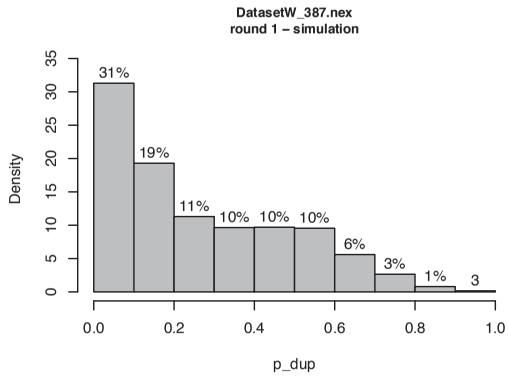
Table 7: Total number of solutions obtained by transforming the probability vectors (Table 6) into cost vectors for the *Wolbachia*-arthropods dataset

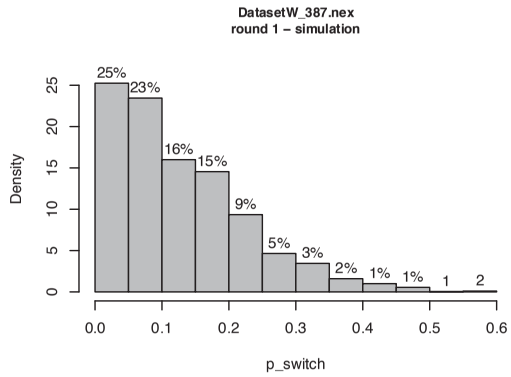
Repeating the analysis that was done with the other biological datasets, we transformed each one of the representative parameter vectors $\langle p_c, p_d, p_s, p_l \rangle$

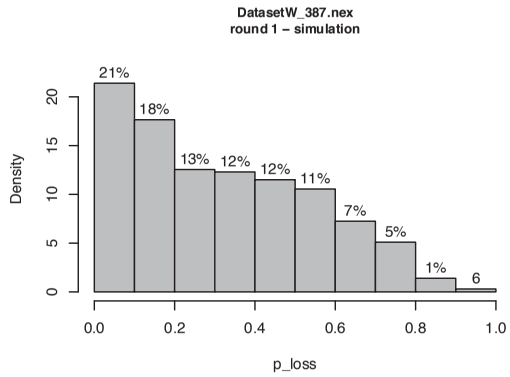
(Table 6) into a vector of costs to compute all the optimal reconciliations between the host and parasite trees given as input.

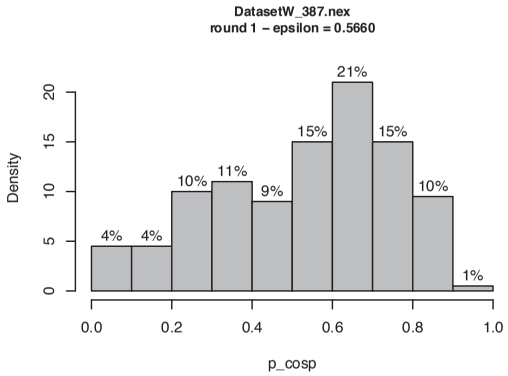
First we ran EUCALYPT to compute the cost and to count the total number of optimal reconciliations. Since the sets of optimal reconciliations for this dataset are huge, we cannot enumerate all of them. For each cost vector, we therefore sampled 10000 solutions and we ran EUCALYPT to filter out the cyclic ones. Table 7 shows the cost vectors, the cost of an optimal solution, the total number of reconciliations and whether we were able to find at least one acyclic solution.

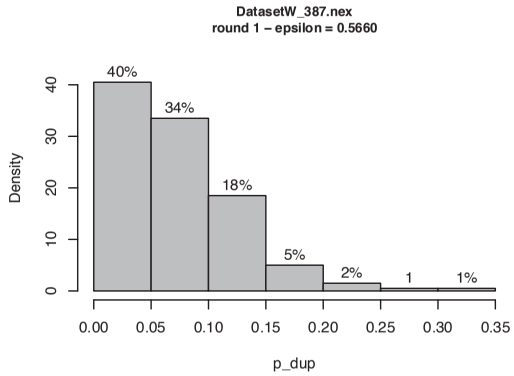


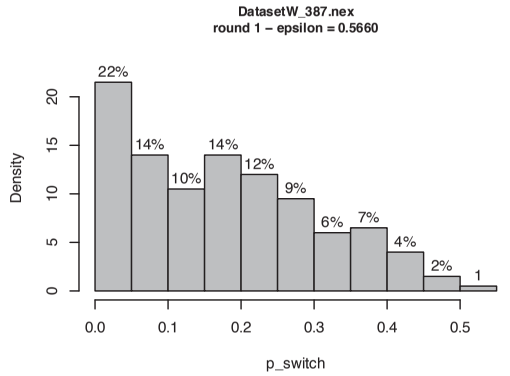


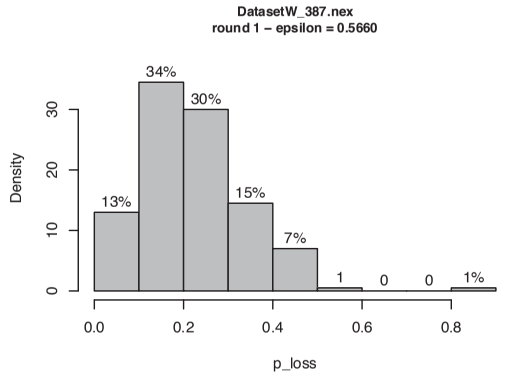


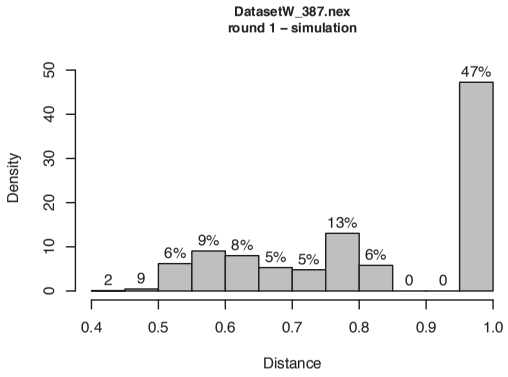


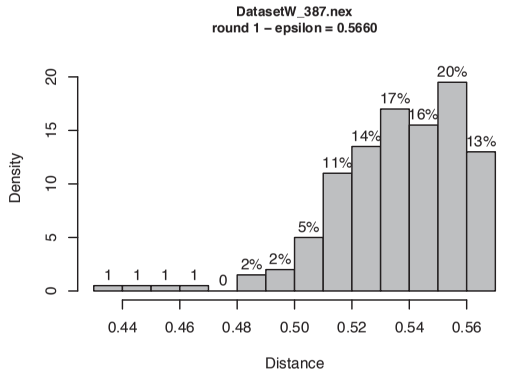




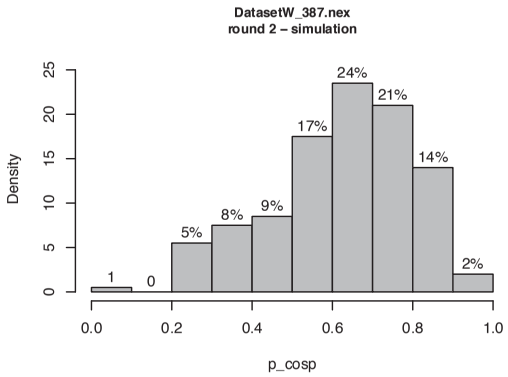


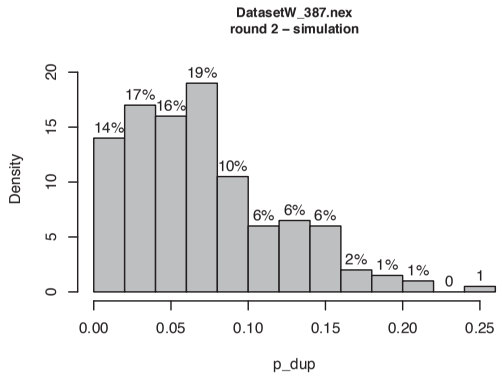


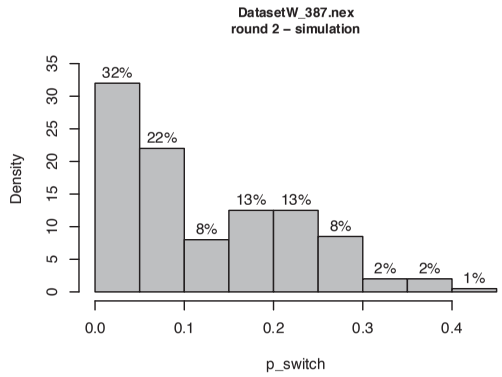


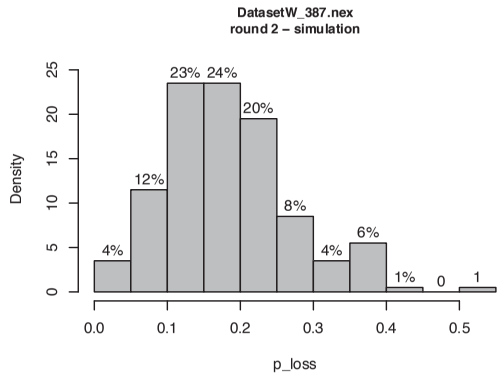


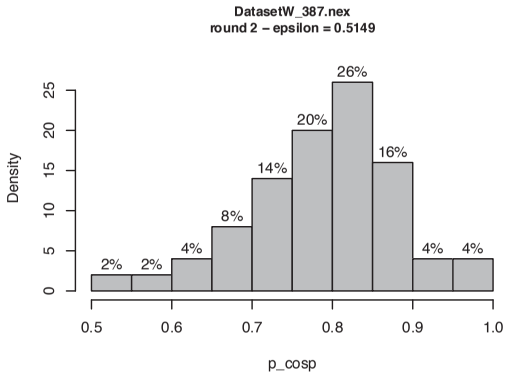
Nexus file: DatasetW_387.nex
Host/Parasite tree: 387/387 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

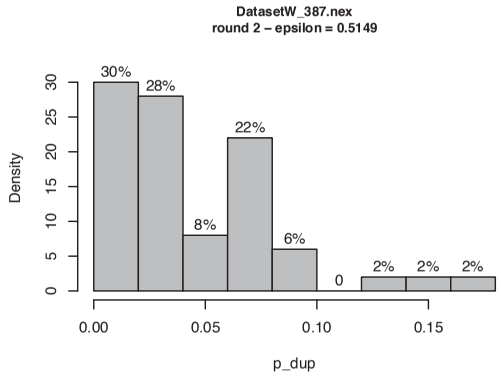


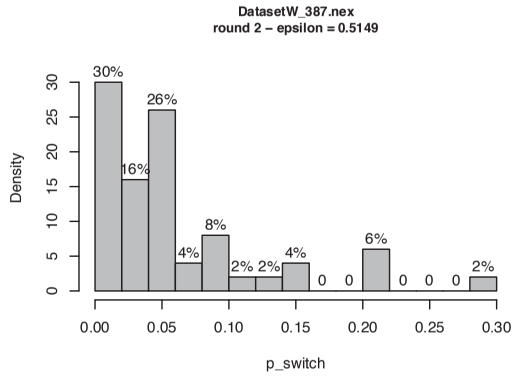


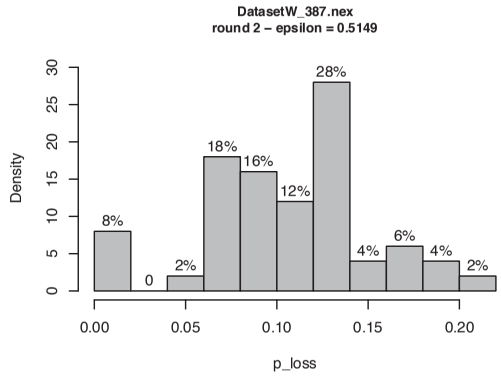


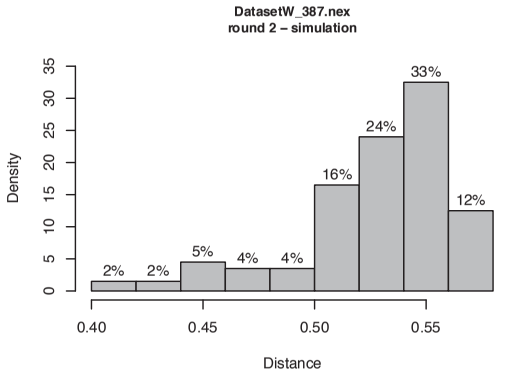


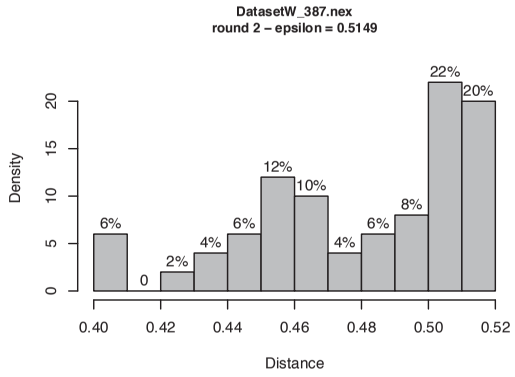




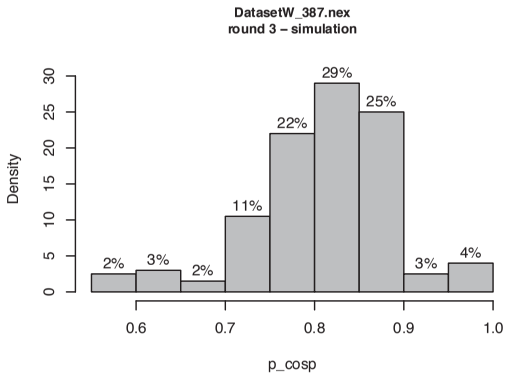


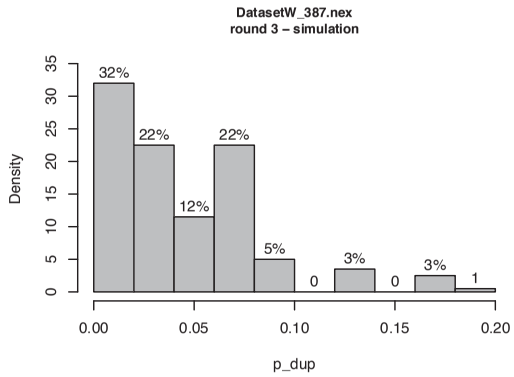


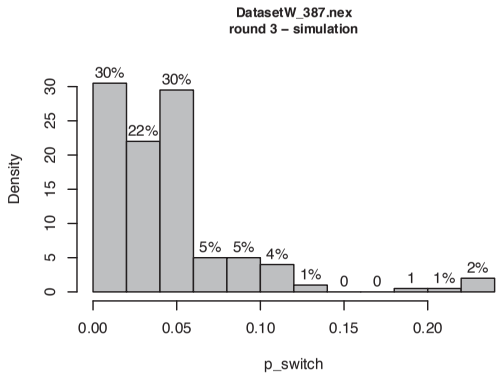


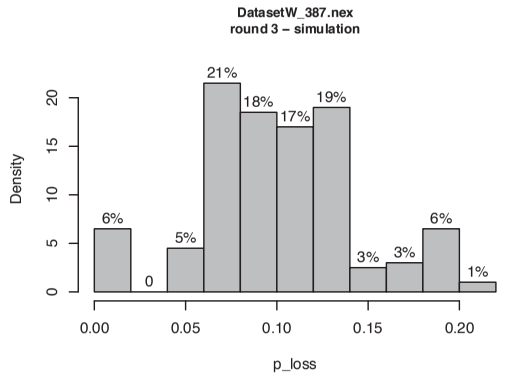


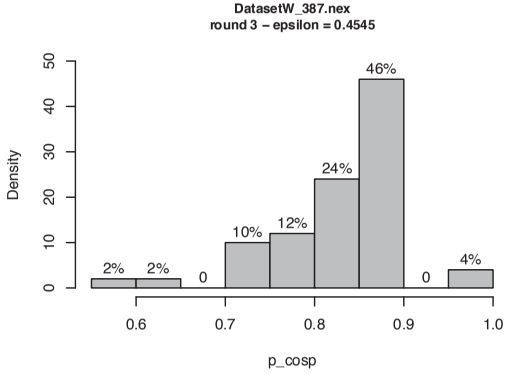
Nexus file: DatasetW_387.nex
Host/Parasite tree: 387/387 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

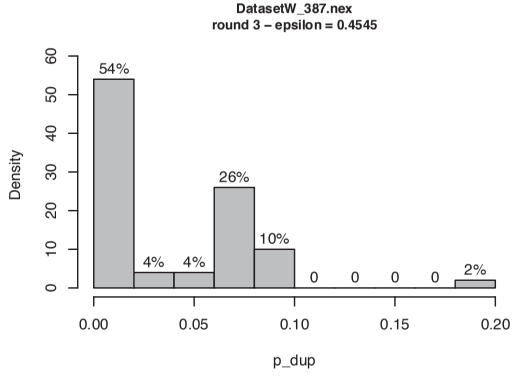


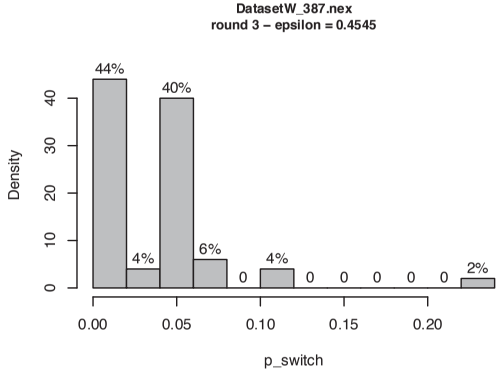


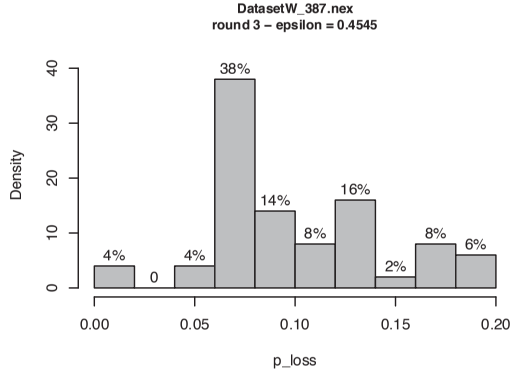


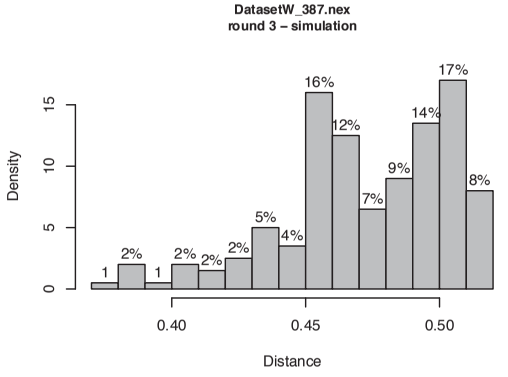


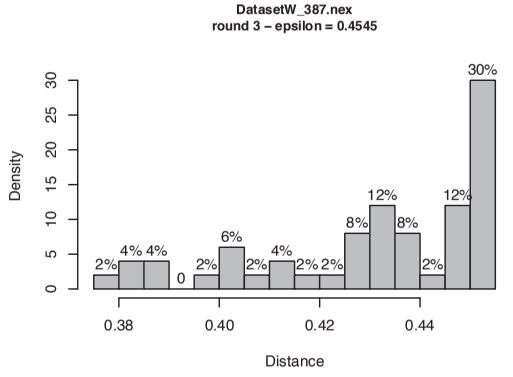




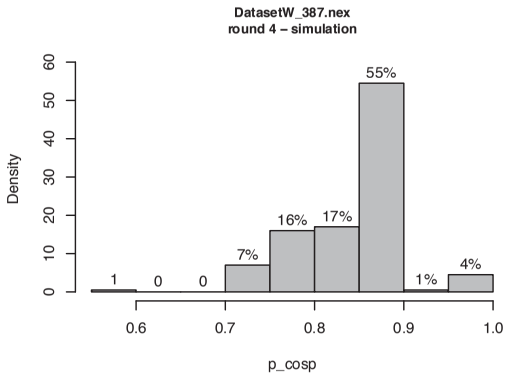


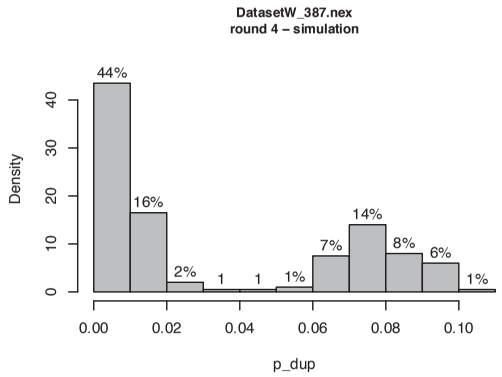


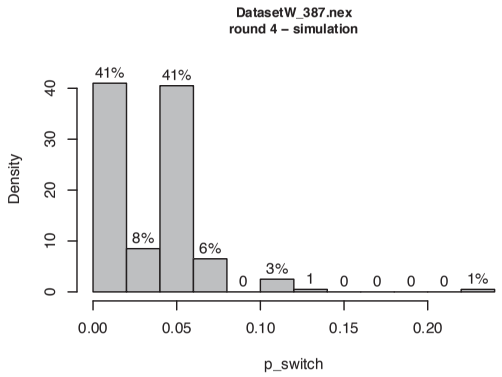


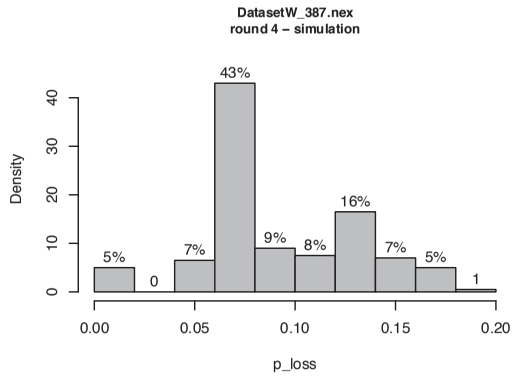


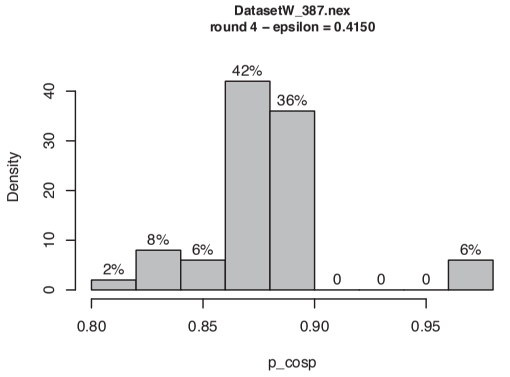
Nexus file: DatasetW_387.nex
Host/Parasite tree: 387/387 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

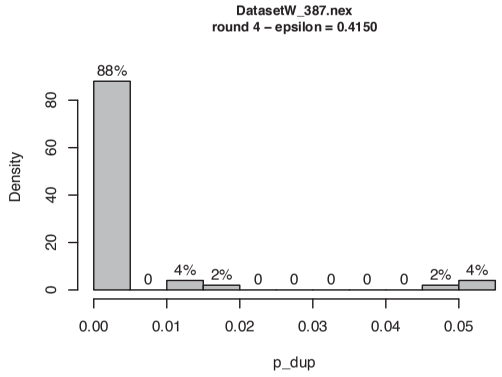


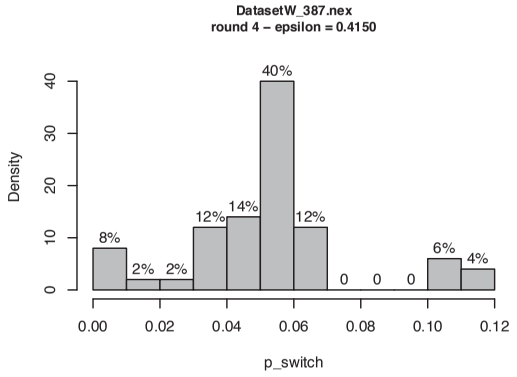


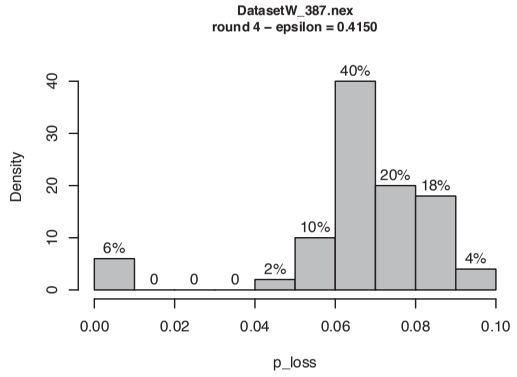


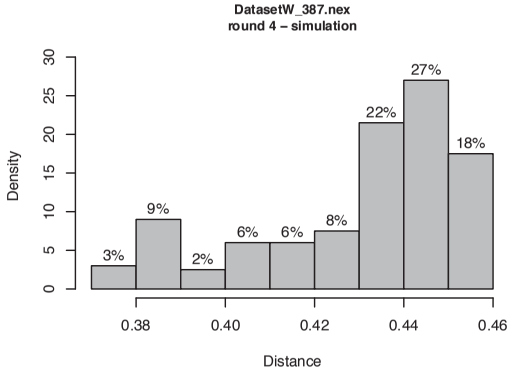


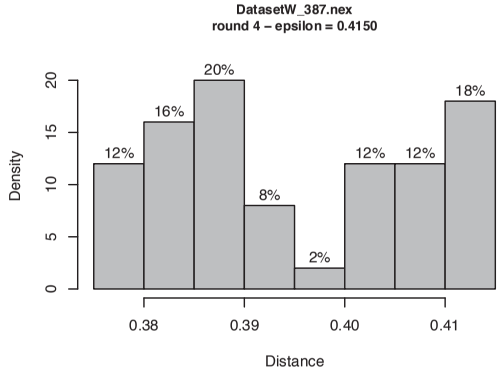




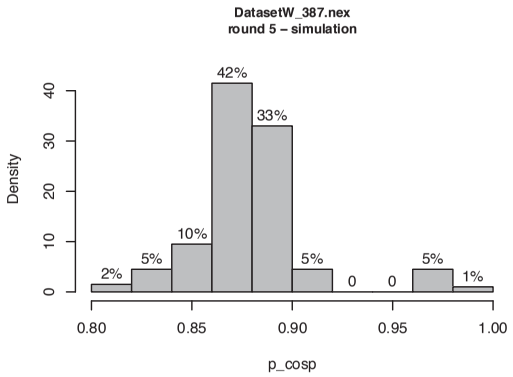


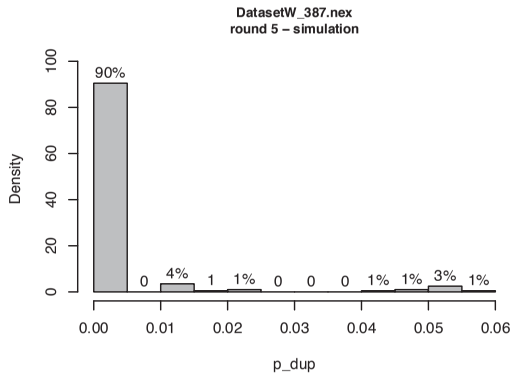


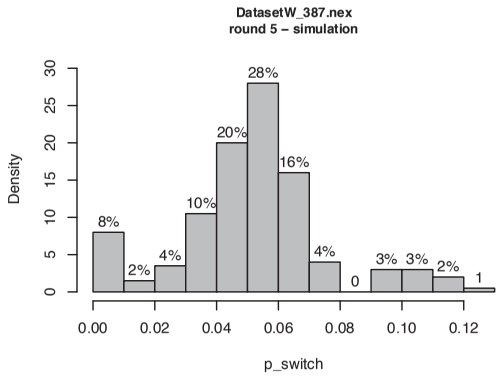


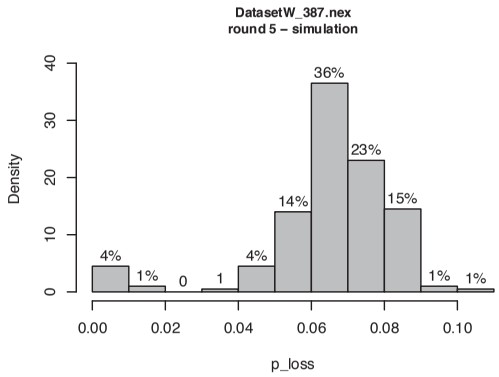


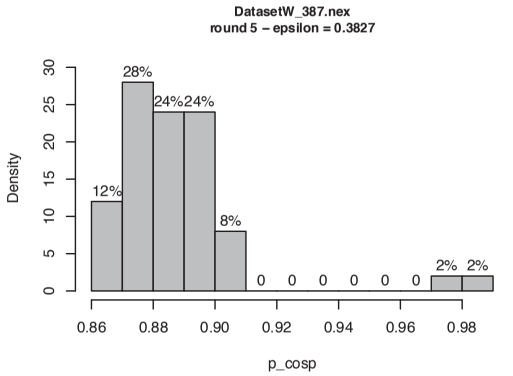
Nexus file: DatasetW_387.nex
Host/Parasite tree: 387/387 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

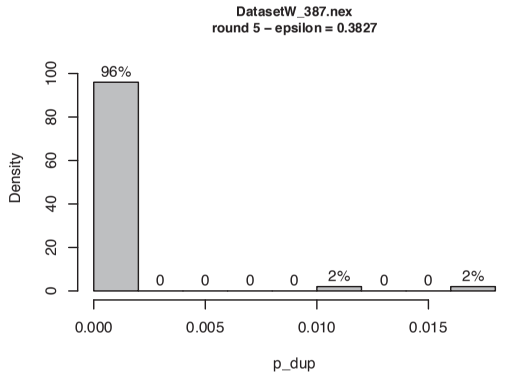


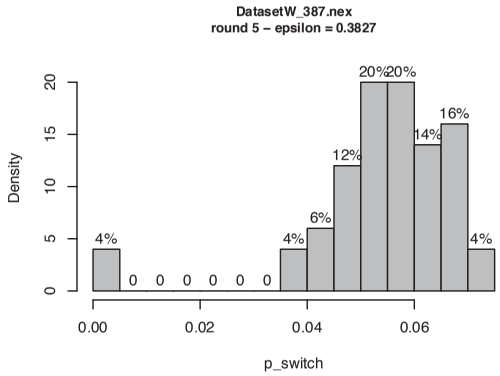


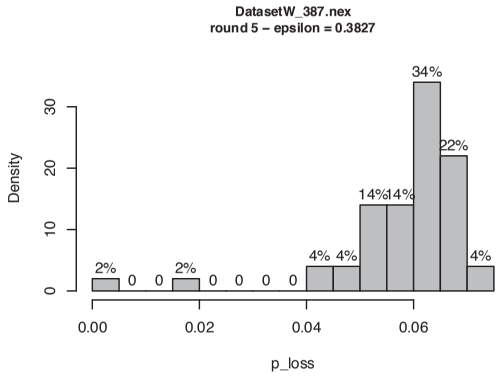


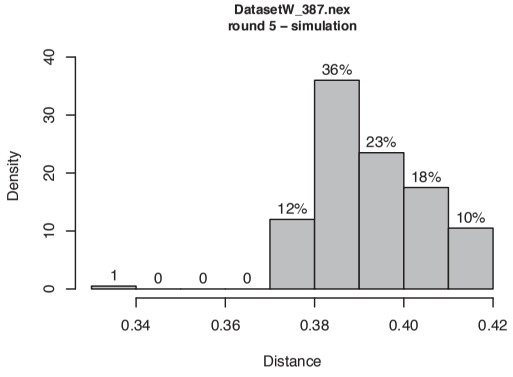


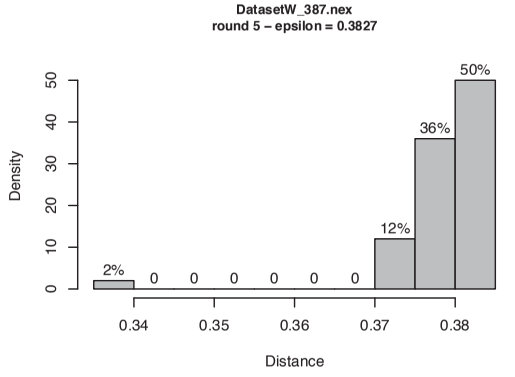












Nexus file: DatasetW_387.nex
Host/Parasite tree: 387/387 leaves
Prior distribution: 2000 vectors
Number of trees: 1000 trees
Maximum number of trees: 5000 trees
Number of rounds: 5 rounds
Perturbation: 0.0100
Tolerances: 0.1000,0.2500,0.2500,0.2500,0.2500
Simulation Model: 1 – From the root to the leaves model
Cyclicity test: 2 – Donati et al., 2014.
Metric: 3 – EVENTS AND MAAC Metric
(alpha1/alpha2): 0.7000/0.3000
Alpha (cosp.,dupl.,switch,loss): (1.0000,1.0000,1.0000,1.0000)

Additional Material for Chapter 5

Examples of orientation assignment procedure.

Consider the following path $v_1 < v_2 < v_3 < v_4$ and suppose that the most frequent orientations for the arcs are:

- for $\langle v_1, v_2 \rangle$: $(o_h^i = true \wedge o_h^j = false)$
- for $\langle v_2, v_3 \rangle$: $(o_h^i = false \wedge o_h^j = true)$
- for $\langle v_3, v_4 \rangle$: $(o_h^i = true \wedge o_h^j = false)$

The orientation assignment will be: $c_1, \bar{c_2}, c_3, \bar{c_4}$.

Observe that for the opposite direction for the path ($v_4 < v_3 < v_2 < v_1$), we have a symmetrical information (since the hits are always symmetric):

- for $\langle v_4, v_3 \rangle$: $(o_h^i = true \wedge o_h^j = false)$
- for $\langle v_3, v_2 \rangle$: $(o_h^i = false \wedge o_h^j = true)$
- for $\langle v_2, v_1 \rangle$: $(o_h^i = true \wedge o_h^j = false)$

We will thus obtain the following consistent orientation: $c_4, \bar{c_3}, c_2, \bar{c_1}$

Consider now the same path $v_1 < v_2 < v_3 < v_4$ but suppose that the most frequent orientations for the arcs are:

- for $\langle v_1, v_2 \rangle$: $(o_h^i = true \wedge o_h^j = false)$
- for $\langle v_2, v_3 \rangle$: $(o_h^i = false \wedge o_h^j = true)$
- for $\langle v_3, v_4 \rangle$: $(o_h^i = false \wedge o_h^j = false)$

There is no way to give a consistent orientation. The arc $\langle v_3, v_4 \rangle$ has to be discarded (the opposite direction is symmetric).

Additional Material for Chapter 6

Enumerating the elements of an implicit inclusion system is NP-hard. (A second strategy).

We present now a second strategy to prove the hardness of Problem 9. In this case, the reduction starts from SATISFIABILITY.

Theorem 5

numerating all the minimal elements of an implicit inclusion set system is NP-hard.

We now show that, if there is an algorithm for enumerating all the elements of a hitting set system in polynomial time, then there is a polynomial-time algorithm for solving SATISFIABILITY.

Proof. Let $F(x_1, ..., x_n)$ be a boolean expression in conjunctive normal form; we construct an inclusion system in the following way:

- $U = \bigcup\{T_i, F_i\}$ for $1 \leq i \leq n$.
- Given a subset $h \subseteq U$, an assignment for the variables can be found as follows:

$$v_h(x_i) = \begin{cases} true & \text{if } T_i \in h \wedge F_i \notin h \\ false & \text{if } T_i \notin h \wedge F_i \in h \\ undefined & otherwise. \end{cases}$$

- The property that implicitly defines $\mathcal{F}$ is the following: $h \in \mathcal{F}$ if and only if:

$$(F(v_h(x_1), ..., v_h(x_n)) = true) \vee (\exists i : \{T_i, F_i\} \subseteq h)$$

$(U, \mathcal{F})$ is clearly a hitting system.

We can now prove that F is not satisfiable if and only if the minimal elements of $\mathcal{F}$ are exactly the n trivial sets of the form $\{T_i, F_i\}$ for all i.

Assume that there exists a procedure to enumerate all the elements of the system in $O(\phi(n, K))$ where $n = |U|$, K is the total number of minimal hitting sets and ϕ is polynomial. Run the procedure to generate the first $n + 1$ hitting sets. If a non trivial set is found, then a valid assignment for F is given, otherwise the procedure will stop at the $n - th$ generated set because no other minimal set exists. □

Bibliography

Arvestad, L. et al. (2003). "Bayesian gene/species tree reconciliation and orthology analysis using MCMC". In: *Bioinformatics* 19.Suppl 1, pp. i7–i15.

Avis, David and Komei Fukuda (1996). "Reverse search for enumeration". In: *Discrete Applied Mathematics* 65.1–3. First International Colloquium on Graphs and Optimization, pp. 21 –46. ISSN: 0166-218X.

Balbuena, J. A., R. Míguez-Lozano, and I. Blasco-Costa (2013). "PACo: A Novel Procrustes Application to Cophylogenetic Analysis". In: *PLoS ONE* 8.4, e61048. URL: http://dx.doi.org/10.1371%2Fjournal.pone.0061048.

Bansal, M. S., E. Alm, and M. Kellis (2012). "Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss". In: *Bioinformatics* 28.12, pp. i283–i291.

Bansal, M. S., E. J. Alm, and M. Kellis (2013). "Reconciliation Revisited: Handling Multiple Optima when Reconciling with Duplication, Transfer, and Loss". In: *Proceedings of the 17th International Conference on Research in Computational Molecular Biology*. RECOMB'13. Beijing, China: Springer-Verlag, pp. 1–13.

Baroni, M. et al. (2005). "Bounding the number of hybridisation events for a consistent evolutionary history". In: *J. Math. Biol.* 51.2, pp. 171–182.

Beaumont, M. A., W. Zhang, and D. J. Balding (2002). "Approximate Bayesian Computation in Population Genetics". In: *Genetics* 162.4, pp. 2025–2035. eprint: http://www.genetics.org/content/162/4/2025.full.pdf+html. URL: http://www.genetics.org/content/162/4/2025.abstract.

Beaumont, M. A. et al. (2009). "Adaptive approximate Bayesian computation". In: *Biometrika* 96, pp. 983–990.

Bille, Philip (2005). "A survey on tree edit distance and related problems". In: *Theoretical Computer Science* 337.1–3, pp. 217 –239. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2004.12.030. URL: http://www.sciencedirect.com/science/article/pii/S0304397505000174.

Boetzer, M. et al. (2011). "Scaffolding pre-assembled contigs using SSPACE". In: *Bioinformatics* 27.4, pp. 578–9.

Bryant, D. and M. Steel (2009). "Computing the distribution of a tree metric". In: *IEEE/ACM Trans. on Comput. Biol. Bioinf.* 6.3, pp. 420–426.

Chandrasekaran, Karthekeyan et al. (2011). "Algorithms for Implicit Hitting Set Problems". In: *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '11. San Francisco, California: SIAM, pp. 614–629. url: http://dl.acm.org/citation.cfm?id=2133036.2133084.

Charleston, M. A. (1998). "Jungles: a new solution to the host/parasite phylogeny reconciliation problem". In: *Math. Biosci.* 149.2, pp. 191–223.

— (2003). "Recent results in cophylogeny mapping". In: *Adv. Parasit.* 54, pp. 303–330.

Conow, C. et al. (2010). "Jane: a new tool for the cophylogeny reconstruction problem". In: *Algorithm. Mol. Biol.* 5.1, p. 16.

Cox, M. P., D. A. Peterson, and P. J. Biggs (2010). "SolexaQA: At-a-glance quality assessment of Illumina second-generation sequencing data". In: *BMC Bioinformatics* 11, p. 485.

D. Swofford G. Olsen, P. Waddell and D. Hillis (1996). "Phylogenetic inference". In: *Molecular Systematics*, pp. 407–513.

Darling, A. C. et al. (2004). "Mauve: multiple alignment of conserved genomic sequence with rearrangements". In: *Genome Res* 14.7, pp. 1394–403.

Darwin, C. (1862). *On the various contrivances by which British and foreign orchids are fertilised by insects, and on the good effects of intercrossing*. Murray, J. London.

David, L. A. and E. J. Alm (2011). "Rapid evolutionary innovation during an Archaean Genetic Expansion". In: *Nature* 469, pp. 93–96.

Day, W. H. E. (1985). "Optimal algorithms for comparing trees with labeled leaves". In: *J. Classif.* 2.1, pp. 7–28.

Dayarian, A., T. P. Michael, and A. M. Sengupta (2010). "SOPRA: Scaffolding algorithm for paired reads via statistical optimization". In: *BMC Bioinformatics* 11, p. 345.

Deacon, Terrence W. (1997). *The Symbolic Species: The Co-evolution of Language and the Brain*. W.W. Norton. url: http://groups.lis.illinois.edu/amag/langev/paper/deacon97theSymbolic.html.

Del Moral, P., A. Doucet, and A. Jasra (2012). "An adaptive sequential Monte Carlo method for approximate Bayesian computation". In: *Stat. Comput.* 22, pp. 1009–1020.

Deng, J. et al. (2013). "Cophylogenetic relationships between Anicetus parasitoids (Hymenoptera: Encyrtidae) and their scale insect hosts (Hemiptera: Coccidae)". In: *BMC Evol. Biol.* 13.1, p. 275.

Donmez, N. and M. Brudno (2013). "SCARPA: scaffolding reads with practical algorithms". In: *Bioinformatics* 29.4, pp. 428–34.

Doyon, J.-P. et al. (2011a). "An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers". In: *Proceedings of the 8th annual RECOMB Satellite Workshop on Comparative Genomics (RECOMB-CG 2010)*. Ed. by E. Tannier. Vol. 6398. Lecture Notes in Bioinformatics. Ottawa, Canada: Spring-Verlag Berlin Heidelberg, pp. 93–108.

Doyon, J.-P. et al. (2011b). "Models, algorithms and programs for phylogeny reconciliation". In: *Brief. Bioinform.* 12.5, pp. 392–400.

Eiter, Thomas (Mar. 1994). "Exact Transversal Hypergraphs and Application to Boolean &Mgr;-functions". In: *J. Symb. Comput.* 17.3, pp. 215–225. issn: 0747-7171. doi: 10.1006/jsco.1994.1013. url: http://dx.doi.org/10.1006/jsco.1994.1013.

Farach-Colton, M., T. M. Przytycka, and M. Thorup (1995). "On the Agreement of Many Trees". In: *Inform. Process. Lett.* 55, pp. 297–301.

Fearnhead, P. and D. Prangle (2012). "Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation". In: *J. R. Stat. Soc.: Series B Stat. Methodol.* 74.3, pp. 419–474. doi: 10.1111/j.1467-9868.2011.01010.x. url: http://dx.doi.org/10.1111/j.1467-9868.2011.01010.x.

Felsenstein, J. (2003). *Inferring phylogenies*. Sunderland, MA: Sinauer Associates, Inc.

Finden, C. R. and A. D. Gordon (1985). "Obtaining Common Pruned Trees". In: *J. Classif.* 2, pp. 255–276.

Fondi, M. et al. (2014). "Draft genomes of three Antarctic Psychrobacter strains producing antimicrobial compounds against Burkholderia cepacia complex, opportunistic human pathogens". In: *Mar Genomics* 13, pp. 37–8.

Galardini, M. et al. (2011). "CONTIGuator: a bacterial genomes finishing tool for structural insights on draft genomes". In: *Source code for biology and medicine* 6, p. 11.

Ganapathy, G. et al. (2006). "Pattern Identification in Biogeography". In: *IEEE/ACM Trans. on Comput. Biol. Bioinf.* 3.4, pp. 334–346.

Gao, S., W. K. Sung, and N. Nagarajan (2011). "Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences". In: *J Comput Biol* 18.11, pp. 1681–91.

Gelman, Andrew et al. (2003). *Bayesian data analysis*. London: Chapman & Hall.

Gritsenko, A. A. et al. (2012). "GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies". In: *Bioinformatics* 28.11, pp. 1429–37.

Hafner, M. and S. Nadler (1988). "Phylogenetic trees support the coevolution of parasites and their hosts". In: *Nature* 332, pp. 258–259.

Hafner, M. S. and R. D. M. Page (1995). "Molecular Phylogenies and Host-Parasite Cospeciation: Gophers and Lice as a Model System". In: *Philos. T. Roy. Soc. B.* 349.1327, pp. 77–83.

Hallett, M. T. and J. Lagergren (2001). "Efficient algorithms for lateral gene transfer problems". In: *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB 2001)*. Ed. by T. Lengauer. ACM. New York, USA, pp. 149–156.

Hanada, K., Y. Suzuki, and T. Gojobori (2004). "A large variation in the rates of synonymous substitution for RNA viruses and its relationship to a diversity of viral infection and transmission modes". In: *Mol. Biol. Evol.* 21, pp. 1074–1080.

Hein, J. (1990). "Reconstructing Evolution of Sequences Subject to Recombination Using Parsimony". In: *Math. Biosci.* 98, pp. 185–200.

Hijum, S. A. van et al. (2005). "Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies". In: *Nucleic Acids Res* 33.Web Server issue, W560–6.

Hird, Myra J. (2010). "Coevolution, Symbiosis and Sociology". In: *Ecological Economics* 69.4. Special Section: Coevolutionary Ecological Economics: Theory and Applications, pp. 737 –742. issn: 0921-8009. doi: http://dx.doi.org/10.1016/j.ecolecon.2008.10.011. url: http://www.sciencedirect.com/science/article/pii/S0921800908004746.

Huelsenbeck, J. P., B. Rannala, and B. Larget (2000). "A Bayesian framework for the analysis of cospeciation". In: *Evolution* 54, pp. 352–364.

Huelsenbeck, J. P., B. Rannala, and Z. Yang (1997). "Statistical tests of host-parasite cospeciation". In: *Evolution* 51.2, 410—419.

Hughes, A. L. and R. Friedman (2000). "Evolutionary diversification of protein-coding genes of hantaviruses". In: *Mol. Biol. Evol.* 17, pp. 1558–1568.

Hughes, J. et al. (2007). "Multiple Cophylogenetic Analyses Reveal Frequent Cospeciation between Pelecaniform Birds and Pectinopygus Lice". In: *Syst. Biol.* 56.2, pp. 232–251.

Hugot, J. P. (1999). "Primates and their pinworm parasites: the Cameron hypothesis revisited". In: *Syst. Biol.* 48.3, pp. 523–546.

— (2003). "New Evidence for Hystricognath Rodent Monophyly from the Phylogeny of their Pinworms". In: *Tangled trees: Phylogeny, cospeciation, and coevolution*. Ed. by R. D. M. Page. Chicago, USA: UC Press, pp. 144–173.

Hunt, M. et al. (2014). "A comprehensive evaluation of assembly scaffolding tools". In: *Genome biology* 15.3, R42.

Husemann, P. and J. Stoye (2010). "Phylogenetic comparative assembly". In: *Algorithms Mol Biol* 5, p. 3.

Jackson, A. P. and M. A. Charleston (2004). "A cophylogenetic perspective of RNA-virus evolution". In: *Mol. Biol. Evol.* 21, pp. 45–57.

Johnson, K. P., D. M. Drown, and D. H. Clayton (2000). "A data based parsimony method of cophylogenetic analysis". In: *Zool. Scr.* 30, pp. 79–87.

Jong, Edwin de (2005). "The MaxSolve Algorithm for Coevolution". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. GECCO '05. Washington DC, USA: ACM, pp. 483–489. isbn: 1-59593-010-8. doi: 10.1145/1068009.1068091. url: http://doi.acm.org/10.1145/1068009.1068091.

Kolmogorov, M. et al. (2014). "Ragout-a reference-assisted assembly tool for bacterial genomes". In: *Bioinformatics* 30.12, pp. i302–i309.

Kurtz, S. et al. (2004). "Versatile and open software for comparing large genomes". In: *Genome Biol* 5.2, R12.

Langfelder, P., B. Zhang, and S. Horvath (2007). "Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R". In: *Bioinformatics* 24.5, pp. 719–720.

Langmead, B. et al. (2009). "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome". In: *Genome Biol* 10.3, R25.

Lawler, E. L., J. K. Lenstra, and A. H. G. Rinnooy Kan (1980). "Generating All Maximal Independent Sets: NP-Hardness and Polynomial-Time Algorithms". In: *SIAM Journal on Computing* 9.3, pp. 558–565. url: http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal\&id=SMJCAT000009000003000558000001\&idtype=cvips\&gifs=yes.

Li, H. and R. Durbin (2009). "Fast and accurate short read alignment with Burrows-Wheeler transform". In: *Bioinformatics* 25.14, pp. 1754–60.

Libeskind-Hadas, R. and M. A. Charleston (2009). "On the computational complexity of the reticulate cophylogeny reconstruction problem". In: *J. Comput. Biol.* 16.1, pp. 105–117.

Maddison, P. W. (1997). "Gene Trees in Species Trees". In: *Systematic Biology* 46.3, pp. 523–536.

Marin, J.-M. et al. (2012). "Approximate Bayesian computational methods". In: *Stat. Comput.* 22.6, pp. 1167–1180. doi: 10.1007/s11222-011-9288-2. url: http://dx.doi.org/10.1007/s11222-011-9288-2.

Merkle, D. and M. Middendorf (2005). "Reconstruction of the cophylogenetic history of related phylogenetic trees with divergence timing information". In: *Theor. Biosci.* 123.4, pp. 277–299.

Merkle, D., M. Middendorf, and N. Wieseke (2010). "A parameter-adaptive dynamic programming approach for inferring cophylogenies". In: *BMC Bioinformatics* 11.Supplementary 1, 10 pages.

Moran, S., I Newman, and Y. Wolfstahl (1990). "Approximation Algorithms for Covering a Graph by Vertex-Disjoint Paths of Maximum Total Weight". In: *NETWORKS* 20.

Moreno-Centeno, Erick and Richard M. Karp (2013). "The Implicit Hitting Set Approach to Solve Combinatorial Optimization Problems with an Application to Multigenome Alignment". In: *Operations Research* 61.2, pp. 453–468.

Nelder, J. and R. Mead (1965). "A simplex method for function minimization". In: *Comput. J.* 7, pp. 308–313.

Nemirov, K., A. Vaheri, and A. Plyusnin (2004). "Hantaviruses: co-evolution with natural hosts". In: *Rec. Res. Dev. Virol.* 6, pp. 201–228.

Page, R. D. and M. A. Charleston (1998). "Trees within trees: phylogeny and historical associations". In: *Trends Ecol. Evol.* 13.9, pp. 356–359.

Page, R. D. M. (1990). "Temporal congruence and cladistic analysis of biogeography and cospeciation". In: *Syst. Biol.* 39.3, pp. 205–226.

— (1994). "Maps between Trees and Cladistic Analysis of Historical Associations among Genes, Organisms, and Areas". In: *Syst. Biol.* 43.1, pp. 58–77.

Paredis, J. (1995). "Coevolutionary Computation". In: *Artificial Life* 2.4, pp. 355–375.

Paterson, A., R. Palma, and R. Gray (2003). "Drowning on arrival, missing the boat, and x-events: How likely are sorting events?" In: *Tangled trees: Phylogeny, cospeciation, and coevolution*. Ed. by R. D. M. Page. Chicago, USA: UC Press, pp. 287–307.

Plyusnin, A. and S. P. Morzunov (2001). "Virus evolution and genetic diversity of hantaviruses and their rodent hosts". In: *Curr. Top. Microbiol. Immunol.* 256, pp. 47–75.

Poulin, R. and D. Mouillot (2003). "Parasite specialization from a phylogenetic perspective: a new index of host specificity". In: *Parasitology* 126 (5), pp. 473–480.

Pritchard, J. et al. (1999). "Population growth of human Y chromosomes: a study of Y chromosome microsatellites". In: *Mol. Biol. Evol.* 16.12, pp. 1791–1798.

Rajaraman, Ashok, Eric Tannier, and Cedric Chauve (2013). "FPSAC: fast phylogenetic scaffolding of ancient contigs." In: *Bioinformatics* 29.23, pp. 2987–2994.

Ramsden, C., E. Holmes, and M. Charleston (2009). "Hantavirus Evolution in Relation to Its Rodent and Insectivore Hosts". In: *Mol. Biol. Evol.* 26.1, pp. 143–153.

Refrégier, G. et al. (2008). "Cophylogeny of the anther smut fungi and their caryophyllaceous hosts: Prevalence of host shifts and importance of delimiting parasite species for inferring cospeciation". In: *BMC Evol. Biol.* 8.1, p. 100.

Robinson, D. F. and L. R. Foulds (1981). "Comparison of Phylogenetic Trees". In: *Math. Biosci.* 55, pp. 131–147.

Ronquist, F. (2002). "Parsimony analysis of coevolving species associations". In: *Tangled trees: Phylogeny, cospeciation and coevolution. University of Chicago Press, Chicago*, pp. 22–64.

— (2003). "Tangled trees: phylogeny, cospeciation, and coevolution". In: ed. by R. D. M. Page. University of Chicago Press. Chap. Parsimony analysis of coevolving species associations, pp. 22–64.

Rosen, D. E. (1978). "Vicariant Patterns and Historical Explanation in Biogeography". In: *Syst. Biol.* 27.2, pp. 159–188.

Rosenblueth, M. et al. (2012). "Evolutionary relationships of flavobacterial and enterobacterial endosymbionts with their scale insect hosts (Hemiptera: Coccoidea)". In: *J. Evolution. Biol.* 25, pp. 2357–2368.

Roth, A. C., G. H. Gonnet, and C. Dessimoz (2008). "Algorithm of OMA for large-scale orthology inference". In: *BMC Bioinformatics* 9, p. 518.

Salmela, L. et al. (2011). "Fast scaffolding with small independent mixed integer programs". In: *Bioinformatics* 27.23, pp. 3259–65.

Schwikowski, Benno and Ewald Speckenmeyer (2002). "On enumerating all minimal solutions of feedback problems". In: *Discrete Applied Mathematics* 117.1–3, pp. 253–265. issn: 0166-218X. doi: http://dx.doi.org/10.1016/S0166-218X(00)00339-5. url: http://www.sciencedirect.com/science/article/pii/S0166218X00003395.

Silva, G. G. et al. (2013). "Combining de novo and reference-guided assembly with scaffold_builder". In: *Source Code Biol Med* 8.1, p. 23.

Simões, P. M. (2012). "Diversity and dynamics of *Wolbachia*-host associations in arthropods from the Society archipelago, French Polynesia". PhD thesis. University of Lyon 1, France.

Simões, P. M. et al. (2011). "Wolbachia detection: an assessment of standard PCR protocols". In: *Mol. Ecol. Resour.* 11.3, pp. 567–572.

Simpson, J. T. et al. (2009). "ABySS: a parallel assembler for short read sequence data". In: *Genome Res* 19.6, pp. 1117–23.

Sisson, S. A., Y. Fan, and M. M.Tanaka (2007). "Sequential Monte Carlo without likelihoods". In: *Proc. Natl. Acad. Sci. USA* 104, pp. 1760–1765.

— (2009). "Sequential Monte Carlo without likelihoods. Erratum 1041760." In: *Proc. Natl. Acad. Sci. USA* 106, pp. 16889–16889.

Steel, M. and D. Penny (1993). "Distributions of tree comparison metrics – Some new results". In: *Syst. Biol.* 42, pp. 126–141.

Stolzer, M. L. et al. (2012). "Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees". In: *Bioinformatics* 28.18, pp. i409–i415. doi: 10.1093/bioinformatics/bts386.

Szöllősi, Gergely J. et al. (2013). "Lateral Gene Transfer from the Dead". In: *Systematic Biology*. doi: 10.1093/sysbio/syt003. eprint: http://sysbio.oxfordjournals.org/content/early/2013/01/25/sysbio.syt003.full.pdf+html. url: http://sysbio.oxfordjournals.org/content/early/2013/01/25/sysbio.syt003.abstract.

Tofigh, A., M. Hallett, and J. Lagergren (2011). "Simultaneous Identification of Duplications and Lateral Gene Transfers". In: *IEEE/ACM Trans. on Comput. Biol. Bioinf.* 8.2, pp. 517–535.

Vienne, D. M. De, T. Giraud, and J. A. Skyhoff (2007). "When can host shifts produce congruent host and parasite phylogenies? A simulation approach". In: *J. Evol. Biol.* 20.4, pp. 1428–1438.

Waterman, M. S. and T. F. Smith (1978). "On the Similarity of Dendrograms". In: *J. Theor. Biol.* 73, pp. 789–800.

Wieseke, N., M. Bernt, and M. Middendorf (2013). "Unifying Parsimonious Tree Reconciliation". In: *13th Workshop on Algorithms in Bioinformatics (WABI 2013)*. Vol. 8126. Lecture Notes in Computer Science. Sophia Antipolis, France: Spring-Verlag Berlin Heidelberg, pp. 200–214.