Computational Biology

Tandy Warnow *Editor*

Bioinformatics and Phylogenetics

Seminal Contributions of Bernard Moret

ISCB International Society for Computational Biology

Springer

Computational Biology

Volume 29

The *Computational Biology* series publishes the very latest, high-quality research devoted to specific issues in computer-assisted analysis of biological data. The main emphasis is on current scientific developments and innovative techniques in computational biology (bioinformatics), bringing to light methods from mathematics, statistics and computer science that directly address biological problems currently under investigation.

The series offers publications that present the state-of-the-art regarding the problems in question; show computational biology/bioinformatics methods at work; and finally discuss anticipated demands regarding developments in future methodology. Titles can range from focused monographs, to undergraduate and graduate textbooks, and professional text/reference works.

More information about this series at http://www.springer.com/series/5769

Tandy Warnow
Editor

Bioinformatics and Phylogenetics

Seminal Contributions of Bernard Moret

Springer

Editor
Tandy Warnow
University of Illinois at Urbana-Champaign
Urbana, IL, USA

ISSN 1568-2684 ISSN 2662-2432 (electronic)
Computational Biology
ISBN 978-3-030-10836-6 ISBN 978-3-030-10837-3 (eBook)
https://doi.org/10.1007/978-3-030-10837-3

Library of Congress Control Number: 2018966860

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Dedicated to Bernard M. E. Moret on his retirement.

Preface

This Festschrift is in honor of Bernard M. E. Moret, whose retirement from Ecole Polytechnique Fédérale de Lausanne (EPFL) in December 2016 culminated a nearly 40-year career. Bernard's research spanned several areas in computer science, including algorithm engineering, high-performance computing, and algorithmic computational biology. Many of Bernard's contributions were concerned with inferring and using phylogenies (i.e., evolutionary trees and phylogenetic networks), especially for large and challenging datasets. His work in genome rearrangement phylogeny is the best known, where he has been one of a small number of leading researchers in establishing theory and developing novel methods and open-source software based on rigorous mathematical theory. In addition, he has also contributed to the understanding and developing divide-and-conquer strategies, supertree construction, absolute fast converging phylogeny estimation, phylogenetic networks, and the use of phylogenies to answer biological questions. He has also trained (both directly and indirectly) many of the leading people in computational biology, including several of the contributors to this volume. For example, Alexandros Stamatakis and Daniel Doerr were postdoctoral researchers of Bernard's; Jijun Tang was his Ph.D. student; and Mark Holder, Luay Nakhleh, and Sébastien Roch were funded by the CIPRES project (see http://www.phylo.org), which Moret directed from 2003–2006.

Two conferences were organized in honor of Bernard's retirement: the first, organized by his current and former students and postdocs, was held at EPFL on November 7–8, 2016 (see https://lcbb.epfl.ch/climb/), and the second, which I organized, was held at Berkeley on June 2, 2017 (see http://tandy.cs.illinois.edu/Moret-Festschrift.html). These two conferences led to this Festschrift.

The chapters in this volume represent some of the areas in which Bernard's work has had an influence, including methods for phylogenetic tree and network estimation, genome rearrangements, cancer phylogeny, species trees, divide-and-conquer strategies, and the use of integer linear programming in computational biology. Each chapter provides an introduction to a cutting-edge problem in computational biology that is computationally and mathematically interesting. Hence, the volume is designed to be useful as a text for a graduate course in

computational biology and bioinformatics, aiming at computer scientists, applied mathematicians, and statisticians. Although much of the work that is described is advanced, each chapter provides references to the literature to enable the reader to obtain additional background, if needed.

The chapters are ordered based on the statistical complexity of the problem they address. The first three chapters address challenges in developing accurate and efficient software for the NP-hard maximum likelihood phylogeny estimation problem. Chapters 1 (by Alexandros Stamatakis) and 2 (by Stéphane Guindon and Olivier Gascuel) focus on optimizing maximum likelihood codes (including but not limited to numerical optimization), while Chap. 3 (by David Bader and Kamesh Madduri) focuses on high-performance computing aspects; together, the three chapters provide complementary insights into how to design codes for this important basic problem in computational phylogenetics. Chapter 4 (by Sébastien Roch) is also about phylogeny estimation from aligned sequences, and addresses a basic statistical question: how much data does a phylogeny estimation method require (or need) to recover the true tree with high probability, as a function of the model tree parameters (e.g., number of leaves and lengths of the branches in the tree)? Roch's chapter comes with a Jupyter notebook and provides scripts for analyses and simulations. Chapter 5 (by Nadia El-Mabrouk and Emmanuel Noutahi) addresses algorithms to infer gene trees when the input includes aligned gene sequences and also a species tree. The inference of species trees is covered in Chaps. 6 and 7. Chapter 6 (which I contributed) is about divide-and-conquer strategies to scale phylogeny estimation methods to large datasets, and specifically addresses limitations in current supertree methods. Chapter 7 (by Benjamin Redelings and Mark Holder) is about taxonomic supertrees and the challenge of constructing them when some taxa in the input have unknown placements within a taxonomic hierarchy. Chapter 8 (by Santos Muños et al.) addresses the inference of ultrametric distances from additive distance matrices, and so is related to the problem of assigning dates to internal nodes in phylogenetic trees. Chapters 9 (by Jiijun Tang) addresses the inference of ancestral genomes under genome rearrangement events. Chapters 10 (by Ron Zeira and Ron Shamir) and 11 (by Russell Schwartz) address complementary approaches to inferring evolutionary histories in cancer; Zeira and Shamir focus on how chromosomal rearrangements can be used as indicators of evolutionary history and Schwartz provides an overview of different approaches to tumor phylogenetics. Chapters 12 (by Louxin Zhang) and 13 (by R. A. Leo Elworth et al.) examine problems in phylogenetic networks, with Zhang focusing on discrete mathematics questions and Elworth et al. addressing statistical estimation issues. Chapter 14 (by Daniel Doerr and Jens Stoye) uses evolution to provide a framework within which to understand comparative and functional genomics. Chapter 15 (by Dan Gusfield) provides an introduction to Integer Linear Programming and its use in computational biology, with specific focus on how ILP can be used to solve the NP-hard Traveling Salesman Problem; his chapter also comes with software and datasets.

While the chapters are designed to be self-contained (and each contains a substantial bibliography to enable the reader to get additional background), some background in computer science (algorithms and running time analysis) and statistics (e.g., the use of probabilistic models and statistical estimation under these models) is assumed. Background in computational phylogenetics is helpful but not required, but many readers may find it helpful to read some of the textbooks in phylogenetics. For a statistical perspective on this research area, see [1] and [3]. A computer science perspective on algorithm design for phylogeny estimation (especially for large datasets) is provided in [2].

Urbana, USA Tandy Warnow

References

1. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Sunderland, Massachusetts (2004)
2. Warnow, T.: Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation. Cambridge University Press, Cambridge, UK (2018)
3. Yang, Z.: Molecular Evolution: A Statistical Approach. Oxford University Press, Oxford (2014)

Contents

Editor and Contributors

About the Editor

Tandy Warnow is the Founder Professor of Computer Science at the University of Illinois at Urbana-Champaign, where she is also an affiliate in the Departments of Mathematics, Statistics, Bioengineering, Electrical and Computer Engineering, Animal Biology, Entomology, and Plant Biology. Tandy received her Ph.D. in Mathematics in 1991 at UC Berkeley under the direction of Gene Lawler and did postdoctoral training with Simon Tavaré and Michael Waterman at USC. Her research combines computer science, statistics, and discrete mathematics, focusing on developing improved models and algorithms for reconstructing complex and large-scale evolutionary histories in biology and historical linguistics. She has published more than 160 papers and one textbook, graduated 11 Ph.D. students, and has 5 current Ph.D. students. Her awards include the NSF Young Investigator Award (1994), the David and Lucile Packard Foundation Award (1996), a Radcliffe Institute Fellowship (2006), and the John Simon Guggenheim Foundation Fellowship (2011). She was elected a Fellow of the Association for Computing Machinery (ACM) in 2015 and of the International Society for Computational Biology (ISCB) in 2017. Warnow succeeded Moret as the director of the NSF-funded CIPRES (Cyber-Infrastructure for Phylogenetic Research) project, whose goal was "To provide the computational infrastructure needed to reconstruct phylogenies for millions of taxa". CIPRES was a multi-institutional project that lasted from 2003 to 2010, and supported several of the contributors to this volume.

Dr. Tandy Warnow is the author of Chapter 6.

Contributors

David A. Bader (PhD 1996, Univ Maryland) is Professor and Chair of the School of Computational Science and Engineering, College of Computing, at Georgia Institute of Technology. He is a Fellow of the IEEE and AAAS, and his research aims to help solve global grand challenges in science, engineering, computing, and data science.

Daniel Doerr (PhD 2015, Bielefeld) is a postdoctoral researcher at Bielefeld University, and a former postdoctoral researcher at EPFL with Bernard Moret.

Nadia El-Mabrouk (PhD 1996, Univ. Paris VII) is Professor at the Computer Science Department (DIRO) of the University of Montreal, where she is also the Director of the Department of Computer Service Education (DESI).

R. A. Leo Elworth (MS 2017, Rice University) is a PhD candidate in the Department of Computer Science at Rice University, working with Prof. Luay Nakhleh.

Olivier Gascuel (PhD 1981, Paris VI) is the head of the Centre for Bioinformatics, Biostatistics and Integrative Biology at Institut Pasteur, Paris. Gascuel was awarded "Le Grand Prix Inria - Académie des sciences" in 2017 for his research contributions to phylogenetics and bioinformatics, and his first PhyML paper (Guindon and Gascuel, Syst Biol 2003) was the most cited paper in environment and ecology from 2007 to 2011.

Stéphane Guindon (PhD 2003, Univ Montpellier) is a researcher at the CNRS (Centre National de la Recherche Scientifique) and the University of Montpellier, France. His research focuses on statistical modeling and computational aspects of statistical inference applied to the analysis of genetic data. His 2003 Systematic Biology article on the PhyML program (co-authored with Olivier Gascuel) was classified as a Fast Breaking Paper and then Current Classic (most cited article in the field) in Environment and Ecology.

Dan Gusfield (PhD 1980, UC Berkeley) is Distinguished Professor of Computer Science at the University of California, Davis, and a Fellow of the IEEE, the ACM, and the International Society of Computational Biology (ISCB). He served as chair of the Computer Science department at UCD (2000-2004), and was the founding Editor-in-Chief of The IEEE/ACM Transactions of Computational Biology and Bioinformatics until January 2009.

Mark T. Holder (PhD 2001, University of Texas) is an Associate Professor in the Department of Ecology and Evolutionary Biology at the University of Kansas, and a senior scientist at the Biodiversity Institute at the University of Kansas. He develops statistical methods and software for phylogenetic estimation.

Eric Lam is a Master's student in Statistics with a specialization in Bioinformatics at the University of Ottawa, where he also received his Bachelor's degree in Statistics.

Kamesh Madduri (PhD 2008, Georgia Inst. of Technology) is Associate Professor at Pennsylvania State University in the School of Electrical Engineering and Computer Science. He is a recipient of the NSF CAREER Award. His research interests are in data science and high-performance computing.

Daniella Santos Muñoz (B.Sc. 2019, University of Ottawa) is an undergraduate student at the University of Ottawa majoring in Biology and Computer Science.

Luay Nakhleh (PhD 2004, University of Texas at Austin) is the J.S. Abercrombie Professor of Computer Science at Rice University, where he is also Chair of the Department of Computer Science. Dr. Nakhleh is the recipient of an Alfred P. Sloan Fellowship and a Guggenheim Fellowship. His main research is in phylogenetic networks.

Emmanuel Noutahi (B.S. 2014, Université de Montreal) is currently a PhD candidate in computational biology at Université de Montréal. His research interests include computational biology, molecular evolution, phylogenetics, and algorithms in general.

Huw A. Ogilvie (PhD 2018, Australian National University) is a Postdoctoral Research Associate and Lecturer in the Department of Computer Science at Rice University. He is researching new phylogenetic methods in the laboratory of Professor Luay Nakhleh.

Benjamin D. Redelings (PhD 2006, UCLA) is a Research Scientist at Duke University and a Senior Assistant Researcher at the University of Kansas. He is a winner of the Mitchell Prize (awarded jointly by the American Statistical Association's Section on Bayesian Statistical Science and the International Society for Bayesian Analysis). Redelings' research is focused on Bayesian inference in phylogenetics.

Sébastien Roch (PhD 2007, Berkeley) is Professor of Mathematics at the University of Wisconsin-Madison, where he is also affiliated with the Department of Statistics and the Institute for Foundations of Data Science. He is the recipient of an NSF CAREER Award, an Alfred P. Sloan Fellowship, and a Simons Fellowship. His current research interests lie at the interface of applied probability, statistics, and theoretical computer science | with an emphasis on biological applications, notably mathematical phylogenetics.

David Sankoff (PhD 1969, McGill University) is the Canada Research Chair in Mathematical Genomics at the University of Ottawa. He is also a Fellow of the Royal Society of Canada and a Fellow of the International Society for Computational Biology (ISCB). His main research interest is comparative genomics.

Russell Schwartz (PhD 2000, MIT) is a Professor of Biological Sciences and Computational Biology at Carnegie Mellon University. He works on various problems in computational genetics and computational biophysics, with primary current focus on computational cancer biology. He is also active in efforts to improve computational and quantitative education in the life sciences.

Ron Shamir (PhD 1984, Berkeley) is Sackler Professor of Bioinformatics in the Blavatnik School of Computer Science at Tel Aviv University (TAU). He is head of the Edmond J. Safra Center for Bioinformatics at TAU. He is a Fellow of the ACM, the International Society for Computational Biology (ISCB), and a winner of the RECOMB 2011 and 2016 Test of Time Awards.

Alexandros Stamatakis (PhD 2004, Technical University of Munich) is a group leader of the scientific computing group at Heidelberg Institute for Theoretical Studies (HITS) and Professor of Computer Science at Karlsruhe Institute of Technology (KIT). Stamatakis is a former postdoc of Bernard Moret, and is the developer of RAxML, ExaML, and other software related to likelihood-based phylogeny estimation.

Jens Stoye (PhD 1997, Bielefeld University) is Professor of Genome Informatics at Bielefeld University, where he is also a Director of the Center for Interdisciplinary Research (ZiF). His current research interests are algorithms for comparative genomics and genome-scale sequence analysis.

Jijun Tang (PhD 2004, Univ New Mexico) is Professor of Computer Science and Engineering at University of South Carolina. Dr. Tang received his Ph.D. from University of New Mexico in Computer Science in 2004, under the supervision of Dr. Moret. Dr. Tang's main research area is Computational Biology, with focus on algorithm development for genome rearrangements.

Ron Zeira is a PhD student at Tel Aviv University working on developing computational models in cancer genomics. Special interests include algorithms and models for genome rearrangements, cancer karyotyping, cancer heterogeneity, and graph algorithms.

Louxin Zhang (PhD 1996, Univ. Waterloo) is Professor of Computational Biology at the National University of Singapore. His current research interests include comparative genomics, phylogenetic networks and cellular protein networks.

Jiafan Zhu (MS 2018, Rice University) is a PhD candidate in the Department of Computer Science at Rice University, working with Professor Luay Nakhleh.

Introduction: A Biography of Bernard Moret

Bernard M. E. Moret, Professor Emeritus at Ecole Polytechnique Fédérale de Lausanne (EPFL), was one of the second wave of computer scientists to enter the field of computational biology. During his scientific career, described in greater detail at the EPFL website, https://people.epfl.ch/bernard.moret, he had several influential positions: Professor and Chair of Computer Science at the University of New Mexico, Professor and Associate Dean for Education at EPFL, Founder of the ACM Journal of Experimental Algorithmics, Founder and Chair of the Steering Committee for the Workshop on Algorithms in Bioinformatics (WABI), and Chair of the Biodata Management and Analysis (BDMA) study section at the US National Institutes of Health. Notably, Moret led a team of over 50 biologists, computer scientists, and mathematicians in the CIPRES (Cyber-Infrastructure for Phylogenetic Research) project, funded by the US National Science Foundation (NSF). He has published over 100 papers in computational biology, under funding from the US NSF, the Alfred P. Sloan Foundation, the IBM Corporation, the US NIH, the Swiss NSF, and SystemsX.ch. He is also an elected Fellow of the ISCB (International Society for Computational Biology).

Before getting involved in computational biology, Moret's main focus was on Algorithm Engineering: the design and assessment of algorithms that are efficient in practice. Professor Moret first realized the gap between theory and practice in algorithms when working with his colleague Henry Shapiro on a textbook in algorithms in the late 1980s. In a loose coalition, he, Kurt Mehlhorn (Max Planck Institut fuer Informatik), Giuseppe Italiano (U. di Roma Tor Vergata), David Johnson (late of Bell Laboratories), and Catherine McGeogh (Amherst College) developed Algorithm Engineering into a rigorous field within computer science. Assessment of algorithms became a particular target, since the theoretical efficiency is based on asymptotic behavior over an infinite set of instances, something very far from practice. This area has become a confirmed junior partner of the more abstract study of algorithms, not just leading to a journal and two annual conferences, but also influencing the work in the theoretical community, giving rise to much work on how to optimize behavior in terms of cache memory, paging, and out-of-memory

access as well as to increased attention on the impact of improvements in asymptotic behavior.

Moret first became involved in computational biology in the late 1990s, as a result of a collaboration on genome rearrangement phylogeny with Tandy Warnow and Bob Jansen (then chair of the Department of Integrative Biology at the University of Texas at Austin). Moret also collaborated with Warnow and Randy Linder (also in the Department of Integrative Biology at UT-Austin) on reconstructing phylogenetic networks (i.e., graphical models of species evolution that are not tree-like, since they model events such as hybrid speciation and horizontal gene transfer). These collaborations resulted in a great deal of mathematical theory and also in the development of open-source software, including GRAPPA (Genome Rearrangement Analysis using Parsimony and other Phylogenetic Algorithms), a collection of programs designed to enable phylogeny estimation for whole genomes based on rearrangement events (e.g., inversions and transpositions of regions within genomes).

From 2001 to 2003, Moret led a core team of researchers from biology and computer science in developing a proposal or an integrative research project to make progress on the development of algorithms, software, and computational infrastructure necessary to infer the Tree of Life. In 2003, these efforts were rewarded by a large ITR grant from the NSF for the CIPRES (Cyber-Infrastructure for Phylogenetic Research, 2003–2010) project. CIPRES involved over a hundred graduate students and many postdocs (including many of the contributors in this volume), and helped the field of computational phylogenetics to flourish within the USA. While the director of CIPRES, Moret's research expanded to include "fast converging" phylogeny estimation methods (i.e., methods that provably recover the true tree with high probability from polynomial length sequences), supertree estimation, exploration of the space of optimal trees under various criteria (including maximum parsimony), orthology detection, and high-performance computing in computational biology (much in collaboration with David Bader).

Moret left the USA in 2006 to join the faculty at EPFL, his alma mater. His favorite contribution during this period has been the work he did with Xiuwei Zhang (initially a Ph.D. student and then a postdoc) on something they termed "phylogenetic transfer of knowledge" (PTK), a type of transfer learning based on a graphical model that leverages known information about evolutionary relationships to improve inference of systems such as transcriptional regulatory networks.

Moret's impact in computational biology, and especially in computational phylogenetics and comparative genomics, has been large. He is well known for developing key algorithms and then implementing the algorithms and testing them rigorously (a consequence of his expertise and commitment to algorithm engineering), and for making his software available in open-source form. However, mentoring young researchers has always been his main priority. He served as a thesis advisor for over 50 MS students, 18 doctoral students, and 6 postdocs. At EPFL, his lab hosted 2–4 young women each summer; over a period of 5 years, 90% of these women went on to Ph.D. programs in Computer Science, Bioinformatics, Physics, and Climatology, at top programs in Europe and North

America. Most of his postdocs and many of his Ph.D. students are faculty members at research institutions in the US and Europe, while most of his MS students at EPFL are now enrolled in Ph.D. programs.

Moret retired from EPFL in 2017, and his retirement was celebrated by two conferences: one at EPFL and another at Berkeley. Many of his former students, postdocs, and colleagues (e.g., Richard Karp, David Sankoff, James Larus, and Willy Zwaenepoel) spoke at these events. Moret now lives on the Big Island of Hawaii, with his wife Carol and cats.

Chapter 1
A Review of Approaches for Optimizing Phylogenetic Likelihood Calculations

Alexandros Stamatakis

Abstract The execution times of likelihood-based phylogenetic inference tools for Maximum Likelihood or Bayesian inference are dominated by the Phylogenetic Likelihood Function (PLF). The PLF is executed millions of times in such analyses and accounts for 85–95% of overall run time. In addition, storing the Conditional Likelihood Vectors (CLVs) required for computing the Phylogenetic Likelihood Function largely determines the associated memory consumption. Storing CLVs accounts for approximately 80% of the overall, and typically large, memory footprint of likelihood-based tree inference tools. In this chapter, we review recent technical as well as algorithmic advances for accelerating PLF calculations and for saving CLV memory. We cover topics such as algorithmic techniques for optimizing PLF computations and low-level optimization on modern x86 architectures. We conclude with an outlook on potential future technical and algorithmic developments.

Keywords Phylogenetic inference · Likelihood calculations · Performance optimization · Parallel computing · Terraces in tree space

1.1 Introduction

A thorough phylogenetic analysis of current whole-transcriptome or whole-genome datasets using likelihood-based methods can require millions or even tens of millions of CPU hours on supercomputers.

The Phylogenetic Likelihood Function (PLF) dominates the memory as well as computational resource requirements in *all* tools for Maximum Likelihood (ML) and Bayesian Phylogenetic inference (BI). The PLF is thus the key target function for (i) optimization using algorithmic and technical means and (ii) parallelization on shared and distributed memory systems.

A. Stamatakis (✉)
Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
e-mail: Alexandros.Stamatakis@h-its.org

A. Stamatakis
Karlsruhe Institute of Technology, Karlsruhe, Germany

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_1

In this chapter, we will review recently introduced techniques for optimizing the sequential and parallel performance of the PLF, both, via technical as well as algorithmic means. Orthogonal numerical approaches to optimizing phylogenetic likelihood calculations are outlined in the chapter by Gascuel and Guindon in this volume. The main focus of this chapter is on tree inference on supermatrices and not on gene tree/species tree reconciliation methods, which constitute a valid alternative and complementary approach to the problem. Gene tree/species tree reconciliation approaches are discussed in the chapter by El-Mabrouk and Noutahhi in this book. As gene tree reconstruction and statistical approaches for gene tree species/tree reconciliation also frequently and heavily rely on efficient phylogenetic likelihood calculations (see, e.g., [4]), several of the approaches discussed here are also relevant for gene tree/species tree reconciliation approaches. A large fraction of the results we present here have been obtained via the algorithm engineering approach.

The chapter is structured as follows: We first provide a brief introduction of how the likelihood score can be computed on phylogenetic trees in Sect. 1.2. In Sect. 1.3, we review algorithmic techniques for accelerating sequential likelihood calculations. Then, we cover technical approaches to accelerating the PLF on single x86 cores (Sect. 1.4). Thereafter, we will outline how full and/or partial terraces in tree space can be used for saving PLF memory and computations (Sect. 1.5). The algorithmic aspects of parallel likelihood computations are covered in Sect. 1.6. We conclude the chapter with a discussion of open problems and future challenges in Sect. 1.7.

1.2 Calculating the Likelihood on Phylogenies

We will initially provide a brief overview of how the PLF is computed. An alternative description and notation is provided in the chapter by Guindon and Gascuel in this volume.

The input for a likelihood-based phylogenetic analysis is a multiple sequence alignment (MSA with n sequences/taxa and m alignment columns). The branch length values on the resulting phylogeny (an unrooted, unordered, leaf-labeled binary tree) represent the mean expected number of substitutions per-MSA site.

In the following, we will outline how to compute the likelihood on a fixed tree topology. Apart from the tree—including $2n - 3$ branch length values—itself, one needs some additional likelihood model parameters. The most important parameter is the Q matrix, which specifies the transition rates for time dt between the character states (e.g., 4 states for DNA data and 20 states for protein data). The Q matrix encodes a time-reversible Markov model of state substitutions. The transition probability matrix P for a given branch length t is obtained via $P(t) = e^{Qt}$. It can be calculated via an eigenvector/eigenvalue decomposition.

To properly initialize the rate matrix Q, we also need the stationary frequencies of the states (e.g., $\pi_A, \pi_C, \pi_G, \pi_T$, for DNA data). These can be obtained empirically by determining the respective frequencies of occurrence in the MSA. Alternatively, they can be considered as free parameters of the model and estimated (ML) or sampled,

accordingly. Finally, if the MSA contains rate heterogeneity (this is the case for most empirical MSAs [27]), that is, distinct MSA sites evolve at different rates, one also needs to optimize/sample the α shape parameter of the Γ model [39] of rate heterogeneity.

Given a value/estimate for all of the above parameters (topology, branch lengths, Q matrix, stationary frequencies, α) one can compute the likelihood of the tree by applying Felsenstein's pruning algorithm [10]. Initially, one needs to place a virtual root anywhere into the tree to define a direction for the tree traversal to compute the likelihood. As the model defined by Q is time-reversible, the likelihood score is invariant with respect to the placement of the virtual root. An important assumption of the PLF is that sites evolve independently. This is important with respect to parallelization as all per-MSA site likelihood scores can, in principle, be computed simultaneously.

Given the virtual root, one conducts a post-order traversal of the tree to calculate the so-called Conditional Likelihood Vectors (CLVs) at each inner node of the tree. A CLV summarizes the signal coming from the left and right subtree of an inner node. For each MSA site i a CLV contains a CLV entry that stores the probability of observing each state of the model (e.g., 4 probabilities for DNA data) conditional on the subtree that is rooted by that CLV. For DNA data, an entry i of a CLV parent node u given two-child CLVs v, w and corresponding branch lengths $b_{u,v}$ and $b_{u,w}$ is computed as follows:

$$\mathbf{CLV}_A^{(u)}(i) = \Big(\sum_{S=A}^{T} P_{AS}(b_{u,v}) \mathbf{CLV}_S^{(v)}(i) \Big) \Big(\sum_{S=A}^{T} P_{AS}(b_{u,w}) \mathbf{CLV}_S^{(w)}(i) \Big) \tag{1.1}$$

Note that the CLV vectors at the tips have a special form (if an A is observed at input sequence position i then the respective CLV entry is set to $CLV_A(i) := 1.0$ and all other entries $CLV_C(i) := CLV_G(i) := CLV_T(i) := 0.0$) if discrete input data is given, which represents the typical use case (but see Sect. 1.3.1). This allows for additional optimizations of the CLV calculations (see Sect. 1.4) that have one or two-child CLVs that are tips. Also, the space for storing CLVs at inner nodes as well as the floating point computations given in Eq. 1.1 largely dominate computational resource requirements of likelihood-based analyses. For instance, the memory footprint for scoring a single tree under ML for the large bird genome dataset [17] amounted to 1TB. The remainder of this chapter focuses on algorithmic and engineering approaches to reduce the computational resource requirements of this equation. Once the CLVs $CLV^{(right)}$, $CLV^{(left)}$ to the left and the right of the virtual root have been computed (i.e., the full tree traversal has been completed) one can compute the per-site likelihood $l(i)$ at site i by applying the following equation:

$$l(i) = \sum_{R=A}^{T} \Big(\pi_R \mathbf{CLV}_S^{(left)}(i) \sum_{S=A}^{T} P_{RS}(b_{left,right}) \mathbf{L}_S^{(right)}(i) \Big) \tag{1.2}$$

The overall likelihood score is then computed by summing over the logarithms (for avoiding numerical underflow) of the $l(i)$:

$$LnL = \sum_{i=1}^{m} log(l(i)) \tag{1.3}$$

As per-site likelihoods can be computed independently and thus concurrently, the calculation of the overall LnL across alignment sites represents the only point in the parallel program flow where cores need to communicate with each other via a classic parallel reduction operation (see Sect. 1.6.2).

One of the key difficulties when implementing ML methods is to devise stable numerical optimization functions (e.g., Brent's method [5], Newton–Raphson method [26], BFGS [11], etc.) for optimizing the parameters in the Q matrix, the α shape parameter, and the branch lengths. For some of these parameters (e.g., branch lengths), it is known that they can exhibit local optima [8]. Cases where the estimates approach the allowed numerical minimum or maximum values are also difficult to cover and handle. Numerical and mathematical approaches for reducing the computational cost of branch length optimization (numerical optimization of Eq. 1.2 with respect to the branch length $b_{left,right}$) are outlined in the chapter by Guindon and Gascuel in this volume, and also in the supplementary material of [13]. The chapter by Gascuel and Guindon in this volume also covers numerical stability issues of optimizers for mixture models.

These numerical difficulties also apply to the design of Markov chain Monte Carlo (MCMC) proposal mechanisms with respect to sampling efficiency *and* the ability to escape local optima of the posterior probability landscape.

1.3 Sequential PLF Optimization via Algorithmic Means

Algorithmic methods for optimizing PLF computations predominantly rely on detection of repeating site patterns in the input data and omitting redundant calculations. Memory-saving approaches are based on omitting storage of such repeating patterns, that is, identical entries in CLVs.

It is important to note that the savings that can be achieved via these techniques are highly implementation dependent. In other words, they depend on how the PLF is applied to trees. In general, the approaches described below perform substantially better for *full* tree traversals, that is, cases where all CLVs of a given phylogenetic tree need to be (re-)computed. This is the case when for instance, the α shape parameter of the Γ model of rate heterogeneity [39] is being optimized (ML) or a new value is being proposed (BI). Full tree traversals are also typically used in methods for divergence time estimation (e.g., mcmctree [40]).

However, tree search algorithms as implemented in MrBayes [28] or, to an even greater extent in RAxML, typically conduct *partial* tree traversals only, that is, after a change in the tree topology only a very small fraction of the CLVs needs to be updated. Moreover, these CLVs are typically located close to the virtual root of the tree that is required for calculating the overall likelihood. This implies that they contain fewer repeating patterns.

1.3.1 Saving Computations

The most straightforward way to save PLF computations for a given multiple sequence alignment (MSA) is through global site pattern compression. Here, exactly identical sites in the input MSA can be compressed into a single- site pattern and be assigned a corresponding weight (number of identical sites represented by this site pattern). This integer weight is then simply multiplied with the per-site log-likelihood score of the pattern to obtain the correct overall log-likelihood score. All current likelihood-based implementations conduct this kind of compression in a preprocessing step. It is worth noting that this compression step can itself be time-consuming for large-scale whole-genome datasets and may even represent a bottleneck in parallel likelihood computations if not handled appropriately (see Sect. 1.6.1).

The key idea for further accelerating likelihood calculations via algorithmic means is to extend this kind of compression to the subtree level, that is, to the subset of MSA sites defined by the taxa that are located in a subtree. This is based on the observation that the closer we are to the tips of the tree, the more identical site patterns we expect to observe in the data and the more unnecessary computations we can hence omit. For example, consider a CLV defining a subtree with two tips as outlined in Fig. 1.1. If the sequences at the tips have an `A` and `C` at MSA positions i and j, respectively, where $i > j$, we can omit calculating (and storing) the CLV entry for MSA site j.

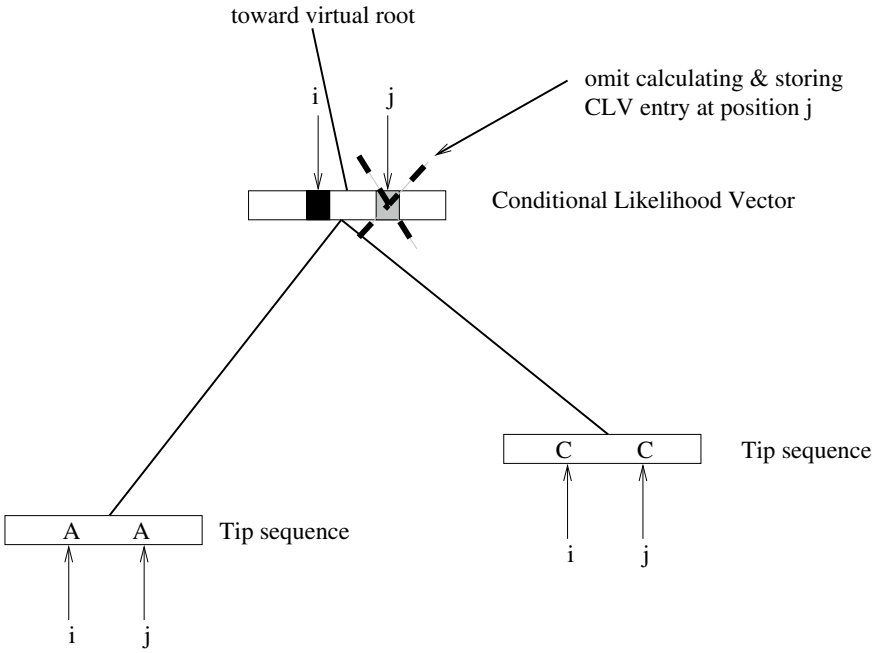


Fig. 1.1 A simple example for a site repeat at positions i and j of the two-child sequences of a CLV

Evidently, the number of such savings (the number of subtree site repeats) depends on the shape of the tree topology whose likelihood is being scored. In addition, savings are greater near the tips of the tree and decrease as we move toward the virtual root. At the virtual root itself, there will be no site repeats at all, as we have already compressed all globally identical sites in the MSA during a preprocessing step. This is also the reason why savings using the site repeats technique for full tree traversals are greater than for partial tree traversals. Moreover, exploiting site repeats on fixed tree topologies that are repeatedly evaluated under different model parameters (e.g., as implemented in FastCodeML [38]) is even cheaper, as the repeating subtree site patterns per CLV only need to be computed only once.

While the idea to detect and omit computations for repeating site patterns in CLVs of subtrees is not new (see [25] and [37] for some early work), the key challenge consisted of designing an appropriate data structure for dynamically updating an index of per-CLV (per-subtree) site repeats for constantly changing trees (e.g., during tree searches). These index and lookup calculations need to be highly efficient, as with increasing vector lengths (128 bit SSE3 instructions and later 256 bit AVX instructions the trade-off margin between detecting repeating site patterns and simply disregarding repeats and computing redundant entries decreases.

A first step in this direction was taken [15] by restraining repeating site patterns to subtree MSA columns consisting entirely of gaps. Such sites often occur in "gappy" supermatrix MSAs (see Fig. 1.3). Typically, patches of gappy data that essentially represent missing molecular data are observed either because (i) specific taxa do not contain a specific gene or (ii) because the specific gene has not been sequenced yet. Restraining the detection of repeating sites to columns consisting entirely of gaps allowed for implementing a simple and thus, highly efficient index data structure for detecting and storing repeating sites that yielded a "good" trade-off. Here, each inner node of the tree is assigned a bit-vector that is as long as the input MSA. A bit i in this bit-vector denotes if the corresponding site i in the subtree below that inner node consists entirely of gaps or not. This also allows to save memory as only CLV entries for non-gap sites need to be stored at inner nodes. The runtime and memory savings that can be achieved were proportional to the amount of missing data in the input MSAs.

For dynamically changing trees, however, these memory savings need to be implemented carefully since some per-node CLV lengths dynamically change every time the virtual root is placed into a different branch as shown in Fig. 1.2. This implies a frequent invocation of `malloc()` and `free()` operations. In the sequential case, the associated overhead is negligible. However, for multithreaded parallel implementations this leads to a bottleneck as each invocation of `malloc()/free()` requires a global lock. Thus, replacing `malloc()` by a corresponding lockless multithreaded implementation yielded a parallel performance improvement of approximately a factor of two (see supplement of [34]).

However, as not all MSAs are sufficiently gappy, especially due to novel sequencing techniques or when genomes of closely related species are being analyzed, there was a need for a more generic solution. As mentioned before, the key challenges consisted in devising an algorithm to rapidly detect subtree site repeats and design-

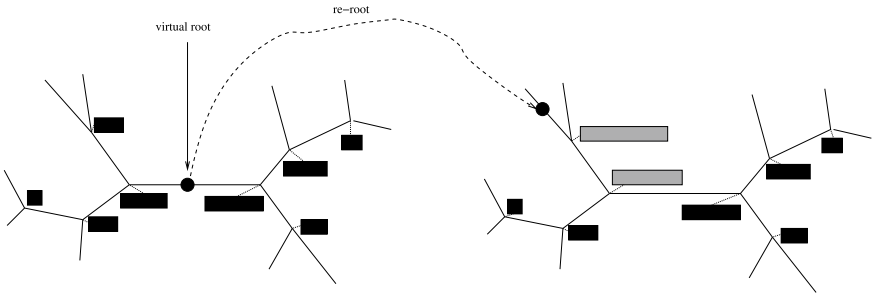


Fig. 1.2 Outline of changing CLV lengths when site repeats are deployed to save memory in a tree that is re-rooted. The lengths of the CLVs in black do not change after re-rooting the tree. The CLV lengths in the shaded boxes *do* change after re-rooting the tree as they will contain less site repeats

ing an appropriate index data structure for storing and looking up repeats. This generic solution was presented in [19] and has now been implemented at production level in RAxML-NG (https://github.com/amkozlov/raxml-ng), Modeltest-NG (https://github.com/ddarriba/modeltest) as well as a low-level phylogenetic likelihood library [12] (https://github.com/xflouris/libpll-2). As expected, site repeats perform best for full tree traversals, but still yield a speedup ranging between a factor of 1.5 and 2 for tree searches in RAxML-NG. Nonetheless, the usage of site repeats might become obsolete as the width of vector registers on x86 systems increases. With the new AVX-512 bit vector instructions, it might be more efficient to carry out redundant calculations than to detect and lookup site repeats. However, such a more straightforward vectorized implementation will not allow for saving memory using the techniques presented here. As we will discuss in Sect. 1.6.3 site repeats also complicate a balanced data distribution for parallel PLF calculations. Finally, site repeats can evidently only be applied to discrete input data, that is, "normal" DNA sequences. There seems to be a trend though toward incorporating uncertainty in the input sequences into the likelihood calculations. Such uncertainty can stem from sequencing error, alignment error, or single-cell sequencing approaches. See Chap. 5 in [20] for some initial models and experiments with uncertain input data. When the input data ceases being discrete site repeat techniques can evidently not be applied anymore.

1.3.2 Saving Memory

As already mentioned above, repeating site patterns at the subtree level can be deployed to substantially reduce the memory requirements for storing CLVs. These savings are directly proportional to the number of repeats in the data.

However, there also exists an orthogonal technique for saving memory in PLF computations that has, unfortunately, not been widely adopted. Initial experiments

with an out-of-core (external memory) algorithm for storing a fraction of the CLVs [16] on disk did not show promising results, albeit the respective algorithm was five times faster than an application-agnostic implementation that relied on the paging mechanism of the operating system.

As a consequence, this direction was abandoned and we investigated trade-offs between storing all CLVs in RAM and recomputing a fraction of CLVs that are not in RAM. This investigation yielded two important results.

First, we showed that for conducting a full tree traversal only the memory for $log(n) + 2$ CLVs needs to be allocated for efficiently computing the likelihood [14], where n is the number of taxa in the MSA. In addition, this procedure does not induce any recomputation cost, that is, requires as many calculations as a full tree traversal with $n - 3$ allocated CLVs (an unrooted phylogenetic tree has $n - 3$ inner nodes). This approach can be combined with the site repeats technique and thereby dramatically reduce the memory requirements for full tree traversals. Moreover, deploying this approach can potentially yield additional speedups as the PLF, in particular for DNA data, is memory bandwidth bound. Thus, a substantially smaller working set could yield improved cache efficiency and, as a consequence, improved performance.

Second, for partial tree traversals, as implemented in RAxML, we were able to develop an efficient strategy for deciding which CLVs to store in RAM and which CLVs to recompute. We showed that storing 50% of the required CLVs only induces a slowdown of 15% due to the required additional CLV recomputations. It must be emphasized though, that this specific slowdown is closely connected to the RAxML search strategy that mostly updates CLVs locally in a neighborhood of the tree. Thus, for less local search strategies or in cases where we cannot control the locality because a stochastic search is conducted (e.g., BI) the associated slowdowns due to recomputations will increase. Nonetheless, we believe that the above techniques are highly useful for accommodating the memory requirements of large datasets or compute as well as memory-intensive complex mixture models such as the C10–C60 family of models [32]. With the exception of IQ-Tree [24] where this technique was used for reducing the memory requirements of the C10–C60 models, we are not aware of any other production-level implementation using this technique. One reason for this might be the increased code complexity and the associated decrease in software maintainability.

1.4 Sequential Optimization via Technical Means

Apart from algorithmic approaches, there are also mostly standard (i.e., implemented in most popular likelihood-based codes) technical approaches to accelerate PLF calculations.

1.4.1 Standard Techniques: Tip–Tip and Tip–Inner Optimizations

A standard technique used in many likelihood-based tools is to use distinct versions of the CLV update kernel depending on the subtree type that is rooted by a CLV.

Developers typically use 3 distinct CLV update kernels. The *tip–tip* kernel is used for calculating CLVs that have two tips (leaves) as children. As the two-child CLVs are normally simply discrete sequences with a limited number of states, a plethora of optimizations can be applied (including lookup tables with precomputed values) to accelerate the CLV computation. The same holds for the *tip–inner* case where one of the two children of the CLV to be computed is a tip. Note that the tip–tip and tip–inner cases account for roughly 50% (evidently, this also depends on the tree shape) of all CLVs that need to be computed during a full tree traversal. Thus, the programming effort for implementing dedicated PLF kernel functions for these cases is justified. With respect to kernel execution times, tip–tip is faster than tip–inner which, in turn, is faster than the general inner–inner case.

The main problem with this approach is that developers typically also implement dedicated PLF kernel functions for each of the most common data types such as DNA, protein, and binary data. This induces increased code complexity and an associated decrease in maintainability.

1.4.2 Vectorization

The use of x86 vector instructions (via 128-bit SSE3 and 256-bit AVX intrinsics) for accelerating PLF calculations is now also common in most widely used likelihood-based tools. Vectorized kernels are also offered by the two libraries for PLF calculations (BEAGLE [2] and PLL [12]). The chapter by Guindon and Gascuel (this volume) provides a more formal description of PLF vectorization.

The main challenge we see consists in the transition to even wider vector widths as x86 CPUs with 512-bit vector instructions are already available now. The reason why this represents a challenge is that the vectorization approach will need to be changed, at least in some implementations, for accommodating wider vectors. Thus far, we vectorized PLF calculations for DNA data on a site-per-site basis. In other words, the per-site log likelihood can be computed independently for each site in a vectorized fashion. With increasing vector widths, this will not be possible anymore. Thus, PLF calculations need to be vectorized across several sites (as implemented in IQ-Tree since version 1.5) to avoid sacrificing performance. The main drawback of this is that the site repeats technique described in Sect. 1.3 will be difficult to apply as it requires each entry i of a CLV corresponding to an MSA site i to be computed independently in order to attain optimal savings. If we vectorize across sites this is not given anymore and we might be conducting redundant computations. While

we suspect that site repeats need to be abandoned for 512-bit vector instructions due to unfavorable trade-offs, this will most likely be the case for 1024-bit vector instructions that will probably become available in the next five years.

1.5 Partial and Full Terraces in Tree Space

It is common practice to infer phylogenies on concatenated multi-gene MSAs, also called supermatrices. In such datasets, one concatenates the DNA or amino acid data for the n taxa under study from several genes G_1, ..., G_p or even entire transcriptomes/genomes into one large supermatrix, where p is the number of genes, or more generally, the number of disjoint partitions of MSA sites in the dataset. Typically, one estimates a separate set of model parameters (e.g., GTR rates, α shape parameter of the Γ model of rate heterogeneity, branch lengths) for each partition of the supermatrix separately. Supermatrices typically contain patches of missing data (see Sect. 1.3.1 above and in Fig. 1.3), that is, sequence data for a specific taxon is not available for some genes/partitions G_i in the supermatrix.

On partitioned datasets, these patches of missing data can induce a potentially misleading effect on the likelihood surface. Under specific partitioning schemes, model settings, and patterns of missing data, different tree topologies might exhibit exactly the same analytical likelihood score. Note that the numerical likelihood scores might deviate slightly because of round-off error propagation. A tree that has the same likelihood value as at least one other, topologically distinct tree, is said to reside on a terrace in phylogenetic tree space.

A simple example for a terrace induced by the MSA and partitioning scheme provided below is shown in Fig. 1.4.

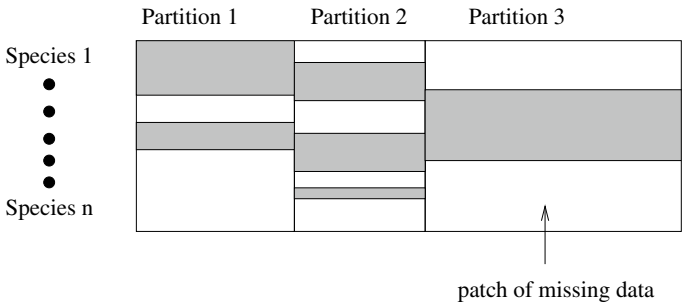


Fig. 1.3 Patches of missing per-partition/gene data in a typical concatenated supermatrix

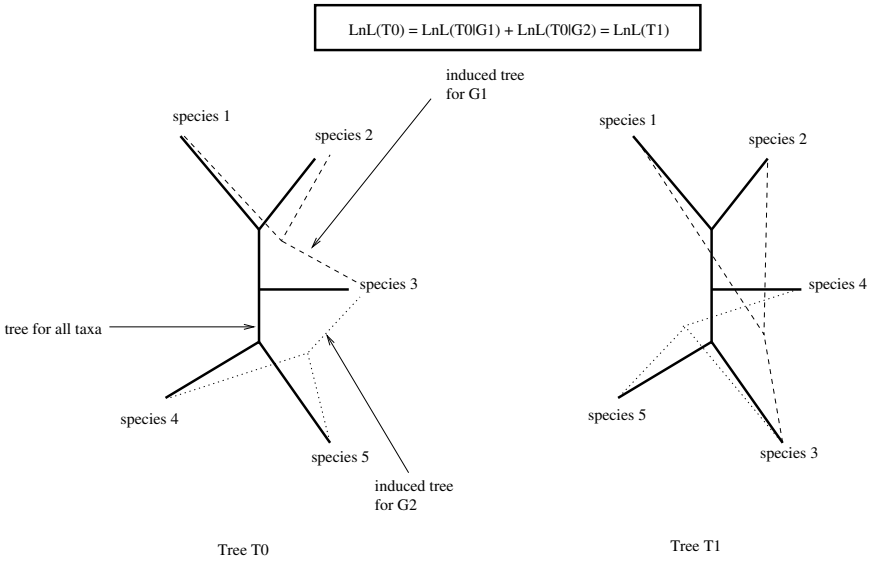


Fig. 1.4 Two topologically distinct tree topologies T_0, T_1 that have exactly identical analytical likelihood scores as the respective induced per-partition tree topologies are identical, that is, $T_0|G_1 = T_1|G_1$ *and* $T_0|G_2 = T_1|G_2$. For the example MSA in the text all 15 possible unrooted binary trees for the 5 species reside on the same terrace

```
index      0123

Species 1 AC--
Species 2 AG--
Species 3 ACTT
Species 4 --AG
Species 5 --GG
```

In the example, MSA denotes missing data. MSA sites `0,1` have been assigned to partition G_1 and MSA sites `2,3` have been assigned to partition G_2. Under the model conditions that give rise to terraces, the log likelihood $LnL(T)$ of a tree T is computed as follows: $LnL(T) = LnL(T|G_1) + LnL(T|G_2)$, where $T|G_i$ denotes the tree topology induced by T for the species/sequences in partition i for which we *have* sequence data. In our example the two induced per-gene/partition trees $T|G_1$ and $T|G_2$ each comprise exactly three taxa (`Species1, Species2, Species3` for G_1 and `Species3, Species4, Species5` for G_2). Hence, every partition only induces a 3-taxon tree. Further, there only exists one possible unrooted tree topology with three taxa. Therefore, all 15 possible 5-taxon trees in our example induce identical per-partition trees and therefore reside on a terrace comprising 15 tree topologies. We denote this as a full or comprehensive terrace.

From a computational point of view, the existence of full terraces allows for substantial computational savings as one only needs to evaluate the likelihood score of one tree on the terrace. However, it is not as straight-forward to detect whether an *unrooted* binary tree is located on a terrace or not [3]. As a work-around, one can design topological moves (e.g., Nearest Neighbor Interchange (NNI) moves or Subtree Pruning and Regrafting (SPR) moves) that are terrace-aware. Such moves will detect if the current tree and the new tree that was obtained, for instance, via a NNI or SPR move reside on a terrace. The current and new tree does reside on a terrace of none of the induced per- partition tree topologies $T|G_i$ was altered by the topological move.

There is additional potential for computational savings by taking the so-called partial terraces into account. A tree resides on a partial terrace if *some* induced per-partition tree topologies $T|G_i$ are invariant under the topological move. As a consequence, the partition likelihood of these unaltered partitions does not need to be recomputed. Apart from the computational savings, this approach also allows for saving memory as only the CLVs of the induced per-partition trees (i.e., the induced tree for which data is available) need to be stored. These induced trees typically have substantially less inner nodes and hence CLVs than the comprehensive tree on which the tree search is conducted. Thus, taking terraces into account allows for memory savings that are proportional to the fraction of missing data in the supermatrix. The computational savings are difficult to characterize or predict mathematically as they depend on the pattern of missing data as well as the terrace size, the heuristic search strategy, and the distribution of partial as well as full terraces in tree space. Experiments reporting empirical run time improvements can be found in [6, 7, 35].

The concept of exploiting full and partial terraces to accelerate likelihood-based tree searches was first implicitly used by Stamatakis and Alachiotis [35] in 2010. In 2011 terraces were mathematically characterized [30]. Some additional properties of terraces, in particular, their impact on support values, are discussed in [29]. More efficient and elegant data structures than in [35] for conducting terrace-aware tree searches were presented in [6, 7]. Due to the increased code complexity the only production-level implementation of terrace-aware moves the author is aware of is provided in IQ-Tree [24].

1.6 Parallel PLF Computations

Apart from algorithmic and technical means for accelerating the PLF a substantial amount of research effort has focused on parallelizing the PLF. The fundamental approach to simply parallelize computations across MSA sites is relatively straightforward. However, as we will outline below, partitioned datasets, algorithmic optimizations such as site repeats, and pure scalability issues require attention and partially rather involved algorithms for attaining optimal data distribution.

1.6.1 Preprocessing and Parallel I/O

For analyzing extremely large empirical datasets on supercomputers (e.g., [17, 22]), we encountered several problems for the first time. One, that can easily be solved is that of parsing and error-checking the MSA which constitutes a hard-to-parallelize process across thousands of cores. The solution for improving scalability is to "cheat a bit" by conducting the checking and parsing (including MSA site pattern compression) in a standalone tool on a server, prior to launching the actual massively parallel likelihood calculations on a supercomputer. This also allows to potentially compress the input data and generate a binary input file that can then be concurrently read by the MPI processes carrying out the likelihood-based search. In the case of ExaML, this approach yielded a reduction of start-up times (I/O, parsing, data distribution), that is of wasted CPU time, by one order of magnitude (see supplement of [21]). What is currently lacking is the specification of a common binary MSA and partition file format for such large-scale phylogenetic inferences.

1.6.2 Parallelization Approaches

As already mentioned, the standard approach consists in parallelizing likelihood computations across sites. For attaining "good" parallel efficiency, this evidently requires the MSA to be long enough. As a rule of thumb, each core needs to work on 1,000–2,000 sites for "good" scalability on DNA data, for instance. This number might increase with improving single core performance (e.g., increasing vector widths). Unfortunately, despite repeated attempts by us and others, there is no scalable solution available for analyzing datasets with thousands of taxa but only a few hundred sites (e.g., large single-gene 16S MSAs). We believe that this is due to the intrinsic sequential dependencies between subsequent tree search steps in most common ML and BI search algorithms.

PLF computations, especially for DNA data, that only require a few floating point operations per-CLV entry while linearly filling the CLV, are memory bandwidth bound. Because of this, we observed impressive super-linear speedups (parallel efficiency of 140% [20]) on large-scale datasets with the hybrid PThreads/MPI version of RAxML-NG.

Another key to improving parallel performance is to avoid the classic master–worker or fork–join approach. In this approach, a dedicated master process/thread will execute the search algorithm and propose new model parameters while the worker threads/processes calculate the PLF. In large partitioned analyses, this can induce a substantial communication bottleneck as the highly frequent collective communication calls (over 1,000 per second) that will be invoked by the master might actually become network bandwidth-bound instead of latency-bound. Consider the example of a large, partitioned dataset with 10,000 partitions. If the search algorithm is in a phase where it optimizes the α shape parameter of the Γ model of rate het-

erogeneity, 10,000 new double precision values will need to be broadcast in each round of the corresponding numerical optimization routine. Note that it is important to optimize all per-partition parameters simultaneously for achieving good parallel efficiency [36]. Given the above, a master–worker approach can easily become network bandwidth-bound using this approach.

To resolve this issue, one can deploy a classic massively parallel approach [33]. Here, each process locally executes a consistent and synchronized copy of the search algorithm while operating only on a part of the data. Consistency can be maintained by ensuring that all processes always have the global likelihood score for the entire MSA by using `MPI_Allreduce()` calls. Apart from increased computational efficiency, this also substantially decreases the code complexity of parallel likelihood implementations, as, apart from a data distribution routine and inserting around a dozen of `MPI_Allreduce()` calls, the sequential code does not need to be changed. For instance, deriving the initial version of ExaML from RAxML only took a day of programming.

1.6.3 Data Distribution Algorithms

While data distribution for a per-MSA site parallelization might appear trivial at first sight (i.e., just distribute m sites to p cores by assigning m/p sites to each core), it is not. First, partitioned datasets complicate the issue substantially. Second, the introduction of site repeats (see Sect. 1.3) means that the per-site computation cost/time can vary substantially among sites *and* that assigning sites that share a large fraction of repeats to distinct cores will increase the computational cost of those sites.

Let us first consider the data distribution problem for partitioned datasets. Without site repeats the computational cost of a partition is proportional to the number of sites. However, there is one additional cost factor, namely the calculation of the transition probability matrix P (see Sect. 1.2) that has a constant cost ρ for each partition. If the sites of a partition are split among two processors, the constant cost ρ for calculating P needs to be paid twice, once at each processor. Note that calculating all r transition probability matrices P for a partitioned dataset with r partitions in parallel and then broadcasting them prior to the actual CLV calculations does not represent a viable solution for avoiding redundant calculations of P. This is because the induced synchronization and communication overhead is too large. In addition, the degree of parallelism is limited by the number of partitions (e.g., analyzing a dataset with 100 partitions on 1,000 cores). While the cost of calculating P is too low for implementing a respective dedicated parallel processing step, the cost of redundant calculations of P can have a substantial negative impact on performance [41]. To address this problem, we initially considered the problem of assigning entire monolithic partitions to cores such that the maximum number of sites assigned to a core is minimized [41]. This guarantees that all per-partition P matrices are only calculated once and at one single core. We found that this data distribution problem is equivalent to the "classic" multiprocessor scheduling which is known to be NP-hard. We, therefore,

tested various well-established heuristics for multiprocessor scheduling. We showed that optimizing the data distribution in this way yields parallel run time improvements by up to one order of magnitude.

Nonetheless, this kind of data distribution strategy might not yield optimal results, in particular when there is a large variance in partition lengths. Consider the case of 100 processors and 100 partitions. Further assume that one "bad" partition is 10,000 sites long while the remaining 99 "good" partitions are just 1,000 sites long. In this case the "bad" partition will be entirely assigned to one core which will slow down all other cores as it has to conduct one order of magnitude more computations. This will result in suboptimal parallel efficiency. Moreover, partitions do not need to be assigned to cores monolithically, but can, as already mentioned above, be split among processors. Evidently, such a splitting will induce redundant calculations of P. We, therefore, obtain a bi-criterion optimization problem: Distribute partitions to cores such that (i) the total number of sites per-core is balanced (each core operates on the same number of sites) and (ii) the number of redundant calculations of P by splitting partitions among cores is minimized. We were able to show that this problem is NP-hard [18]. In addition, we developed an approximation algorithm with a very tight bound. The data distribution computed by our approximation algorithm only incurs one redundant calculation of P more than the optimal solution in the worst case. We also implemented this algorithm in our dedicated BI (ExaBayes [1]) and ML (ExaML [21]) phylogenetic inference tools for supercomputers. We were able to show that this data distribution strategy performs best in all cases compared to previous data distribution strategies because it is independent of the given partition length distribution. In the best case, this near-optimal data distribution yielded parallel run time improvements of a factor of 6. This is "classic" algorithm engineering work as it combines theory, practice, and implementation following Bernard Moret's spirit.

As already mentioned in Sect. 1.3.1, when using site repeats, the above algorithm cannot be applied, since (i) the cost of computation is not linear to the number of sites anymore, (ii) the cost of computation depends on the actual tree shape (the number of repeats varies among trees), and (iii) splitting a partition among cores might yield an increased computational cost for that partition as some repeats between sites assigned to distinct cores are lost. Initial work [23, 31] on developing dedicated data distribution heuristics that do take into account site repeats, has shown that the number of repeats only varies slightly among distinct tree topologies and, that repeats need to be taken into account for attaining good load balance. This is particularly true when the fraction of repeats varies substantially among partitions.

A nice property of the problem is that it is easy to compute a lower bound for the data distribution heuristics as this is simply the fraction of site repeats in the sequential case. Thus, one can easily assess how many site repeats are lost due to a given data distribution. We suspect that the corresponding formal data distribution problem is NP-hard. In addition, the problem could be analyzed in a more theoretical context by transforming it into a graph partitioning problem and applying the broad graph algorithm machinery to it.

1.7 Open Problems and Future Challenges

There are several future challenges for efficient likelihood calculations and related problems as described in this chapter.

One challenge are growing vector widths which might yield the site repeats technique and related algorithmic tricks for accelerating likelihood calculations obsolete. Another question is, if there exist notable differences in energy consumption between AVX (256 bit) with site repeats enabled and AVX-512 with site repeats disabled. This set of alternative PLF kernels could also be extended to include the numerical tricks outlined in the chapter by Guindon and Gascuel (this volume). Thus, apart from pure computation speed, there might be other factors to consider. Also, given the entire zoo of alternative likelihood kernel implementations (e.g., various memory-saving techniques, site repeats, terrace-aware moves), a dynamic selection of the most efficient kernel for specific tools or even distinct algorithmic phases (e.g., tree search with partial tree traversals versus parameter optimization with full tree traversals) of phylogenetic inference tools might yield additional gains in efficiency.

A related technical topic is that of parallel fault tolerance. With increasing data set sizes (e.g., an insect transcriptome dataset that is one order of magnitude larger than the one published in *Science* in the late 2014 [17]) and improving scalability of tools, processor failures in large computations will become more common. Some key questions to be answered in this context are (i) if a reliable fault-tolerant version of `MPI_Allreduce()` is available and (ii) how failures can best be recovered. Regarding the latter, one would not rely on standard checkpointing but consider the trade-offs between lightweight in-memory rollback mechanism versus only recomputing the computations that failed. This also gives rise to interesting theoretical questions. When a core fails, the data assigned to this core needs to be redistributed to the remaining cores. This redistribution cost could be alleviated by a redundant data distribution algorithm that might be able to handle a certain number of processor failures without having to reread data from disk.

There are also some other open theoretical questions. The algorithm [9] for counting and enumerating trees on a terrace requires rooted trees as input. However, phylogenetic trees are typically unrooted and the algorithm can currently only be applied to unrooted per-partition trees $T|P$ if they all have at least one taxon in common that allows for a consistent rooting. However, many empirical datasets do not contain such a common taxon that appears (has data) for all partitions. Another open problem is the theoretical time complexity of data distribution under site repeats.

Acknowledgements The author gratefully acknowledges the support of the Klaus Tschira Foundation and the support he received from Bernard Moret over all those years.

References

1. Aberer, A.J., Kobert, K., Stamatakis, A.: ExaBayes: massively parallel Bayesian tree inference for the whole-genome era. Mol. Biol. Evol. **31**(10), 2553–2556 (2014)
2. Ayres, D.L., Darling, A., Zwickl, D.J., Beerli, P., Holder, M.T., Lewis, P.O., Huelsenbeck, J.P., Ronquist, F., Swofford, D.L., Cummings, M.P., et al.: BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. Syst. Biol. **61**(1), 170–173 (2011)
3. Biczok, R., Bozsoky, P., Eisenmann, P., Ernst, J., Ribizel, T., Scholz, F., Trefzer, A., Weber, F., Hamann, M., Stamatakis, A.: Two C++ libraries for counting trees on a phylogenetic terrace. bioRxiv, p. 211276 (2017)
4. Boussau, B., Szöllősi, G.J., Duret, L., Gouy, M., Tannier, E., Daubin, V.: Genome-scale coestimation of species and gene trees. Genome Res. **23**(2), 323–330 (2013)
5. Brent, R.P.: An algorithm with guaranteed convergence for finding a zero of a function. Comput. J. **14**(4), 422–425 (1971)
6. Chernomor, O., von Haeseler, A., Minh, B.Q.: Terrace aware data structure for phylogenomic inference from supermatrices. Syst. Biol. **65**(6), 997–1008 (2016)
7. Chernomor, O., Minh, B.Q., von Haeseler, A.: Consequences of common topological rearrangements for partition trees in phylogenomic inference. J. Comput. Biol. **22**(12), 1129–1142 (2015)
8. Chor, B., Hendy, M.D., Holland, B.R., Penny, D.: Multiple maxima of likelihood in phylogenetic trees: an analytic approach. Mol. Biol. Evol. **17**(10), 1529–1541 (2000)
9. Constantinescu, M., Sankoff, D.: An efficient algorithm for supertrees. J. Class. **12**(1), 101–112 (1995)
10. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**(6), 368–376 (1981)
11. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (1987)
12. Flouri, T., Izquierdo-Carrasco, F., Darriba, D., Aberer, A., Nguyen, L.T., Minh, B., Von Haeseler, A., Stamatakis, A.: The phylogenetic likelihood library. Syst. Biol. **64**(2), 356–362 (2014)
13. Hoang, D.T., Chernomor, O., von Haeseler, A., Minh, B.Q., Vinh, L.S.: UFBoot2: improving the ultrafast bootstrap approximation. Mol. Biol. Evol. **35**(2), 518–522 (2018). https://doi.org/10.1093/molbev/msx281
14. Izquierdo-Carrasco, F., Gagneur, J., Stamatakis, A.: Trading memory for running time in phylogenetic likelihood computations. Heidelberg Institute for Theoretical Studies (2011)
15. Izquierdo-Carrasco, F., Smith, S.A., Stamatakis, A.: Algorithms, data structures, and numerics for likelihood-based phylogenetic inference of huge trees. BMC Bioinform. **12**(1), 470 (2011)
16. Izquierdo-Carrasco, F., Stamatakis, A.: Computing the phylogenetic likelihood function out-of-core. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Ph.D Forum (IPDPSW), pp. 444–451. IEEE (2011)
17. Jarvis, E., Mirarab, S., Aberer, A.J., Li, B., Houde, P., Li, C., Ho, S., Faircloth, B.C., Nabholz, B., Howard, J.T., Suh, A., Weber, C.C., da Fonseca, R.R., Li, J., Zhang, F., Li, H., Zhou, L., Narula, N., Liu, L., Ganapathy, G., Boussau, B., Bayzid, M.S., Zavidovych, V., Subramanian, S., Gabaldón, T., Capella-Gutiérrez, S., Huerta-Cepas, J., Rekepalli, B., Munch, K., Schierup, M., Lindow, B., Warren, W.C., Ray, D., Green, R.E., Bruford, M.W., Zhan, X., Dixon, A., Li, S., Li, N., Huang, Y., Derryberry, E.P., Bertelsen, M.F., Sheldon, F.H., Brumfield, R.T., Mello, C.V., Lovell, P.V., Wirthlin, M., Schneider, M.P.C., Prosdocimi, F., Samaniego, J.A., Velazquez, A.M.V., Alfaro-Núnez, A., Campos, P.F., Petersen, B., Sicheritz-Ponten, T., Pas, A., Bailey, T., Scofield, P., Bunce, M., Lambert, D.M., Zhou, Q., Perelman, P., Driskell, A.C., Shapiro, B., Xiong, Z., Zeng, Y., Liu, S., Li, Z., Liu, B., Wu, K., Xiao, J., Yinqi, X., Zheng, Q., Zhang, Y., Yang, H., Wang, J., Smeds, L., Rheindt, F.E., Braun, M., Fjeldsa, J., Orlando, L., Barker, F.K., Jonsson, K.A., Johnson, W., Koepfli, K.P., O'Brien, S., Haussler, D., Ryder, O.A., Rahbek, C., Willerslev, E., Graves, G.R., Glenn, T.C., McCormack, J., Burt, D., Ellegren, H., Alstrom, P., Edwards, S.V., Stamatakis, A., Mindell, D.P., Cracraft, J., Braun, E.L., Warnow,

T., Jun, W., Gilbert, M.T.P., Zhang, G.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science **346**(6215), 1320–1331 (2014)
18. Kobert, K., Flouri, T., Aberer, A., Stamatakis, A.: The divisible load balance problem and its application to phylogenetic inference. In: International Workshop on Algorithms in Bioinformatics, pp. 204–216. Springer (2014)
19. Kobert, K., Stamatakis, A., Flouri, T.: Efficient detection of repeating sites to accelerate phylogenetic likelihood calculations. Syst. Biol. **66**(2), 205–217 (2017)
20. Kozlov, A.: Models, optimizations, and tools for large-scale phylogenetic inference, handling sequence uncertainty, and taxonomic validation. Ph.D. thesis, Karlsruhe Institute of Technology (2017)
21. Kozlov, A.M., Aberer, A.J., Stamatakis, A.: ExaMl version 3: a tool for phylogenomic analyses on supercomputers. Bioinformatics **31**(15), 2577–2579 (2015)
22. Misof, B., Liu, S., Meusemann, K., Peters, R.S., Donath, A., Mayer, C., Frandsen, P.B., Ware, J., Flouri, T., Beutel, R.G., Niehuis, O., Petersen, M., Izquierdo-Carrasco, F., Wappler, T., Rust, J., Aberer, A., Aspöck, U., Aspöck, H., Bartel, D., Blanke, A., Berger, S., Calcott, B., Chen, J., Friedrich, F., Fukui, M., Fujita, M., P., Gu, S., Huang, Y., Jermiin, L., Kawahara, A., Krogmann, L., Lanfear, R., Letsch, H., Li, Y., Li, Z., Li, J., Lu, H., Machinda, R.Y.M., Kapli, P., McKenna, D., Meng, G., Nakagaki, Y., Navarrete-Heredia, J., Ott, M., Ou, Y., Pass, G., Podsiadlowski, L., Pol, H., von Reumont, B., Schutte, K., Sekiya, K., Shimizu, S., Slipinski, A., Stamatakis, A., Song, W., Su, X., Szucsich, N., Tan, M., Tan, X., Tan, M.G., Tomizuka, S., Trautwein, M., Tong, X., Wilbrandt, J., Wipfler, B., Wong, T., Wu, Q., Wu, G., Xie, Y., Yang, S., Yang, Q.Y.: The timing and pattern of insect evolution. Science **346**(6210), 763–767 (2014)
23. Morel, B., Flouri, T., Stamatakis, A.: A novel heuristic for data distribution in massively parallel phylogenetic inference using site repeats. In: The IEEE International Conference on High Performance Computing and Communications (HPCC). IEEE (2017)
24. Nguyen, L.T., Schmidt, H.A., von Haeseler, A., Minh, B.Q.: IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol. Biol. Evol. **32**(1), 268–274 (2014)
25. Pond, S.L.K., Muse, S.V.: Column sorting: rapid calculation of the phylogenetic likelihood function. Syst. Bio. **53**(5), 685–692 (2004)
26. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C, 2nd edn. Cambridge University Press, New York (1992)
27. Ripplinger, J., Sullivan, J.: Does choice in model selection affect maximum likelihood analysis? Syst. Biol. **57**(1), 76–85 (2008)
28. Ronquist, F., Huelsenbeck, J.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**, 1572–1574 (2003)
29. Sanderson, M.J., McMahon, M.M., Stamatakis, A., Zwickl, D.J., Steel, M.: Impacts of terraces on phylogenetic inference. Syst. Biol. **64**(5), 709–726 (2015)
30. Sanderson, M.J., McMahon, M.M., Steel, M.: Terraces in phylogenetic tree space. Science **333**(6041), 448–450 (2011)
31. Scholl, C., Kobert, K., Flouri, T., Stamatakis, A.: The divisible load balance problem with shared cost and its application to phylogenetic inference. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 408–417. IEEE (2016)
32. Si Quang, L., Gascuel, O., Lartillot, N.: Empirical profile mixture models for phylogenetic reconstruction. Bioinformatics **24**(20), 2317–2323 (2008)
33. Stamatakis, A., Aberer, A.J.: Novel parallelization schemes for large-scale likelihood-based phylogenetic inference. In: 2013 IEEE 27th International Symposium on Parallel & Distributed Processing (IPDPS), pp. 1195–1204. IEEE (2013)
34. Stamatakis, A., Aberer, A.J., Goll, C., Smith, S.A., Berger, S.A., Izquierdo-Carrasco, F.: RAxML-Light: a tool for computing terabyte phylogenies. Bioinformatics **28**(15), 2064–2066 (2012)
35. Stamatakis, A., Alachiotis, N.: Time and memory efficient likelihood-based tree searches on phylogenomic alignments with missing data. Bioinformatics **26**(12), i132–i139 (2010)

36. Stamatakis, A., Ott, M.: Load balance in the phylogenetic likelihood kernel. In: International Conference on Parallel Processing, 2009, ICPP'09, pp. 348–355. IEEE (2009)
37. Stamatakis, A.P., Ludwig, T., Meier, H., Wolf, M.J.: Accelerating parallel maximum likelihood-based phylogenetic tree calculations using subtree equality vectors. In: ACM/IEEE 2002 Conference on Supercomputing, pp. 1–16. IEEE (2002)
38. Valle, M., Schabauer, H., Pacher, C., Stockinger, H., Stamatakis, A., Robinson-Rechavi, M., Salamin, N.: Optimization strategies for fast detection of positive selection on phylogenetic trees. Bioinformatics **30**(8), 1129–1137 (2014)
39. Yang, Z.: Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J. Mol. Evol. **39**(3), 306–314 (1994)
40. Yang, Z., Rannala, B.: Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds. Mol. Biol. Evol. **23**(1), 212–226 (2005)
41. Zhang, J., Stamatakis, A.: The multi-processor scheduling problem in phylogenetics. In: 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum (IPDPSW), pp. 691–698. IEEE (2012)

Chapter 2
Numerical Optimization Techniques in Maximum Likelihood Tree Inference

Stéphane Guindon and Olivier Gascuel

Abstract In this chapter, we present recent computational and algorithmic advances for improving the inference of phylogenetic trees from the analysis of homologous genetic sequences under the maximum likelihood criterion. In particular, we detail how the use of matrix algebra at the core of Felsenstein's pruning algorithm, combined with the architecture of modern day computer processors, leads to efficient techniques for optimizing edge lengths. We also discuss some properties of the likelihood function when considering the optimization of the parameters of mixture models that are used to describe the variation of rates-across sites.

Keywords Maximum likelihood · Markov processes · Optimization · Mixture models

2.1 Introduction

Software programs for inferring phylogenies are among the most frequently used tools in bioinformatics with thousands of citations every year (e.g., PAUP* [29], PhyML [8, 9], RAxML [26], IQ-TREE [21], MEGA [30], or MrBayes [24], totaling more than 140,000 citations to date). All these methods aim at reconstructing a phylogenetic tree given a multiple alignment of sequences (DNA, proteins, or codons). The first approaches were based on evolutionary distances between pairs of sequences and parsimony [6, 35]. Today, most trees are inferred using approaches that rely on probabilistic models of sequence evolution. A breakthrough in the application of these techniques was the article of Joseph Felsenstein in 1981 [5], describing

S. Guindon
Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, CNRS and Université Montpellier (UMR 5506), Montpellier, France
e-mail: guindon@lirmm.fr

O. Gascuel (✉)
Unité Bioinformatique Evolutive, C3BI Institut Pasteur and CNRS (USR 3756), Paris, France
e-mail: olivier.gascuel@pasteur.fr

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_2

the "pruning" (or "peeling", or "sum-product") algorithm to efficiently compute the likelihood of a tree with branch lengths given a multiple alignment of sequences and a model of sequence evolution along with the parameter values of this model. An essential goal in phylogenetics then became to design algorithms for inferring trees from sequences, using the maximum likelihood (ML) principle. The first algorithms were slow due to the difficulty of the optimization task, which combines combinatorial optimization to select the best tree topology among an exponential number of possible solutions [6], and numerical optimization to estimate branch lengths and substitution model parameters. In addition, these two numerical and combinatorial facets are inseparable and cannot be performed independently. Computers were also slow compared to nowadays high-performance clusters, and most analyses were limited to a few dozens of taxa, the analysis of more than 100 taxa using any model-based approach being extremely rare until the end of the 90s .

In 2003, we published the first fast algorithm to infer trees using the maximum likelihood principle, implemented in PhyML [9]. The idea was to combine within the same algorithm the optimization of branch lengths and the search for the optimal tree topology using multiple simultaneous "Nearest Neighbor Interchanges" (NNIs). These (local) topological moves exchange subtrees separated by three branches. NNIs were performed in a hill-climbing manner, until no more move was able to improve the likelihood of the current tree. The numerical parameters were optimized along the way. Although this first version of PhyML was not performing an extensive tree search, it was fairly accurate and rapidly became popular.

Since then, a number of algorithms have been designed to perform a more extensive exploration of the space of trees. RAxML-III [27] implemented "Subtree Pruning and Regrafting" (SPR) topological moves, where a subtree is pruned from the whole tree and regrafted on another edge. To avoid evaluating all possible SPRs, RAxML concentrates on those moves where the pruned subtree is re-inserted at a limited distance (in number of edges) from its initial position. We proposed alternative solutions, where distance- and parsimony-based filtering was applied to determine the most "promising" SPRs [8, 13]. IQPNNI [32] combined an NNI-based approach similar to that of PhyML with the pruning and regrafting of individual taxa. Several alternative approaches have been proposed since then, to introduce some stochasticity in the hill-climbing scheme [21], maintain a population of solutions that are combined using a genetic algorithm [11, 38], and to use several starting trees to avoid being trapped too early in poor local optima of the likelihood function. Last but not least, parallel versions of these various approaches were designed and implemented in a number of environments, including GPUs, thanks to dedicated libraries [2, 23]. The chapter by Stamatakis in this book reviews these advanced computing methods and their implementations, which today make it possible to analyze huge data sets comprising dozens to thousands of sequences and whole genomes in a few large-scale studies (e.g., [14]) or in one of Bernard Moret's important work on the topic [20]).

However, while a large number of articles describe the combinatorial side of phylogenetic inference, aimed at exploring the tree space, very few articles and book chapters deal with the numerical side, which forms the basis of all ML approaches.

The goal of this chapter is to review the main advances in numerical computation and optimization, which makes it possible to analyze large data sets nowadays. In the following, we first present the basic statistical principles underlying models of genetic sequence evolution. We then show how the pruning algorithm can be accelerated in the context of individual edge length optimization through proper algebraic factorization of the various matrix components of the models. The impact of these progresses on branch length optimization is presented, along with other ideas to accelerate the inference under mixture models that are very popular in phylogenetics. These advanced numerical methods are implemented in recent versions of PhyML and also in similar forms in other software programs, most notably RAxML and IQ-TREE.

2.2 Modeling Sequence Evolution

Molecular sequences lend themselves very well to probabilistic modeling. In fact, the building of mechanistic models of molecular evolution is at the core of major advances in our understanding of evolution and biological diversity in general. There are mainly two reasons to that. First, genetic sequences are abundant. The rise of ever-improving modern sequencing technologies makes it incredibly fast and cheap to sequence complete genomes nowadays compared to two decades ago. Large amounts of data limit the impact of sampling errors, thereby giving the opportunity to fit realistic models of evolution and derive precise parameter estimates. Second, genomes are made of very simple building blocks, i.e., the four nucleotides (adenine, cytosine, guanine and thymine, or A, C, G, and T) so that the state space of the proposed models is generally well-defined, as explained below.

A typical data set suitable for phylogenetic analysis consists in an alignment of homologous nucleotide or amino acid sequences collected in multiple individuals. Depending on the evolutionary time-scale of interest, these individuals may belong to distinct species (when one is interested in ancient evolutionary events) or to the same population (when one focuses on recent events). Two or more sequences are said to be homologous whenever they have a common origin, i.e., these contemporaneous sequences all descend from a single ancestral sequence. Homologous sequences are aligned beforehand so that each site, i.e., each column of the alignment, consists in homologous nucleotide or amino acid characters.

As is the case with most if not all probabilistic models used in biology, stochastic models of sequence evolution rely on several simplifying assumptions. First, the different sites of the alignment are assumed to be independent realizations of the same data-generating process. In the statistical jargon, sites are said to be i.i.d., i.e., independent and identically distributed. The i.i.d. assumption is very convenient as the modeling then takes place at the individual-site level. The state-space of the models of interest thus become the set of four nucleotides (61 coding triplets of nucleotides when analyzing coding sequences using codon models) or 20 amino acids, depending on the type of data under scrutiny. The model describes how the

random accumulation of substitutions between these limited number of states during the course of evolution gave rise to the observed data.

Let $\mathscr{A}$ be the state-space of interest and $w \in \mathscr{A}$ a particular state. We consider that w is the realization of the random variable $W(t)$ which returns the state a given sequence is in at time t, at a particular site. We assume that the time to the next substitution event, where state w is replaced by another state, distinct from w, is exponentially distributed. More precisely, we assume that $|\mathscr{A}| - 1$ exponential clocks are running in parallel, each clock being associated to a state distinct from w. The timing of the next substitution event is given by the minimum of $|\mathscr{A}| - 1$ exponential draws. Let r_{wv} be the rate of substitution from state w to v. The time to the next substitution event is thus exponential with parameter $r_w := \sum_{v \neq w, v \in \mathscr{A}} r_{wv}$.

The simplest models assume that the substitution rates between characters are all equal. It is indeed the case with the first Markov model of substitution between nucleotides, proposed by Jukes and Cantor in 1969 [16]. In that particular situation, $r_{wv} = 1/3$ for all $w \neq v \in \mathscr{A}$, such that $r_w = r_v$ and the number of substitutions in a time interval t is described by a Poisson distribution with parameter rt, where r is the (common) rate of substitution "away" from any state (i.e., $r = r_w = r_v$). Let $\mathbf{R}$ be the one-step transition probability matrix between characters, i.e., $\mathbf{R}_{wv} = 1/3$ if $v \neq w$ and $\mathbf{R}_{wv} = 0$ otherwise, for the Jukes and Cantor model. The matrix $\mathbf{P}(t)$ of transition between states over a period of time t is thus expressed as follows:

$$\mathbf{P}(t) = \sum_{n=0}^{\infty} \mathbf{R}^n \frac{(rt)^n e^{-rt}}{n!}, \tag{2.1}$$

where the sum is over the (Poisson distributed) number of substitutions that could have occurred in t. We thus have:

$$\begin{aligned} \mathbf{P}(t) &= e^{-rt} \sum_{n=0}^{\infty} \frac{\mathbf{R}^n (rt)^n}{n!} && (2.2)\\ &= e^{-rt} \sum_{n=0}^{\infty} \frac{(\mathbf{R}rt)^n}{n!} && (2.3)\\ &= e^{-rt} e^{\mathbf{R}rt} && (2.4)\\ &= e^{(\mathbf{R}-\mathbf{I})rt}. && (2.5) \end{aligned}$$

The matrix $\mathbf{Q} := (\mathbf{R} - \mathbf{I})r$ is the infinitesimal generator of the Markov process. It gives the rates at which the different substitutions take place.

Not all models satisfy the constraint $r_w = r_v$ however. In fact, the most popular substitution models do not satisfy this constraint when fitted to most data sets. Nonetheless, the equality $\mathbf{P}(t) = e^{\mathbf{Q}t}$ still holds. A relatively straightforward way to verify that this is true indeed is to consider now that substitutions are described by a Markov process that authorizes self-substitutions, i.e., substitutions between identical states. Substitutions, including self-substitutions, are then governed by a Poisson

process with parameter λ, taken as any real value greater than r_w for all $w \in \mathscr{A}$. The one-step transition probability matrix of this new process is obtained by letting

$$\widetilde{\mathbf{R}} := \mathbf{I} + \lambda^{-1}\mathbf{Q}. \tag{2.6}$$

With this new process, the transition probabilities are given by

$$\widetilde{\mathbf{P}}(t) = \sum_{n=0}^{\infty} \widetilde{\mathbf{R}}^n \frac{(\lambda t)^n e^{-\lambda t}}{n!}, \tag{2.7}$$

and one can verify that $\widetilde{\mathbf{P}}_{vw}(h) = \mathbf{Q}_{vw}h + \mathscr{O}(h) = \mathbf{P}_{vw}(h)$ and $\widetilde{\mathbf{P}}_{ww}(h) = \mathbf{Q}_{ww}h + \mathscr{O}(h) = \mathbf{P}_{ww}(h)$ as $h \downarrow 0$, thus showing that the two processes, the one authorizing self-substitutions and the other that does not, have the same probability law $\Pr(W(t_1) = w_1, W(t_2) = w_2, \ldots, W(t_k) = w_k)$ for states $w_i \in \mathscr{A}$ and times t_i satisfying $0 \leq t_1 < t_2 < \ldots < t_k$.

Nucleotide and codon substitution models are "mechanistic" in the sense that the different parameters involved in the rate matrix correspond to particular biological properties. For instance, transitions and transversions generally have distinct rates in order to reflect the specificities of the biochemical processes involved in point mutations. Similarly, codon models distinguish between non-synonymous and synonymous substitutions in order to accommodate for the potential impact of natural selection acting on coding sequences. In contrast, models of substitutions between amino acids are constructed in an empirical manner. It is computationally infeasible and generally risky from a statistical perspective to estimate substitution rates from the analysis of individual data sets. In fact, precise and accurate estimation of 190 (i.e., $20 \times 19/2$) substitution rate parameters require large numbers of sequences, displaying reasonable amounts of divergence. Rather than estimating these rates on a routine basis as is done with nucleotide models, relative rates of substitutions were estimated from the analysis of data bases of multiple homologous sequence alignments [4, 15, 18, 33] using ad hoc inference techniques. The rate matrices hence obtained are then used "as is" in the analysis of individual data sets, keeping the rate parameters fixed throughout the inference.

The vast majority of substitution models are homogeneous, i.e., a single rate matrix applies everywhere in the phylogeny. As a consequence, these models are also stationary, i.e., the vector of state frequencies $\boldsymbol{\Pi}$ that satisfies $\boldsymbol{Q\Pi} = \boldsymbol{\Pi}$ is the same everywhere in the tree. This vector also corresponds to the long-term (or limiting) frequency the substitution process spends in each state. We refer the reader to [7] for further detail on the statistical properties of these models, although some of these properties are also described in subsequent sections of the present chapter.

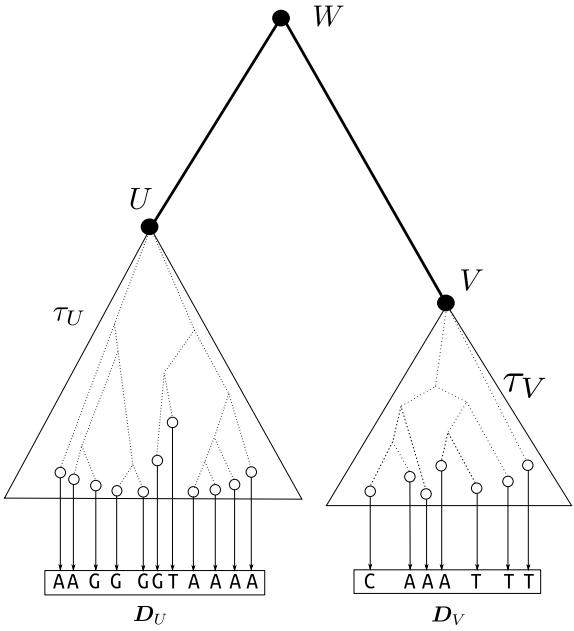


Fig. 2.1 Elements of notation. W is the root of the binary tree. The observed data corresponds to $D_U \cup D_V$. τ_U and τ_V are the topologies of the two subtrees rooted at nodes U and V, respectively

2.3 Matrix-Based Calculation of the Likelihood Function

Figure 2.1 serves as a basis to introduce the notation used in this chapter. We consider a binary tree with node W as its root. Nodes U and V are sister nodes that are the direct descendants of W. τ_W (or, simply, τ) denotes the topology of the whole tree while τ_U and τ_V are the topologies of the subtrees rooted with nodes U and V, respectively. We assume that the substitution process giving rise to a datum at the tips of the tree, noted as $\boldsymbol{D}_W$, or simply $\boldsymbol{D}$, is a continuous-time Markov chain. As before, the state space for this Markov process is denoted as $\mathscr{A}$ and corresponds to the set of four nucleotides or twenty amino acids, depending on the type of sequences under scrutiny. Let $\boldsymbol{D}_U$ and $\boldsymbol{D}_V$ be the discrete character states observed at the tips of the subtrees rooted by U and V, respectively, so that $\boldsymbol{D} = \boldsymbol{D}_W = \boldsymbol{D}_U \cup \boldsymbol{D}_V$. The Markov process is homogeneous and stationary with $\boldsymbol{\Pi}$ the column vector of equilibrium frequencies of character states. $\boldsymbol{P}(l_U)$ is the transition probability matrix on the edge of length l_U between nodes W and U. $\boldsymbol{P}(l_V)$ and l_V are defined in a similar fashion. In a slight abuse of notation, we will also use W to denote the random variable giving the character state at that node. Hence, $W = w$ indicates that the state $w \in \mathscr{A}$ is observed at node W. Finally, let $\mathscr{M}$ denote an instance of the phylogenetic model. $\mathscr{M}$ is a composite set of parameters that includes the tree topology and the set of substitution matrices on every edge of that tree. Note that these substitution matrices are not independent since they derive from a unique rate matrix (e.g., HKY [10] or LG [18]) corresponding to the generator of the Markov process. As will be explained later

in this chapter, mixture models apply several potentially independent rate matrices on all edges of the tree, each matrix applying with a given probability.

The likelihood of the phylogenetic model for a given datum is as follows:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \boldsymbol{\Pi}^t \times \boldsymbol{L}_W, \tag{2.8}$$

where $\boldsymbol{L}_W$ is the column vector $(\Pr(\boldsymbol{D}|W = \cdot, \tau_W)) \in [0, 1]$, i.e., the vector of likelihoods conditioned on the state observed at node W and the current phylogenetic tree τ_W (i.e., the whole phylogeny τ), where τ_W can also be considered as the "union" $\tau_U \cup \tau_V$ (plus the edge connecting nodes U to W and that between V and W). The calculation of $\boldsymbol{L}_W$ can be done in a recursive manner, involving the equivalent vectors of conditional likelihoods at nodes U and V. We then have:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \boldsymbol{\Pi}^t \times \big((\boldsymbol{P}(l_U) \times \boldsymbol{L}_U) \circ (\boldsymbol{P}(l_V) \times \boldsymbol{L}_V)\big), \tag{2.9}$$

where "$\circ$" denotes the Hadamard product between two matrices, i.e., $(\boldsymbol{A} \circ \boldsymbol{B})_{vw} = (\boldsymbol{A})_{vw}(\boldsymbol{B})_{vw}$, while the symbol "$\times$" corresponds to the standard product between matrices/vectors. $\boldsymbol{L}_U := \Pr(\boldsymbol{D}_U|U = \cdot, \tau_U)$ is the likelihood of the phylogenetic model defined by the subtree τ_U. $\boldsymbol{L}_V$ is defined in a similar manner.

Most Markov models used in phylogenetics are time-reversible. For this particular class of models, we have $\boldsymbol{\Pi}_w \boldsymbol{P}_{wv} = \boldsymbol{\Pi}_v \boldsymbol{P}_{vw}$. In matrix notation, we thus have $(\boldsymbol{\Pi} \times \mathbf{1}^t) \circ \boldsymbol{P} = (\mathbf{1} \times \boldsymbol{\Pi}^t) \circ \boldsymbol{P}^t$, where $\mathbf{1}$ denotes a column vector of all ones of length $|\mathscr{A}|$. Using standard rules of matrix algebra and the properties of Hadamard products plus the reversibility equality aforementioned, we rewrite Eq. 2.9 as follows:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)^t \times \boldsymbol{P}(l) \times \boldsymbol{L}_V, \tag{2.10}$$

where $\boldsymbol{P}(l)$ is the transition probability matrix along an edge of length $l = l_U + l_V$.

2.4 Likelihood Calculation and Pruning Algorithm Using Vector Operations Only

The pruning algorithm is a recursive method where vectors of conditional likelihoods are computed in a post-order tree traversal. Taking the tree in Fig. 2.1, we have:

$$\boldsymbol{L}_W = (\boldsymbol{P}(l_U) \times \boldsymbol{L}_U) \circ (\boldsymbol{P}(l_V) \times \boldsymbol{L}_V), \tag{2.11}$$

where both $\boldsymbol{L}_U$ and $\boldsymbol{L}_V$ are computed using the conditional likelihood vectors found on the pairs of nodes corresponding to the immediate descendants of U and V, if any. The recursion is initialized by instantiating the conditional likelihood vectors at the tips using the data observed in the corresponding sequences.

Evaluating the likelihood function then relies on two types of core computations corresponding to the calculation of conditional likelihood vectors using Eq. 2.11 and the calculation of the likelihood itself using Eq. 2.10. These calculations involve the standard and Hadamard matrix product between matrices and vectors. In practice, these two operations are generally performed using series of core operations involving scalars. For instance, the Hadamard product between two matrices could be obtained by taking each element of the first matrix and multiplying it by the corresponding element in the second matrix. The same applies to the standard product between two matrices, although this operation also requires adding scalars.

The present section explains how Hadamard and standard product applied to matrices and vectors can also be conducted using core operations that involve vectors instead of scalars. As detailed in the chapter by Stamatakis in the present book, modern computers provide tools to perform standard operations on vectors whereby adding or taking the dot-product between pairs of vectors entails the same computational load as adding or multiplying two scalars, thereby giving the opportunity to considerably speed up matrix operations.

The pruning algorithm requires multiplying a transition matrix, noted as $\boldsymbol{P}$, by a vector $\boldsymbol{L}$ of conditional likelihoods (see Eq. 2.11). Considering the product $\boldsymbol{P} \times \boldsymbol{L}$ naively, one would multiply each element on the first row of $\boldsymbol{P}$ by the corresponding element in $\boldsymbol{L}$ and add the $|\mathscr{A}|$ scalars hence obtained in order to work out the first element of the vector $\boldsymbol{P} \times \boldsymbol{L}$. The same series of operations would be performed for the second row of $\boldsymbol{P}$ and so on. Yet, this approach is not fully satisfactory if one wants to deal with vectors only. Indeed, the series of operations performed by multiplying the i-th row of $\boldsymbol{P}$ by $\boldsymbol{L}$ results in a scalar, i.e., the i-th element of the vector resulting from $\boldsymbol{P} \times \boldsymbol{L}$. Our objective here is that every operand in our calculations and the corresponding results are all vectors.

One way to make sure vectors are used throughout is to rewrite $\boldsymbol{P} \times \boldsymbol{L}$ as $\big(\boldsymbol{P} \circ (\mathbf{1} \times \boldsymbol{L}^t)\big) \times \mathbf{1}$. Indeed, taking the product of a matrix by the vector $\mathbf{1}$ amounts to adding the columns of this matrix, which can be done by adding vectors of length $|\mathscr{A}|$. Similarly, calculating $\boldsymbol{P} \circ (\mathbf{1} \times \boldsymbol{L}^t)$ can be done column-wise by taking the Hadamard product between the vector made of the i-th column of $\boldsymbol{P}$ and a vector made of $|\mathscr{A}|$ repeats of the i-th element of $\boldsymbol{L}$.

In general though, the length of vectors core operations apply to may be different from $|\mathscr{A}|$. In order to clarify the explanations, we will refer to a core vector as a *block* in what follows. The length of a block, i.e., the number of elements in one such vector, is denoted as v, with $v \leq |\mathscr{A}|$ and, in the particular case described here, $v \bmod |\mathscr{A}| = 0$. This last equality may not always hold however, depending on the type of data and vectorization considered, and data structure padding may thus be required. $\underline{V}$ will denote a single block while $\underline{\boldsymbol{V}}$ will denote a vector of more than one block. Algorithm 1 details the different steps in the calculation of $\big(\boldsymbol{P} \circ (\mathbf{1} \times \boldsymbol{L}^t)\big) \times \mathbf{1}$ using blocks and vectors of blocks. When considering scalar operations only, evaluating $\boldsymbol{P} \times \boldsymbol{L}$ or $\big(\boldsymbol{P} \circ (\mathbf{1} \times \boldsymbol{L}^t)\big) \times \mathbf{1}$ requires $\mathscr{O}(|\mathscr{A}|^2)$ scalar operations. Algorithm 1 involves only $\mathscr{O}(|\mathscr{A}^2|/v)$ core vector operations where each core vector operation has the same computational cost as a scalar operation. This improvement in terms of computational time complexity comes at the cost of an increased memory

traffic, however. Indeed, vector computation requires loading series of scalars into dedicated data structure. These loading operations, which are not required when dealing with scalars throughout the analysis, incur extra computational costs that should not be ignored when assessing the pros and cons of using vectorization.

```
input : L, P
output: R                                   // R is a vector of b blocks
v ← length of a block;
b ← |𝒜|/v;                                  // Number of blocks in 𝒜
R ← 0;                                      // Initialize R
for i ← 1 to |𝒜| do
    Fill V with L_i;                        // V is a block
    for j ← 1 to b do
        x ← (j − 1)v + 1; y ← jv + 1;
        Fill W with P_{x:y,i};              // W is a block
        R_j ← R_j + (W ∘ V);                // Update j-th block of R.
    end
end
```

Algorithm 1: Calculation of $\left(\boldsymbol{P} \circ (\boldsymbol{1} \times \boldsymbol{L}^t)\right) \times \boldsymbol{1}$, where $\boldsymbol{P}$ and $\boldsymbol{L}$ are the transition probability matrix and the conditional likelihood vector at the bottom of a given branch, respectively (e.g., one may have $\boldsymbol{P} = \boldsymbol{P}(l_U)$ and $\boldsymbol{L} = \boldsymbol{L}_U$ or $\boldsymbol{P} = \boldsymbol{P}(l_V)$ and $\boldsymbol{L} = \boldsymbol{L}_V$).

2.5 Inferring Edge Lengths

2.5.1 Speeding Up the Likelihood Calculation

The transition probability matrix $\boldsymbol{P}(l)$ is derived from the infinitesimal generator of the Markov process, or rate matrix, noted as $\boldsymbol{Q}$, with $\boldsymbol{P}(l) = \exp(\boldsymbol{Q}l)$. Exponentiating the rate matrix can be done through its eigendecomposition. We have $\boldsymbol{Q} = \boldsymbol{U} \times \boldsymbol{\Lambda} \times \boldsymbol{U}^{-1}$, where $\boldsymbol{U}$ is the matrix of right eigenvectors of $\boldsymbol{Q}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of the corresponding eigenvalues. We thus have: $\boldsymbol{P}(l) = \boldsymbol{U} \times \exp(\boldsymbol{\Lambda}l) \times \boldsymbol{U}^{-1}$. Plugging this last equality into Eq. 2.10, we obtain:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \left((\boldsymbol{\Pi} \circ \boldsymbol{L}_U)^t \times \boldsymbol{U}\right) \times \exp(\boldsymbol{\Lambda}l) \times \left(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\right). \tag{2.12}$$

Let $\boldsymbol{X}$ be a vector of size n and $\boldsymbol{M}$ be a n by n diagonal matrix. Then it is straightforward to verify that the following equality holds $\boldsymbol{X}^t \times \boldsymbol{M} = (\boldsymbol{X} \circ diag(\boldsymbol{M}))^t$, where $diag(\boldsymbol{M})$ is the vector of size n made of the diagonal entries of $\boldsymbol{M}$. Furthermore, let $\boldsymbol{Y}$ be another vector of size n, then one can verify that $\boldsymbol{X}^t \times \boldsymbol{Y} = \boldsymbol{1}^t \times (\boldsymbol{X} \circ \boldsymbol{Y})$ holds as well. Using these two properties, Eq. 2.12 can be re-written as follows:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \mathbf{1}^t \times \Big(\big(\boldsymbol{U}^t \times (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)\big) \circ \exp(\boldsymbol{\lambda} l) \circ \big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\big)\Big), \quad (2.13)$$

where $\exp(\boldsymbol{\lambda} l) = diag(\exp(\boldsymbol{\Lambda} l))$, and $\boldsymbol{\lambda}$ is the column vector of eigenvalues of $\boldsymbol{Q}$.

Considering Eq. 2.9, the number of scalar operations (product and addition) corresponding to the matrix-vector product $\boldsymbol{P}(l_U) \times \boldsymbol{L}_U$ is $2|\mathscr{A}|^2 - |\mathscr{A}|$. The same applies to the product $\boldsymbol{P}(l_V) \times \boldsymbol{L}_V$. The Hadamard product requires $|\mathscr{A}|$ scalar operations while the product involving $\boldsymbol{\Pi}^t$ requires $2|\mathscr{A}| - 1$ operations. In total, the computational cost involved in the calculation of the likelihood (assuming $\boldsymbol{L}_U$ and $\boldsymbol{L}_V$ are known) is that associated to $4|\mathscr{A}|^2 + |\mathscr{A}| - 1$ scalar operations. When taking Eq. 2.10 instead, the computational cost is reduced to $2|\mathscr{A}|^2 + 2|\mathscr{A}| - 1$. Importantly, each step in the optimization of l requires applying every scalar operation involved in the calculation of Eq. 2.9 or Eq. 2.10.

Considering now Eq. 2.13, the calculation of $\big(\boldsymbol{U}^t \times (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)\big)$ requires $2|\mathscr{A}|^2$ scalar operations while that of $\big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\big)$ requires $2|\mathscr{A}|^2 - |\mathscr{A}|$ operations. The two Hadamard products require $|\mathscr{A}|$ each, and the product involving $\mathbf{1}^t$ costs $2|\mathscr{A}| - 1$ so that, in total, $4|\mathscr{A}|^2 + 3|\mathscr{A}| - 1$ operations are required, which does not make this approach particularly computationally effective compared to the two others if applied naively. However, it is important to note that the terms $\big(\boldsymbol{U}^t \times (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)\big)$ and $\big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\big)$ are not functions of l. The corresponding vectors are constant throughout the optimization of that edge length. Therefore, the computational burden involved in Eq. 2.13 is that corresponding to the two Hadamard products around the exponential term and the product involving $\mathbf{1}^t$ only, i.e., $4|\mathscr{A}| - 1$. Leaving aside the pre-computations, the computational complexity of the likelihood calculation during a branch length optimization therefore drops from $\mathscr{O}(|\mathscr{A}|^2)$ when using Eq. 2.10 to $\mathscr{O}(|\mathscr{A}|)$ when using Eq. 2.13.

2.5.2 *Optimizing One Length*

The calculation of the first and second derivative of the likelihood at a given site with respect to the length of an edge is given by two expressions very similar to that of Eq. 2.13. Indeed, we have:

$$\frac{d \Pr(\boldsymbol{D}|\mathscr{M})}{dl} = \mathbf{1}^t \times \Big(\big(\boldsymbol{U}^t \times (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)\big) \circ \boldsymbol{\lambda} \circ \exp(\boldsymbol{\lambda} l) \circ \big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\big)\Big)$$

$$\frac{d^2 \Pr(\boldsymbol{D}|\mathscr{M})}{dl^2} = \mathbf{1}^t \times \Big(\big(\boldsymbol{U}^t \times (\boldsymbol{\Pi} \circ \boldsymbol{L}_U)\big) \circ \boldsymbol{\lambda} \circ \boldsymbol{\lambda} \circ \exp(\boldsymbol{\lambda} l) \circ \big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_V\big)\Big).$$

The first and second derivatives of the log-likelihood with respect to the same edge length are then obtained as follows:

$$\frac{d \log \Pr(\boldsymbol{D}|\mathscr{M})}{dl} = \frac{1}{\Pr(\boldsymbol{D}|\mathscr{M})} \frac{d \Pr(\boldsymbol{D}|\mathscr{M})}{dl}$$
$$\frac{d^2 \log \Pr(\boldsymbol{D}|\mathscr{M})}{dl^2} = \frac{1}{\Pr(\boldsymbol{D}|\mathscr{M})} \frac{d^2 \Pr(\boldsymbol{D}|\mathscr{M})}{dl^2} - \left(\frac{d \log \Pr(\boldsymbol{D}|\mathscr{M})}{dl} \right)^2,$$

and the first and second derivatives of the log-likelihood for a whole sequence alignment are obtained by applying these last two equations at each site of the alignment and then summing the obtained values. The ability to evaluate quickly and without approximation the derivatives of the log-likelihood with respect to a given edge length is paramount to the efficient optimization of this model parameter. The Newton–Raphson method for finding roots of functions could then be applied to the first derivative function in order to obtain a maximum likelihood estimate of the edge length. One would simply need to make sure that the optimization converged to a maximum of the log-likelihood function rather than a minimum by verifying that the second derivative at the stationary point is negative.

We experienced difficulties with this technique, however, especially in cases where the initial value for the optimization routine is far from the optimum, which is a common issue that affects many optimization methods. Instead, we considered optimization techniques that rely only on the first derivative. Brent's method [3] was the technique used in PhyML until 2017. It is a combination of a quadratic interpolation method, which works well in the vicinity of an optimum of the function of interest, and other optimization tricks that ensure that the maximum of the function is always bracketed by an interval of decreasing width between successive iterations of the algorithm. We recently switched to an original and more efficient method that combines the same two ingredients as Brent's that are the bracketing of a maximum and interpolation. This new approach relies on a cubic spline interpolation, which can be asymmetric in contrast to a parabola as that used in Brent's method. The derivatives of the cubic spline at the bracketing points are equal to that of the target function, thereby providing a sound approximation of the optimized function near a maximum even in cases where this target function is not symmetrical around that point.

Let $\ell(l_u)$ and $\ell(l_v)$ be the log-likelihoods calculated for a given phylogeny with the length of the edge to be optimized set to l_u and l_v, respectively. These lengths are chosen such that $l_u < l_v$. Also, we have $d\ell(l_u)/dl > 0$ and $d\ell(l_v)/dl < 0$ so that a maximum of the likelihood function exists in the $[l_u, l_v]$ interval. We then fit a cubic spline to the log-likelihood function in the interval aforementioned. This spline is a polynomial of degree three that has only one maximum in $[l_u, l_v]$ and has its derivative equal to that of the function of interest at l_u and l_v (i.e., $d\ell(l_u)/dl$ and $d\ell(l_v)/dl$, respectively). The argument, l^*, that maximizes the spline function is calculated analytically. The first derivative of the log-likelihood is next evaluated at that point (i.e., l^*). We then have $l_u \leftarrow l^*$ if $d\ell(l^*)/dl > 0$ or $l_v \leftarrow l^*$ if $d\ell(l^*)/dl < 0$. A new bracketing interval is then defined that way and the algorithm moves on to the next iteration. The optimization stops whenever the width of the interval $[l_u, l_v]$ is below a certain threshold.

We compared the number of times the likelihood function was evaluated during the optimization of edge lengths using the spline interpolation technique described above and Brent's method. We considered two real-world data sets. The first is an alignment of 1,566 nucleotide sequences, 915 bp-long. The second has 62 amino acid sequences, each of them being 230,322 residue-long. The GTR + G4 [31, 34] and LG + G4 [18] substitution models were used in the first and second cases, respectively. A phylogeny was first estimated using a distance-based tree reconstruction technique. Edge lengths were then optimized on a fixed tree topology. For the nucleotide data set, the optimization required 781,996 calls to the likelihood function when using Brent's method while it required only 176,938 calls to the same function when using spline interpolation. For the protein data set, the equivalent figures were 20,471 for Brent and 4,022 for spline interpolation. Although these results concern a very limited number of examples and parameter settings, they strongly suggest that the spline interpolation technique does a better job at fitting the likelihood function, thereby requiring significantly less computation in order to reach a maximum compared to Brent's approach.

2.5.3 *Optimizing All Lengths*

The optimization of all edge lengths in the tree can be performed in an efficient manner using a pre-order traversal algorithm. Let us consider the tree depicted in Fig. 2.2. A post-order tree traversal is conducted in the first place so that the conditional likelihood vectors $\boldsymbol{L}_X$, $\boldsymbol{L}_U$, $\boldsymbol{L}_V$ and $\boldsymbol{L}_W$ are already known. This traversal was initiated at a tip node taken in τ_Z, i.e., the subtree in the top triangle of Fig. 2.2 (the root node corresponds to the green disk in the figure). Let $\boldsymbol{L}_Y^{\uparrow} := \Pr(\boldsymbol{D}_Z | Z = \cdot, \tau_Z)$ correspond to a partial likelihood vector "looking up" from node Y, i.e., looking toward the subtree containing the root of the tree (hence the $\uparrow$ symbol). Optimizing the length of the edge between vertices W and Y, i.e., l_W, requires the calculation of the partial likelihood vector $\boldsymbol{L}_W^{\uparrow} := \Pr(\boldsymbol{D}_Z \cup \boldsymbol{D}_X | Y = \cdot, \tau_Y)$, whereby $\tau_Y = \tau_X \cup \tau_Z$ is the subtree made of the union of subtrees τ_X and τ_Z. Once $\boldsymbol{L}_W^{\uparrow}$ is known, optimizing the length of the edge under scrutiny can be done by applying Eq. 2.13 as follows:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \mathbf{1}^t \times \Big(\big((\boldsymbol{\Pi} \circ \boldsymbol{L}_W)^t \times \boldsymbol{U} \big) \circ \exp(\boldsymbol{\lambda} l_W) \circ \big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_W^{\uparrow} \big) \Big). \quad (2.14)$$

The same two steps are applied next to optimize the length of the edge between U and W, i.e., l_U. $\boldsymbol{L}_U^{\uparrow}$ is calculated first (it is derived using $\boldsymbol{L}_W^{\uparrow}$ and $\boldsymbol{L}_V$ along with $\boldsymbol{P}(l_W)$ and $\boldsymbol{P}(l_V)$ and applying Eq. 2.11). The optimization then takes place by applying Eq. 2.13 as follows:

$$\Pr(\boldsymbol{D}|\mathscr{M}) = \mathbf{1}^t \times \Big(\big((\boldsymbol{\Pi} \circ \boldsymbol{L}_U)^t \times \boldsymbol{U} \big) \circ \exp(\boldsymbol{\lambda} l_U) \circ \big(\boldsymbol{U}^{-1} \times \boldsymbol{L}_U^{\uparrow} \big) \Big). \quad (2.15)$$

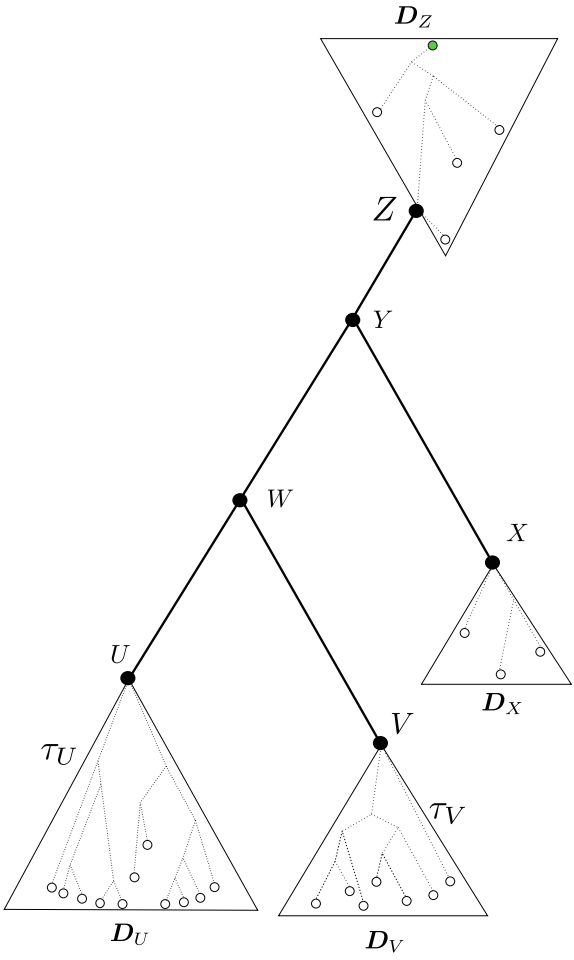


Fig. 2.2 Elements of notations used to present the recursive optimization of edge lengths. The root of the tree is taken as a tip here (in green). The observed data is given by $D_U \cup D_V \cup D_X \cup D_Z$

Once all the edges under node U are optimized by applying this pre-traversal algorithm, the partial likelihood vector $\boldsymbol{L}_U$ is updated and the length of the edge connecting U and V is optimized once more. The same steps are then followed for the subtree rooted by node V after updating $\boldsymbol{L}_V^{\uparrow}$ and optimizing the edge between nodes W and V. Once all the edges under node W are optimized, the partial likelihood vector $\boldsymbol{L}_W$ is updated using Eq. 2.11. Doing so guarantees that the optimization of l_X is performed using an up-to-date vector $\boldsymbol{L}_X^{\uparrow}$ (which requires having an up-to-date version of $\boldsymbol{L}_W$). A very similar approach that combines edge length optimization and updating of the partial likelihood vector in a post- and pre-traversal algorithm was implemented early on in the software package MOLPHY [1].

2.6 Inferring Parameters of Mixture Models

We previously considered that $\boldsymbol{D}$ referred to a datum, i.e., a single column of a nucleotide or amino acid sequence alignment. We hereby slightly modify this notation so that $\boldsymbol{D}_s$ now corresponds to the data at the column s of the alignment. Because sites are assumed to be independent and identically distributed, we have:

$$\ell = \prod_{s=1}^{L} \Pr(\boldsymbol{D}_s | \mathscr{M}), \tag{2.16}$$

where L is the number of columns in the alignment. Because evolutionary processes vary along the sequences (e.g., the third position in coding genes is fast-evolving compared to the other two positions), mixture models generally provide substantially better fit to the data than the default approach. Mixture models are standard probabilistic models whereby each observation making up the data is assumed to have been generated by a model involving *distributions* of parameters rather than single values. Different observations in the sample can thus be generated under distinct values of the parameter(s) of interest. In the case under scrutiny here, the mixture defines a probabilistic distribution over (relative) rates of evolution, thereby authorizing different sites to have distinct rates of substitution. Mixture models have also been used extensively to model the variability of selective regimes across amino acids in coding sequences (e.g., [22, 36, 37]), to accommodate for the variability of secondary structures along proteins sequences [19] or to take into account different patterns of substitutions in protein regions evolving at distinct rates [17].

We consider a particular type of mixture model with K classes where only the frequencies of each class are free to vary. For instance, in the mixture below:

$$\ell = \prod_{s=1}^{L} \sum_{c=1}^{K} \Pr(\boldsymbol{D}_s | R = w_c, \mathscr{M}) \Pr(R = w_c), \tag{2.17}$$

or, using a more concise notation:

$$\ell = \prod_{s=1}^{L} \sum_{c=1}^{K} \Pr(\boldsymbol{D}_s | w_c) p_c, \tag{2.18}$$

the values of $w_1, \ldots, w_K$ are fixed, while that of the corresponding frequencies $p_1, \ldots, p_K$ are free to vary and can be adjusted so as to maximize the fit of the model to the data available. In this last expression, $\Pr(\boldsymbol{D}_s | w_c)$ corresponds to the probability of observing $\boldsymbol{D}_s$ under a phylogenetic model whereby the length of every edge is multiplied by w_c.

For any $K \geq 1$, the log-likelihood is written as follows:

$$\begin{aligned}\log \ell &= \sum_{s=1}^{L} \log \sum_{c=1}^{K} p_c \Pr(\boldsymbol{D}_s|w_c) \\ &= \sum_{s=1}^{L} \log \left[\sum_{c=1}^{K-1} p_c \big((\Pr(\boldsymbol{D}_s|w_c) - \Pr(\boldsymbol{D}_s|w_K) \big) + \Pr(\boldsymbol{D}_s|w_K) \right].\end{aligned}$$

Let $f_s(\boldsymbol{p})$ be the argument of the logarithm function above for a given site s. Taking the second row of the last equation above, it is straightforward to verify that the following equality holds: $f_s(\frac{\boldsymbol{p}+\boldsymbol{p}'}{2}) = \frac{1}{2}(f_s(\boldsymbol{p}) + f_s(\boldsymbol{p}'))$, so that $f_s(\boldsymbol{p})$ is linear (and thus both concave and convex) and its logarithm is strictly concave. We then have:

$$\log \Big(f_s\big(\frac{\boldsymbol{p}+\boldsymbol{p}'}{2}\big) \Big) = \log \Big(\frac{1}{2} \big(f_s(\boldsymbol{p}) + f_s(\boldsymbol{p}') \big) \Big) > \frac{1}{2} \Big(\log \big(f_s(\boldsymbol{p}) \big) + \log(f_s(\boldsymbol{p}')) \Big).$$

Finally, the sum of strictly concave functions is also strictly concave, making $\log \ell$ a strictly concave function too.

The concavity of the logarithm function guarantees that the optimization of the frequency parameters does not encounter local optima. Note that this conclusion is applicable to the particular situation where only the frequencies of the mixture classes are adjusted while the relative rates are fixed. The +I model, where a proportion of sites do not evolve, belongs to this class of model. For a larger number of classes, this model contrasts with the standard approach for modeling the variability of rates-across sites. A discrete gamma distribution where the class frequencies are fixed while the relative rates are given by the (estimated) shape of the gamma distribution corresponds to the most popular approach. Similarly, the FreeRate model [25] (but see also [17]) adjusts both the relative rates and the corresponding frequencies. There is no guarantee that the likelihood function is concave in this context.

Empirical evidence indicates that the likelihood function is not strictly concave when using the standard approximation of the gamma distribution [12]. The fact that this function does not have local maxima when the gamma distribution is discretized by fixing the relative rates to sensible values and adjusting the rate class frequencies suggests that this approximation could in fact be preferable to the standard approach. Similar observations were made before by Susko et al. in 2003 [28]. These authors considered a discrete gamma distribution whereby the relative rates of evolution were fixed and equally spaced while the probability of the corresponding rate classes was adjusted so as to maximize the likelihood. They pointed out that this approximation of the continuous gamma distribution is easier to deal with from a computational perspective since adjusting the class frequencies can be done very quickly compared to the optimization of the relative rates (the pruning algorithm can be avoided when optimizing the frequencies). They also showed that more accurate rate estimates could be obtained using their approximation compared to the standard one.

2.7 Conclusion

The present chapter gives an overview of the specifics underlying the optimization of edge lengths in a phylogeny under the maximum likelihood criterion. We show how to best exploit the architecture of modern computer processors to speed up the calculation of the likelihood itself, along with partial likelihood vectors. These calculations are performed using vector operations only—a context in which the application of vector intrinsics is particularly suitable. We also show that the amount of computation required for the evaluation of the likelihood during the optimization of a given edge length can drop from $|\mathscr{A}|^2$ to $|\mathscr{A}|$, which is particularly useful when considering amino acid substitution models and even more so with codon models. We present a post- and pre-traversal algorithm for optimizing all edge lengths in the tree that minimizes the amount of computation required for updating partial likelihood vectors. These techniques are implemented in several state-of-the-art maximum likelihood tree building software programs. Nonetheless, to the best of our knowledge, they have not been described in detail in the literature. We then make the point that the most popular approach to model the variability of substitution rates-across sites could be improved by fixing the rates and optimizing the corresponding frequencies rather than the opposite. We show indeed that the likelihood function has only one maximum when rates are fixed (while it is sometimes otherwise when they are not), thereby providing a guarantee of "good behavior".

Maximum likelihood tree estimation was introduced about four decades ago. This chapter illustrates the fact that, despite its relatively old age, this approach is still a source of interesting mathematical developments that can be used to further improve the inference techniques, in particular in terms of the speed of calculation. In conjunction with important initiatives aimed at making large-scale computing services available to the public—the Cyber-Infrastructure for Phylogenetic Research (CIPRES) Project, originally created by Bernard Moret amongst others, being arguably the most popular of them—maximum likelihood tree inference has helped improve our understanding of molecular evolution and biology in general. While current maximum likelihood software programs cannot cope with the analysis of thousands of complete genomes, there are hopes that new likelihood-based techniques capable of dealing with massive amounts of data, which rely on some of the advances described in this book, will see the light in the near future.

Acknowledgements We would like to thank Alexandros Stamatakis for helpful suggestions on how to improve this chapter and Tandy Warnow for inviting us to celebrate Bernard Moret's contributions to the field of computational evolution.

References

1. Adachi, J., Hasegawa, M.: MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. Institute of Statistical Mathematics Tokyo (1996)
2. Ayres, D.L., Darling, A., Zwickl, D.J., Beerli, P., Holder, M.T., Lewis, P.O., Huelsenbeck, J.P., Ronquist, F., Swofford, D.L., Cummings, M.P., Rambaut, A., Suchard, M.A.: BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. Syst. Biol. **61**(1), 170–173 (2011)
3. Brent, R.P.: An algorithm with guaranteed convergence for finding a zero of a function. Comput. J. **14**(4), 422–425 (1971)
4. Dayhoff, M., Schwartz, R., Orcutt, B.: A model of evolutionary change in proteins. In: Dayhoff, M. (ed.) Atlas of Protein Sequence and Structure, vol. 5, pp. 345–352. National Biomedical Research Foundation, Washington, D.C. (1978)
5. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**, 368–376 (1981)
6. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Sunderland, MA (2004)
7. Gascuel, O., Guindon, S.: Modelling the variability of evolutionary processes. In: Gascuel, O., Steel, M. (eds.) Reconstructing Evolution: New Mathematical and Computational Advances, pp. 65–99. Oxford University Press (2007)
8. Guindon, S., Dufayard, J.F., Lefort, V., Anisimova, M., Hordijk, W., Gascuel, O.: New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. Syst. Biol. **59**(3), 307–321 (2010)
9. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. **52**(5), 696–704 (2003)
10. Hasegawa, M., Kishino, H., Yano, T.: Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol. **22**(2), 160–174 (1985)
11. Helaers, R., Milinkovitch, M.C.: MetaPIGA v2.0: maximum likelihood large phylogeny estimation using the metapopulation genetic algorithm and other stochastic heuristics. BMC Bioinform. **11**(1), 379 (2010)
12. Hoang, D.T., Chernomor, O., von Haeseler, A., Minh, B.Q., Le, S.V.: UFBoot2 improving the ultrafast bootstrap approximation. Mol. Biol. Evol. **35**(2), 518–522 (2018)
13. Hordijk, W., Gascuel, O.: Improving the efficiency of SPR moves in phylogenetic tree search methods based on maximum likelihood. Bioinformatics **21**(24), 4338–4347 (2005)
14. Jarvis, E., Mirarab, S., Aberer, A., Li, B., Houde, P., Li, C., Ho, S., Faircloth, B., Nabholz, B., Howard, J., Suh, A., Weber, C., da Fonseca, R., Li, J., Zhang, F., Li, H., Zhou, L., Narula, N., Liu, L., Ganapathy, G., Boussau, B., Bayzid, M., Zavidovych, V., Subramanian, S., Gabaldón, T., Capella-Gutiérrez, S., Huerta-Cepas, J., Rekepalli, B., Munch, K., Schierup, M., Lindow, B., Warren, W., Ray, D., Green, R., Bruford, M., Zhan, X., Dixon, A., Li, S., Li, N., Huang, Y., Derryberry, E., Bertelsen, M., Sheldon, F., Brumfield, R., Mello, C., Lovell, P., Wirthlin, M., Schneider, M., Prosdocimi, F., Samaniego, J., Vargas Velazquez, A., Alfaro-Núñez, A., Campos, P., Petersen, B., Sicheritz-Ponten, T., Pas, A., Bailey, T., Scofield, P., Bunce, M., Lambert, D., Zhou, Q., Perelman, P., Driskell, A., Shapiro, B., Xiong, Z., Zeng, Y., Liu, S., Li, Z., Liu, B., Wu, K., Xiao, J., Yinqi, X., Zheng, Q., Zhang, Y., Yang, H., Wang, J., Smeds, L., Rheindt, F., Braun, M., Fjeldsa, J., Orlando, L., Barker, F., Jønsson, K., Johnson, W., Koepfli, K., O'Brien, S., Haussler, D., Ryder, O., Rahbek, C., Willerslev, E., Graves, G., Glenn, T., McCormack, J., Burt, D., Ellegren, H., Alström, P., Edwards, S., Stamatakis, A., Mindell, D., Cracraft, J., Braun, E., Warnow, T., Jun, W., Gilbert, M., Zhang, G.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science **346**(6215), 1320–1331 (2014)
15. Jones, D.T., Taylor, W.R., Thornton, J.M.: The rapid generation of mutation data matrices from protein sequences. Bioinformatics **8**(3), 275–282 (1992)
16. Jukes, T., Cantor, C.: Evolution of protein molecules. In: Munro, H. (ed.) Mammalian Protein Metabolism, vol. III, chap. 24, pp. 21–132. Academic Press, New York (1969)
17. Le, S.Q., Dang, C.C., Gascuel, O.: Modeling protein evolution with several amino acid replacement matrices depending on site rates. Mol. Biol. Evol. **29**(10), 2921–2936 (2012)

18. Le, S.Q., Gascuel, O.: An improved general amino acid replacement matrix. Mol. Biol. Evol. **25**(7), 1307–1320 (2008)
19. Le, S.Q., Gascuel, O.: Accounting for solvent accessibility and secondary structure in protein phylogenetics is clearly beneficial. Syst. Biol. **59**(3), 277–287 (2010)
20. Lin, Y., Hu, F., Tang, J., Moret, B.M.: Maximum likelihood phylogenetic reconstruction from high-resolution whole-genome data and a tree of 68 eukaryotes. In: Biocomputing 2013, pp. 285–296. World Scientific (2013)
21. Nguyen, L.T., Schmidt, H.A., von Haeseler, A., Minh, B.Q.: IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol. Biol. Evol. **32**(1), 268–274 (2014)
22. Nielsen, R., Yang, Z.: Likelihood models for detecting positively selected amino acid sites and application to the HIV-1 envelope gene. Genetics **148**, 929–936 (1998)
23. Pratas, F., Trancoso, P., Stamatakis, A., Sousa, L.: Fine-grain parallelism using multi-core, cell/be, and GPU systems: accelerating the phylogenetic likelihood function. In: International Conference on Parallel Processing, 2009, ICPP'09, pp. 9–17. IEEE (2009)
24. Ronquist, F., Huelsenbeck, J.P.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**(12), 1572–1574 (2003)
25. Soubrier, J., Steel, M., Lee, M.S., Der Sarkissian, C., Guindon, S., Ho, S.Y., Cooper, A.: The influence of rate heterogeneity among sites on the time dependence of molecular rates. Mol. Biol. Evol. **29**(11), 3345–3358 (2012)
26. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21), 2688–2690 (2006)
27. Stamatakis, A., Ludwig, T., Meier, H.: RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics **21**(4), 456–463 (2004)
28. Susko, E., Field, C., Blouin, C., Roger, A.J.: Estimation of rates-across-sites distributions in phylogenetic substitution models. Syst. Biol. **52**(5), 594–603 (2003)
29. Swofford, D.: PAUP*: phylogenetic analysis using parsimony (* and other methods) Ver. 4. Sinauer Associates, Sunderland, Massachusetts (2002)
30. Tamura, K., Peterson, D., Peterson, N., Stecher, G., Nei, M., Kumar, S.: MEGA5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. Mol. Biol. Evol. **28**(10), 2731–2739 (2011)
31. Tavaré, S.: Some probabilistic and statistical problems in the analysis of DNA sequences. Lectures on Mathematics in the Life Sciences, vol. 17, pp. 57–86. American Mathematical Society (1986)
32. Vinh, L.S., von Haeseler, A.: IQPNNI: moving fast through tree space and stopping in time. Mol. Biol. Evol. **21**(8), 1565–1571 (2004)
33. Whelan, S., Goldman, N.: A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. Mol. Biol. Evol. **18**(5), 691–699 (2001)
34. Yang, Z.: Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J. Mol. Evol. **39**, 306–314 (1994)
35. Yang, Z.: Computational molecular evolution. Oxford University Press (2006)
36. Yang, Z., Nielsen, R.: Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. Mol. Biol. Evol. **17**, 32–43 (2000)
37. Yang, Z., Nielsen, R.: Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. Mol. Biol. Evol. **19**, 908–917 (2002)
38. Zwickl, D.: GARLI: genetic algorithm for rapid likelihood inference (2006). http://www.bio.utexas.edu/faculty/antisense/garli/Garli.html

Chapter 3
High-Performance Phylogenetic Inference

David A. Bader and Kamesh Madduri

Abstract Software tools based on the maximum likelihood method and Bayesian methods are widely used for phylogenetic tree inference. This article surveys recent research on parallelization and performance optimization of state-of-the-art tree inference tools. We outline advances in shared-memory multicore parallelization, optimizations for efficient Graphics Processing Unit (GPU) execution, as well as large-scale distributed-memory parallelization.

Keywords Phylogenetic tree inference · Maximum likelihood · Bayesian inference · Parallel algorithms · Algorithm engineering

3.1 Introduction

Computational phylogenetics is an active research area. A variety of algorithms and software tools exist for the compute-intensive task of tree inference. Early methods were based on distance-based similarity clustering [18, 43, 46] and on the Maximum Parsimony principle [17, 20]. These simple methods are now subsumed by more sophisticated algorithms. Probabilistic approaches, specifically Maximum Likelihood (ML)-based [15] methods and Bayesian inference methods [25, 42], currently dominate the landscape of tree inference software. As of October 2018, the OMICtools website [39] lists 266 software tools in the Phylogenetic Inference category. Felsenstein's Phylogeny Programs web page [14] lists more than 90 ML-based methods and more than 25 Bayesian inference methods. The Cyberinfrastructure for Phylogenetic Research (CIPRES) Science Gateway Version 3.3 [9, 30] currently supports 15 parallel programs for tree inference and sequence alignment. Phylogeny.fr [10] is another long-running web portal for phylogenetic analysis.

D. A. Bader (✉)
Georgia Institute of Technology, Atlanta, GA, USA
e-mail: bader@cc.gatech.edu

K. Madduri
Pennsylvania State University, University Park, PA, USA
e-mail: madduri@cse.psu.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_3

Popular, free, and open-source tools include PHYLIP [13], RAxML [47, 48], PhyML [22, 23], MrBayes [42], and BEAST 2 [6]. Nearly all of these tools support some form of parallelism.

Moret played a seminal role in establishing the research area of high-performance computational phylogenetics by leading the development of GRAPPA and associated algorithms [21, 32–34]. GRAPPA is a maximum parsimony-based suite of programs for phylogeny reconstruction using genome rearrangements. For breakpoint phylogeny reconstruction, using efficient data structures and optimizations, GRAPPA was engineered to perform nearly 2500 times faster than the original Sankoff–Blanchette algorithm [44] on a single processor. When executed on a 512-processor cluster, GRAPPA achieved an awe-inspiring million-fold speedup [5]. GRAPPA is a significant milestone in the areas of algorithm engineering and parallel phylogenetic inference. Many of the current probabilistic inference methods take aligned sequences, typically DNA or amino acid sequences, as input. The quality of multiple sequence alignment will thus directly impact the quality of trees generated. The methods also assume a model for site evolution and estimate model parameters. The Generalized Time Reversible (GTR) model [51] is a commonly used model for inference on DNA and amino acid sequences. For additional background on statistical methods, please refer to [24, 52]. Current software tools support a wide variety of evolutionary models.

Likelihood calculations [48] constitute a significant fraction of the overall running time of both ML and Bayesian inference methods. We first discuss performance optimizations and parallelization strategies to speed up likelihood calculations. In Sect. 3.3, we discuss miscellaneous execution time-reducing implementation changes and approaches to improve multi-node performance. (See also the chapters by Stamatakis and Guindon & Gascuel in this book for more about this subject.).

3.2 Faster Likelihood Calculations

ML-based tree reconstruction has been shown to be an NP-hard optimization problem under various assumptions [8, 41]. An exponential number of tree configurations need to be evaluated in order to find the optimal solution, and this is intractable with even a modest number of organisms. Thus, software tools employ a variety of heuristics to reduce the search space. For each tree topology, evaluating the likelihood function involves postorder tree traversal and propagating likelihood values from the tips to the root according to Felsenstein's pruning algorithm [15]. Likelihood computations also appear in Bayesian inference methods. These computations are both floating-point operation and memory-intensive, and take up a dominant fraction of the running time in state-of-the-art programs.

Fortunately, there is abundant fine-grained parallelism to exploit in these likelihood calculations. The partial likelihood scores at each site can be computed independent of other sites. Since the number of sites can vary from thousands to millions, the multiple sequence alignment output can be further split into partitions that

can be evaluated independently. Likelihood calculations are also prone to floating-point rounding errors and need to be evaluated carefully. The community is moving away from monolithic codes and transitioning to using library-based approaches. Bio++ [12] is an early example of a C++ library with optimized implementations of key phylogenetic primitives. BEAGLE (Broad-platform Evolutionary Analysis General Likelihood Evaluator) [4, 50] is a library and an application programming interface for parallel likelihood calculations. BEAGLE routines can be used in both ML-based inference methods and Bayesian methods. In addition to partitioning of alignment sites, fine-grained data parallelism is possible across rate categories and state values. BEAGLE includes SSE implementations for CPUs, as well as CUDA and OpenCL implementations of routines for GPUs.

BEAGLE also provides interfaces to the inference tools BEAST 2 [6], BEAST [11], MrBayes [42], and GARLI [54]. It is shown that the library-based approaches outperform the standalone implementations, and that the GPU-based approach delivers a significant performance boost over a CPU implementation. Recent work by Ayres and Cummings [3] explores additional tuning opportunities to further improve the performance of BEAGLE routine.

Phylogenetic Likelihood Library (PLL) [19] is another open-source library inspired by Bio++ and BEAGLE. PLL is used by ExaML [29] and RAxML-NG [28], two recent and modern implementations of RAxML, and also interfaces with IQ-TREE [37], a recent ML-based inference package. PLL has a backend for the Intel Xeon Phi accelerator, Python bindings, includes many SIMD implementations, and also supports MPI parallelization. It is shown to be 1.9–4× faster [19] than BEAGLE on benchmarks.

3.3 Performance Optimizations and Multi-node Parallelism

Bayesian methods [7, 52] approximate the posterior distribution of evolutionary parameters using Bayes' theorem. The methods rely on sampling approaches such as the Metropolis-coupled Markov chain Monte Carlo (MCMC) algorithm, give probability distributions for model parameters, and allow incorporation of prior assumptions. Altekar et al. [2] discuss shared-memory and distributed-memory parallelization of the sampling scheme used in MrBayes. ExaBayes [1] also uses distributed Metropolis-coupled chains, and further proposes chain swaps using nonblocking communication messages. This nonblocking communication optimization is shown to reduce running time by up to 19% [1]. ExaBayes also includes a memory-saving technique by recomputing partial results on-demand. ExaML uses a similar recomputation optimization to reduce inter-node communication. When likelihood calculations are parallelized based on partitions, the P_i matrix calculations are redundantly performed by every process. Kobert et al. [27] formulate a bi-criterion data distribution problem to determine the optimal distribution of partitions and sites to processes, and show that their new implementation is up to 3× faster than the implementation with the prior data distribution scheme. Other notable multi-node parallelizations

include the master–worker strategy to parallelize the IQPNNI approach [31] and the Java-based DPRml [26] method. I/O optimizations and checkpointing are other important considerations in parallel environments. ExaML and Beast 2 include support for periodic disk-based checkpointing. ExaML converts the text-based input file to a binary format to permit parallel I/O.

In addition to parallelism, algorithmic changes also contribute to significant speedups. For instance, FastTree [40] employs several novel optimizations and is shown to be two orders of magnitude faster than RAxML version 7. A recent evaluation by Zhou et al. [53] shows that FastTree continues to be faster than recent versions of RAxML/ExaML, PhyML, and IQ-TREE, while also producing trees that are more dissimilar to trees generated using the other tool.

3.4 Conclusions

We have witnessed dramatic advances since early work on parallel phylogenetic inference [16, 45, 49]. Software development for computational phylogenetics is thriving [36], and performance optimization continues to be a focal area. It is now possible to achieve significant performance improvements for phylogenetic likelihood function calculations by leveraging modern libraries such as BEAGLE and PLL. Moret et al. [35] review methods for phylogenetic inference from rearrangement data, and describe an ML-based method that is competitive with approaches for sequence data. For the problem of supertree estimation, Nguyen et al. [38] show that Matrix Representation with Likelihood (MRL), an ML-based approach, is fast and outperforms leading alternative supertree methods (see chapter by Warnow in this book for more about MRL and supertree methods). Parallel algorithms and optimizations to improve scaling of these recent ML-based methods could be a promising future research direction.

Acknowledgements This work is supported in part by the National Science Foundation awards #1339745, #1439057, and #1535058.

References

1. Aberer, A.J., Kobert, K., Stamatakis, A.: ExaBayes: massively parallel Bayesian tree inference for the whole-genome era. Mol. Biol. Evol. **31**(10), 2553–2556 (2014). https://doi.org/10.1093/molbev/msu236
2. Altekar, G., Dwarkadas, S., Huelsenbeck, J.P., Ronquist, F.: Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics **20**(3), 407–415 (2004). https://doi.org/10.1093/bioinformatics/btg427
3. Ayres, D.L., Cummings, M.P.: Rerooting trees increases opportunities for concurrent computation and results in markedly improved performance for phylogenetic inference. In: Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 247–256 (2018). https://doi.org/10.1109/IPDPSW.2018.00049

4. Ayres, D.L., Darling, A., Zwickl, D.J., Beerli, P., Holder, M.T., Lewis, P.O., Huelsenbeck, J.P., Ronquist, F., Swofford, D.L., Cummings, M.P., Rambaut, A., Suchard, M.A.: BEAGLE: an application programming interface and high-performance computing library for statistical phylogenetics. Syst. Biol. **61**(1), 170–173 (2012). https://doi.org/10.1093/sysbio/syr100
5. Bader, D.A., Moret, B.M.E.: GRAPPA runs in record time. HPC Wire **9**, 47 (2000)
6. Bouckaert, R., Heled, J., Kühnert, D., Vaughan, T., Wu, C.H., Xie, D., Suchard, M.A., Rambaut, A., Drummond, A.J.: BEAST 2: a software platform for Bayesian evolutionary analysis. PLOS Comput. Biol. **10**(4), 1–6 (2014). https://doi.org/10.1371/journal.pcbi.1003537
7. Box, G.E.P., Tiao, G.C.: Bayesian Inference in Statistical Analysis, vol. 40. Wiley (2011)
8. Chor, B., Tuller, T.: Maximum likelihood of evolutionary trees: hardness and approximation. Bioinformatics **21**(suppl1), i97–i106 (2005). https://doi.org/10.1093/bioinformatics/bti1027
9. CIPRES Cyberinfrastructure for Phylogenetic Research. http://www.phylo.org/. Accessed Oct 2018
10. Dereeper, A., Guignon, V., Blanc, G., Audic, S., Buffet, S., Chevenet, F., Dufayard, J.F., Guindon, S., Lefort, V., Lescot, M., Claverie, J.M., Gascuel, O.: Phylogeny.fr: robust phylogenetic analysis for the non-specialist. Nucleic Acids Res. **36**(suppl2), W465–W469 (2008). https://doi.org/10.1093/nar/gkn180
11. Drummond, A.J., Rambaut, A.: BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evol. Biol. **7**(1), 214 (2007). https://doi.org/10.1186/1471-2148-7-214
12. Dutheil, J., Gaillard, S., Bazin, E., Glémin, S., Ranwez, V., Galtier, N., Belkhir, K.: Bio++: a set of C++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. BMC Bioinform. **7**(1), 188 (2006). https://doi.org/10.1186/1471-2105-7-188
13. Felsenstein, J.: PHYLIP version 3.697. http://evolution.genetics.washington.edu/phylip.html. Accessed Oct 2018
14. Felsenstein, J.: Phylogeny programs. http://evolution.genetics.washington.edu/phylip/software.html. Accessed Oct 2018
15. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**(6), 368–376 (1981). https://doi.org/10.1007/BF01734359
16. Feng, X., Buell, D.A., Rose, J.R., Waddell, P.J.: Parallel algorithms for Bayesian phylogenetic inference. J. Parallel Distrib. Comput. **63**(7), 707–718 (2003). https://doi.org/10.1016/S0743-7315(03)00079-0
17. Fitch, W.M.: On the problem of discovering the most parsimonious tree. Am. Nat. **111**(978), 223–257 (1977). https://doi.org/10.1086/283157
18. Fitch, W.M., Margoliash, E.: Construction of phylogenetic trees. Science **155**(3760), 279–284 (1967)
19. Flouri, T., Izquierdo-Carrasco, F., Darriba, D., Aberer, A., Nguyen, L.T., Minh, B., Von Haeseler, A., Stamatakis, A.: The phylogenetic likelihood library. Syst. Biol. **64**(2), 356–362 (2015). https://doi.org/10.1093/sysbio/syu084
20. Foulds, L.R., Graham, R.L.: The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. **3**(1), 43–49 (1982)
21. GRAPPA genome rearrangements analysis under parsimony and other phylogenetic algorithms. https://www.cs.unm.edu/~moret/GRAPPA/. Accessed Oct 2018
22. Guindon, S., Dufayard, J.F., Lefort, V., Anisimova, M., Hordijk, W., Gascuel, O.: New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. Syst. Biol. **59**(3), 307–321 (2010). https://doi.org/10.1093/sysbio/syq010
23. Guindon, S., Gascuel, O.: Recent computational advances in maximum-likelihood phylogenetic inference. In: Warnow, T. (ed.) Bioinformatics and Phylogenetics—Seminal Contributions of Bernard Moret. Springer International Publishing AG (2018)
24. Holder, M., Lewis, P.O.: Phylogeny estimation: traditional and Bayesian approaches. Nat. Rev. Genet. **4**(4), 275–284 (2003)
25. Huelsenbeck, J.P., Ronquist, F., Nielsen, R., Bollback, J.P.: Bayesian inference of phylogeny and its impact on evolutionary biology. Science **294**(5550), 2310–2314 (2001). https://doi.org/10.1126/science.1065889

26. Keane, T.M., Naughton, T.J., Travers, S.A.A., McInerney, J.O., McCormack, G.P.: DPRml: distributed phylogeny reconstruction by maximum likelihood. Bioinformatics **21**(7), 969–974 (2005). https://doi.org/10.1093/bioinformatics/bti100
27. Kobert, K., Flouri, T., Aberer, A., Stamatakis, A.: The divisible load balance problem and its application to phylogenetic inference. In: Brown, D., Morgenstern, B. (eds.) Algorithms in Bioinformatics, pp. 204–216. Springer, Berlin Heidelberg (2014)
28. Kozlov, A.: amkozlov/raxml-ng: RAxML-NG v0.6.0 BETA (2018). https://doi.org/10.5281/zenodo.1291478
29. Kozlov, A.M., Aberer, A.J., Stamatakis, A.: ExaML version 3: a tool for phylogenomic analyses on supercomputers. Bioinformatics **31**(15), 2577–2579 (2015). https://doi.org/10.1093/bioinformatics/btv184
30. Miller, M.A., Schwartz, T., Pfeiffer, W.: User behavior and usage patterns for a highly accessed science gateway. In: Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale, pp. 46:1–46:8. ACM (2016). https://doi.org/10.1145/2949550
31. Minh, B.Q., Vinh, L.S., von Haeseler, A., Schmidt, H.A.: pIQPNNI: parallel reconstruction of large maximum likelihood phylogenies. Bioinformatics **21**(19), 3794–3796 (2005). https://doi.org/10.1093/bioinformatics/bti594
32. Moret, B.M., Tang, J., Wang, L.S., Warnow, T.: Steps toward accurate reconstructions of phylogenies from gene-order data. J. Comput. Syst. Sci. **65**(3), 508–525 (2002). https://doi.org/10.1016/S0022-0000(02)00007-7
33. Moret, B.M., Wang, L.S., Warnow, T., Wyman, S.K.: New approaches for reconstructing phylogenies from gene order data. Bioinformatics **17**(suppl1), S165–S173 (2001). https://doi.org/10.1093/bioinformatics/17.suppl_1.S165
34. Moret, B.M.E., Bader, D.A., Warnow, T.: High-performance algorithm engineering for computational phylogenetics. J. Supercomput. **22**(1), 99–111 (2002). https://doi.org/10.1023/A:1014362705613
35. Moret, B.M.E., Lin, Y., Tang, J.: Rearrangements in phylogenetic inference: compare, model, or encode? In: Chauve, C., El-Mabrouk, N., Tannier, E. (eds.) Models and Algorithms for Genome Evolution, pp. 147–171. Springer, London (2013). https://doi.org/10.1007/978-1-4471-5298-9_7
36. Nekrutenko, A., Galaxy Team, Goecks, J., Taylor, J., Blankenberg, D.: Biology needs evolutionary software tools: let's build them right. Mol. Biol. Evol. **35**(6), 1372–1375 (2018). https://doi.org/10.1093/molbev/msy084
37. Nguyen, L.T., Schmidt, H.A., von Haeseler, A., Minh, B.Q.: IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol. Biol. Evol. **32**(1), 268–274 (2015). https://doi.org/10.1093/molbev/msu300
38. Nguyen, N., Mirarab, S., Warnow, T.: MRL and SuperFine+MRL: new supertree methods. Algorithms Mol. Biol. **7**(1), 3 (2012). https://doi.org/10.1186/1748-7188-7-3
39. OMICtools: phylogenetic inference software tools. https://omictools.com/phylogenetic-inference-category?tab=software&page=1. Accessed Oct 2018
40. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree 2 approximately maximum-likelihood trees for large alignments. PLOS ONE **5**(3), 1–10 (2010). https://doi.org/10.1371/journal.pone.0009490
41. Roch, S.: A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**(1), 92 (2006). https://doi.org/10.1109/TCBB.2006.4
42. Ronquist, F., Huelsenbeck, J.P.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**(12), 1572–1574 (2003). https://doi.org/10.1093/bioinformatics/btg180
43. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4), 406–425 (1987)
44. Sankoff, D., Blanchette, M.: The median problem for breakpoints in comparative genomics. In: Jiang, T., Lee, D.T. (eds.) Computing and Combinatorics, pp. 251–263. Springer, Berlin, Heidelberg (1997)
45. Snell, Q., Whiting, M., Clement, M., McLaughlin, D.: Parallel phylogenetic inference. In: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing. IEEE Computer Society (2000)

46. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationship. Univ. Kansas Sci. Bull. **28**, 1409–1438 (1958)
47. Stamatakis, A.: RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics **30**(9), 1312–1313 (2014). https://doi.org/10.1093/bioinformatics/btu033
48. Stamatakis, A.: A review of approaches for optimizing phylogenetic likelihood calculations. In: Warnow, T. (ed.) Bioinformatics and Phylogenetics—Seminal Contributions of Bernard Moret. Springer International Publishing AG (2018)
49. Stewart, C.A., Hart, D., Berry, D.K., Olsen, G.J., Wernert, E.A., Fischer, W.: Parallel implementation and performance of fastDNAml: a program for maximum likelihood phylogenetic inference. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing. ACM (2001). https://doi.org/10.1145/582034.582054
50. Suchard, M.A., Rambaut, A.: Many-core algorithms for statistical phylogenetics. Bioinformatics **25**(11), 1370–1376 (2009). https://doi.org/10.1093/bioinformatics/btp244
51. Tavaré, S.: Some probabilistic and statistical problems in the analysis of DNA sequences. Lect. Math. Life Sci. **17**(2), 57–86 (1986)
52. Yang, Z.: Computational Molecular Evolution. Oxford University Press (2006)
53. Zhou, X., Shen, X.X., Hittinger, C.T., Rokas, A.: Evaluating fast maximum likelihood-based phylogenetic programs using empirical phylogenomic data sets. Mol. Biol. Evol. **35**(2), 486–503 (2018). https://doi.org/10.1093/molbev/msx302
54. Zwickl, D.J.: Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. Ph.D. thesis, The University of Texas at Austin (2006)

Chapter 4
Hands-on Introduction to Sequence-Length Requirements in Phylogenetics

Sébastien Roch

Abstract In this tutorial, through a series of analytical computations and numerical simulations, we review many known insights into a fundamental question: how much data is needed to reconstruct the Tree of Life? A Jupyter notebook and code for this tutorial are provided in Python.

Keywords Phylogenetics · Sequence-length requirements · Distance-based methods · Maximum likelihood estimation

4.1 Introduction

Phylogeny estimation is a central problem in evolutionary biology and beyond [29]. In the most basic form of the problem, one has access to aligned homologous DNA sequences, say from a common gene, across multiple species. The goal is to output a phylogeny that describes the underlying evolutionary relationships. A large number of inference methods have been developed for this problem [31]. Often one relies on the assumption that the data fits a stochastic model of sequence evolution on a tree, under which many methods have been proven to be statistically consistent, i.e., as the amount of data increases, the estimated phylogeny converges to the true phylogeny with probability one.

In order to compare the statistical accuracy of different methods, however, a natural theoretical approach is to analyze the rate at which this convergence occurs. Through a series of analytical computations and numerical simulations, we review some known insights into this fundamental question: how much data is needed to reconstruct the Tree of Life? After some basic definitions, we analyze in detail a simple setting: the three-leaf rooted case under the Cavender–Farris model. Despite its simplicity, this setting already brings to light the important role played by various parameters, in particular, the shortest branch length and the depth, in the difficulty of reconstructing phylogenies. We consider both distance-based and likelihood-based

S. Roch (✉)
Department of Mathematics, University of Wisconsin, Madison, USA
e-mail: roch@math.wisc.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_4

methods, as well as some information-theoretic lower bounds. We subsequently extend these observations to larger trees, emphasizing the role of a different parameter, the branching rate.

Inspired by Bernard Moret's work in this area, which bridges theoretical [32] and empirical [15, 16, 24] perspectives, we also test all mathematical predictions against (admittedly limited) numerical simulations. Code is provided in Python and a Jupyter notebook is available at

`https://github.com/sebroc/seq-len`

Finally, bibliographic information can be found in the last section.

4.2 Definitions

The unknown phylogeny is a tree $T = (V, E)$ whose root R has degree 2 and whose internal vertices have degree 3. We let $\mathbf{T}_n$ be the set of such phylogenies with n leaves.

The sequence data at the leaves $L = \{X_1, \ldots, X_n\}$ is assumed to be generated under the Cavender–Farris (CF) model [2, 11] (also referred to as the Cavender–Farris–Neyman (CFN) model in [31]). The CF model is a two-state model, usually defined on the state space $\{0, 1\}$, but we use instead $\{-1, +1\}$ for reasons that will become clear below. Formally, given branch lengths $l_e \in \mathbb{R}_+$ for $e \in E$, every site $i = 1, \ldots, k$ is distributed independently according to the following process. Pick the root state σ_R^i uniformly at random in $\mathbf{S} = \{-1, +1\}$. A substitution occurs independently on edge e with probability

$$p(l_e) := \frac{1}{2}\left(1 - e^{-2l_e}\right).$$

Let $\tau_e^i = -1$ if a substitution occurs on e on site i, and let $\tau_e^i = +1$ otherwise. The state at U on site i is

$$\sigma_U^i = \sigma_R^i \prod_{e \in P(R,U)} \tau_e^i,$$

where $P(R, U)$ is the set of edges on the path from root R to vertex U. While this representation of the CF model may be unfamiliar to the reader, it will make both analytical derivations and numerical computations more straightforward. Denoted by

$$\sigma_U^{(k)} = (\sigma_U^1, \ldots, \sigma_U^k),$$

the resulting sequence at U and let $\sigma_L^{(k)} = \{\sigma_X^{(k)} : X \in L\}$ be the set of sequences at the leaves. We write $\sigma_L^{(k)} \sim (T, l)^{\otimes k}$ for a sequence dataset with k sites generated at the leaves L of T with branch lengths $l = (l_e : e \in E)$.

A phylogenetic reconstruction algorithm is a collection of maps $\{\mathbf{R}_n^k : \mathbf{S}^{L\times[k]} \to \mathbf{T}_n\}$ from sequence datasets of length k on L to phylogenies with n leaves, for all $n, k \in \mathbb{N}$ (where we used the notation $[k] = \{1, \ldots, k\}$). Such an algorithm is statistically consistent, if for any number of leaves n and any weighted phylogeny on n leaves (T, l), the probability of correct reconstruction goes to 1 as the sequence length k goes to $+\infty$, i.e.,

$$\lim_k \mathbb{P}\left[\mathbf{R}_n^k(\sigma_L^{(k)}) = T\right] = 1,$$

where $\sigma_L^{(k)} \sim (T, l)^{\otimes k}$.

The sequence-length requirement of a consistent reconstruction algorithm $\mathbf{R} = \{\mathbf{R}_n^k :, n, k \in \mathbb{N}\}$ is a natural way to quantify the convergence rate of the success probability as $k \to +\infty$. Fix $\delta \in (0, 1)$. Formally, we define the sequence-length requirement of $\mathbf{R}$ at (T, l) as the smallest integer $K_{\mathbf{R}}(T, l)$ such that

$$\mathbb{P}\left[\mathbf{R}_n^k(\sigma_L^{(k)}) = T\right] > 1 - \delta,$$

for all $k \geq K_{\mathbf{R}}(T, l)$, where $\sigma_L^{(k)} \sim (T, l)^{\otimes k}$. The requirement at a given model (T, l) is not particularly meaningful and we can always achieve perfect reconstruction by simply outputting (T, l) on any dataset. We instead consider a class of phylogenetic models $\mathscr{P}$, e.g., all phylogenies with n leaves and branch lengths in some set of allowed values. We will then define the sequence-length requirement over $\mathscr{P}$ as

$$K_{\mathbf{R}}(\mathscr{P}) = \sup_{(T,l)\in\mathscr{P}} K_{\mathbf{R}}(T, l).$$

Rather than computing $K_{\mathbf{R}}$ explicitly, one typically looks for upper and lower bounds that depend on structural parameters that affect the accuracy of $\mathbf{R}$, namely, the size of the tree, its shortest branch length as well as its depth.

4.3 A Simple Setting

We will mostly focus on the simplest setting, a three-leaf phylogeny under the molecular clock assumption. Despite its simplicity, this setting suffices to illustrate key elementary insights about sequence-length requirements.

Definition 1 *(Three-species setting).* On the set of leaves $L = \{A, B, C\}$, there are three possible rooted topologies denoted, respectively, by $AB|C$, $AC|B$, and $BC|A$, where the first two leaves are "closest." For $T = XY|Z$, let M be the most recent common ancestor of X and Y. We denote by g the lengths $l_{XM} = l_{YM}$ and we denote by f the length l_{MR}, where R is the root. We further assume that $l_{ZR} = g + f$. Notice that all paths from the root to the leaves have the same length—this is the

so-called molecular clock case. We refer to this model as $XY|Z_{g,f}$ and we write $\sigma_L^{(k)} \sim XY|Z_{g,f}^{\otimes k}$ for a corresponding dataset of length k.

We will use numerical simulations (embedded in the text) to illustrate some basic results on sequence-length requirements. Below the function `AB_C` generates `N` sequence datasets of length `k` at the leaves of the tree $T = AB|C$ with parameters `g` and `f`, as defined above. Rather than outputting the sequences themselves, it returns what will turn out to be a more convenient representation, for each pair of leaves X, Y, each site i and each sample the quantity

$$s_{XY}^i = \sigma_X^i \sigma_Y^i,$$

which is -1 if X and Y disagree, and $+1$ otherwise. Note that each assignment of values $s_{AB}^i, s_{AC}^i, s_{BC}^i$, in fact, corresponds to two different sites (by flipping all the signs), but this will not be an issue below. To see how these s_{XY}^i values are generated, note that a different but equivalent expression for s_{XY}^i is $s_{MX}^i s_{MY}^i$, where we use the notation $s_{U_1U_2}^i = \sigma_{U_1}^i \sigma_{U_2}^i$ for any two vertices $U_1, U_2 \in V$. Further, observe that $s_{MA}^i = \tau_{MA}^i$ and $s_{MB}^i = \tau_{MB}^i$ while

$$s_{MC}^i = \tau_{RM}^i \tau_{RC}^i,$$

i.e., there is a substitution between M and C if there is an odd number of substitutions on the path RM, RC. The total length of this path is $g + 2f$ and, as a result, it can be checked (using the computations later in this section, for instance) that the probability that $s_{MC}^i = -1$ is $p(g + 2f)$.

```
from math import exp, sqrt
import numpy as np
np.random.seed(0)

def l2p(l): # branch length to substitution probability
    return (1-exp(-2*l))/2

def sub(p,k,N): # output -1 indicates substitution (o.w. 1)
    return 1 - 2*(np.random.rand(N,k)<p)

def AB_C(g,f,k,N): # generate dataset under AB|C
    sMA, sMB = sub(l2p(g),k,N), sub(l2p(g),k,N)
    sMC = sub(l2p(g+2*f),k,N)
    return sMA*sMB, sMA*sMC, sMB*sMC
```

The class of reconstruction methods that is perhaps easiest to analyze are the distance-based methods, i.e., loosely speaking those methods based on pairwise sequence comparisons. Let

$$\Sigma_{XY}^k = \sum_{i=1}^k s_{XY}^i.$$

Observe that this quantity is positive if and only if X and Y agree on a majority of sites. The uncorrected distance formula under the CF model, i.e., the fraction of differences between the sequences at X and Y is then given by $\frac{1}{2}\left(1-\frac{1}{k}\Sigma_{XY}^k\right)$.

To provide some insights into the sequence-length requirements of distance-based methods, we begin with the following intuitive algorithm **D** over three-species datasets:

Definition 2 *(Algorithm* **D***).* Given sequence data $\sigma_L^{(k)}$, we set $\mathbf{D}(\sigma_L^{(k)}) = XY|Z$ if

$$\min\left\{\Sigma_{XY}^k - \Sigma_{XZ}^k, \Sigma_{XY}^k - \Sigma_{YZ}^k\right\} > 0;$$

and we return a failure if no such pair exists. Notice that at most one pair can satisfy this property. In words, we choose the closest pair to be that whose sequences are most similar.

Numerical simulations: The function `test_pairwise` below implements this method and estimates its accuracy under sequence data of length up to `k` generated under $T = AB|C$ with parameters `g`, `f`. The number of repetitions is `N`. For speed, we reuse the data for sequence length $k'-1$ in the simulation for sequence length k'.

```
def comp(criABvAC,criABvBC): # cumulative comparison across seq
    return np.cumsum(criABvAC,axis=1), np.cumsum(criABvBC,axis=1)

def test_pairwise(g,f,k,N): # testing D under AB|C
    sAB, sAC, sBC = AB_C(g,f,k,N)
    ABvAC, ABvBC = comp(sAB-sAC,sAB-sBC)
    return np.sum(np.logical_and(ABvAC>0, ABvBC>0),axis=0)/N
```

As the next experiment illustrates, the frequency of successful reconstruction by **D** increases to 1 as $k \to +\infty$. That is, the simulation supports (but does not prove) the claim that **D** is a consistent reconstruction algorithm in this simple setting (Fig. 4.1).

```
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 300 # high-resolution figures

# EXP 1: accuracy of pairwise comparisons v. k
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 300 # high-resolution figures

g, f, k, N = 0.1, 0.05, 500, 1000

freq_succ_pw = test_pairwise(g, f, k, N)

plt.plot(np.arange(1,k+1),freq_succ_pw);
plt.xlabel('Sequence Length'), plt.ylabel('Success Probability');
```

Analytical derivation: In fact, consistency is straightforward to establish analytically in this case. Indeed, recall that $s_{AB}^i = s_{MA}^i s_{MB}^i = \tau_{MA}^i \tau_{MB}^i$. Define

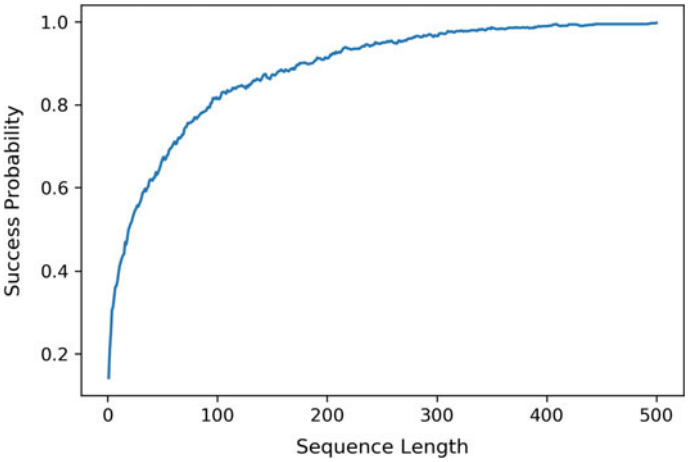


Fig. 4.1 Success probability of **D** as function of sequence length

$$\mathbb{E}\left[\tau^i_{MA}\right] = [+1](1 - p(g)) + [-1]p(g) = e^{-2g} =: \theta(g).$$

Because edge substitutions are independent, it follows that

$$\mathbb{E}\left[s^i_{AB}\right] = \mathbb{E}\left[\tau^i_{MA}\tau^i_{MB}\right] = \theta(g)^2.$$

Similarly, $\mathbb{E}\left[s^i_{AC}\right] = \theta(g + 2f)^2$ and $\mathbb{E}\left[s^i_{BC}\right] = \theta(g + 2f)^2$. Hence, by the law of large numbers, as $k \to +\infty$ it holds that

$$\frac{1}{k}\Sigma^k_{AB} \to \theta(g), \quad \frac{1}{k}\Sigma^k_{AC} \to \theta(g + 2f), \quad \frac{1}{k}\Sigma^k_{BC} \to \theta(g + 2f).$$

Since θ is strictly decreasing in its argument, the last observation implies that Σ^k_{AB} is eventually larger than both Σ^k_{AC} and Σ^k_{BC} with probability 1, establishing consistency.

Claim 1 (Consistency of **D**). In the three-species setting, the algorithm **D** is statistically consistent.

In the next section, we consider the rate of convergence of **D**.

4.4 Phylogenetic Signal

As pointed out earlier, two structural parameters that affect the sequence-length requirement of reconstruction algorithms are the shortest branch length and the depth of a phylogeny. We study them, in turn, in the three-leaf case. We do not compute the

sequence-length requirement explicitly—rather we obtain upper bounds depending on g and f. In a subsequent section, we also provide lower bounds.

Analytical derivation: Let $T = AB|C$, assume that $\sigma_L^{(k)} \sim AB|C_{g,f}^{\otimes k}$ and let Σ_{XY}^k be defined as above. For the distance-based method **D** to succeed, it must be that events $\mathscr{E}_{AC} = \{\Sigma_{AB}^k - \Sigma_{AC}^k > 0\}$ and $\mathscr{E}_{BC} = \{\Sigma_{AB}^k - \Sigma_{BC}^k > 0\}$ hold simultaneously. To get an upper bound on this probability, we appeal to a standard concentration result, Hoeffding's inequality (see e.g. [23]), which states that if $W_1, \ldots, W_k$ are independent, respectively, $[\alpha_i, \beta_i]$-valued random variables, then for all $\varepsilon > 0$

$$\mathbb{P}\left[\sum_{i=1}^{k}(W_i - \mathbb{E}[W_i]) \geq k\varepsilon\right] \leq \exp\left(-\frac{2k^2\varepsilon^2}{\sum_{i=1}^{k}(\beta_i - \alpha_i)^2}\right).$$

Hence, rewriting

$$\mathbb{P}\left[\mathscr{E}_{AC}^c\right] = \mathbb{P}\left[\sum_{i=1}^{k}\left(s_{AC}^i - \theta(g+f) - s_{AB}^i + \theta(g)\right) \geq k\left(\theta(g) - \theta(g+2f)\right)\right],$$

and applying Hoeffding's inequality, we obtain

$$\mathbb{P}\left[\mathscr{E}_{AC}^c\right] \leq \exp\left(-\frac{2k^2\left[\theta(g) - \theta(g+2f)\right]^2}{k(2)^2}\right) = \exp\left(-\frac{k}{2}\left[\theta(g) - \theta(g+2f)\right]^2\right). \tag{4.1}$$

By a union bound,

$$\mathbb{P}\left[\mathbf{D}(\sigma_L^{(k)}) = T\right] = 1 - \mathbb{P}\left[\mathscr{E}_{AC}^c \cup \mathscr{E}_{BC}^c\right] \geq 1 - 2\exp\left(-\frac{k}{2}\left[\theta(g) - \theta(g+2f)\right]^2\right).$$

Observe that

$$\theta(g) - \theta(g+2f) = e^{-2g}\left(1 - e^{-4f}\right),$$

so that if

$$k \geq \bar{\kappa}_{\mathbf{D}}(g, f) := \frac{2\ln(2/\delta)}{e^{-4g}\left(1 - e^{-4f}\right)^2},$$

then **D** succeeds with probability greater than $1 - \delta$. That is, the sequence-length requirement of **D** at $AB|C_{g,f}$ is smaller than $\bar{\kappa}_{\mathbf{D}}(g, f)$.

This bound extends to a much larger class of phylogenetic models by symmetry over the topologies and by the monotonicity of $\bar{\kappa}_{\mathbf{D}}(g, f)$ in g and f.

Claim 2 (Sequence-length requirement of **D**). Let

$$\mathscr{P} = \bigcup_{g' \leq g, f' \geq f} \{AB|C_{g',f'}, AC|B_{g',f'}, BC|A_{g',f'}\}.$$

We have the following upper bound on the sequence-length requirement of **D** over $\mathscr{P}$

$$K_{\mathbf{D}}(\mathscr{P}) \leq \frac{2\ln(2/\delta)}{e^{-4g}\left(1-e^{-4f}\right)^2}.$$

4.4.1 Short Branches

Short branches affect sample complexity, as we show next.

Numerical simulations: When fixing δ, g and taking $f \to 0$, a Taylor expansion of the denominator shows that $\bar{\kappa}_{\mathbf{D}}(g, f)$ scales like $\propto f^{-2}$. The next experiment illustrates this point. Here, under $T = AB|C$ for a fixed value of g and an array `f_arr` of values of f, the smallest sequence length to achieve the `target` value for $1 - \delta$ is identified. That produces an empirical estimate of $K_{\mathbf{D}}(AB|C_{g,f})$. In a plot of $\log K_{\mathbf{D}}$ v. $\log f$, the slope can be seen to be somewhat close to -2, the theoretical prediction (Fig. 4.2).

```
# EXP 2: requirement for pairwise comparisons v. f
g, k, N, target = 0.1, 1500, 1000, 0.95
f_arr_min, f_arr_max, f_arr_len = 0.025, 0.075, 10
f_arr = np.linspace(f_arr_min, f_arr_max, num=f_arr_len)

k_thres_f = np.zeros(f_arr_len)
for i in range(f_arr_len):
    freq_succ = test_pairwise(g, f_arr[i], k, N)
    k_thres_f[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(np.log(f_arr), np.log(k_thres_f));
plt.xlabel('Log f'), plt.ylabel('Log Length Required');
```

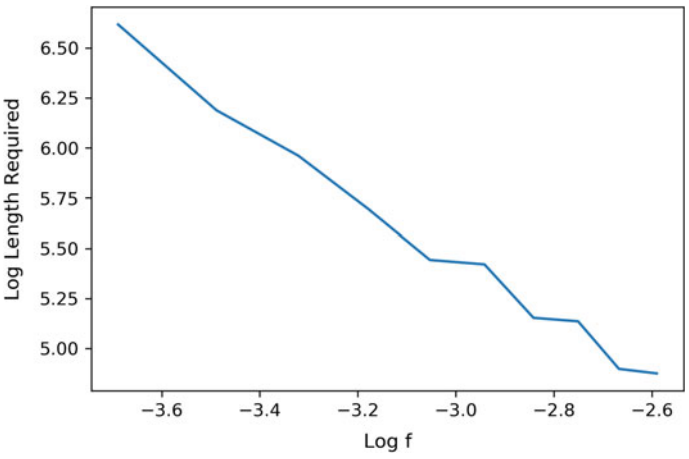


Fig. 4.2 Sequence-length requirement of **D** as function of f

4.4.2 Depth

The depth of the phylogeny also affects sample complexity. By "depth," in the current setting we mean the length of the paths from the root to the leaves, which when f is small is roughly g.

Numerical simulations: Similarly, fixing δ, f in the previous expression gives that $\bar{\kappa}_{\mathbf{D}}(g, f)$ scales like $\propto e^{4g}$. The next experiment illustrates this point. Here, for an array `g_arr` of values of g, the smallest sequence length to achieve the `target` value for $1 - \delta$ is identified. In a plot of $\log K_{\mathbf{D}}$ v. g, we observe a roughly linear relationship, as expected (Fig. 4.3).

```
# EXP 3: requirement for pairwise comparisons v. g
f, k, N = 0.05, 1500, 1000
g_arr_min, g_arr_max, g_arr_len, target = 0.01, 0.2, 10, 0.95
g_arr = np.linspace(g_arr_min, g_arr_max, num=g_arr_len)

k_thres_g = np.zeros(g_arr_len)
for i in range(g_arr_len):
    freq_succ = test_pairwise(g_arr[i], f, k, N)
    k_thres_g[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(g_arr, np.log(k_thres_g));
plt.xlabel('g'), plt.ylabel('Log Length Required');
```

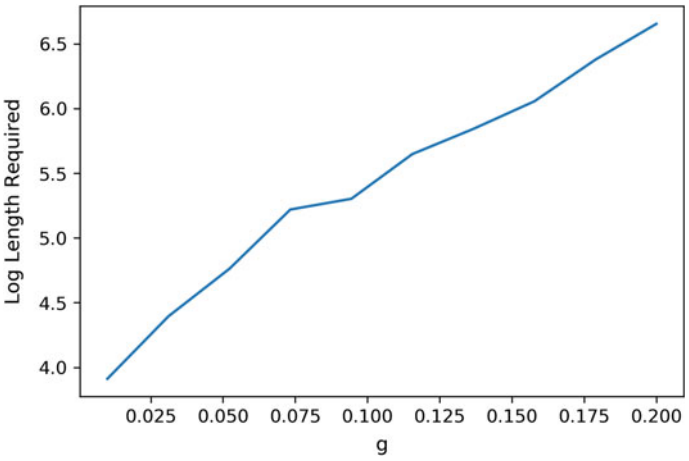


Fig. 4.3 Sequence-length requirement of **D** as function of g

4.5 Not All Reconstruction Methods are Created Equal

Sequence-length requirements are useful to compare reconstruction methods: by definition, a higher requirement indicates more data is needed to achieve the same accuracy. We give a simple (albeit artificial) example.

Consider the following modification of the distance-based method **D**. Assuming k is even. Define

$$\Pi^k_{XY\vee XZ} = \sum_{\substack{i=1\\ i \text{ odd}}}^{k} (s^i_{XY} - s^i_{XZ})(s^{i+1}_{XY} - s^{i+1}_{XZ}),$$

and

$$\Pi^k_{XY\vee YZ} = \sum_{\substack{i=1\\ i \text{ odd}}}^{k} (s^i_{XY} - s^i_{YZ})(s^{i+1}_{XY} - s^{i+1}_{YZ}).$$

The reconstruction algorithm $\mathbf{D}^2$ then proceeds as follows:

Definition 3 *(Algorithm $\mathbf{D}^2$).* Given sequence data $\sigma_L^{(k)}$, we set $\mathbf{D}^2(\sigma_L^{(k)}) = XY|Z$ if

$$\min\left\{\Pi^k_{XY\vee XZ}, \Pi^k_{XY\vee YZ}\right\} > \eta_{g,f} := \frac{1}{2}e^{-4g}\left(1 - e^{-4f}\right)^2;$$

and we return a failure if no such pair exists or if more than one pair satisfies this property. Note that this reconstruction algorithm requires knowledge of (or bounds on) g and f.

Analytical derivation: To see that $\mathbf{D}^2$ is consistent, let again $T = AB|C$ and $\sigma_L^{(k)} \sim AB|C^{\otimes k}_{g,f}$. Notice that, by independence of the odd and even sites, it holds that for i odd

$$\mathbb{E}\left[(s^i_{AB} - s^i_{AC})(s^{i+1}_{AB} - s^{i+1}_{AC})\right] = \mathbb{E}\left[s^i_{AB} - s^i_{AC}\right]\mathbb{E}\left[s^{i+1}_{AB} - s^{i+1}_{AC}\right] = [\theta(g) - \theta(g+2f)]^2.$$

Similarly

$$\mathbb{E}\left[(s^i_{AB} - s^i_{BC})(s^{i+1}_{AB} - s^{i+1}_{BC})\right] = [\theta(g) - \theta(g+2f)]^2$$

and

$$\mathbb{E}\left[(s^i_{AC} - s^i_{BC})(s^{i+1}_{AC} - s^{i+1}_{BC})\right] = 0.$$

By the law of large xfnumbers, we obtain

$$\frac{2}{k}\Pi^k_{AC\vee AB} = \frac{2}{k}\Pi^k_{AB\vee AC} \to [\theta(g) - \theta(g+2f)]^2$$

and

$$\frac{2}{k}\Pi^k_{BC\vee AB} = \frac{2}{k}\Pi^k_{AB\vee BC} \to [\theta(g) - \theta(g+2f)]^2,$$

while

$$\frac{2}{k}\Pi^k_{BC\mathrm{v}AC} = \frac{2}{k}\Pi^k_{AC\mathrm{v}BC} \to 0.$$

Since

$$[\theta(g) - \theta(g+2f)]^2 > \frac{1}{2}e^{-4g}\left(1-e^{-4f}\right)^2 > 0,$$

that establishes consistency.

Claim 3 (Consistency of $\mathbf{D}^2$). In the three-species setting, the algorithm $\mathbf{D}^2$ is statistically consistent.

We now derive an upper bound on the sequence-length requirement of $\mathbf{D}^2$. Consider the events

$$\mathscr{E}_{AB\mathrm{v}AC} = \left\{\Pi^k_{AB\mathrm{v}AC} > \eta_{g,f}\right\}, \quad \mathscr{E}_{AB\mathrm{v}BC} = \left\{\Pi^k_{AB\mathrm{v}BC} > \eta_{g,f}\right\},$$

and

$$\mathscr{E}_{AC\mathrm{v}BC}\left\{\Pi^k_{AC\mathrm{v}BC} < \eta_{g,f}\right\}.$$

The event $\mathscr{E}_{AB\mathrm{v}AC} \cup \mathscr{E}_{AB\mathrm{v}BC} \cup \mathscr{E}_{AC\mathrm{v}BC}$ implies that the output of $\mathbf{D}^2$ is correct. On rewriting, we obtain

$$\mathbb{P}\left[\mathscr{E}^c_{AB\mathrm{v}AC}\right] = \mathbb{P}\left[\Pi^k_{AB\mathrm{v}AC} - \mathbb{E}\left[\Pi^k_{AB\mathrm{v}AC}\right] \le -\frac{1}{2}e^{-4g}\left(1-e^{-4f}\right)^2\right],$$

and, applying Hoeffding's inequality, we get the upper bound

$$\mathbb{P}\left[\mathscr{E}^c_{AB\mathrm{v}AC}\right] \le \exp\left(-\frac{2(k/2)^2\left[\frac{1}{2}e^{-4g}\left(1-e^{-4f}\right)^2\right]^2}{k/2\times(2)^2}\right),$$

and similarly for the events $\mathscr{E}^c_{AB\mathrm{v}BC}$, $\mathscr{E}^c_{AC\mathrm{v}BC}$. By a union bound,

$$\mathbb{P}\left[\mathbf{D}^2(\sigma^{(k)}_L) = T\right] \ge 1 - 3\exp\left(-\frac{k}{16}e^{-8g}\left(1-e^{-4f}\right)^4\right).$$

So if

$$k \ge \bar{\kappa}_{\mathbf{D}^2}(g,f) := \frac{16\ln(3/\delta)}{e^{-8g}\left(1-e^{-4f}\right)^4},$$

then $\mathbf{D}^2$ succeeds with probability greater than $1-\delta$. That is, the sequence-length requirement of $\mathbf{D}^2$ at $AB|C_{g,f}$ is at most $\bar{\kappa}_{\mathbf{D}^2}(g,f)$.

Claim 4 (Sequence-length requirement of $\mathbf{D}^2$). Let

$$\mathscr{P} = \bigcup_{g' \le g, f' \ge f} \{AB|C_{g',f'}, AC|B_{g',f'}, BC|A_{g',f'}\}.$$

We have the following upper bound on the sequence-length requirement of $\mathbf{D}^2$ over $\mathscr{P}$

$$K_{\mathbf{D}^2}(\mathscr{P}) \le \frac{16 \ln(3/\delta)}{e^{-8g}\left(1 - e^{-4f}\right)^4}.$$

Numerical simulations: Now, notice that as $f \to 0$ (leaving g and δ fixed), we have the asymptotic behavior $\bar{\kappa}_{\mathbf{D}^2}(g, f) \propto f^{-4}$, which is worse than what we obtained for $\mathbf{D}$ (see Claim 2). In words, this bound suggests that $\mathbf{D}^2$ requires significantly more data than $\mathbf{D}$ to achieve the same accuracy, if we divide f by 2, for instance, $\mathbf{D}$ requires roughly four times as much data, while $\mathbf{D}^2$ requires 16 times as much. This is only an upper bound of course. We use a numerical simulation to test the theoretical prediction. The following function `test_two_site` implements $\mathbf{D}^2$ on $T = AB|C$ with parameters `g` and `f`, and tests it `N` times across sequences of length up to `k`.

```
def test_two_site(g, f, k, N): # testing D^2 under AB|C
    sABo, sACo, sBCo = AB_C(g,f,k//2,N)
    sABe, sACe, sBCe = AB_C(g,f,k//2,N)
    ABvAC, ABvBC = comp((sABo-sACo)*(sABe-sACe),
                        (sABo-sACo)*(sABe-sACe))
    eta = (1/2)*exp(-4*g)*(1-exp(-4*f))**2
    return np.sum(np.logical_and(ABvAC>eta, ABvBC>eta),axis=0)/N
```

The following experiment is consistent with our bounds on the sequence-length requirements of $\mathbf{D}$ and $\mathbf{D}^2$. Below, a plot of $\log K_{\mathbf{D}^2}$ v. $\log f$ (solid line) shows a roughly linear behavior with slope close to -4. A plot of $\log K_{\mathbf{D}}$ v. $\log f$ (dotted line) is also reproduced for comparison (Fig. 4.4).

```
# EXP 4: D^2 has requirement f^-4 (takes a minute or two)
g, f, k, N = 0.1, 0.05, 240000, 200

k_thres_f2 = np.zeros(f_arr_len)
for i in range(f_arr_len):
    freq_succ = test_two_site(g, f_arr[i], k, N)
    k_thres_f2[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(np.log(f_arr),np.log(k_thres_f2));
plt.plot(np.log(f_arr),np.log(k_thres_f),':');
plt.xlabel('Log f'), plt.ylabel('Log Length Required');
```

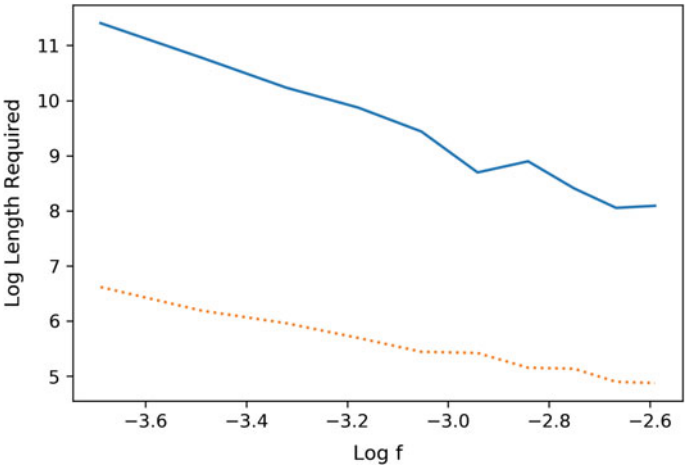


Fig. 4.4 Sequence-length requirement of $\mathbf{D}^2$ (solid) and $\mathbf{D}$ (dotted) as functions of f

4.6 What About Maximum Likelihood Estimation?

So far, we have analyzed pairwise comparison methods. Another common approach in practice is maximum likelihood estimation (MLE), which takes into account the full empirical distribution at the leaves. In the three-leaf case considered previously, the MLE is defined as follows:

$$\mathbf{L}(\sigma_L^{(k)}) = \operatorname{argmax} \left\{ \sup_{g_0, f_0} \log \mathscr{L}^k(\sigma_L^{(k)}; T_0, g_0, f_0) : T_0 \in \{AB|C, AC|B, BC|A\} \right\},$$

where $\mathscr{L}^k(\sigma_L^{(k)}; T_0, g_0, f_0)$ is the likelihood, i.e., the probability of observing the data $\sigma_L^{(k)}$ under the tree T_0 with parameters g_0, f_0. For simplicity, if more than one tree achieves the maximum, we return a failure. How does the sequence-length requirement of $\mathbf{L}$ compared to that of $\mathbf{D}$?

To study this question, we note first that, in this case, the log-likelihood takes a simple analytical form. Let $T_0 = XY|Z$ with parameters g_0 and f_0 and let M be the most recent common ancestor of X and Y. By independence of the sites, the probability of observing $\sigma_L^{(k)}$ is the product of the probabilities of observing the σ_L^i's, which after taking a logarithm becomes a sum

$$\log \mathscr{L}^k(\sigma_L^{(k)}; T_0, g_0, f_0) = \sum_{i=1}^{k} \log \mathscr{L}^1(\sigma_L^i; T_0, g_0, f_0).$$

Let $p_0 = p(g_0)$ and $q_0 = p(g_0 + 2f_0)$, and define

$$I^i_{XY} = \frac{1 + s^i_{XY}}{2}, \qquad I^i_{XZ} = \frac{1 + s^i_{XZ}}{2},$$

i.e., $I^i_{XY} = 1$ if X and Y agree on site i and $I^i_{XY} = 0$ otherwise. In terms of I^i_{XY} and I^i_{XZ} (which are functions of the data σ^i_L), the log-likelihood is

$$\log \mathscr{L}^1(\sigma^i_L; T_0, g_0, f_0) = \log\left(\frac{1}{2}\Lambda_1 + \frac{1}{2}\Lambda_2\right),$$

where

$$\Lambda_1 = (1-p_0)(1-p_0)^{I^i_{XY}} p_0^{1-I^i_{XY}} (1-q_0)^{I^i_{XZ}} q_0^{1-I^i_{XZ}}$$

and

$$\Lambda_2 = p_0(1-p_0)^{1-I^i_{XY}} p_0^{I^i_{XY}} (1-q_0)^{1-I^i_{XZ}} q_0^{I^i_{XZ}}.$$

The above expression is obtained by considering whether or not there is a substitution along the edge XM, followed by whether or not there are substitutions along edge MY and path MR, RZ. Note that all these substitutions are independent.

4.6.1 Likelihood Ratio Test

Analyzing the behavior of **L** is somewhat complicated by the need to optimize over the nuisance parameters g and f. To get some insight, it is easier to start by assuming that g and f are known and that the true topology is either $AB|C$ or $AC|B$. That is, we consider the class of phylogenetic models $\mathscr{P} = \{AB|C_{g,f}, AC|B_{g,f}\}$. Then, the MLE is obtained by identifying the topology among these two with largest log-likelihood. To simplify the notation, define

$$\mathscr{L}^k_{XY}(\sigma^{(k)}_L) = \mathscr{L}^k(\sigma^{(k)}_L; XY|Z, g, f).$$

Consider the following modification $\mathbf{L}'$ of **L**.

Definition 4 *(Algorithm $\mathbf{L}'$).* Given sequence data $\sigma^{(k)}_L$, we return $AB|C$ if

$$\log \mathscr{L}^k_{AB}(\sigma^{(k)}_L) > \log \mathscr{L}^k_{AC}(\sigma^{(k)}_L);$$

$AC|B$ if

$$\log \mathscr{L}^k_{AC}(\sigma^{(k)}_L) > \log \mathscr{L}^k_{AB}(\sigma^{(k)}_L);$$

or, otherwise, we choose uniformly at random between the two. This is also known as a likelihood ratio test (LRT); see for example [1].

Analytical derivation: While we could use Hoeffding's inequality again to analyze the sequence-length requirement of $\mathbf{L}'$, we will introduce instead a comparison argu-

ment to distance-based methods that generalizes more easily to larger, more complex phylogenies. The comparison works as follows. Let Ψ be an arbitrary randomized test for deciding whether sequence data $\sigma_L^{(k)}$ was generated by $AB|C_{g,f}$ or $AC|B_{g,f}$, i.e., for each $\sigma_L^{(k)} \in S^{L\times[k]}$, $\Psi(\sigma_L^{(k)})$ is a $\{AB|C, AC|B\}$-valued random variable. The sum of so-called Type I and Type II errors is defined as

$$\mathscr{W}_{I,II}[\Psi] = \mathscr{W}_{E,AB}[\Psi] + \mathscr{W}_{E,AC}[\Psi],$$

where $\mathscr{W}_{E,AB}[\Psi]$ is the probability of error under $AB|C_{g,f}$

$$\mathscr{W}_{E,AB}[\Psi] = \sum_{\sigma_L^{(k)}} \mathscr{L}_{AB}^k(\sigma_L^{(k)})(1 - \mathbb{P}[\Psi(\sigma_L^{(k)}) = AB|C]),$$

and, similarly, $\mathscr{W}_{E,AC}[\Psi]$ is the probability of error if the data had been generated instead under $AC|B_{g,f}$

$$\mathscr{W}_{E,AC}[\Psi] = \sum_{\sigma_L^{(k)}} \mathscr{L}_{AC}^k(\sigma_L^{(k)})\mathbb{P}[\Psi(\sigma_L^{(k)}) = AB|C].$$

It is a standard fact of statistical theory that $\mathscr{W}_{I,II}[\Psi]$ is minimized by $\Psi = \mathbf{L}'$ (as can easily be derived by inspecting the expression for $\mathscr{W}_{I,II}$).

We use this fact to get a bound on the probability of error of $\mathbf{L}'$. We will need a simple observation first. Notice that, by symmetry, for $\Psi = \mathbf{L}'$, we have $\mathscr{W}_{E,AB}[\Psi] = \mathscr{W}_{E,AC}[\Psi]$. Hence,

$$\mathbb{P}[\mathbf{L}'(\sigma_L^{(k)}) \neq AB|C] = \frac{1}{2}\mathscr{W}_{I,II}[\mathbf{L}'] \leq \frac{1}{2}\mathscr{W}_{I,II}[\Psi],$$

for any Ψ. Now choose Ψ to be the following modification $\mathbf{D}'$ of $\mathbf{D}$: we return $AB|C$ if $\Sigma_{AB}^k > \Sigma_{AC}^k$; $AC|B$ if $\Sigma_{AC}^k > \Sigma_{AB}^k$; or otherwise, we choose uniformly at random between the two. Because the symmetry argument above holds for $\mathbf{D}'$ as well, we finally get

$$\mathbb{P}[\mathbf{L}'(\sigma_L^{(k)}) \neq AB|C] \leq \mathbb{P}[\mathbf{D}'(\sigma_L^{(k)}) \neq AB|C].$$

In words, the probability of failure of $\mathbf{L}'$ is at most that of $\mathbf{D}'$. We have already shown (see (4.1)) that the right-hand side is $\leq \exp\left(-\frac{k}{2}[\theta(g) - \theta(g+2f)]^2\right)$, which is less than δ if

$$k \geq \frac{2\ln(1/\delta)}{e^{-4g}\left(1 - e^{-4f}\right)^2}.$$

In essence, this argument implies that over this restricted class $\mathscr{P}$ the sequence-length requirement of the MLE is at most that of the distance-based method.

Claim 5 (Sequence-length requirement of $\mathbf{L}'$). Let

$$\mathscr{P} = \{AB|C_{g,f}, AC|B_{g,f}\}.$$

We have the following upper bound on the sequence-length requirement of $\mathbf{L}'$ over $\mathscr{P}$

$$K_{\mathbf{L}'}(\mathscr{P}) \leq \frac{2\ln(1/\delta)}{e^{-4g}\left(1-e^{-4f}\right)^2}.$$

Numerical simulations: We further explore this prediction in a simulation. The function `test_lrt` below is, in fact, more general than the LRT described above (in order to obtain a fair comparison to `test_pairwise`), as it compares the log-likelihood of all three topologies $AB|C$, $AC|B$, and $BC|A$ for fixed parameters `g` and `f` (which we refer to as the "three-way" LRT.). It tests how often the log-likelihood of $AB|C$ is strictly larger than that of the other two.

```
def s2i(s): # converts {-1,+1} to {0,1}
    return (1+s)//2

def llXY_Z(g,f,sXY,sXZ): # log-likelihood under XY|Z
    p, q, iXY, iXZ = l2p(g), l2p(g+2*f), s2i(sXY), s2i(sXZ)
    L1=(1-p)*((1-p)**iXY)*(p**(1-iXY))*((1-q)**iXZ)*(q**(1-iXZ))
    L2=p*((1-p)**(1-iXY))*(p**iXY)*((1-q)**(1-iXZ))*(q**iXZ)
    return np.log((1/2)*L1+(1/2)*L2)

def test_lrt(g,f,g0,f0,k,N): # testing LRT under AB|C
    sAB, sAC, sBC = AB_C(g,f,k,N)
    llAB = llXY_Z(g,f,sAB,sAC)
    llAC, llBC = llXY_Z(g0,f0,sAC,sAB), llXY_Z(g0,f0,sBC,sAB)
    ABvAC, ABvBC = comp(llAB-llAC,llAB-llBC)
    return np.sum(np.logical_and(ABvAC>0, ABvBC>0),axis=0)/N
```

The following experiment indicates that the probabilities of success of the basic likelihood-based (solid line) and distance-based (dotted line) methods are very similar in this setting for all sequence lengths (Fig. 4.5).

```
# EXP 5: success of LRT v. k
g, f, k, N = 0.1, 0.05, 500, 1000 # params used for freq_succ_pw

freq_succ_ll = test_lrt(g,f,g,f,k,N)

plt.plot(np.arange(1,k+1),freq_succ_ll);
plt.plot(np.arange(1,k+1),freq_succ_pw,':');
plt.xlabel('Sequence Length'), plt.ylabel('Success Probability');
```

We plot next the sequence length required for the three-way LRT to succeed with probability at least `target` as f varies over an array of values `f_arr`. The results are consistent with a requirement scaling as $\propto f^{-2}$ (Fig. 4.6).

```
# EXP 6: requirement for LRT v. f
g, f, k, N, target = 0.1, 0.05, 2500, 1000, 0.95
f_arr_min, f_arr_max, f_arr_len = 0.025, 0.075, 10 #as k_thres_f
f_arr = np.linspace(f_arr_min, f_arr_max, num=f_arr_len)
```

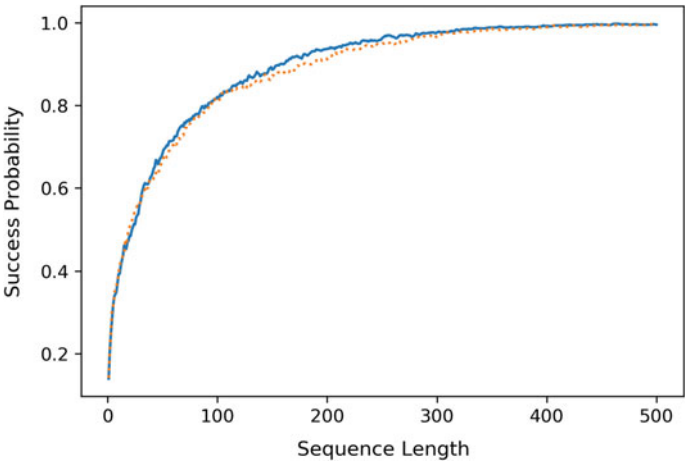


Fig. 4.5 Success probability of likelihood-based (solid) and distance-based (dotted) approaches as functions of sequence length

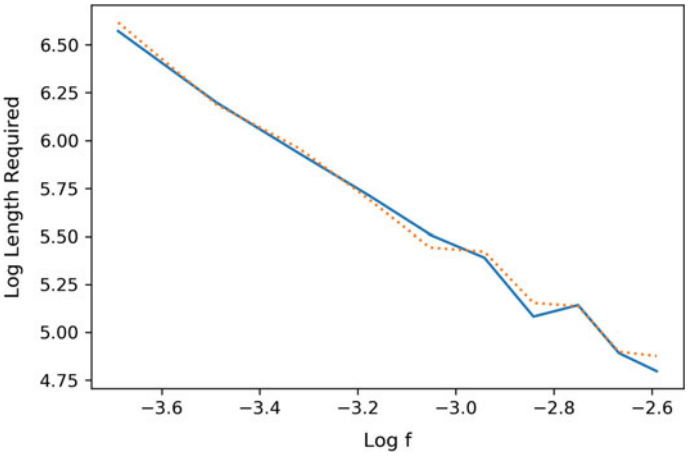


Fig. 4.6 Sequence-length requirement of likelihood-based (solid) and distance-based (dotted) approaches as functions of f

```
k_thres_fll = np.zeros(f_arr_len)
for i in range(f_arr_len):
    freq_succ = test_lrt(g,f_arr[i],g,f_arr[i],k,N)
    k_thres_fll[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(np.log(f_arr), np.log(k_thres_fll));
plt.plot(np.log(f_arr),np.log(k_thres_f),':');
plt.xlabel('Log f'), plt.ylabel('Log Length Required');
```

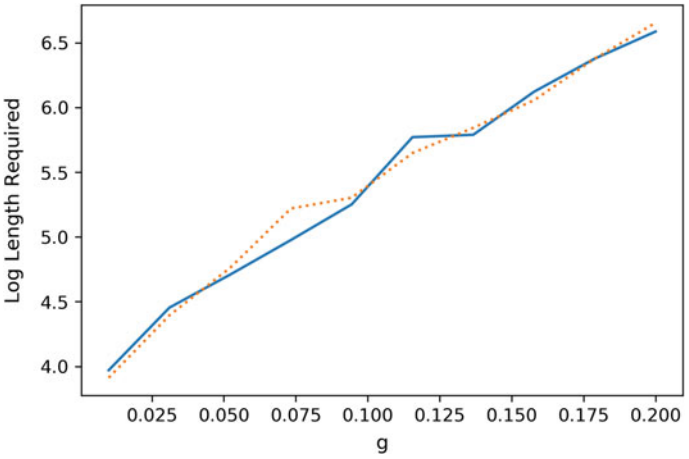


Fig. 4.7 Sequence-length requirement of likelihood-based (solid) and distance-based (dotted) approaches as functions of g

We plot next the sequence length required for the three-way LRT to succeed with probability at least `target` as g varies over an array of values `g_arr`. The results are consistent with a requirement scaling as exponentially in g (Fig. 4.7).

```
# EXP 7: requirement for LRT v. f
f, k, N, target = 0.05, 2500, 1000, 0.95
g_arr_min, g_arr_max, g_arr_len = 0.01, 0.2, 10 #as k_thres_g
g_arr = np.linspace(g_arr_min, g_arr_max, num=g_arr_len)


k_thres_gll = np.zeros(g_arr_len)
for i in range(g_arr_len):
    freq_succ = test_lrt(g_arr[i],f,g_arr[i],f,k,N)
    k_thres_gll[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(g_arr,np.log(k_thres_gll));
plt.plot(g_arr,np.log(k_thres_g),':');
plt.xlabel('g'), plt.ylabel('Log Length Required');
```

4.6.2 *Optimizing the Branch Lengths*

Up to this point, we have ignored the effect of branch length estimation on the MLE. To get some partial insight into this difficult, but important issue, we consider a modified setting: we generate sequence datasets according to $AB|C_{g,f}$ and study how often the log-likelihood under $AB|C$ exceeds that of the alternative $AC|B$ — with the optimal choice of branch lengths in both cases in the limit $k \to +\infty$. For

$AB|C$, the choices g and f are optimal under the expected log-likelihood by standard results in statistical theory (namely, Gibbs' or information inequality; see e.g. [3]). For $AC|B$, however, it is not immediate what the right choice of branch lengths is when the data is generated under $AB|C$.

Numerical simulations: We first run an experiment which estimates the log-likelihood for the model $AC|B$ over a grid of branch lengths g and f with a large value of k. The contour plot below, obtained under $AB|C$ with parameters `0.1` and `0.05`, suggests that, in this case, the optimal f is 0 while the optimal g is somewhat larger than 0.1 (Fig. 4.8).

```
# EXP 8: a better choice of branch lengths for alternative
g, f, k, N, m_gr = 0.1, 0.05, 10000, 1, 50
f_gr = np.linspace(0, 0.05, num=m_gr)
g_gr = np.linspace(0.05, 0.20, num=m_gr)

sAB, sAC, sBC = AB_C(g,f,k,N)
ll_gf = np.zeros((m_gr,m_gr))
for i_f in range(m_gr):
    for i_g in range(m_gr):
        ll_gf[i_f,i_g] = np.sum(llXY_Z(g_gr[i_g],
                                f_gr[i_f],sAC,sAB))/k

optf=f_gr[np.unravel_index(np.argmax(ll_gf),np.shape(ll_gf))[0]]
optg=g_gr[np.unravel_index(np.argmax(ll_gf),np.shape(ll_gf))[1]]
print(f'Optimal f = {optf}')
print(f'Optimal g = {optg}')
[X, Y], Z = np.meshgrid(g_gr,f_gr), ll_gf
CS = plt.contour(X,Y,Z);
plt.clabel(CS), plt.xlabel('g'), plt.ylabel('f');


Optimal f = 0.0
Optimal g = 0.12959183673469388
```

Analytical derivation: In other words, the experiment above indicates that for those parameters the star tree achieves the optimum under the alternative topology. We confirm this heuristically in the limit of small branch lengths. Note that, for g, f small, we have $p(g) = g + O(g^2)$ and $p(f) = f + O(f^2)$. Second, in this asymptotic setting, the first-order contributions to the log-likelihood are those realizations involving a single substitution (except for the constant site). Using these two observations and the expression for the log-likelihood derived at the beginning of the section, it can be shown that the expected log-likelihood under the model $AC|B_{g_0,f_0}$ (for g_0, f_0 small) given data generated under $AB|C_{g,f}$ is to the first order

$$\widetilde{\log \mathscr{L}}_0(g_0, f_0) = (-3g_0 - 2f_0) + (2g + 2f)\log g_0 + g\log(g_0 + 2f_0),$$

where, for example, the last term corresponds to A disagreeing with B but agreeing with C. The first term corresponds to the constant sites, where we used $\log(1 -$

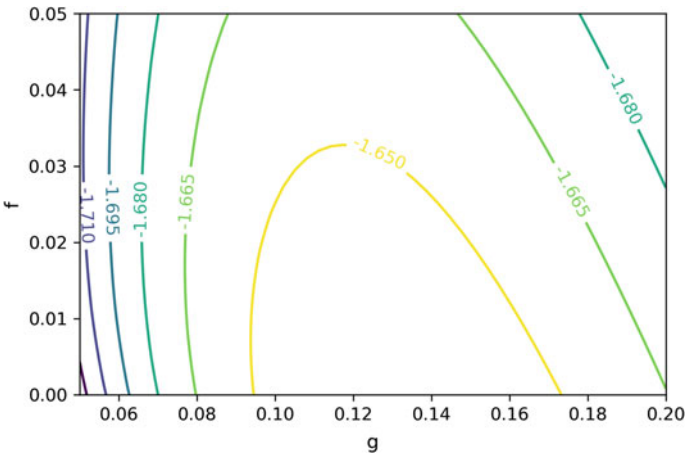


Fig. 4.8 Approximate expected log-likelihood for alternative topology as function of parameters g and f

$x) = -x + O(x^2)$ and ignored second-order contributions. We seek to maximize $\widetilde{\log \mathscr{L}}_0(g_0, f_0)$ for fixed g and f. The partial derivative with respect to f_0 is

$$\partial_{f_0} \widetilde{\log \mathscr{L}}_0(g_0, f_0) = -2 + 2\frac{g}{g_0 + 2f_0}.$$

Hence, for $g_0 < g$, $\partial_{f_0} \widetilde{\log \mathscr{L}}_0(g_0, f_0) = 0$ when f_0 satisfies $g = g_0 + 2f_0$. While for $g_0 \geq g$, $\partial_{f_0} \widetilde{\log \mathscr{L}}_0(g_0, f_0) < 0$ for all $f_0 \geq 0$ and the optimal f_0 for fixed g_0 is 0. We plug back this optimal f_0 into $\widetilde{\log \mathscr{L}}_0(g_0, f_0)$ and consider the two cases again, when $g_0 < g$,

$$\frac{\mathrm{d}}{\mathrm{d}g_0} \widetilde{\log \mathscr{L}}_0(g_0, (g - g_0)/2) = -2 + \frac{2g + 2f}{g_0} > 0,$$

when $g_0 \geq g$,

$$\frac{\mathrm{d}}{\mathrm{d}g_0} \widetilde{\log \mathscr{L}}_0(g_0, 0) = -3 + \frac{3g + 2f}{g_0},$$

which is 0 for g_0 satisfying $3g_0 = 3g + 2f$. To summarize, the optimal choice of branch lengths is, therefore,

$$g_0^* = g + \frac{2}{3}f, \qquad f_0^* = 0.$$

That is consistent with the contour plot above.

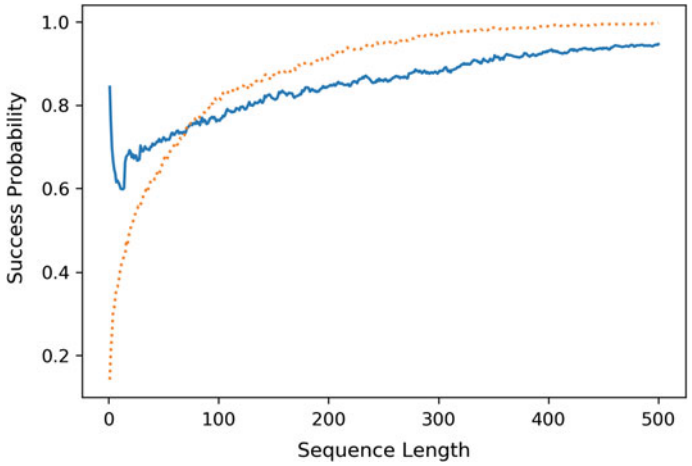


Fig. 4.9 Success probability of likelihood-based (solid) and distance-based (dotted) approaches as functions of sequence length with an asymptotically optimal choice of parameters g and f for the alternative topologies

Numerical simulations: In the next experiment, we use parameters g_0^* and f_0^* for the alternatives and we plot the success probability of the three-way LRT. For large k, observe that the distance-based method (dotted line) performs better than the three-way LRT (solid line). For small k, however, the distance-based method performs worse, possibly because our optimal choice is only valid in the limit $k \to +\infty$ (Fig. 4.9).

```
# EXP 9: accuracy of LRT v. k for better choice of lengths
g, f, k, N = 0.1, 0.05, 500, 1000 # params used for freq_succ_pw

freq_succ_ll = test_lrt(g,f,g+(2/3)*f,0,k,N)

plt.plot(np.arange(1,k+1),freq_succ_ll);
plt.plot(np.arange(1,k+1),freq_succ_pw,':');
plt.xlabel('Sequence Length'), plt.ylabel('Success Probability');
```

Using again parameters g_0^* and f_0^* for the alternative topologies, we plot the sequence length required for the three-way LRT to succeed with probability at least `target`. The results are consistent once again with a requirement scaling as $\propto f^{-2}$. This time, however, the requirement for LRT (solid line) is significantly higher than that of the distance-based approach (dotted line). While likelihood-based methods use the joint distribution of the data and, therefore, may be expected to perform better than distance-based methods which rely on pairwise distributions, in the three-species settings there may not be enough "extra information" in the joint distribution to compensate for the downsides of nuisance parameters (Fig. 4.10).

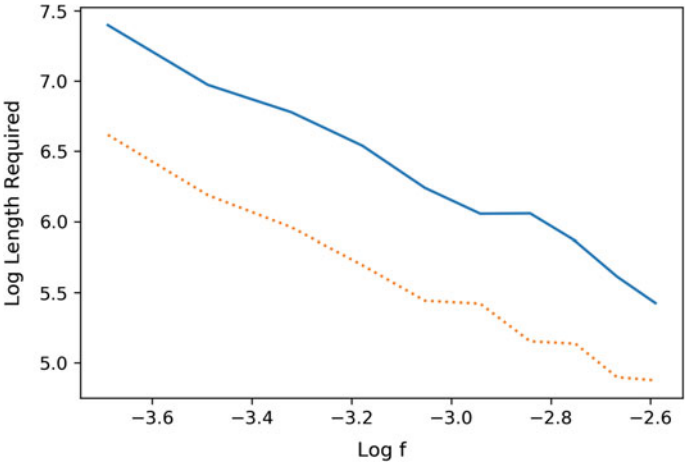


Fig. 4.10 Sequence-length requirement of likelihood-based (solid) and distance-based (dotted) approaches as functions of f with an asymptotically optimal choice of parameters g and f for the alternative topologies

```
# EXP 10: requirement for LRT v. f for better choice of lengths
g, f, k, N, target = 0.1, 0.05, 2500, 1000, 0.95
f_arr_min, f_arr_max, f_arr_len = 0.025, 0.075, 10 #as k_thres_f
f_arr = np.linspace(f_arr_min, f_arr_max, num=f_arr_len)

k_thres_fll = np.zeros(f_arr_len)
for i in range(f_arr_len):
    freq_succ = test_lrt(g,f_arr[i],g+(2/3)*f_arr[i],0,k,N)
    k_thres_fll[i] = np.min(np.nonzero(freq_succ>target))

plt.plot(np.log(f_arr), np.log(k_thres_fll));
plt.plot(np.log(f_arr),np.log(k_thres_f),':');
plt.xlabel('Log f'), plt.ylabel('Log Length Required');
```

4.7 Lower Bound on the Best Achievable Requirement

In this section, we consider lower bounds on the sequence-length requirement. In particular, we show—both analytically and numerically—that, in the three-species setting, the requirement we derived for distance-based and likelihood-based reconstruction approaches in previous sections cannot be improved (up to constants). These lower bounds are information theoretic, i.e., they apply to any reconstruction method.

The standard way to obtain such a lower bound is to "make the problem easier" by considering the two-topology setup of the previous section. Namely, suppose the sequence dataset $\sigma_L^{(k)}$ is generated by a model in the class $\mathscr{P} = \{AB|C_{g,f}, AC|B_{g,f}\}$. Our goal again is to guess which one of the two models

the data came from. How large does k need to be for there to exist a reconstruction method that succeeds with probability $1-\delta$? A lower bound on the required k for $\mathscr{P}$ automatically gives a lower bound on the required k for the larger class $\cup_{g'\leq g, f'\geq f}\{AB|C_{g',f'}, AC|B_{g',f'}, BC|A_{g',f'}\}$—since it includes $\mathscr{P}$.

Analytical derivation: Recall the definitions of $\mathscr{W}_{E,AB}[\Psi]$ and $\mathscr{W}_{E,AC}[\Psi]$ and let

$$\mathscr{W}_{E,\max}[\Psi] = \max\{\mathscr{W}_{E,AB}[\Psi], \mathscr{W}_{E,AC}[\Psi]\}$$

be the maximum probability of error for Ψ under models in $\mathscr{P}$. We seek to establish a lower bound on $\mathscr{W}_{E,\max}[\Psi]$ that applies to any Ψ. We already know that

$$\mathscr{W}_{E,\max}[\Psi] \geq \frac{1}{2}\mathscr{W}_{I,II}[\mathbf{L}'].$$

We first relate the r.h.s. to a standard notion of distance on probability measures. If $(\lambda_x : x \in \mathscr{X})$ and $(\gamma_x : x \in \mathscr{X})$ are probability measures over the discrete space $\mathscr{X}$, then their total variation distance (see e.g. [25]) is defined as

$$\mathrm{TV}(\lambda, \gamma) = \frac{1}{2}\sum_{x\in\mathscr{X}} |\lambda_x - \gamma_x|,$$

which is always between 0 and 1. We express $\mathscr{W}_{I,II}[\mathbf{L}']$ in terms of the total variation distance between $\mathscr{L}^k_{AB}$ and $\mathscr{L}^k_{AC}$. By definition of $\mathbf{L}'$ and the fact that $\mathscr{L}^k_{AB}$ sums to 1, we get

$$\mathscr{W}_{I,II}[\mathbf{L}'] = 1 - \sum_{\sigma_L^{(k)}} (\mathscr{L}^k_{AB}(\sigma_L^{(k)}) - \mathscr{L}^k_{AC}(\sigma_L^{(k)}))\mathbf{1}[\mathscr{L}^k_{AB}(\sigma_L^{(k)}) > \mathscr{L}^k_{AC}(\sigma_L^{(k)})],$$

where $\mathbf{1}[\mathscr{E}]$ is the indicator of the event $\mathscr{E}$. Using symmetry under the interchanging of the role of B and C, we get further

$$\mathscr{W}_{I,II}[\mathbf{L}'] = 1 - \frac{1}{2}\sum_{\sigma_L^{(k)}} \left|\mathscr{L}^k_{AB}(\sigma_L^{(k)}) - \mathscr{L}^k_{AC}(\sigma_L^{(k)})\right| = 1 - \mathrm{TV}(\mathscr{L}^k_{AB}, \mathscr{L}^k_{AC}).$$

By combining this with the inequality above, we have reduced the problem of deriving a lower bound on $\mathscr{W}_{E,\max}[\Psi]$ for any Ψ to that of deriving an upper bound on the total variation distance between $\mathscr{L}^k_{AB}$ and $\mathscr{L}^k_{AC}$.

Computing $\mathrm{TV}(\mathscr{L}^k_{AB}, \mathscr{L}^k_{AC})$ for arbitrary k turns out to be tricky because of the underlying combinatorial complexity. Therefore, the next step is to further reduce the problem to a single site. We use a standard trick in statistical theory, moving to the Hellinger distance. Let

$$\mathrm{H}^2(\lambda,\gamma)=\frac{1}{2}\sum_{x\in\mathscr{X}}\left(\sqrt{\lambda_x}-\sqrt{\gamma_x}\right)^2$$

be the Hellinger distance (see e.g. [25]) between probability measures λ and γ, which is always between 0 and 1. This unintuitive distance has two useful properties. First, it is closely related to the more natural total variation distance, through the following inequality which we derive for completeness. (There is also an inequality in the other direction, which we omit.) Writing

$$\sum_{x\in\mathscr{X}}|\lambda_x-\gamma_x|=\sum_{x\in\mathscr{X}}\left(\sqrt{\lambda_x}-\sqrt{\gamma_x}\right)\left(\sqrt{\lambda_x}+\sqrt{\gamma_x}\right)$$

and applying the Cauchy–Schwarz inequality, we get

$$\sum_{x\in\mathscr{X}}|\lambda_x-\gamma_x|\le\sqrt{\sum_{x\in\mathscr{X}}\left(\sqrt{\lambda_x}-\sqrt{\gamma_x}\right)^2}\sqrt{\sum_{x\in\mathscr{X}}\left(\sqrt{\lambda_x}+\sqrt{\gamma_x}\right)^2}.$$

Using the fact that λ and γ sum to 1 and applying the Cauchy–Schwarz inequality again,

$$\sum_{x\in\mathscr{X}}\left(\sqrt{\lambda_x}+\sqrt{\gamma_x}\right)^2=2+2\sum_{x\in\mathscr{X}}\sqrt{\lambda_x}\sqrt{\gamma_x}\le 2+2\sqrt{\sum_{x\in\mathscr{X}}\lambda_x}\sqrt{\sum_{x\in\mathscr{X}}\gamma_x}=4.$$

Hence,

$$\mathrm{TV}(\lambda,\gamma)\le\sqrt{2\mathrm{H}^2(\lambda,\gamma)}.$$

Second, the Hellinger distance (somewhat magically) "tensorizes", let $\lambda^{\otimes k}$ be the distribution of k independent samples from λ, we have

$$1-\mathrm{H}^2(\lambda^{\otimes k},\gamma^{\otimes k})=\sum_{\mathbf{x}\in\mathscr{X}^k}\prod_{i=1}^{k}\sqrt{\lambda_{x_i}\gamma_{x_i}}=\left(\sum_{x\in\mathscr{X}}\sqrt{\lambda_x\gamma_x}\right)^k=\left(1-\mathrm{H}^2(\lambda,\gamma)\right)^k.$$

Putting everything together, we finally get the following key result for any randomized test Ψ

$$\mathscr{W}_{E,\max}[\Psi]\ge\frac{1}{2}\left[1-\sqrt{2\left[1-\left(1-\mathrm{H}^2(\mathscr{L}^1_{AB},\mathscr{L}^1_{AC})\right)^k\right]}\right]. \tag{4.2}$$

Numerical computations: So it remains to bound $\mathrm{H}^2(\mathscr{L}^1_{AB},\mathscr{L}^1_{AC})$, which can be estimated both numerically and analytically. The numerical experiment below computes the Hellinger-based lower bound on $\mathscr{W}_{E,\max}[\Psi]$ for a fixed $\mathtt{g}$ and for k scaling like $\propto f^{-2}$, for an array `f_arr` of values of f. The plots indicate that the lower bound increases toward $1/2$ when $k=bf^{-2}$ for b ranging between 0.01 and 1. Note

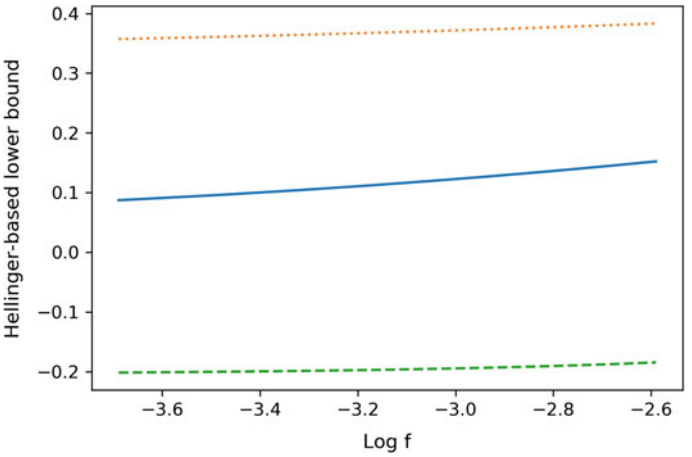


Fig. 4.11 Hellinger-based lower bound as function of f for sequence length $k = bf^{-2}$ for $b = 0.01, 0.1, 1$ (dotted, solid, dashed)

that, when $b = 1$, the lower bound on the probability of error is negative, which is of course useless. This is because our bound relating TV and H^2 is off by a factor of $\sqrt{2}$ when the probability measures are close to maximally distinct (Fig. 4.11).

```
# EXP 11: Hellinger distance between AB|C and AC|B v. f
g = 0.1
f_arr_min, f_arr_max, f_arr_len = 0.025, 0.075, 10
f_arr = np.linspace(f_arr_min, f_arr_max, num=f_arr_len)

hell, s = np.zeros(f_arr_len), [-1,1]
for i in range(f_arr_len):
    for jAB in range(2):
        for jAC in range(2):
            lAB = exp(llXY_Z(g,f_arr[i],s[jAB],s[jAC]))
            lAC = exp(llXY_Z(g,f_arr[i],s[jAC],s[jAB]))
            hell[i] = hell[i] + 2*((sqrt(lAB)-sqrt(lAC))**2)/2

base = (1-hell)**(1/(f_arr**2))
plt.plot(np.log(f_arr),(1/2)*(1-np.sqrt(2*(1-base**0.1))));
plt.plot(np.log(f_arr),(1/2)*(1-np.sqrt(2*(1-base**0.01))),':');
plt.plot(np.log(f_arr),(1/2)*(1-np.sqrt(2*(1-base**1))),'--');
plt.xlabel('Log f'), plt.ylabel('Hellinger-based lower bound');
```

Analytical derivation (continued): We confirm the results above analytically. Fix g and consider the limit $f \to 0$. By symmetry, for constant sites σ_L^1, $\mathscr{L}_{AB}^1(\sigma_L^1) = \mathscr{L}_{AC}^1(\sigma_L^1)$ so such sites contribute nothing to $H^2(\mathscr{L}_{AB}^1, \mathscr{L}_{AC}^1)$. The same holds for the sites $\sigma_L^1 = (+1, -1, -1)$ and $\sigma_L^1 = (-1, +1, +1)$. For the sites $\sigma_L^1 = (+1, +1, -1)$ and $\sigma_L^1 = (-1, -1, +1)$, we use the expression for the likelihood derived in the previous section with $I_{AB}^1 = 1$ and $I_{AC}^1 = 0$. Let $p = p(g)$ and $q = p(g + 2f)$ and expand to the first order in f to get that $q = p + c_1 f + O(f^2)$ where $c_1 = 2e^{-2g}$. Then, we have

$$\mathscr{L}^1_{AB}(\sigma^1_L) = \frac{1}{2}(1-p)^2 q + \frac{1}{2}p^2(1-q) = \frac{1}{2}(1-p)^2 p + \frac{1}{2}p^2(1-p) + c_2 f + O(f^2),$$

where $c_2 = \frac{c_1}{2}[(1-p)^2 - p^2] = \frac{c_1}{2}(1-2p)$ and

$$\begin{aligned}
&\mathscr{L}^1_{AC}(\sigma^1_L) \\
&= \frac{1}{2}(1-p)p(1-q) + \frac{1}{2}p(1-p)q \\
&= \frac{1}{2}(1-p)^2 p + \frac{1}{2}p^2(1-p) + c_3 f + O(f^2),
\end{aligned}$$

where $c_3 = \frac{c_1}{2}[-(1-p)p + p(1-p)] = 0$. Using $\sqrt{z+x} = \sqrt{z} + x/(2\sqrt{z}) + O(x^2)$ and letting $z = \frac{1}{2}(1-p)^2 p + \frac{1}{2}p^2(1-p) = \frac{1}{2}p(1-p)$, we get

$$\sqrt{\mathscr{L}^1_{AB}(\sigma^1_L)} - \sqrt{\mathscr{L}^1_{AC}(\sigma^1_L)} = c_4 f + O(f^2),$$

where $c_4 = c_2/(2\sqrt{z})$. So the contribution of σ^1_L to $\mathrm{H}^2(\mathscr{L}^1_{AB}, \mathscr{L}^1_{AC})$ is $\frac{1}{2}c_4^2 f^2 + O(f^3)$. By interchanging the role of B and C, we see that the contribution from the sites $\sigma^1_L = (+1,-1,+1)$ and $\sigma^1_L = (-1,+1,-1)$ is the same. So, finally

$$\mathrm{H}^2(\mathscr{L}^1_{AB}, \mathscr{L}^1_{AC}) = cf^2 + O(f^3),$$

where $c = 2c_4^2$. Note that c is strictly positive and depends only on g.

Claim 6 (Lower bound on probability of error). Taking $k = bf^{-2}$ in the Hellinger-based lower bound on the maximum probability of error (see (4.2)), we get that

$$\mathscr{W}_{E,\max}[\Psi] \geq \frac{1}{2}\left[1 - \sqrt{2\left[1 - \left(1 - cf^2 + O(f^3)\right)^{bf^{-2}}\right]}\right],$$

where the c depends on g. As $f \to 0$, the r.h.s. converges to

$$\frac{1}{2}\left[1 - \sqrt{2\left[1 - e^{-cb}\right]}\right].$$

This last expression is $1/2$ when $b = 0$, and it decreases monotonically as b gets larger.

To summarize, no method can have a maximum probability of error bounded away from $1/2$ unless k scales at least like $\propto f^{-2}$.

4.8 Scaling Up to Large Trees

Until now, we have restricted ourselves to small phylogenies. The results we have derived in the previous sections can be used as building blocks to obtain some bounds on sequence-length requirements for large trees as well.

In the molecular clock case, one can reconstruct all three-leaf subtrees of an n-species phylogenies T using the simple distance-based method described earlier. Once all such "triplets" have been reconstructed correctly, it is straightforward to infer the full rooted phylogeny. What is the sequence-length requirement in this case? Assume g and f are, respectively, the longest and shortest branch lengths in T. Recall (see Claim 2) that if the sequence length k satisfies

$$k \geq \frac{2\ln(2/\delta)}{e^{-4G}\left(1-e^{-4F}\right)^2}, \tag{4.3}$$

then the pairwise comparison test **D** succeeds at reconstructing a fixed triplet over $\{A, B, C\}$ with probability greater than $1-\delta$. Here, G is the length of the path to the most recent common ancestor of the two closest leaves in $\{A, B, C\}$ and F is the length of the path from that vertex to the root of the triplet. To obtain a bound on the sequence-length requirement, we need to bound F, G, and δ in terms of f, g, and n.

We use that necessarily $F \geq f$. As for δ, since there are at most n^3 triplets, for all of them to be reconstructed correctly with probability $1-\delta'$ we require $\delta = \delta'/n^3$. It remains to upper bound G, which depends on the minimum number of edges h from a leaf to the root. Because T is binary and has n leaves, we have

$$n \geq 2^h.$$

Hence,

$$G \leq gh \leq g\log_2 n.$$

Putting everything together, when the sequence length satisfies

$$k \geq \frac{6\ln(2n/\delta')}{e^{-4g\log_2 n}\left(1-e^{-4f}\right)^2},$$

reconstruction of T with probability $1-\delta'$ is possible. This bound differs from that of a three-species phylogeny (see Claim 2) in two ways: (1) a factor of $\log n$ accounts for the fact that a polynomial in n number of triplets must be correctly reconstructed; (2) a polynomial factor in n (namely, $e^{4g\log_2 n}$) accounts for the depth of the phylogeny. As it turns out, the latter—the role of the depth (i.e, in this setting, the length of the paths from the root to the leaves)—is more intricate than our naive analysis suggests. We discuss this next.

4.8.1 Signal Decay

The extent to which the depth of a phylogeny affects the sequence-length requirement of reconstruction methods depends strongly on the branching rate. To highlight this subtle phenomenon, we first consider a different problem, reconstructing an ancestral state.

Numerical simulations: We begin with a numerical simulation. The function `full` generates `N` samples of sequence length `k` at the leaves of a full binary tree with `h` levels where all branch lengths are equal to `b`. More specifically what is generated is, for each site, the total number of substitutions on level `h` compared to the root state. While this does not fully characterize the sequences at the leaves, it will suffice for our purposes. The function `test_maj` then infers the root sequence by majority vote over the leaves on a single site and outputs the fraction of successful reconstructions over `N` attempts.

```
def child(s,i,p): # number of subs to one child of each parent
    return np.random.binomial(s,1-p)+np.random.binomial(2**i-s,p)

def full(b,k,h,N): # number of subs at leaves of full binary tree
    ns_root = np.zeros((N,k),dtype=int)
    for i in range(h):
        ns_root = child(ns_root,i,l2p(b))+child(ns_root,i,l2p(b))
        return ns_root

def test_maj(b,h,N): # ancestral reconstruction by majority
    return np.sum(full(b,1,h,N)<2**(h-1))/N
```

The following experiment tests the accuracy of ancestral state reconstruction by majority vote as the number of levels increases for two different values of branch lengths, `b0` and `b1`. In both cases, the probability of correct reconstruction roughly decreases with the number of levels. However, we see that for the longer branch length `b1` (dotted line), the probability of correct reconstruction appears to converge to $1/2$, while that probability settles on a much larger value for the shorter branch length `b0` (solid line). Note that a success probability of $1/2$ corresponds to guessing at random (Fig. 4.12).

```
# EXP 12: accuracy of ancestral reconstruction by majority v. h
b0, b1, k, h, N = 0.1, 0.3, 1, 12, 1000

freq_succ0, freq_succ1 = np.zeros(h), np.zeros(h)
for i in range(h):
    freq_succ0[i] = test_maj(b0,i,N)
    freq_succ1[i] = test_maj(b1,i,N)

plt.plot(np.arange(h),freq_succ0);
plt.plot(np.arange(h),freq_succ1,':');
plt.xlabel('Height'), plt.ylabel('Success probability');
```

Analytical derivation: We next explain this significant difference analytically. Precisely, we compute the variance of the ancestral state estimator above and show that

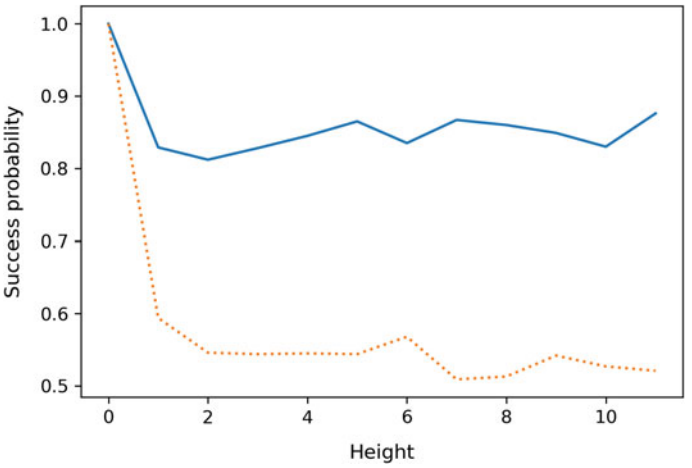


Fig. 4.12 Success probability of root reconstruction by majority as function of height for branch lengths $b = 0.1, 0.3$ (solid, dotted)

it undergoes a phase transition as the branch length b increases. Let $T = (V, E)$ be a full binary tree with h levels and all branch lengths equal to b. Let R be its root and $L = \{A_1, \ldots, A_{2^h}\}$ be its leaves. Assume that $(\sigma_v : v \in V)$ is a single site on T generated under the CF model. In particular, σ_L denotes the states at the leaves. We are interested in the following natural estimator of the root state σ_R from σ_L: take a majority vote over the states at the leaves (or pick uniformly at random in case of a tie). In our setting, this estimator is equivalent to the sign of the average state at the leaves, which for convenience we normalize as follows:

$$\mathscr{A}_{h,\theta} = \frac{1}{2^h \theta^h} \sum_{i=1}^{2^h} \sigma_{A_i},$$

where $\theta := \theta(b)$.

To analyze this estimator, we first show that $\mathscr{A}_{h,\theta}$ is conditionally unbiased, given the state at the root. Indeed, we get by symmetry

$$\mathbb{E}\left[\mathscr{A}_{h,\theta} \middle| \sigma_R\right] = \frac{1}{\theta^h} \mathbb{E}\left[\sigma_{A_1} \middle| \sigma_R\right].$$

Recalling that $P(R, A_1)$ are the edges on the path between R and A_1 and using the formulas derived previously, we then get

$$\mathbb{E}\left[\mathscr{A}_{h,\theta} \middle| \sigma_R\right] = \frac{1}{\theta^h} \mathbb{E}\left[\sigma_R \prod_{e \in P(R,A_1)} \tau_e \middle| \sigma_R\right] = \frac{1}{\theta^h} \sigma_R \theta^h = \sigma_R.$$

Next, we study the variance of $\mathscr{A}_{h,\theta}$. By a standard formula,

$$\mathrm{Var}\left[\mathscr{A}_{h,\theta}\middle|\sigma_R\right] = \frac{1}{2^{2h}\theta^{2h}}\mathbb{E}\left[\left(\sum_{i=1}^{2^h}\sigma_{A_i}\right)^2\middle|\sigma_R\right] - \left(\mathbb{E}\left[\mathscr{A}_{h,\theta}\middle|\sigma_R\right]\right)^2.$$

We have already computed the second term on the r.h.s., which is 1. For the first term, we observe that the expectation is equal to

$$\sum_{i,j=1}^{2^h}\mathbb{E}\left[\sigma_{A_i}\sigma_{A_j}\middle|\sigma_R\right] = \sum_{i,j=1}^{2^h}\mathbb{E}\left[\left(\sigma_R\prod_{e\in P(R,A_i)}\tau_e\right)\left(\sigma_R\prod_{e\in P(R,A_j)}\tau_e\right)\middle|\sigma_R\right].$$

Let $A_i \wedge A_j$ be the most recent common ancestor of A_i and A_j and let $h_{i\wedge j}$ be the graph distance from the root to $A_i \wedge A_j$. Then, cancellations on the path from the root to $A_i \wedge A_j$ lead to the simplified expression

$$\begin{aligned}&\mathbb{E}\left[\left(\sigma_R\prod_{e\in P(R,A_i)}\tau_e\right)\left(\sigma_R\prod_{e\in P(R,A_j)}\tau_e\right)\middle|\sigma_R\right]\\ &= \mathbb{E}\left[\prod_{e\in P(A_i\wedge A_j,A_i)}\tau_e\prod_{e\in P(A_i\wedge A_j,A_j)}\tau_e\right]\\ &= \theta^{2h-2h_{i\wedge j}}.\end{aligned}$$

Plugging this back above and decomposing the sum over the levels of T, we get

$$\sum_{i,j=1}^{2^h}\mathbb{E}\left[\sigma_{A_i}\sigma_{A_j}\middle|\sigma_R\right] = \sum_{m=0}^{h}2^m\left(2^{h-m-1}\right)^2\theta^{2h-2m} = \frac{1}{4}\sum_{m=0}^{h}2^{2h-m}\theta^{2h-2m},$$

where the term $\left(2^{h-m-1}\right)^2$ counts the number of pairs A_i, A_j whose most recent common ancestor $A_i \wedge A_j$ is a fixed vertex v on level m, while the term 2^m counts the number of such vertices v. Finally,

$$\mathrm{Var}\left[\mathscr{A}_{h,\theta}\middle|\sigma_R\right] = \frac{1}{4}\sum_{m=0}^{h}(2\theta^2)^{-m} - 1.$$

The key observation is that the limit of this variance as the height goes to $+\infty$ depends crucially on the quantity $2\theta^2$.

Claim 7 As $h \to +\infty$,

$$\mathrm{Var}\left[\mathscr{A}_{h,\theta}|\sigma_R\right] \to \begin{cases} +\infty & \text{if } 2\theta^2 \leq 1, \\ \frac{1}{4(1-(2\theta^2)^{-1})} - 1 & \text{if } 2\theta^2 > 1. \end{cases}$$

Intuitively, this can be interpreted as follows: when the variance goes to $+\infty$, the estimator $\mathscr{A}_{h,\theta}$ is essentially unable to distinguish between the cases $\sigma_R = +1$ and $\sigma_R = -1$.

This is indeed what we observe on the plot above. Note that, in terms of branch lengths, the critical threshold is

$$2(e^{-2b})^2 = 1 \iff b = \frac{1}{2}\log\sqrt{2} = 0.173...$$

Hence, `b0=0.1` above (solid line) is below the critical threshold corresponding to a finite variance in the limit, while `b1=0.3` (dotted line) is above the threshold corresponding to an infinite variance. Notice, moreover, that while our analysis is asymptotic in h, the previous experiment suggests that convergence occurs after a small number of levels.

4.8.2 Depth v. Branching

How is this related to sequence-length requirements? The results in the previous section indicate that the decay of the signal along a phylogeny presents two regimes, as illustrated by the ability of majority voting to reconstruct the state at the root. It is natural to expect that this phenomenon may have a significant impact on phylogeny reconstruction. We first test this hypothesis through a simulation.

We consider the following generalization of our previous simple setting.

Definition 5 *(Deep setting).* We start with triplet $AB|C$ with parameters g and f as before and we add a full binary tree with h levels below each of A, B, and C. Let $T = [AB|C]^{h,b}_{g,f}$ be the resulting tree and let

$$L = \{A_1, \ldots, A_{2^h}, B_1, \ldots, B_{2^h}, C_1, \ldots, C_{2^h}\}$$

be the corresponding leaves, where the first batch of size 2^h are descendants of A, and so on. For $X \in \{A, B, C\}$, let T_X be the subtree below (and including) X. We assume that the branch lengths on T_A, T_B, and T_C are all equal to b. Our goal is to infer the deep triplet $AB|C$ from sequence data $\sigma_L^{(k)}$ at the leaves.

Numerical simulations: We begin with a simple test. We perform our previous pairwise comparison test **D** on the sub-dataset $(\sigma^{(k)}_{A_1}, \sigma^{(k)}_{B_1}, \sigma^{(k)}_{C_1})$, i.e., we only use the data from one leaf in each subtree. The function `test_deep_naive` below performs this test:

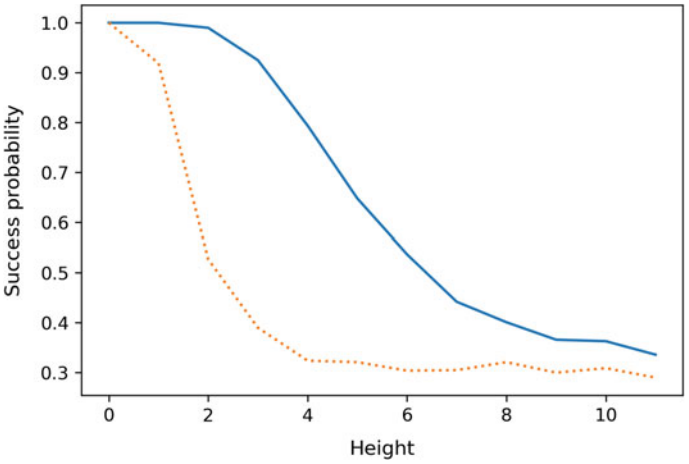


Fig. 4.13 Success probability of deep triplet reconstruction as function of height for branch lengths $b = 0.1, 0.3$ (solid, dotted) using the naive pairwise comparison approach

```
def test_deep_naive(g,f,b,h,k,N): # pairwise, deep triplet AB|C
    sAB, sAC, sBC = AB_C(g+h*b,f,k,N)
    ABvAC, ABvBC = comp(sAB-sAC,sAB-sBC)
    return np.sum(np.logical_and(ABvAC[:,-1]>0,
                                 ABvBC[:,-1]>0),axis=0)/N
```

In the following experiment, we plot the success probability of this test as the number of levels h increases for two different values of branch length b, one on each side of the critical threshold. In both cases, the success probability rapidly converges to $1/3$, although that convergence is somewhat slower for the smaller branch length (solid line) (Fig. 4.13).

```
# EXP 13: accuracy of pairwise comparisons for deep triplet
g, f, b0, b1, h, k, N = 0.01, 5, 0.1, 0.3, 12, 75, 1000

freq_succ0, freq_succ1 = np.zeros(h), np.zeros(h)
for i in range(h):
    freq_succ0[i] = test_deep_naive(g,f,b0,i,k,N)
    freq_succ1[i] = test_deep_naive(g,f,b1,i,k,N)

plt.plot(np.arange(h),freq_succ0);
plt.plot(np.arange(h),freq_succ1,':');
plt.xlabel('Height'), plt.ylabel('Success probability');
```

That first test was somewhat naive, in that it used a single leaf per subtree. It is not surprising that the rate of signal decay only has a mild effect on its behavior; in our observations on ancestral state reconstruction, it was crucial to use all the leaves. A more sophisticated estimator is obtained by averaging over all pairs of leaves between each pair of subtrees. Namely, we consider the following distance-based algorithm $\overline{\mathbf{D}}$.

Definition 6 *(Algorithm $\overline{\mathbf{D}}$).* Given sequence data $\sigma_L^{(k)}$, we set $\overline{\mathbf{D}}(\sigma_L^{(k)}) = XY|Z$ if

$$\min\left\{\sum_{i,j=1}^{2^h} \Sigma^k_{X_iY_j} - \sum_{i,j=1}^{2^h} \Sigma^k_{X_iZ_j}; \sum_{i,j=1}^{2^h} \Sigma^k_{X_iY_j} - \sum_{i,j=1}^{2^h} \Sigma^k_{Y_iZ_j}\right\} > 0,$$

and we return a failure if no such pair exists.

In the function `avg_dst` below, we rewrite

$$\sum_{i,j=1}^{2^h} \Sigma^k_{X_iY_j} = \sum_{m=1}^{k}\left(\sum_{i=1}^{2^h} s^m_{X_iX}\right) s^m_{XY}\left(\sum_{j=1}^{2^h} s^m_{YY_j}\right),$$

and we note that the expressions in parentheses on the r.h.s. can be expressed in terms of the total number of substitutions between X (respectively Y) and $X_1, \ldots, X_{2^h}$ (respectively $X_1, \ldots, X_{2^h}$). The latter quantities are of course not known from the data at the leaves—we only use this convenient representation for the sake of speedy simulation. The function `test_deep_avg` implements the estimator $\overline{\mathbf{D}}$ and tests it for different values of the depth h.

```
def avg_dst(ns2X,sXY,ns2Y,h): # mean distance, subtrees X and Y
    return (-ns2X+(2**h-ns2X))*sXY*(-ns2Y+(2**h-ns2Y))

def test_deep_avg(g,f,b,h,k,N): # mean pairwise, triplet AB|C
    sAB, sAC, sBC = AB_C(g,f,k,N)
    ns2A,ns2B,ns2C=full(b,k,h,N),full(b,k,h,N),full(b,k,h,N)
    criABvAC = avg_dst(ns2A,sAB,ns2B,h)-avg_dst(ns2A,sAC,ns2C,h)
    criABvBC = avg_dst(ns2A,sAB,ns2B,h)-avg_dst(ns2B,sBC,ns2C,h)
    ABvAC, ABvBC = comp(criABvAC,criABvBC)
    return np.sum(np.logical_and(ABvAC[:,-1]>0,
                                 ABvBC[:,-1]>0),axis=0)/N
```

The following experiment shows a drastically different outcome. In the plot, the solid line is the probability of success of $\overline{\mathbf{D}}$ when b is below the critical threshold (here, `b0=0.1`) while the dotted line shows the same quantity above the threshold (here, `b1=0.3`). Below the threshold, the probability of success remains 1 no matter how deep the tree is (here, up to `h=12`). On the other hand, above the threshold, the success deteriorates fast with h. Morally, in the first case, the phylogeny appears "shallow" (information-theoretically speaking) independently of its true depth (combinatorially speaking). The sequence length was chosen so that, in both cases, the success probability is 1 when $h = 0$ (trial and error not shown). Also `g` and `f` were chosen to ensure that somewhat short sequences suffice to allow for a fast simulation (Fig. 4.14).

```
# EXP 14: accuracy of mean pairwise comp for deep triplet
g, f, b0, b1, h, k, N = 0.01, 5, 0.1, 0.3, 12, 75, 1000

freq_succ0, freq_succ1 = np.zeros(h), np.zeros(h)
```

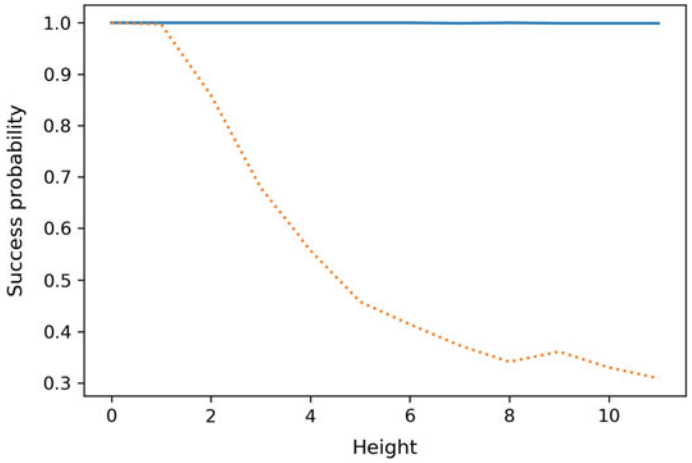


Fig. 4.14 Success probability of deep triplet reconstruction as function of height for branch lengths $b = 0.1, 0.3$ (solid, dotted) using the averaged pairwise comparison approach

```
for i in range(h):
    freq_succ0[i] = test_deep_avg(g,f,b0,i,k,N)
    freq_succ1[i] = test_deep_avg(g,f,b1,i,k,N)

plt.plot(np.arange(h),freq_succ0);
plt.plot(np.arange(h),freq_succ1,':');
plt.xlabel('Height'), plt.ylabel('Success probability');
```

Analytical derivation: We confirm this picture analytically. We assume the data is generated under $AB|C$. We want an upper bound on the probability of error, i.e., the probability of the intersection of the events

$$\mathscr{E}_{AC} = \left\{ \sum_{i,j=1}^{2^h} \Sigma^k_{A_i B_j} - \sum_{i,j=1}^{2^h} \Sigma^k_{A_i C_j} > 0 \right\}$$

and

$$\mathscr{E}_{BC} = \left\{ \sum_{i,j=1}^{2^h} \Sigma^k_{A_i B_j} - \sum_{i,j=1}^{2^h} \Sigma^k_{B_i C_j} > 0 \right\}.$$

By symmetry, these have the same probability. We follow the argument used in the analysis of **D**, with one modification. Rather than using Hoeffding's inequality (which is valid for bounded variables—not the case here), we use Chebyshev's inequality (see e.g. [23]), one form of which is the following: if $W_1, \ldots, W_k$ are independent random variables with respective variances α_i, $i = 1, \ldots, k$, then for all $\varepsilon > 0$

$$\mathbb{P}\left[\sum_{i=1}^{k}(W_i - \mathbb{E}[W_i]) \geq k\varepsilon\right] \leq \frac{\sum_{i=1}^{k}\alpha_i}{k^2\varepsilon^2}.$$

Hence, using the expression for $\sum_{i,j=1}^{2^h}\Sigma_{A_iB_j}^k$ derived above, it remains to compute the mean and variance of $\left(\sum_{i=1}^{2^h}s_{A_iA}^m\right)s_{AB}^m\left(\sum_{j=1}^{2^h}s_{BB_j}^m\right)$. By definition, $s_{XY}^m = \sigma_X^m\sigma_Y^m$ so that by cancellation

$$\widetilde{\Sigma}_{AB}^m := \left(\sum_{i=1}^{2^h}s_{A_iA}^m\right)s_{AB}^m\left(\sum_{j=1}^{2^h}s_{BB_j}^m\right) = \left(\sum_{i=1}^{2^h}\sigma_{A_i}^m\right)\left(\sum_{j=1}^{2^h}\sigma_{B_j}^m\right).$$

The law of total expectation allows to condition on the states at A and B as follows:

$$\mathbb{E}\left[\left(\sum_{i=1}^{2^h}\sigma_{A_i}^m\right)\left(\sum_{j=1}^{2^h}\sigma_{B_j}^m\right)\right] = \mathbb{E}\left[\mathbb{E}\left[\left(\sum_{i=1}^{2^h}\sigma_{A_i}^m\right)\left(\sum_{j=1}^{2^h}\sigma_{B_j}^m\right)\middle|\sigma_A^m,\sigma_B^m\right]\right].$$

This is useful because, once we condition on σ_A^m and σ_B^m, the states at leaves of T_A and T_B are independent. This is the so-called Markov property. Hence, we get

$$\mathbb{E}\left[\mathbb{E}\left[\left(\sum_{i=1}^{2^h}\sigma_{A_i}^m\right)\left(\sum_{j=1}^{2^h}\sigma_{B_j}^m\right)\middle|\sigma_A^m,\sigma_B^m\right]\right] = \mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{2^h}\sigma_{A_i}^m\middle|\sigma_A^m\right]\mathbb{E}\left[\sum_{j=1}^{2^h}\sigma_{B_j}^m\middle|\sigma_B^m\right]\right].$$

Using our previous formula for the conditional expectations above, we get finally

$$\begin{aligned}&\mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{2^h}\sigma_{A_i}^m\middle|\sigma_A^m\right]\mathbb{E}\left[\sum_{j=1}^{2^h}\sigma_{B_j}^m\middle|\sigma_B^m\right]\right]\\ &= \mathbb{E}\left[2^h\theta(b)^h\sigma_A^m \times 2^h\theta(b)^h\sigma_B^m\right]\\ &= (2\theta(b))^{2h}\theta(2g).\end{aligned}$$

That is,

$$\mathbb{E}\left[\widetilde{\Sigma}_{AB}^m\right] = (2\theta(b))^{2h}\theta(2g).$$

As for the variance, we first use the conditional variance formula

$$\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m\right] = \mathrm{Var}\left[\mathbb{E}\left[\widetilde{\Sigma}_{AB}^m|\sigma_A^m,\sigma_B^m\right]\right] + \mathbb{E}\left[\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m|\sigma_A^m,\sigma_B^m\right]\right].$$

From the computation above, the first term is

$$\mathrm{Var}\left[\mathbb{E}\left[\widetilde{\Sigma}_{AB}^m|\sigma_A^m,\sigma_B^m\right]\right] = \mathrm{Var}\left[2^h\theta(b)^h\sigma_A^m \times 2^h\theta(b)^h\sigma_B^m\right] = (2\theta(b))^{4h}\mathrm{Var}\left[\sigma_A^m\sigma_B^m\right].$$

Using that $(\sigma_A^m)^2 = (\sigma_B^m)^2 = 1$,

$$\mathrm{Var}\left[\sigma_A^m \sigma_B^m\right] = 1 - \left(\mathbb{E}\left[\sigma_A^m \sigma_B^m\right]\right)^2 = 1 - \theta(2g)^2.$$

For the second term in the conditional variance formula, we use that

$$\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m \middle| \sigma_A^m, \sigma_B^m\right] = \mathbb{E}\left[\left(\widetilde{\Sigma}_{AB}^m\right)^2 \middle| \sigma_A^m, \sigma_B^m\right] - \left(\mathbb{E}\left[\widetilde{\Sigma}_{AB}^m \middle| \sigma_A^m, \sigma_B^m\right]\right)^2.$$

The second term on the r.h.s. is equal to $(2\theta(b))^{4h}$, while the first term is, by the Markov property, again,

$$\mathbb{E}\left[\left(\sum_{i=1}^{2^h} \sigma_{A_i}^m\right)^2 \middle| \sigma_A^m\right] \mathbb{E}\left[\left(\sum_{j=1}^{2^h} \sigma_{B_j}^m\right)^2 \middle| \sigma_B^m\right] = \left(\frac{1}{4}\sum_{m=0}^{h} 2^{2h-m}\theta(b)^{2h-2m}\right)^2,$$

where the last expression was derived in the section on ancestral state reconstruction. Note that both terms in our derived expression for $\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m \middle| \sigma_A^m, \sigma_B^m\right]$ do not, in fact, depend on σ_A^m, σ_B^m, and therefore are unaffected by taking an expectation. Putting everything together, the variance is

$$\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m\right] = (2\theta(b))^{4h}\left(1 - \theta(2g)^2\right) + \left(\frac{1}{4}\sum_{m=0}^{h} 2^{2h-m}\theta(b)^{2h-2m}\right)^2 - (2\theta(b))^{4h}.$$

After simplification that becomes

$$\mathrm{Var}\left[\widetilde{\Sigma}_{AB}^m\right] = \left(\frac{1}{4}\sum_{m=0}^{h} 2^{2h-m}\theta(b)^{2h-2m}\right)^2 - (2\theta(b))^{4h}\theta(2g)^2.$$

We will bound $\mathbb{P}[\mathscr{E}_{AC}^c]$ as follows:

$$\mathbb{P}[\mathscr{E}_{AC}^c] \leq \mathbb{P}[\mathscr{F}_{AB}] + \mathbb{P}[\mathscr{F}_{AC}],$$

where

$$\mathscr{F}_{AB} = \left\{\sum_{m=1}^{k}\left(\widetilde{\Sigma}_{AB}^m - \mathbb{E}[\widetilde{\Sigma}_{AB}^m]\right) \leq -k(2\theta(b))^{2h}\frac{\theta(2g) - \theta(2g+2f)}{2}\right\}$$

and

$$\mathscr{F}_{AC} = \left\{\sum_{m=1}^{k}\left(\widetilde{\Sigma}_{AC}^m - \mathbb{E}[\widetilde{\Sigma}_{AC}^m]\right) \geq k(2\theta(b))^{2h}\frac{\theta(2g) - \theta(2g+2f)}{2}\right\}.$$

In words, if $\sum_{i,j=1}^{2^h} \Sigma^k_{A_iB_j} - \sum_{i,j=1}^{2^h} \Sigma^k_{A_iC_j} \leq 0$, then one of the two terms must be away from its expectation by more than half the gap between the expectations. By Chebyshev's inequality, we have the bound

$$\mathbb{P}[\mathscr{F}_{AB}] \leq \frac{\left(\frac{1}{4}\sum_{m=0}^{h} 2^{2h-m}\theta(b)^{2h-2m}\right)^2 - (2\theta(b))^{4h}\theta(2g)^2}{k(2\theta(b))^{4h}(\theta(2g)-\theta(2g+2f))^2/4}.$$

After simplification, we get

$$\mathbb{P}[\mathscr{F}_{AB}] \leq \frac{1}{k}\frac{\frac{1}{4}\left(\sum_{m=0}^{h}(2\theta(b)^2)^{-m}\right)^2}{(\theta(2g)-\theta(2g+2f))^2}.$$

It can be checked that the same bound holds for $\mathbb{P}[\mathscr{F}_{AC}]$. Applying the same argument to $\mathbb{P}[\mathscr{E}^c_{BC}]$, we finally get the following bound:

$$\mathbb{P}\left[\overline{\mathbf{D}}(\sigma_L^{(k)}) \neq T\right] \leq \frac{1}{k}\frac{\left(\sum_{m=0}^{h}(2\theta(b)^2)^{-m}\right)^2}{(\theta(2g)-\theta(2g+2f))^2}.$$

Claim 8 (Sequence-length requirement of $\overline{\mathbf{D}}$). Let

$$\mathscr{P} = \{[AB|C]^{h,b}_{g,f}, [AC|B]^{h,b}_{g,f}, [BC|A]^{h,b}_{g,f}\}.$$

We have the following upper bound on the sequence-length requirement of $\overline{\mathbf{D}}$ over $\mathscr{P}$

$$K_{\overline{\mathbf{D}}}(\mathscr{P}) \leq \frac{1}{\delta}\frac{\left(\sum_{m=0}^{h}(2\theta(b)^2)^{-m}\right)^2}{(\theta(2g)-\theta(2g+2f))^2}.$$

The behavior of the sequence length required depends crucially on $2\theta(b)^2$. When $2\theta(b)^2 > 1$, the numerator on the r.h.s. is at most $1/(1-(2\theta(b)^2)^{-1})^2$ and we require

$$k \geq \frac{1}{\delta}\frac{1}{(\theta(2g)-\theta(2g+2f))^2(1-(2\theta(b)^2)^{-1})^2},$$

which does not depend on h. That is, in that regime, the sequence length requirement of this method is not sensitive to the depth of the tree. On the other hand, when $2\theta(b)^2 < 1$ (we omit the equality case), the numerator on the r.h.s. of our bound on the error probability now grows exponentially with h and we require

$$k \geq \frac{1}{\delta}\frac{(1/2\theta(b)^2)^{2h+2}}{(\theta(2g)-\theta(2g+2f))^2((1/2\theta(b)^2)-1)^2}.$$

4.9 Bibliographic Remarks

While for simplicity, we have focused exclusively on the Cavender–Farris model under a molecular clock, sequence-length requirement results have been derived in much more general contexts—using some of the insights described here as well as many other ideas. We give a brief, non-extensive review of these results below.

Under a general Markov model on a general phylogeny with branch lengths bounded between two constants, distance-based methods have been developed that have the same type of dependence on shortest branch length and depth that we previously described, although branch length and depth must be defined with some care [9, 10, 12, 20]. In particular, the sequence-length requirement of these so-called fast-converging methods is polynomial in the number $\{n\}$ of leaves under these assumptions [32]. It should be noted that not all distance-based methods are fast converging. Most prominently, the popular neighbor-joining method has been shown to have an exponential requirement in n [13]. Results on fast-converging distance-based methods have also been extended to partial forest reconstruction [5, 7, 19].

Phase transition results on general phylogenies have also been obtained, albeit under more restrictive assumptions. In the "reconstruction regime," i.e., when branch lengths are below a critical threshold that depends on the model, the sequence-length requirement has been shown to scale logarithmically in n for certain *ad hoc* methods [6, 14, 18], as well as distance-based methods similar to the one described above [26] and maximum likelihood estimation [27]. Currently, these results have been rigorously established under simpler models, such as Jukes–Cantor, and further require that branch lengths are discretized. It is a (potentially difficult) open problem to obtain logarithmic in n sequence-length requirements without this discretization assumption. Lower bounds have been derived in [17, 22, 30].

Some limited amount of work has been dedicated to deriving sequence-length requirements in more complex models, including models of insertions and deletions [8] and multilocus coalescent-based models [4, 21, 28].

Acknowledgements This work is supported by NSF grants DMS-1149312 (CAREER), DMS-1614242, and CCF-1740707 (TRIPODS).

When I was first introduced to the field of computational phylogenetics in graduate school, I had the privilege of being supported by the NSF-funded CIPRES project—of which Bernard Moret was a leader—which had a significant impact on my early career.

References

1. Casella, G., Berger, R.: Statistical Inference. Duxbury Resource Center (2001)
2. Cavender, J.A.: Taxonomy with confidence. Math. Biosci. **40**(3–4) (1978)
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley-Interscience. Wiley, Hoboken, NJ (2006)

4. Dasarathy, G., Nowak, R., Roch, S.: Data requirement for phylogenetic inference from multiple loci: a new distance method. IEEE/ACM Trans. Comput. Biol. Bioinform. **12**(2), 422–432 (2015)
5. Daskalakis, C., Hill, C., Jaffe, A., Mihaescu, R., Mossel, E., Rao, S.: Maximal accurate forests from distance matrices. In: Apostolico, A., Guerra, C., Istrail, S., Pevzner, P.A., Waterman, M. (eds.) Research in Computational Molecular Biology, pp. 281–295. Springer, Berlin, Heidelberg (2006)
6. Daskalakis, C., Mossel, E., Roch, S.: Evolutionary trees and the ising model on the bethe lattice: a proof of steel's conjecture. Probab. Theory Relat. Fields **149**(1), 149–189 (2011)
7. Daskalakis, C., Mossel, E., Roch, S.: Phylogenies without branch bounds: contracting the short, pruning the deep. SIAM J. Discret. Math. **25**(2), 872–893 (2011)
8. Daskalakis, C., Roch, S.: Alignment-free phylogenetic reconstruction: sample complexity via a branching process analysis. Ann. Appl. Probab. **23**(2), 693–721 (2013)
9. Erdős, P.L., Steel, M.A., Székely, L., Warnow, T.J.: A few logs suffice to build (almost) all trees (i). Random Struct. Algorithms **14**(2), 153–184 (1999)
10. Erdős, P.L., Steel, M.A., Székely, L., Warnow, T.J.: A few logs suffice to build (almost) all trees: part II. Theor. Comput. Sci. **221**(1), 77–118 (1999)
11. Farris, J.S.: A probability model for inferring evolutionary trees. Syst. Zool. **22**(4), 250–256 (1973)
12. Huson, D.H., Nettles, S.M., Warnow, T.J.: Disk-covering, a fast-converging method for phylogenetic tree reconstruction. J. Comput. Biol. **6**(3–4), 369–386 (1999)
13. Lacey, M.R., Chang, J.T.: A signal-to-noise analysis of phylogeny estimation by neighbor-joining: Insufficiency of polynomial length sequences. Math. Biosci. **199**(2), 188–215 (2006)
14. Mihaescu, R., Hill, C., Rao, S.: Fast phylogeny reconstruction through learning of ancestral sequences. Algorithmica **66**(2), 419–449 (2013)
15. Moret, B.M., Roshan, U., Warnow, T.: Sequence-length requirements for phylogenetic methods. In: Guigó, R., Gusfield, D. (eds.) In: International Workshop on Algorithms in Bioinformatics (WABI), pp. 343–356. Springer, Berlin, Heidelberg (2002)
16. Moret, B.M.E., Wang, L.S., Warnow, T.: Toward new software for computational phylogenetics. Computer **35**(7), 55–64 (2002). https://doi.org/10.1109/MC.2002.1016902
17. Mossel, E.: On the impossibility of reconstructing ancestral data and phylogenies. J. Comput. Biol. **10**(5), 669–676 (2003)
18. Mossel, E.: Phase transitions in phylogeny. Trans. Am. Math. Soc. **356**(6), 2379–2404 (2004)
19. Mossel, E.: Distorted metrics on trees and phylogenetic forests. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**(1), 108–116 (2007)
20. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden Markov models. Ann. Appl. Probab. **16**(2), 583–614 (2006)
21. Mossel, E., Roch, S.: Distance-based species tree estimation under the coalescent: information-theoretic trade-off between number of loci and sequence length. Ann. Appl. Probab. **27**(5), 2926–2955 (2017)
22. Mossel, E., Roch, S., Sly, A.: On the inference of large phylogenies with long branches: how long is too long? Bull. Math. Biol. **73**(7), 1627–1644 (2011)
23. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
24. Nakhleh, L., Moret, B.M.E., Roshan, U., John, K.S., Sun, J., Warnow, T.: The accuracy of fast phylogenetic methods for large datasets. In: Altman, R., Dunker, A., Hunter, L., Lauderdale, K., Klein, T. (eds.) In: Pacific Symposium on Biocomputing 2002, pp. 211–222. World Scientific Press, Singapore
25. Pollard, D., Gill, R., Ripley, B.: A User's Guide to Measure Theoretic Probability. Cambridge Series in Statistica. Cambridge University Press (2002)
26. Roch, S.: Toward extracting all phylogenetic information from matrices of evolutionary distances. Science **327**(5971), 1376–1379 (2010)
27. Roch, S., Sly, A.: Phase transition in the sample complexity of likelihood-based phylogeny inference. Probab. Theory Relat. Fields **169**(1), 3–62 (2017)

28. Roch, S., Warnow, T.: On the robustness to gene tree estimation error (or lack thereof) of coalescent-based species tree methods. Syst. Biol. **64**(4), 663–676 (2015)
29. Steel, M.: Phylogeny. Society for Industrial and Applied Mathematics, Philadelphia, PA (2016)
30. Steel, M., Székely, L.: Inverting random functions II: explicit bounds for discrete maximum likelihood estimation, with applications. SIAM J. Discret. Math. **15**(4), 562–575 (2002)
31. Warnow, T.: Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation. Cambridge University Press (2017)
32. Warnow, T., Moret, B.M.E., St. John, K.: Absolute convergence: true trees from short sequences. In: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01, pp. 186–195. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2001)

Chapter 5
Gene Family Evolution—An Algorithmic Framework

Nadia El-Mabrouk and Emmanuel Noutahi

Abstract Most biological discoveries can only be made in light of evolution. In particular, functional annotation of genes is usually deduced from the orthology, paralogy, or xenology relations between genes, which are inferred from the comparison of a gene tree with a species tree. As sequence-only gene tree reconstruction methods often do not allow to confidently discriminate between trees, recent "integrative methods" include information from the species tree. The idea is to consider, in addition to a value measuring the fitness of a tree to a sequence alignment, a measure reflecting the evolution of a whole gene family through gene gain and loss. One such measure is the "reconciliation" cost, i.e., the cost of a gain and loss scenario explaining the incongruence between the gene and species tree. This chapter begins with a review of deterministic algorithms for computing reconciliation distances under various evolutionary models of gene family evolution. We then review integrative methods for correcting a gene tree, based on various strategies for exploring its neighborhood. The considered algorithms are those based on polytomy resolution, tree amalgamation and supertree reconstruction. The goal is to provide a comprehensive overview of existing methods with algorithms presented in concise form. The reader is referred to original papers for more details and proofs of complexity.

Keywords Phylogeny · Gene tree · Duplication · Loss · Horizontal gene transfer · Incomplete lineage sorting · Reconciliation

N. El-Mabrouk (✉) · E. Noutahi
Département d'Informatique et de Recherche Opérationnelle (DIRO),
Université de Montréal, Montreal, Canada
e-mail: mabrouk@iro.umontreal.ca

E. Noutahi
e-mail: fmr.noutahi@umontreal.ca

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_5

5.1 Introduction

Genes are the molecular units of heredity holding the information to build and maintain cells. They are key to understanding biological mechanisms, identifying genetic variation, and designing appropriate gene therapies.

In the course of evolution, genes are mutated, duplicated, lost, and passed to organisms through speciation or Horizontal Gene Transfer (HGT), the exchange of genetic material among coexisting species. Therefore, most biological discoveries can only be made in the light of evolution. Genes originating from the same ancestral copy are called *homologs*. Homologous genes are grouped into *gene families*, usually via sequence similarity methods. Moreover, they can be *orthologs* if their most recent common ancestor has been subjected to a speciation event, *paralogs* if it has been subjected to a duplication event and *xenologs* if they diverged via a HGT event.

Homologous sequences tend to have similar structure and function, and are often located in homologous genomic regions. These properties can be exploited in various biological applications, making deciphering the relation between genes essential for several biological analyses. For example, because homologous genes can be used as markers, they are essential in comparative genomics studies based on gene order, a field widely explored by renowned researchers in computational biology. In particular, Bernard Moret has led the development of highly efficient tools for comparing gene orders [5, 54, 55].

Methods for inferring gene relations are subdivided into tree-based and tree-free methods. Tree-free methods are mostly based on gene clustering according to sequence similarity, (cf., e.g., the COG database [87], OrthoMCL [50], InParanoid [10]). They are often unable to detect the full set of relations between members of a gene family and fail to differentiate orthologs from paralogs and xenologs. On the other hand, tree-based methods consist in reconstructing a phylogenetic tree for the gene family and then inferring the nature of internal nodes (duplication, speciation or HGT) from a *reconciliation*, i.e., an embedding of the gene tree into the species tree. Methods relying on reconciliation, the focus of this chapter, usually yield more accurate gene relations. However, they are very sensitive to the quality of the input trees, a single misplaced branch likely leading to a completely different evolutionary scenario.

Tree reconciliation can be performed through different biological models of evolution, the most common being the Duplication (D), Duplication-Loss (DL) or Duplication-Loss and Transfer (DTL) models. *Incomplete lineage sorting* (ILS), i.e., imperfect segregation of alleles has also been considered, mainly for reconciliation with a non-binary species tree. While most reconciliation methods are based on the parsimony principle of minimizing the number or the cost of induced operations, probabilistic models seeking for a reconciliation with maximum likelihood or maximum posterior probability have also been developed [2, 76, 84] (see [85] for a review). Although relying on more realistic models of gene family evolution through gains and losses, these methods are much slower than parsimony methods. This chapter is dedicated to parsimony methods for reconciliation.

As mentioned above, accurate inference of the true evolutionary history of a gene family through reconciliation strongly depends on the accuracy of the considered gene and species trees. This is the main reason for the continuing effort made to reduce errors in gene tree reconstruction. In particular, standard phylogenetic methods standing solely on sequence alignment (e.g., PhyML [33], RAxML [78], MrBayes [71], PhyloBayes [48]) are often error-prone as they are subject to, among other systematic errors [69], errors arising from the quality of the dataset (e.g., quality of gene annotations, gene family clustering, and alignment). In addition, gene sequences often do not contain enough differentiation (substitutions) to resolve all the branches of a phylogeny, or alternatively, too much such that the substitution history is saturated. The resulting low resolution of gene relations can usually be assessed with measures of statistical support (e.g., bootstrap and posterior probability) on tree branches.

To address the limitation of standard methods, other reconstruction methods, accounting for fitness with the species tree, have been developed. These methods, designated as *integrative methods*, report gene trees with better accuracy compared to sequence-only methods [14, 59, 84, 89]. Most of them rely on a two-steps approach: first compute a tree, or a set of trees, with the best fit to the sequences, and then "correct" the initial tree, or set of trees, according to the reconciliation cost. Four main strategies are considered for the second step: (1) Select neighboring gene trees of an initial tree by performing some branch swapping, typically Nearest Neighbor Interchange (NNI), Subtree Pruning and Regrafting (SPR) or Tree Bisection and Reconnection (TBR) (e.g., GeneTree [62], TreeFix [94], TreeFix-DTL [8], MowgliNNI [58], Notung [18]); (2) Contract branches of weak support and resolve the obtained polytomies (non-binary nodes) (e.g., NOTUNG [18], ProfileNJ [60]); Finally, select a set of trees or clades (leafsets) and construct (3) an amalgamated tree (e.g., ecceTERA [36], ALE [84] or (4) a supertree (e.g., MinSGT [41, 43]).

The first strategy, relying on tree rearrangement events (NNI, SPR, TBR) near poorly supported branches, consists of searching for alternative topologies of an initial gene tree with a better fit to the species tree. Methods based on this strategy explore the tree space often by using search heuristics such as branch-and-bound and hill-climbing. Some of them restrict the candidate alternative topologies to those that cannot be rejected by sequence data. Their main drawback stems from the performance of the criteria used to stop the tree space exploration, which in the worst case can result in exploring the complete exponential-size tree space.

In this chapter, while we focus on the second step of integrative methods, we only present the less straightforward methods based on strategies (2), (3) and (4). After introducing the preliminary notations in Sect. 5.2, the following sections are dedicated to the various formulations of the reconciliation problem depending on the considered trees and evolutionary model (with or without HGTs, considering or disregarding ILS). Section 5.3 is dedicated to the classical reconciliation between a binary gene tree and a binary species tree, Sect. 5.4 presents an extension to non-binary species trees, and Sect. 5.5 deals with the polytomy resolution problem, namely, the reconciliation of a non-binary gene tree with a binary species tree. This latter section is related to strategy (2) described above for integrative methods. We then move, in

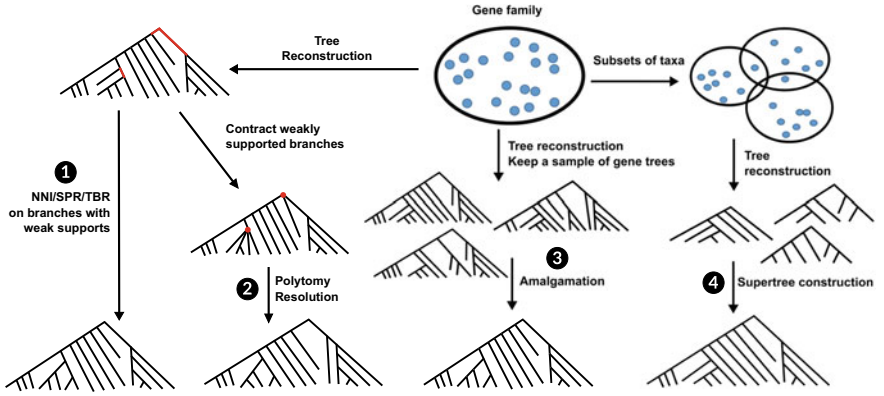


Fig. 5.1 Different strategies for gene tree construction and correction. A single gene tree is constructed from the sequences of all the genes of the gene family: in (1), tree rearrangement methods around weakly supported branches are used to search an alternative tree minimizing with a better reconciliation cost; in (2), branches with weak support are rather contracted and the obtained non-binary nodes resolved according to the reconciliation cost with the species tree. (3) *Amalgamation:* a sample of gene trees is first reconstructed from a single gene family, then a single gene tree is reconstructed based on "trusted" clusters of the tree sample. (4) *Supertree*: The gene family is first subdivided into a set of, possibly overlapping, groups of genes (usually, groups of orthologs), a tree is reconstructed for each group and these trees are then combined into a single supertree displaying all of them

Sect. 5.6, to strategy (3) and (4), taking advantage of a set of gene trees rather than a single input gene tree, through amalgamation or supertree methods, as illustrated in Fig. 5.1. Section 5.7 then presents, for the DL model, a unifying view simultaneously considering polytomy resolution and supertree reconstruction in a single framework for gene tree correction. We end this chapter with a discussion in Sect. 5.8.

5.2 Trees

We denote, respectively, by $V(T)$, $E(T)$, and $L(T)$ the set of nodes, edges and leaves of a tree T. Notice that $L(T) \subset V(T)$. We say that T is a *tree on* $L(T)$. Unless stated differently, all trees considered in this chapter are *rooted*, i.e., they admit a single node $r(T)$ called the root of T.

Let x be a node of $V(T)$; y is an *ancestor* of x if y is on the path from x to the root; y is a *descendant* (respectively, proper descendant) of x if y is on the path from x to a leaf of T including x (respectively, excluding x); x and y are *incomparable* if y is neither an ancestor nor a descendant of x. If (x, y) is an edge of T, then x is the *parent* $p(y)$ of y and y is a *child* of x ($y \in Ch(x)$).

For a tree T, we denote by T_x the subtree of T rooted at $x \in V(T)$. Two subtrees T_x and T_y of T are *separated* iff x and y are incomparable nodes of T. Given a subset

L of leaves, we call the *lowest common ancestor* (LCA) of L in T and denote by $lca_T(L)$ the common ancestor of L in T that is the farthest from the root. We also denote by $T|_L$ the tree with leafset $L \cap L(T)$ obtained from the subtree of T rooted at $lca_T(L \cap L(T))$ by removing all leaves that are not in both L and $L(T)$, and then all internal nodes with a single child.

A tree T' is said to be an *extension* of a tree T if it can be obtained by a sequence of graftings, where each *grafting* consists of subdividing an edge (x, y) of $E(T)$ by creating a new node z between x and y, then adding a leaf l with parent z.

In this chapter, all considered trees have internal nodes with at least two descendants. An internal node x of T is *binary* if it has exactly two descendants. A *binary tree* is a tree with all internal nodes being binary nodes. A *non-binary tree* has at least one internal node which is a *polytomy*, i.e., a node with more than two descendants.

Definition 1 (*binary refinement*) A *binary refinement* $B = B(T)$ of a tree T, is a binary tree such that $V(T) \subseteq V(B)$ and such that for every $x \in V(T)$, $L(T_x) = L(B_x)$.

In other words, a binary tree $B(T)$ is a binary refinement of T if whenever a node x is an ancestor of y in T, x is also an ancestor of y in $B(G)$.

Gene and species trees: Two types of trees are considered: species trees and gene trees (see Fig. 5.2). A *species tree* S for a set $\Sigma = \{\sigma_1, \ldots, \sigma_t\}$ of species represents an ordered set of *speciation events* (the separation of one species into two different species) that have led to Σ.

Inside the species' genomes, genes undergo speciation when the species to which they belong to speciate, but also *duplication* i.e., the creation of a new locus, *loss* of a locus, and *Horizontal Gene Transfer* (HGT) when a gene is transmitted from a source species to a different, coexisting target species.

A *gene family* Γ is a set of genes sharing a common ancestor, and a *gene tree* G is a tree on a gene family Γ. We denote by $s(x)$ the genome of Σ to which x belongs.

When no distinction needs to be made between gene copies in the same genome, genes can just be identified by their corresponding genome, and thus a gene tree can be represented as a tree on Σ with possibly repeated leaf-labels (see Fig. 5.3).

5.3 Reconciliation of a Binary Gene Tree with a Binary Species Tree

The evolutionary history of a gene family is usually inferred from the embedding of its corresponding gene tree into the species tree, through a process called reconciliation explaining incongruities between gene and species trees by gene evolution events.

More precisely, a *reconciliation* $R(G, S)$ of a gene tree G with a species tree S (if no ambiguity arises, we will just write R) is a node-labeled extension of the gene tree G reflecting a history of speciation and gene gain and loss in agreement with S (see Fig. 5.2). Each node x of $V(R)$ (internal or leaf) is mapped to a node

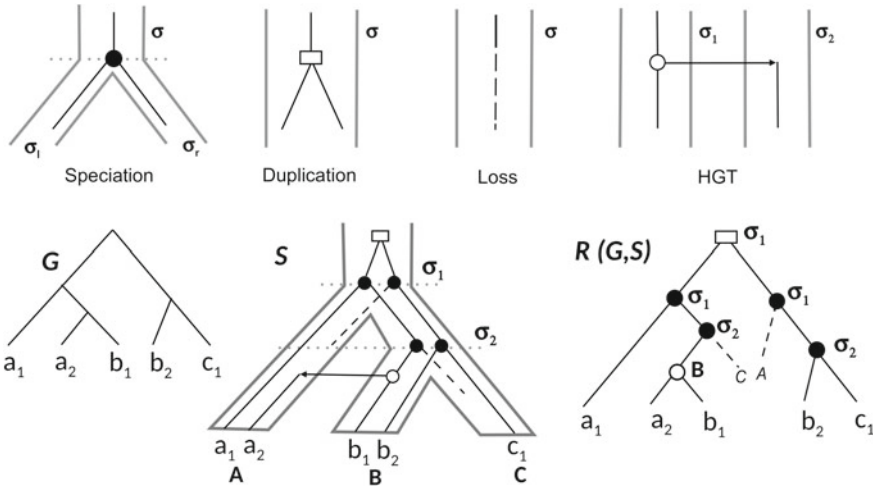


Fig. 5.2 *Top*: A speciation (black circle), duplication (white rectangle), loss (dotted line) and HGT (white circle) events. For the speciation event, σ_l and σ_r refer to the two species descendent from the species σ; for the HGT event, σ_1 is the source and σ_2 the target unrelated species. *Bottom*: (left) A gene tree G for the gene family $\Gamma = \{a_1, a_2, b_1, b_2, c_1\}$, where each lower case denotes a gene belonging to the corresponding genome in upper case; (middle) an evolutionary history of Γ embedded in the species tree $S = (A, (B, C))$; (right) the reconciliation $R(G, S)$ corresponding to the given evolutionary history. Each internal node and grafted leaf x of $R(G, S)$ is labeled with $s(x)$. The edge (B, a_2) is a HGT edge

$s(x) \in V(S)$. Some branches of R may also be labeled as transfer edges. A formal definition follows.

Definition 2 (*Reconciled gene tree*) Let G be a binary gene tree and S be a binary species tree. A *reconciliation* $R(G, S)$ of G with S is an extension of G such that, for each internal node x of $R(G, S)$ with two children x_l and x_r, one of the following cases holds:

1. $s(x_l)$ and $s(x_r)$ are the two children of $s(x)$, in which case x is a speciation node;
2. $s(x_l) = s(x_r) = s(x)$ in which case x is a duplication node representing a duplication in $s(x)$;
3. one of $s(x_l)$ and $s(x_r)$ is equal to $s(x)$ and the other is incomparable to $s(x)$. Let y corresponds to the element of $\{x_l, x_r\}$ such that $s(y)$ is incomparable to $s(x)$. Then x is a HGT node representing a HGT event with source genome $s(x)$ and target genome $s(y)$, and (x, y) is a HGT edge.

Each grafted leaf x corresponds to a loss in $s(x)$.

Two genes are said *orthologs* if their LCA in $R(G, S)$ is a speciation event, *paralogs* if it is a duplication event and *xenologs* if it is a HGT. For example in Fig. 5.2, b_2, c_1 are orthologs, b_2, a_1 are paralogs and a_2, b_1 are xenologs.

Remark 1 A more flexible definition of xenologs, where two genes are said to be xenologs if the history since their LCA involves a HGT, is also considered in the literature [27]. With this definition, a pair of xenologous genes can diverge through speciation, duplication or transfer. For example with this definition, genes a_1, b_1 in Fig. 5.2 are xenologs that diverged through a speciation. To avoid further ambiguity, a new classification of xenologs into subtypes, which takes into account the evolutionary events at the divergence of gene pairs and the relative timing of transfer and speciation events was also recently proposed [20]. In this chapter, we will consider the simplest event-based definition of xenologs through divergence via a transfer event, inducing a unique assignment type for each pair of genes into orthologs, paralogs or xenologs. Notice that with this definition, orthologs are not restricted anymore to genes from different species (see [20] for a discussion). For example, in Fig. 5.2, a_1 and a_2 are orthologs although they are found in the same present-day species A.

The standard parsimony criteria used to choose among the large set of possible reconciliations are the minimum number of duplications (D), duplications and losses (DL), or duplications, losses and HGTs (DTL) events induced by the reconciliation. The first two distances can be computed in linear time using the *LCA mapping* [30, 96, 99] (see Sect. 5.3.1 below). An algorithm enumerating all solutions for general costs with different event penalties was described in [22] for the DL model and extended to DTL in [15].

5.3.1 DL Reconciliation

The *LCA-mapping* between a gene tree G and a species tree S maps each node $x \in V(G)$ toward a genome $s(x) \in V(S)$, such as $L(S_{s(x)})$ is the smallest set of genomes to which all genes in $L(G_x)$ belong. Formally, $s(x) = lca_S(\{s(y) : y \in L(G_x)\})$ in the species tree. Note that the LCA-mapping is unique for any given pair (G, S).

Given that mapping, each internal node x of G can be labeled as a duplication node if $s(x_l) = s(x)$ and/or $s(x_r) = s(x)$, otherwise it is a speciation node. The total number of losses correspond to the minimum number of grafting on G required to have a reconciliation $R(G, S)$. The reconciliation induced by the LCA-mapping, called *LCA-reconciliation* is optimal for both D and DL distances. It is also the unique reconciliation minimizing the DL distance (see Fig. 5.3(1) for an example).

We highlight two types of duplication nodes inferred from LCA mapping. Consider each gene of G as simply identified by the genome it belongs to. Let x be a duplication node of G with children x_l and x_r. It is a *Nonapparent duplication* (*NAD*) iff $L(G_{x_l}) \cap L(G_{x_r}) = \emptyset$. In other words, the reason for x being a duplication node is not the presence of paralogs in the same genome, but rather an inconsistency with the species tree. A duplication which is not a NAD is an *Apparent Duplication* (*AD*) node, i.e., a node with the left and right subtrees sharing a common leaf-label.

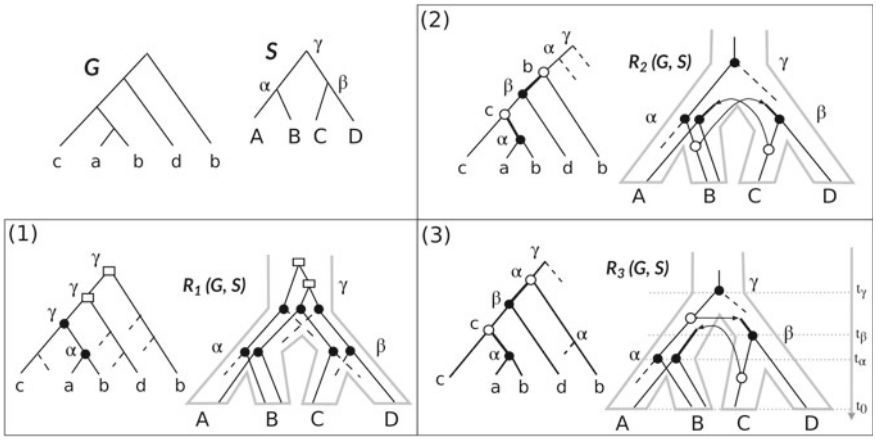


Fig. 5.3 Three different reconciliations for the species tree S and the gene tree G, for the gene family $\Gamma = \{a, b, b, c, d\}$, where each lower case denotes a gene belonging to the corresponding genome in upper case; (1) An evolutionary scenario optimal for the D and DL distances (two duplications and five losses); G is labeled according to the LCA-mapping; (2) A DTL-scenario with two HGTs and two losses. This scenario is cyclic, and is therefore infeasible; (3) An alternative and acyclic DTL-scenario with two HGTs and one loss; this scenario is also biologically unfeasible as it is not date-respecting, according to the considered speciation times

For example in Fig. 5.3(1), the lower duplication node of G is a NAD, while the upper duplication node is an apparent duplication, as its left and right subtrees each contains a leaf labeled b.

While apparent duplications are supported by the presence of paralogs, in the same genome, that are necessarily the result of duplication, NAD nodes have been flagged as potential errors in many studies, and in particular in the Ensembl Compara gene tree database [28]. The distinction between these two types of duplication nodes is required for certain formulations of the gene tree correction problem [40], or for considering an optimal history accounting for ILS, as we will see later.

5.3.2 DTL Reconciliation

In contrast with the DL reconciliation framework, the optimal DTL reconciliation is not unique, and cannot be computed by means of the LCA-mapping. With HGTs, a gene evolution is not restricted anymore within the parental edges of its genome in the species tree. As such, to the standard vertical transmission of genes from one ancestor genome to its descendants, there is an additional need to consider transmission between incomparable nodes of the species tree. Such transmissions are represented in the reconciliation by a transfer edge (x, y) corresponding to a gene transfer from a source genome $s(x)$ to a target genome $s(y)$. For a HGT to

be biologically feasible, both genomes are required to be contemporary at the time of the transfer event. Therefore, a "consistent" HGT scenario should allow a total temporal ordering of the internal nodes of the species tree S. As demonstrated by Tofigh et al. [91], this requires the DTL-reconciliation to be *acyclic*, as defined below.

Definition 3 A reconciliation $R(G, S)$ is acyclic if and only if there is a total order $<$ on $V(S)$ such that:

(1) if $(s, s') \in E(S)$ then $s < s'$ and
(2) if (x, y) and (x', y') are transfer edges in G such that y' is a descendant of y in $R(G, S)$, then $p(s(x)) < s(y')$.

For example, scenario 2 in Fig. 5.3(2) is a cyclic DTL-scenario, as the ordering defined by the above definition would lead, for the two transfer edges of G, to $\alpha < \alpha$. On the other hand, scenario 3 (Fig. 5.3(3)) is acyclic.

The problem of finding a most parsimonious acyclic (i.e., time-consistent) DTL-scenario is NP-hard [23, 24, 34, 61]. However, the problem becomes polynomial if the acyclicity requirement is dropped [6, 91]. In that case, the main idea for computing an optimal DTL-reconciliation is to consider all possible mappings of G nodes to S nodes, using a dynamic programming approach.

More precisely, let $c(x, s)$ be the minimum cost of a reconciliation of G_x with S such that x is mapped to $s \in V(S)$. The gene tree G is processed in post-order traversal, with the base case corresponding to leaves $x \in L(G)$, treated as follows:

$$For \;\; x \in L(G),\; c(x, s) = \begin{cases} 0, & \text{If } s = s(x), \\ +\infty, & \text{Otherwise.} \end{cases}$$

As for an internal node x with children y and z, we have to consider the three possibilities of x being labeled as a speciation, duplication or HGT node, with $c_s(x, s)$, $c_d(x, s)$, and $c_t(x, s)$ representing these three mutually exclusive cases. Then, $c(x, s) = \min\{c_s(x, s), c_d(x, s), c_t(x, s)\}$. Finally, the minimum cost of a reconciliation of G with S is $\min_{s \in V(S)} c(r(G), s)$.

For simplicity, we report below the recurrences when considering the cost of reconciliation as being the number of duplications and HGT [91].

$$c_s(x, s) = \begin{cases} \min\{c(y, t) + c(z, u) \text{ for all } t, u \\ \quad \text{incomparable and such that } lca(t, u) = s\}, & \text{If } s \text{ is an internal node of } S, \\ +\infty, & \text{Otherwise.} \end{cases}$$

$$c_d(x, s) = \min\{1 + c(y, t) + c(z, u) \text{ for all descendants } t,\; u \text{ of } s \text{ in } S\}$$

$$c_t(x, s) = \min\{1 + c(y, t) + c(z, u) \text{ for all } t \text{ being descendant of } s \text{ in } S \text{ and all } u \text{ being incomparable to } s\}$$

A straightforward implementation of these recurrences lead to an algorithm in $O(mn^2)$ time, where $m = |V(G)|$ and $n = |V(S)|$. This time complexity has been further improved to $O(mn)$ [90].

Notice that losses may be essential for distinguishing between duplications and HGT events. The above recurrences have to be adapted to handle losses. David and Alm [21] have described an algorithm for the DTL distance running in $O(mn^2)$, while Bansal et al. [6] described RANGER-DTL, an algorithm running in $O(mn)$.

When divergence time information, or a temporal ordering of internal nodes, is available for S, then the DTL-scenario must respect this ordering (i.e., HGT events are constrained to occur only between coexisting species). A DTL-scenario respecting a dated tree is called a *date-respecting DTL-scenario*. Bansal et al. [6] show how the definition of a reconciliation and the above recurrences can be adapted to solve this problem. They give an algorithm with $O(mn \log n)$ time complexity.

For example, scenario 3 of Fig. 5.3 is not date-respecting. Notice that a date-respecting DTL-scenario is not necessarily time-consistent. In fact, scenarios may be locally consistent (i.e., HGT events occurring between coexisting species), but globally inconsistent. Global consistency may be obtained by subdividing the species tree S into slices and exploring slices one after the other. This strategy has been first used in [51], leading to an $O(nm^4)$ algorithm. Later, Doyon et al. [24] have improved the computation of a most parsimonious time-consistent DTL-reconciliation with a dated species tree to $O(mn^2)$.

5.3.3 *Binary Gene Tree Reconciliation in Presence of ILS*

When a population of individuals undergoes a series of speciations in a short period of time, different alleles for the same gene locus may remain present in a given lineage, and then eventually fixed differently in descendant lineages [52]. This phenomenon, known as *deep coalescence* or *Incomplete Lineage Sorting* (ILS) may also explain discrepancies between a gene tree and a species tree. For example in Fig. 5.4, the subtree $((a, b_1), c_1)$ of G, which is incongruent with the species tree $(A, (B, C))$, may be explained from the history depicted in the left backbone of (i), which involves no duplication, but simply the fact that the allele inherited in C is different from the one inherited in A and B.

In the absence of paralogous genes in the same genome, inconsistencies between a gene tree and a species tree can always be explained through ILS. Wu and Zhang [93] have shown that a unique reconciliation with minimum deep coalescence cost can be obtained in that case, using LCA-mapping. It is, however, necessary to take duplication events into account as ILS cannot explain the presence of additional loci. For example, in Fig. 5.4, while the NAD (nonapparent duplication) in G can be adequately explained through ILS, the apparent duplication node above it necessarily involves the creation of a second locus. As seen in Fig. 5.4(iii), (iv), ILS-aware reconciliation methods may produce evolutionary histories with fewer losses, highlighting the need of models jointly considering duplication, loss, HGT and ILS events. In a recent paper, Bork et al. [13] have shown that the duplication-loss-ILS reconciliation problem is NP-hard, even when only duplications are to be minimized.

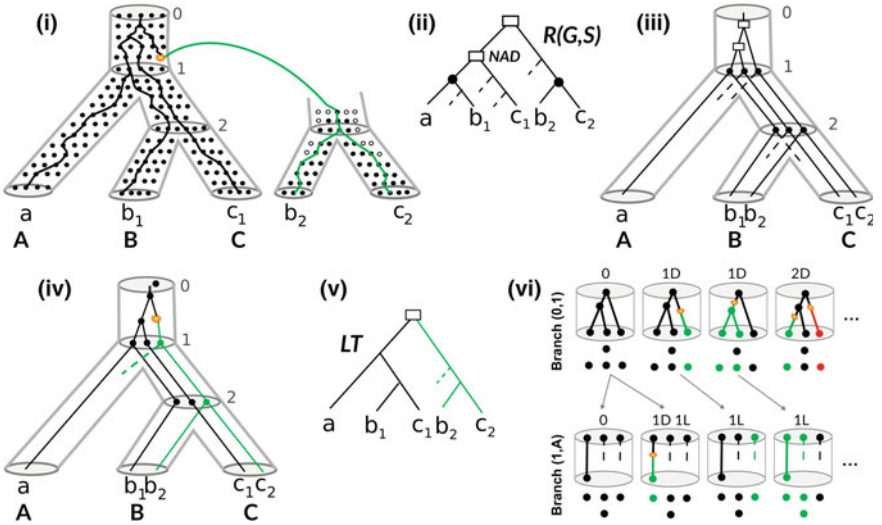


Fig. 5.4 Gene family evolution and incomplete lineage sorting. (i) Evolution of a gene family inside a species tree $S = (A, (B, C))$, in the context of a population. Each species tree backbone contains the evolution of a single locus and each row represents a generation of individuals in a population. The lines inside the tree backbones represent the evolution of the gene family leading to the tree G in (ii). In this example, the evolution of two loci (black and green) are depicted. Two alleles of the black locus are present at the time of speciation 1. The first allele is fixed in A and B, whereas the second is only fixed in C. The green locus was created after an ancestral duplication occurring just before speciation 1, and was lost in genome A; (ii) The resulting gene tree G for the gene family $\Gamma = \{a, b_1, b_2, c_1, c_2\}$ is the represented reconciled tree $R(G, S)$, ignoring losses (dotted lines) and internal node labeling. Duplication nodes, inferred from the LCA-mapping, are not coherent with the true evolutionary history of the gene family. (iii) A different representation of $R(G, S)$ reflecting the number n_s of gene copies in each genome s. For example, for the branch (0, 1), we have $n_0 = 1$ and $n_1 = 3$. (iv) A different scenario able to explain incongruities between the gene and species tree through duplication, loss and deep coalescence. This more parsimonious history involves one duplication, a loss, and a deep coalescence event. It relies on the labeled coalescent tree model which simultaneously describes the species, locus and gene trees, as well as the reconciliations between them. (v) The locus tree (LT) induced by the scenario shown in (iv). (vi) Enumeration of the possible locus maps for each branch of the species tree. Each locus is shown with a different color and new locus are created by duplications. Only some locus maps for branches (0, 1) and (1, A) are shown. The mapping is based on the total number of gene lineages at the start and end of each edge of the species tree, which can be determined with LCA-mapping

Very few papers have attempted to jointly model ILS and other macro-evolutionary events during gene and species tree reconciliation. In two papers by Durand's group [81, 92], the problem is reformulated as a reconciliation between a binary gene tree and a non-binary species tree minimizing the DL/DTL cost. Their algorithm first requires contraction of short branches of the species tree into polytomies and ILS are only allowed at those unresolved nodes and remain unpenalized. Section 5.4 is dedicated to this algorithm.

On the other hand, Kellis et al. [67, 95] have considered a coalescent model for reconciling a binary gene tree with a binary species tree, accounting for duplications, losses and deep coalescence. They first devised a probabilistic algorithm, called DLCoal [67]. Although efficient, this algorithm is highly parameterized, making it impracticable. Subsequently, they proposed a parsimony-based algorithm, called DLCpar [95], introducing the concept of a *label coalescent tree (LCT)* (see Fig. 5.4(iv)), which simultaneously describes the reconciliation between a gene tree, a locus tree, and a species tree. This latter algorithm proceeds in the following steps:

1. Use the LCA-mapping between G and S to determine all implied speciation nodes and count, for each branch (x, y) of the species tree, the numbers n_x and n_y of gene copies at x and y.
2. For each branch (x, y), in a pre-order traversal of S, enumerate all possible scenarios of DL and ILS events leading from n_x to n_y gene copies (see Fig. 5.4(vi)). This yields the set of possible species-specific locus maps that associates each node of the gene tree to the locus in which it evolves. The event cost for each branch of S can be computed by counting the number of additional loci and lost loci, respectively corresponding to duplications and losses, as well as the number of extra lineages caused by deep coalescence (see Fig. 5.4(vi)). In practice, some histories are not considered since they are never most parsimonious.
3. Perform a post-order traversal of S, and for each branch (x, y) and each assignment (n_x, n_y), use dynamic programming to determine the minimum cost on the subtree of S rooted as x, computed as the cost of the branch (x, y) plus the minimum cost of the left and right subtrees rooted at y, where y is assigned n_y loci. The minimum among all possible choices is selected as the most parsimonious reconciliation. Optimal loci at the start and end of each branch can then be assigned with a traceback starting from the root of the species tree.

Although not explicitly given in the paper, the complexity of the algorithm strongly depends on the size of the locus maps set and on the choices considered for each branch of the species tree. This part is not detailed in the paper. In particular, the method is supposed to search over the entire space of reconciliations, but it is not clear whether it leads to a heuristic or to an exact algorithm.

In a follow-up paper, Rogers et al. [70] further attempt to extend the LCT model in order to address one of its shortcomings, namely the assumption of a single haploid sample for each species. More recently, Chan et al. [16] have proposed the first FPT (fixed-parameter-tractable) algorithm that computes the most parsimonious time-consistent reconciliation fully accounting for ILS, duplications, HGTs and losses (IDTL). This algorithm is an extension of the DTL-reconciliation described in [24] with modifications to allow ILS, and has a total complexity of $O(|V_G|(|V_S|^2 + |V_S|n_k 2^{k_s})2_s^k)$, where k is the number of branches in the largest ILS subtree (i.e., subtrees of the species tree where ILS occur) and n_k the number of ILS subtrees.

5.4 Reconciliation with a Non-binary Species Tree

The LCA-mapping can naturally be generalized to a non-binary species tree. However, the LCA-reconciliation used for binary gene and species trees will not produce correct gene evolution history when applied to non-binary species trees. In fact, a node of G and its child mapping to the same non-binary node of the species tree does not necessarily indicate a duplication. In [97], Zheng et al. proved that the general reconciliation problem of a gene tree G with a non-binary species tree S via binary refinement is NP-hard, even when only duplications are considered. In the same paper, they proposed a heuristic for the problem also allowing for polytomies in the gene tree.

We can distinguish two reasons for the presence of non-binary nodes in a species tree. They can either represent "true" evolutionary events, i.e., adaptive radiations leading to the emergence of a set of species from a single ancestral one, or can be caused by a lack of resolution in the species tree, due to methodological reasons. Such non-binary nodes are called *hard* in the former case and *soft* in the latter case. A soft polytomy may be due to short time since speciation, leading to genetic drift.

In either case, non-binary nodes of a species tree often correspond to populations with substantial genetic diversity, and coexisting multiple alleles. It is expected that some gene families might exhibit imperfect segregation of all their alleles (in other words ILS) at these nodes. Therefore, a subtree of the gene tree whose root maps to a polytomy in the species tree may be differently explained by speciation, duplication or ILS, depending on the considered resolution of that polytomy.

Vernot et al. [92] have considered the problem of finding a most parsimonious DL scenario explaining the differences between a binary gene tree G and a non-binary species tree S, assuming that disagreements between the two trees can stem from either duplication or ILS. Their algorithm only considers the possibility of ILS at non-binary nodes of S. The main idea of their algorithm is to identify *required duplications*, i.e., those disagreements with the species tree that can only be explained by a duplication. Clearly, these nodes are those in G that would be labeled as duplication in all resolutions of S. However, as shown in [92], there is no need to try all the resolutions of S.

The procedure described in [92] consists of a post-traversal of G during which each node x of $V(G) \setminus \{r(G)\}$ is labeled by the set $N(x)$, which is the subset of $\{h : h \in Ch(s(p(x)))\}$ such that each element $h \in N(x)$ has at least one descendant in $\{s(l) : l \in L(G_x)\}$. This set represents the minimum set of nodes in $V(S)$ that would be traversed from $s(x)$ to the mapping of x's children, regardless of the resolution of S. Consequently, a node x with children x_l and x_r is a required duplication if and only if $N(x_r) \cap N(x_l) \neq \emptyset$ (see Fig. 5.5 for an example).

The set labeling a node of G is of size $O(k_S)$ where k_S is the maximum outdegree in S. Based on this fact, Vernot et al. [92] have described an algorithm for the D distance running in $O(|V(G)|(k_S + h_S))$ time, where h_S is the height of S (i.e., maximum number of nodes from the root to any leaf of S). However, inferring the induced minimum number of losses is not as straightforward as for binary species trees. In

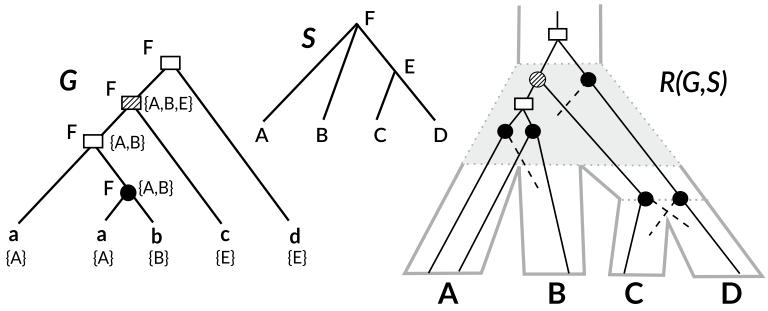


Fig. 5.5 A species tree S for the genome set $\Sigma = \{A, B, C, D\}$; A gene tree G for the gene family $\Gamma = \{a, a, b, c, d\}$, where each small letter designs a gene belonging to the corresponding genome in upper case. The tree G is labeled according to LCA-mapping suggesting three duplication nodes (rectangles). However, according to the $N(x)$ labeling in brackets, only two duplications are required, while the third (striped rectangle/circle) can be explained through ILS instead (see history in the right side), leading to a most parsimonious DL scenario involving two duplications and four losses

fact, for a loss associated to a polytomy, it is not generally possible to determine the exact lineage in the gene tree in which the loss has occurred, and several edges of G have to be tested. An exponential algorithm running in $O(|V(G)|k_S 2^{2k_S})$ was described.

In [81], Stolzer et al. further extended the framework to HGT events and developed an algorithm running in $O(|V(G)|(h_S + k_S)(V|S| + n_k 2^{k_S})^2)$. Although their algorithm does not guarantee a time-consistent reconciliation, temporal feasibility of each scenario is evaluated a posteriori. Both DL and DTL algorithms are implemented in NOTUNG.

5.5 Reconciliation of a Non-binary Gene Tree with a Binary Species Tree

We will detail the most efficient algorithms for DL reconciliation, and end up with a brief discussion on extensions to DTL reconciliation of a non-binary gene tree G with a binary species tree S. This problem is motivated by the gene tree correction problem, where a non-binary gene tree can be obtained from an initial tree by contracting weakly supported branches. In other words, the polytomies of G are considered soft, i.e., reflect non-resolved parts of the tree. The goal is then to find an appropriate refinement (as defined in Sect. 5.2) of this non-binary gene tree.

Definition 4 (*Resolution*) A *resolution* of G with respect to S is a reconciliation $R(B, S)$ between a binary refinement B of G and S. The set of all possible resolutions of a gene tree G is denoted $\mathscr{R}(G)$.

The optimization problem follows.

MINIMUM RESOLUTION PROBLEM:
Input: A binary species tree S and a non-binary gene tree G.
Output: A *Minimum Resolution* of G with respect to S (or simply *Minimum Resolution of* G), e.g., a resolution of G of minimum reconciliation cost with respect to S.

As first noticed by Chang and Eulenstein [17], each polytomy of G can be considered independently and a minimum resolution of G can be obtained by a depth-first procedure that iteratively solves each polytomy G_x for each internal node x of G.

An $O\left(|V(S)||V(G)|^3\right)$ algorithm for the resolution of a non-binary gene tree minimizing duplications and losses was first considered in NOTUNG [25]. The same year, Chang and Eulenstein [17] also described an algorithm with a better complexity, running in $O\left(|V(S)||V(G)|^2\right)$. In 2012 [45], we developed the first linear-time algorithm for resolving a polytomy (a single unresolved node), leading to an overall quadratic-time algorithm for a whole tree. An algorithmic result extending linearity to a whole gene tree was later obtained by Zheng and Zhang [98]. The key idea is to resolve each polytomy with a species tree restricted to the smallest necessary set of genomes. Their algorithm does not allow, however, to output all solutions and is restricted to unit cost for duplications and losses. Based on the same optimization idea, we developed PolytomySolver [42] which is a generalization of the dynamic programming algorithm given in [45], allowing for both event-specific and species-specific costs. The time complexity of PolytomySolver is linear for the unit cost and quadratic for the general cost, which outperforms the best-known solutions so far by a linear factor.

In the rest of this section, we describe the dynamic programming technique in PolytomySolver for the resolution of a single polytomy under the DL distance with unitary event costs. More details, complexity improvement, extension to other costs and to a full non-binary gene tree, can be found in [42].

5.5.1 PolytomySolver

In the following, to prevent penalizing losses in genomes with no descendant genes in G, the species tree is restricted to $S|_{\{s(x)\,:\,x\in L(G)\}}$ and we will simply continue to refer to it as S.

PolytomySolver proceeds with a recursion made on the subtrees of S. Define the multiplicity $m(s)$ of $s \in V(S)$ in G as the number of times it appears in G, i.e., $m(s) = |\{x \in L(G) : s(x) = s\}|$. An (s,k)-*resolution* of G is a forest of k reconciled gene trees $\mathscr{T} = \{T_1, \ldots, T_k\}$ s.t. $\forall\ 1 \leq i \leq k, s(r(T_i)) = s$, and each leaf x of G with $s(x)$ being a descendant of s is present as a leaf of some tree of $\mathscr{T}$ (see Fig. 5.6 for an example). Leaves of trees in $\mathscr{T}$ that do not appear in $L(G)$ represent losses. We denote by $c(\mathscr{T})$ the reconciliation cost of the forest $\mathscr{T}$. This cost is the sum of the reconciliation costs of all $T_i \in \mathscr{T}$.

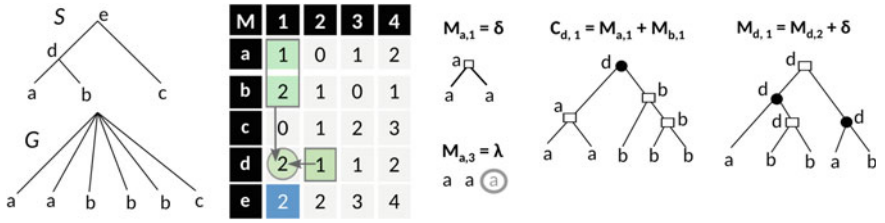


Fig. 5.6 (Figure from [42]; use permitted under the Creative Commons Attribution License CC-BY 3.0) A polytomy G and a species tree S. Squares on trees illustrate duplications, whereas speciation are denoted by a black circle. To the right of table M, the forests corresponding to an $(a, 1)$ and $(a, 3)$-resolution are given, where the gray circled a illustrates a loss. We illustrate the $(d, 1)$-resolution, rooted at a speciation node, corresponding to $C_{d,1} = 3$ (obtained from the vertical arrows in table M), and an optimal $(d, 1)$-resolution, obtained from a $(d, 2)$-resolution (horizontal arrow in M). The optimal cost for the resolution of G ($M_{e,2} = 2$) is highlighted in blue

The cost of a minimum resolution of G can be computed using a dynamic programming algorithm that fills a table M. Each cell $M_{s,k}$ of M corresponds to the minimum cost of an (s, k)-resolution for a given node s of S and a given integer $k \geq 1$ ($M_{s,k} = +\infty$ for $k < 1$). The final cost of a minimum resolution of G is given by $M_{r(S),1}$. The table M can be computed, line by line, in a bottom-up traversal of S. Although k is unlimited (number of gene losses is unlimited), we have shown in [42] that there is no need to consider values larger than $|V(G)| - 1$.

Lemma 1 gives the base case to compute $M_{s,k}$ when $s \in L(S)$. It follows from the fact that, if k is larger than the number of available leaves, then additional leaves corresponding to gene losses are required; otherwise, leaves have to be joined under duplication nodes. An illustration of this lemma is shown in Fig. 5.6 where it is used to compute the first three lines of M.

Lemma 1 *(Base case) For a leaf node s of S, if $k > m(s)$ then $M_{s,k} = k - m(s)$; otherwise $M_{s,k} = m(s) - k$.*

For an internal node s of S, speciation events also need to be considered. We require an intermediate cost table C where each entry $C_{s,k}$ represents the minimum cost of an (s, k)-resolution in which every tree is rooted at a speciation node with two children or is a leaf of G already mapped to s. For $k > m(s)$, an (s, k)-resolution of cost $C_{s,k}$ can only be obtained from an $(s_l, k - m(s))$-resolution and an $(s_r, k - m(s))$-resolution by first generating $k - m(s)$ speciation nodes, mapped to s, each joining a pair (s_l, s_r), then adding the $m(s)$ trees already available (see for example the $(d, 1)$-resolution corresponding to $C_{d,1}$ in Fig. 5.6; in this case $m(d) = 0$). Thus, we define:

$$C_{s,k} = M_{s_l,k-m(s)} + M_{s_r,k-m(s)} \text{ if } k > m(s) \text{ and } C_{s,k} = +\infty \text{ otherwise} \qquad (5.1)$$

As nodes mapped to s are not necessarily speciation nodes but can also correspond to duplications, it is readily seen that $M_{s,k} \leq C_{s,k}$. A recurrence for computing $M_{s,k}$ follows.

Lemma 2 *For an internal node s of S, $M_{s,k} = \min(M_{s,k-1} + 1, M_{s,k+1} + 1, C_{s,k})$.*

In Lemma 2, the first term of $M_{s,k}$ corresponds to a loss, while the second corresponds to a duplication at s.

Since $M_{s,k}$ depends on $M_{s,k+1}$ and vice-versa, the recurrence cannot be used to compute C and M. This dependency can, however, be avoided due to a strong property on lines of M. In [45] we have shown that each line M_s is characterized by two values k_1 and k_2 such that, for any $k_1 \leq k \leq k_2$, all $M_{s,k}$ have a single minimum value γ, for any $k \leq k_1$, $M_{s,k-1} = M_{s,k} + 1$, and for any $k \geq k_2$, $M_{s,k+1} = M_{s,k} + 1$. In other words, M_s can be treated as a convex function fully determined by k_1, k_2 and its minimum value γ. We say M_s has a *minimum plateau* between k_1 and k_2. For example, line M_d in Fig. 5.6 is fully determined by $k_1 = 2$ and $k_2 = 3$ and its minimum value $\gamma_d = 1$.

Theorem 1 (Recurrence 1) *Let k_1 and k_2 be the smallest and largest values, respectively, such that $C_{s,k_1} = C_{s,k_2} = \min_k C_{s,k}$. Then,*

$$M_{s,k} = \begin{cases} C_{s,k} & \text{if } k_1 \leq k \leq k_2 \\ \min(C_{s,k}, M_{s,k+1} + 1) & \text{if } k < k_1 \\ \min(C_{s,k}, M_{s,k-1} + 1) & \text{if } k > k_2 \end{cases}$$

Theorem 1 shows how a row M_s for an internal node s of S can be computed: for each k, compute $C_{s,k}$ using recurrence Theorem 1 and keep the two columns k_1 and k_2 setting the bounds of the convex function's plateau. The $M_{s,k}$ values at the left and right of the minimum plateau can then be easily computed from the value of the minimum plateau. These recurrences, with the base case for S leaves given in Lemma 1, describe how the dynamic programming algorithm of PolytomySolver works.

Algorithm 1 describes the computation of table M. We refer the reader to [45] for the reconstruction of a solution from M, which is accomplished using a standard backtracking procedure. Moreover, we show in [42] that k_1 and k_2 for each $M(s)$ can be computed in constant time from M_{s_l} and M_{s_r} vectors. This implies a linear-time algorithm for the computation of $M_{root(S),k}$.

Unrooted trees: If the gene tree is unrooted, an exhaustive testing of all roots can be done with PolytomySolver, ProfileNJ [60] and NOTUNG [18]. A series of papers by Gorecki et al. also consider the properties of the plateau to avoid exploring all branches [31, 32] of unrooted gene trees.

Algorithm 1 Compute $M(G, S)$

for each node $s \in V(S)$ visited in post-order **do**
  if s is a leaf **then**
    $M_{s,k} = |k - m(s)|$ for each k;
  else
    Compute $C_{s,k} = M_{s_1,k-m(s)} + M_{s_2,k-m(s)}$ for each k;
    find k_1, the smallest index such that C_{s,k_1} is minimum;
    find k_2, the largest index such that C_{s,k_2} is minimum;
    $M_{s,k} = C_{s,k}$ for each $k_1 \leq k \leq k_2$;
    for each $k < k_1$ **do**
      $M_{s,k} = \min(C_{s,k}, M_{s,k+1} + 1)$
    end for
    for each $k > k_2$ **do**
      $M_{s,k} = \min(C_{s,k}, M_{s,k-1} + 1)$
    end for
  end if
end for

5.5.2 *Extensions to DTL Reconciliation*

The dated and undated formulations of the DTL reconciliation have been shown to be NP-hard for non-binary gene trees [38]. Kordi and Bansal [39] have also shown that the problem is Fixed-Parameter-Tractable (FPT) in the maximum degree k of the gene tree, and explored a $O\left(2^k k^k (|V(S)| + |V(G)|)^{o(1)}\right)$ algorithm testing all possible resolutions of the gene tree. A similar algorithm, implemented in NOTUNG [47], also tries all possible resolutions of each polytomy before computing the DTL distance for each resolution. Heuristics for the problem, including exploration of the tree space surrounding an initial resolution were also implemented in NOTUNG. One such possibility consists of selecting a best tree for the DL reconciliation, and then exploring alternative topologies at a given maximum NNI distance from the initial topology. Finally, Jacox et al. [37] have also proposed an algorithm improving the time complexity to $O\left((3^k - 2^{k+1})(V(|S|) + V(|G|))^{o(1)}\right)$ by using amalgamation principles (see Sect. 5.6). Although this algorithm improves the running time by an exponential factor, it runs in $O(2^k)$ space compared to the algorithm described in [39] requiring polynomial space complexity.

5.6 Inferring a Gene Tree from a Set of Trees

We now move to a slightly different gene tree correction strategy, which consists of taking advantage of a set of gene trees rather than a single input gene tree.

5.6.1 Amalgamation: Gene Tree Inference from a Set of Clades

As sequence information may contain limited signal, phylogenetic reconstruction often involves choosing among a set of equally likely trees. This idea has inspired the amalgamation procedure for reconstructing a tree from the clades, i.e., subtree leafsets, of a set of gene trees. This principle was first introduced by David and Alm [21] and a heuristic for correcting an initial gene tree based on this idea has been described. The amalgamation principle was extended by Szöllősi et al. [84] in a probabilistic method called ALE (for Amalgamated Likelihood Estimation) considering conditional clade probabilities (introduced in [35]) and a joint sequence-reconciliation likelihood score.

An alternative deterministic algorithm, called TERA (for Tree Estimation using Reconciliation) has been developed by Scornavacca et al. [74]. This algorithm "amalgamates" the most parsimonious DTL reconciled gene tree from an initial set of gene trees and achieves similar accuracy than ALE, while being much faster.

We start with some definitions, before presenting the outline of TERA.

Definition 5 Given a tree T and a node x of T, we call $L(T_x)$ the *clade* of T at x and denote by $\mathscr{C}(T)$ the set of all clades of T. If x is an internal node with children x_l and x_r, a tripartition at x is defined as $\pi_x = (\pi_x[1], \pi_x[2], \pi_x[3])$ with $\pi_x[1] = L(T_x)$, $\pi_x[2] = L(T_{x_l})$ and $\pi_x[3] = L(T_{x_r})$. Given a set $\mathscr{G}$ of k gene trees on the same gene family Γ, we denote by $\mathscr{C}(\mathscr{G})$ the set of all the clades of $\mathscr{G}$, and by $\Pi(\mathscr{G})$ the union of all tripartitions of $\mathscr{G}$. For a given clade $c \in \mathscr{C}(\mathscr{G})$, $\Pi(c)$ corresponds to the set of tripartitions π of $\Pi(\mathscr{G})$ such that $\pi[1] = c$.

Definition 6 (*Amalgamation*) An amalgamation of $\mathscr{G}$ is any gene tree G on Γ such that $\mathscr{C}(G) \subset \mathscr{C}(\mathscr{G})$.

MOST PARSIMONIOUS AMALGAMATION PROBLEM
Input: A set $\mathscr{G}$ of gene trees on the gene family Γ, and $\mathscr{C}(\mathscr{G})$ the set of all the clades of $\mathscr{G}$.
Output: An *amalgamation* of $\mathscr{G}$ minimizing the reconciliation cost with respect to S.

The TERA algorithm solves the amalgamation problem by computing the optimal reconciliation of each clade (i.e., polytomy with clade as leafset) with each node of S. For that purpose, the algorithm performs a joint traversal of the species tree S and the clades of $\mathscr{C}(\mathscr{G})$. In an initial step, it computes the reconciliation of each clade $c \in \mathscr{C}(\mathscr{G})$ with the leaves of S. Then S is traversed bottom-up, and for each node $s \in V(S)$, the reconciliation cost of each tripartition of c with s is computed. For each pair (c, s), the algorithm computes the cost of reconciling the clade c with s by testing all possible tripartitions π in $\Pi(c)$. As each non-trivial tripartition π can be seen as an internal node of an amalgamated tree with children $\pi[2]$ and $\pi[3]$, the cost of reconciling a tripartition π with s can be computed, using the recurrences of the DTL-reconciliation algorithm [24] (see Sect. 5.3), from the cost of reconciling

$\pi[2]$ and $\pi[3]$ respectively with nodes of $V(S_s)$. The output of TERA is the most parsimonious reconciliation at one of the root clades.

The TERA algorithm is part of a unifying software called ecceTERA [36] accounting for a variety of evolutionary events including duplications, losses, transfers, transfer-loss and transfers from/to an unsampled species (not represented by the set of genes). The software also handles fully or partially dated, as well as undated, species trees.

5.6.2 *Supertree: Inferring a Tree from a Set of Subtrees*

Homology-based search tools are usually used to seek all homologs of a given gene in a set of genomes. The resulting gene family may be very large, involving distant gene sequences that may be hard to align, leading to weakly supported trees. Alternatively, gene copies may be grouped into smaller sets of orthologs and inparalogs, using clustering algorithms such as OrthoMCL [50], InParanoid [10], Proteinortho [49] or many others.[1] Trees obtained for such partial gene families should then be combined into a single one using a *supertree method*.

Supertree methods have been mainly designed to reconstruct a species tree from gene trees obtained for various gene families (see for example [7, 11, 57, 64, 65, 72, 80, 83]). However, they can have applications for gene tree reconstruction as well. In this case, a gene tree is constructed from a set of subtrees for partial, possibly overlapping, subsets of the gene family. Ideally, the obtained tree should display each of the input trees, which is only possible if the partial trees are *consistent*, i.e., exhibit the same topology for each triplet of genomes (assuming genes are simply represented by the genome they belong to).

The simplest formulation of the supertree problem is therefore to state whether an input set of trees is consistent, and if so, find a *compatible* tree, called a *supertree*, displaying them all. This problem is NP-complete for unrooted trees [73, 79], but solvable in polynomial time for rooted trees [1, 19, 56, 75]. The BUILD algorithm [1] can be used to test, in polynomial time, whether a collection of rooted trees is consistent, and if so, construct a compatible, not necessarily fully resolved, supertree. This algorithm has been generalized to output all supertrees [19, 56, 75], which may be exponential in the number of genes.

Supertree methods can also be used to correct gene trees, by removing weakly supported upper branches and then constructing a supertree from the set of terminal subtrees. In contrast with the polytomy resolution approach, neither the input subtrees, nor the gene clusters of those subtrees are necessarily preserved. In other words, the exhibited monophyly of input gene clusters can be challenged. This is particularly relevant because it has been shown that genes under negative selection, while exhibiting the true topology, might be wrongly grouped into monophyletic groups (see for example [53, 77, 82, 88]). Using a supertree method might, there-

[1] See Quest for Orthologs links at http://questfororthologs.org/.

fore, be beneficial, as it preserves the topology of subtrees, while allowing to group genes from different subtrees.

In [41, 43], we introduced the $MinSGT$ problem defined as follows.

MINIMUM SUPERGENETREE ($MinSGT$) PROBLEM:
Input: A species set Σ and a species tree S for Σ; a gene family Γ of size n, a set $\Gamma_{i,1\leq i\leq k}$ of potentially intersecting subsets of Γ such that $\bigcup_{i=1}^{k} \Gamma_i = \Gamma$, and a consistent set $\mathscr{G} = \{G_1, G_2, \ldots, G_k\}$ of gene trees such that, for each $1 \leq i \leq k$, G_i is a tree for Γ_i.
Output: Among all trees G for Γ and compatible with $\mathscr{G}$, one of minimum reconciliation cost.

Under the D distance, we have shown that this problem is NP-hard to approximate within a $n^{1-\varepsilon}$ factor, for any $0 < \varepsilon < 1$, even for instances in which there is only one gene per species in the input trees, and even if each gene appears in at most one input tree. Although it has not been proven yet, $MinSGT$ is conjectured NP-hard for the DL reconciliation cost, as accounting for losses in addition to duplications is unlikely to make the problem simpler.

We developed a dynamic programming algorithm for $MinSGT$ with the DL reconciliation cost, which has a time complexity exponential in the number of input trees. The algorithm constructs the supertree G from the root to the leaves. At each step, i.e., for each internal node x being constructed in G, all possible bipartitions $(B_l(x), B_r(x))$ that could be induced by x are tried, and the iteration continues on each of $B_l(x)$ and $B_r(x)$. The goal is to find the bipartition of Γ, that leads to the minimum DL reconciliation cost at the root. At each step, corresponding to a node x, the reconciliation cost is computed from a local reconciliation cost at x, and from the best reconciliation cost of the two clusters of the considered bipartition. Because of the constraint of being compatible with the input gene trees only a subset of the bipartition set need to be tested at each step.

Property 1 Let $\mathscr{G} = \{G_1, \ldots, G_k\}$ be a set of gene trees. The root of a supertree G compatible with $\mathscr{G}$ subdivides $\bigcup_{i=1}^{k} L(G_i)$ into a *compatible bipartition* (B_l, B_r), i.e., a bipartition such that, for each i s.t. $1 \leq i \leq k$, either: (1) $L(G_i) \subseteq B_l$; or (2) $L(G_i) \subseteq B_r$; or (3) $L(G_{i_l}) \subseteq B_l$ and $L(G_{i_r}) \subseteq B_r$; or (4) $L(G_{i_l}) \subseteq B_r$ and $L(G_{i_r}) \subseteq B_l$.

Let $\mathscr{B}(G_1, \ldots, G_k)$ be the set of all possible combinations of choices resulting from Property 1 (see Fig. 5.7 for an example). Notice that not all such combinations are valid bipartitions. For instance in Fig. 5.7, the first bipartition (top-left) cannot be valid if G_1 and G_2 share a leaf with the same label, as a gene cannot be sent both left and right. These cases, however, can be detected easily by verifying the leafset of B_l and B_r.

Denote by $MinSGT(G_1, \ldots, G_k)$ the minimum DL reconciliation cost of a supertree compatible with $\mathscr{G} = \{G_1, \ldots, G_k\}$. The main recurrence formula of the dynamic programming algorithm is stated as follows.

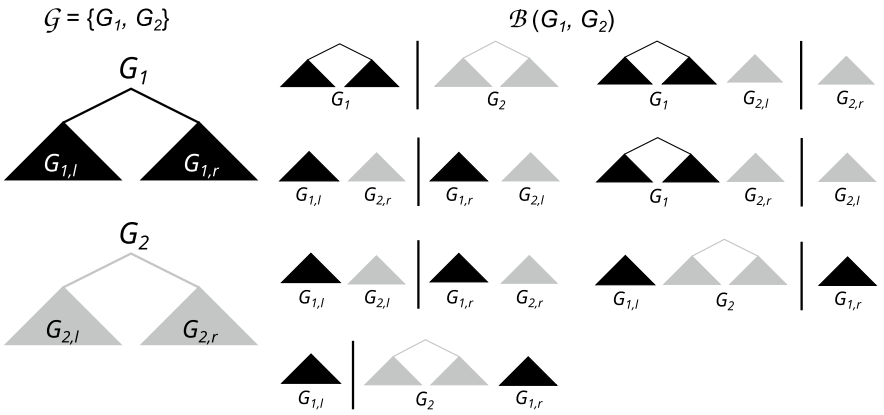


Fig. 5.7 An illustration of the seven valid bipartitions for two trees G_1 and G_2. Each bipartition is obtained by "sending" $L_1 \in \{L(G_1), L(G_{1,l}), L(G_{1,r}), \emptyset\}$ in the left part, and the complement $L(G_1) \setminus L_1$ in the right part. The same process is then applied to G_2. The set $\mathscr{B}(G_1, G_2)$ consists of the set of all possible combinations of choices, after eliminating symmetric cases and partitions with an empty side

Theorem 2 *Let $\mathscr{G} = \{G_1, \ldots, G_k\}$ be a set of gene trees.*

1. $MinSGT(G_1, \ldots, G_k) = 0$ *if* $| \bigcup_{i=1}^{k} L(G_i) | = 1$ *(Stop condition);*
2. *Otherwise,*
   $MinSGT(G_1, \ldots, G_k) =$

$$\min_{(B_l, B_r) \in \mathscr{B}(G_1, \ldots, G_k)} \left\{ \begin{array}{l} cost(B_l, B_r) + \\ MinSGT(G_{1|B_l}, \ldots, G_{k|B_l}) + \\ MinSGT(G_{1|B_r}, \ldots, G_{k|B_r}) \end{array} \right\}$$

Note that, given a bipartition $(B_l, B_r) \in \mathscr{B}(G_1, \ldots, G_k)$, for each i such that $1 \leq i \leq k$, $G_{i|B_l}$ and $G_{i|B_r}$ are equal either to $\emptyset$ or G_i or G_{i_l} or G_{i_r}. Thus, $G_{i|B_l}$ and $G_{i|B_r}$ are always either empty trees or complete subtrees of G_i. Furthermore, the existence of a compatible bipartition, at each step, follows from the fact that the input gene trees are assumed to be consistent.

In [41] we show how Theorem 2 can be modified to account for inconsistencies between gene trees, by adding a third equation: If $| \bigcup_{i=1}^{k} L(G_i) | > 1$ and $| \mathscr{B}(G_1, \ldots, G_k) | = 0$, $MinSGT(G_1, \ldots, G_k) = +\infty$. We also show that $|\mathscr{B}(G_1, \ldots, G_k)| \leq (\frac{4^k}{2}) - 1$, resulting in the time complexity of the overall algorithm being $O((n+1)^k \times 4^k \times k)$, where n is the maximum number of nodes in a tree G_i.

5.7 A Unifying View for the DL Model

The polytomy-based and supertree-based framework for gene tree correction have been developed separately, considering separate assumptions and constraints. In the absence of a unifying model, the conservative or permissive nature of each framework with respect to the other can only be tested empirically. A conceptual breakthrough is the discovery that, for the DL model, the two frameworks are in fact two special cases of a more general one: LabelGTC expressed in terms of a 0–1 edge-labeled gene tree [26], and TripletGTC expressed in terms of preserving triplets [26]. Here, we focus on LabelGTC.

Given an initial tree G for a gene family Γ, the correction problem can be defined as finding a "better tree" G' according to a reconciliation cost. The various versions of the problem differ on the flexibility we have in modifying G. Regarding which parts of G should be preserved, an intuitive way is to take advantage of the support on each branch (x, y) which reflects the confidence we have on $L(G_y)$ being a separate clade in the gene family. Hence, we could allow modifications only on weakly supported branches, i.e the ones with a support below a given threshold, while preserving all well-supported branches. Using a threshold, we therefore obtain a 0–1 edge-labeling of $E(G)$, where 0 indicates a low support and 1 a high support.

If G further contains a set of separated subtrees whose topologies are to be "trusted", they should also be preserved during correction. For example, ortholog groups that agree with the species tree and were separately obtained to build G may be trusted.

Accordingly, we describe below the most general gene tree correction problem (see Fig. 5.8 for an illustration), where a *covering set of subtrees* $\mathscr{C}_G$ for G is a set of separated subtrees of G, $\mathscr{C}_G = \{G_{x_1}, G_{x_2}, \ldots, G_{x_n}\}$ such that $\bigcup_{i=1}^{n} L(G_{x_i}) = L(G)$, and a 0–1 edge-labeling for G is a function f defined from the set of edges $E(G)$ to $\{0, 1\}$. In the following formulation, edge labels are ignored for the trees of $\mathscr{C}_G$. For an extension that considers edge-labeling inside the covering set, see [26].

LABEL RESPECTING GENE TREE CORRECTION (LABELGTC) PROBLEM:
Input: A species tree S, a gene tree G, a covering set of trees $\mathscr{C}_G$ for G and a 0–1 edge-labeling f for G.
Output: A supertree G' for $\mathscr{C}_G$ of minimum reconciliation cost such that: if $(x, y) \in E(G) \setminus E(\mathscr{C}_G)$ and $f(x, y) = 1$, then there is an edge (x', y') in $E(G')$ such that $L(G_y) = L(G'_{y'})$.

When no information on "trusted" separated subtrees is available, each tree of $\mathscr{C}_G$ is simply restricted to a leaf of G, and $\mathscr{C}_G$ thus refers to the leafset of G.

In the following, we reformulate the polytomy-related (Sect. 5.5) and supertree-related (Sect. 5.6.2) correction problems according to a 0–1 edge-labeled gene tree (see Fig. 5.9 for an illustration of the problems). We then show that they are special cases of the general LabelGTC problem.

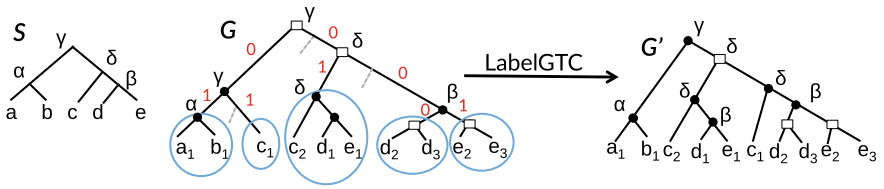


Fig. 5.8 (Figure modified from [26]; use permitted under the Creative Commons Attribution License CC-BY 3.0) **Left**. A species tree S for $\Sigma = \{a, b, c, d, e\}$, a reconciled 0–1 edge-labeled gene tree G for $\Gamma = \{a_1, b_1, c_1, c_2, d_1, d_2, d_3, e_1, e_2, e_3\}$ where each leaf x_i denotes a gene belonging to genome x, and a covering set $\mathscr{C}_G$ of subtrees for G indicated by blue circles around each subtree. Rectangular nodes represent duplications, black dots are speciations and dotted lines are losses. **Right**. A supertree for $\mathscr{C}_G$ of minimum reconciliation cost (cost of 3) respecting the edge-labeling of G

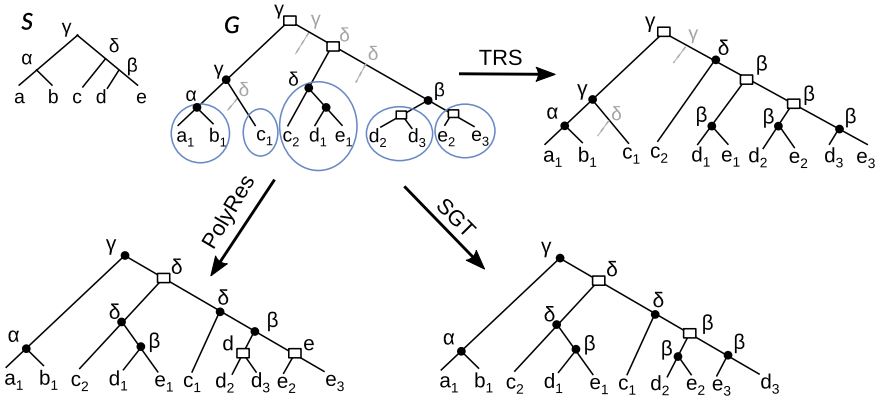


Fig. 5.9 (Figure from [26]; use permitted under the Creative Commons Attribution License CC-BY 3.0) A species tree S for $\Sigma = \{a, b, c, d, e\}$ and a gene tree G for $\Gamma = \{a_1, b_1, c_1, c_2, d_1, d_2, d_3, e_1, e_2, e_3\}$ with a covering set $\mathscr{C}_G$ of subtrees for G as in Fig. 5.8 (without the 0–1 labeling of edges). **Bottom left**. A polytomy resolution for $\mathscr{C}_G$ of minimum reconciliation cost (cost of 3). **Bottom right**. A supertree for $\mathscr{C}_G$ of minimum reconciliation cost (cost of 2). **Top right**. A triplet-respecting supertree for $\mathscr{C}_G$ of minimum reconciliation cost (cost of 5). Note that the solutions for the TRS, SGT and PolyRes problems may differ from the optimal supertree for the LabelGTC problem, because of the 0–1 edge-labeling. In this particular case, the optimal supertree for the SGT problem is identical to the one returned for LabelGTC in Fig. 5.8

In the general version of the polytomy resolution problem, all weakly supported internal branches of G are contracted, leading to a non-binary tree G^{nb}. The goal is then to find a binary refinement of G^{nb} minimizing the reconciliation cost.

MUTIPLE POLYTOMY RESOLUTION (M- POLYRES) PROBLEM:
Input: A species tree S and a 0–1 edge-labeled gene tree G and the tree G^{nb} obtained from G by contracting edges labeled 0;
Output: A binary refinement of G^{nb} minimizing the reconciliation cost.

In the simplest form of the polytomy resolution problem, we have a single polytomy which consists of a non-binary node at the root of G^{nb}. The subtrees rooted at the children of $r(G^{nb})$ are the "trusted" partial trees that should remain subtrees of the final tree (see the tree obtained from PolyRes in Fig. 5.9).

POLYTOMY RESOLUTION (POLYRES) PROBLEM:
Input: A species tree S, a gene tree G and a covering set of trees $\mathscr{C}_G$ for G.
Output: A supertree G' for $\mathscr{C}_G$ of minimum reconciliation cost such that for any tree $G_i \in \mathscr{C}_G$, $G'_{|L(G_i)} = G_i$.

Now recall the $MinSGT$ correction problem introduced in Sect. 5.6.2, but in the simplest case of separated gene trees.

SUPERGENETREE (SGT) PROBLEM:
Input: A species tree S, a gene tree G and a covering set of trees $\mathscr{C}_G$ for G.
Output: A supertree G' for $\mathscr{C}_G$ of minimum reconciliation cost.

To avoid having a supertree grouping genes that are far apart in the original tree, we also introduced, in [41], an alternative version of the problem restricting the output space to supertrees preserving the topology of any triplet of genes taken from three different input subtrees of $\mathscr{C}_G$. A formulation of the triplet-based constrained supertree problem follows.

TRIPLET- RESPECTING SUPERGENETREE (TRS) PROBLEM:
Input: A species tree S, a gene tree G and a covering set of trees $\mathscr{C}_G$ for G.
Output: A supertree G' for $\mathscr{C}_G$ of minimum reconciliation cost respecting the following property: for any triplet (a, b, c) where a, b and c are genes of Γ being leaves of three different trees of $\mathscr{C}_G$, $G'_{|\{a,b,c\}} = G_{|\{a,b,c\}}$.

The difference between the TRS and SGT problems is illustrated in Fig. 5.9. The solution of the SGT Problem shown in that figure is not a solution of the TRS problem as the triplet (a_1, c_1, c_2), where each gene belongs to a separate subtree of $\mathscr{C}_G$, has the topology $(a_1, (c_1, c_2))$ in the SGT tree while it has the topology $((a_1, c_1), c_2)$ in G.
A unifying view: Theorem 3 shows that the polytomy-related and supertree-related problems are in fact special cases of the general LabelGTC problem. We begin by introducing some notation.

Given a covering set of subtrees $\mathscr{C}_G$ for G, we say that an edge (x, y) of $E(G) \setminus E(\mathscr{C}_G)$ is a *terminal edge* if y is the root of a tree in $\mathscr{C}_G$. Any other edge in $E(G) \setminus E(\mathscr{C}_G)$ is called a *non-terminal edge* (see Fig. 5.10 for an illustration).

Theorem 3 *Let G be a 0–1 edge-labeled gene tree and $\mathscr{C}_G$ be a covering set for G. Then the LabelGTC Problem is reduced to the:*

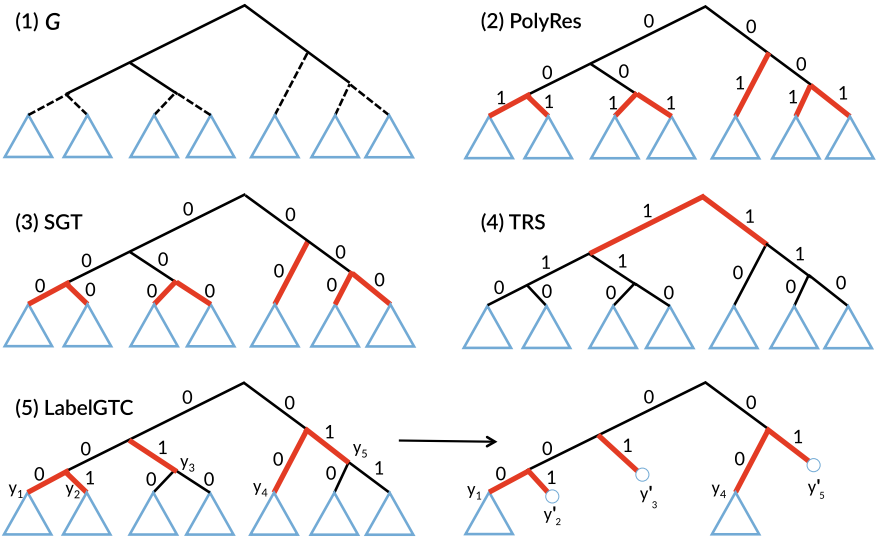


Fig. 5.10 (Figure from [26]; use permitted under the Creative Commons Attribution License CC-BY 3.0) (1) A gene tree G with a covering set $\mathscr{C}_G$ composed of 7 subtrees indicated as triangles. The set $E(G) \setminus E(\mathscr{C}_G)$ contains 7 terminal edges (dotted lines) and 5 non-terminal edges (solid lines). (2), (3) and (4) are three 0–1 edge-labeling corresponding respectively to the PolyRes, SGT and TRS problems. (5) is a general input of the LabelGTC problem

1. *M-PolyRes Problem if* $\mathscr{C}_G = L(G)$*; Otherwise:*
2. *PolyRes Problem if all non-terminal edges are labeled* 0*, and all terminal edges are labeled* 1*;*
3. *SGT Problem if all non-terminal and terminal edges are labeled* 0*;*
4. *TRS Problem if all non-terminal edges are labeled* 1*, and all terminal edges are labeled* 0*.*

Finally, we have developed an algorithm, called LabelGTC, handling the general version of the problem, not represented by any of the special cases reflected in Theorem 3. For any edge (x, y) in $E(G) \setminus E(\mathscr{C}_G)$ labeled 1, there should exist a node y' in the final corrected tree G' such that $L(y') = L(y)$. So the subtree $G'_{y'}$ of G' for the subset $L(G_y)$ can first be constructed independently of the remaining nodes of G', and then grafted at the appropriate location in a way minimizing the reconciliation cost. The LabelGTC algorithm proceeds iteratively, in a bottom-up order, on subtrees G_y with parental edge (x, y) fitting the above criterion, and is recursively called to reconstruct $G'_{y'}$. Each solution $G'_{y'}$ is implicitly treated as a leaf in subsequent calls to avoid modifying its content.

In [26], we showed that the time complexity of the algorithm is related to the time complexity of $MinSGT$, which makes it exponential in the number of terminal subtrees. More precisely, the algorithm runs in time $O(4^k \cdot (n+1)^k \cdot k)$, where $n = |\Gamma|$ and $k = |\mathscr{C}_G|$.

5.8 Discussion

Efficient pipelines for gene tree inference should typically include accurate gene sequence alignment tools and use inference methods combining information from both micro-evolutionary (sequence level) and macro-evolutionary (genome level) information. In the recent years, new algorithms improving the accuracy of sequence alignment and gene tree inference have been described.

In particular, probabilistic gene tree construction methods relying on complex evolutionary models that account for both sequence and species tree data have been developed [2, 4, 66, 67, 76, 84, 86]. These methods unfortunately present some drawbacks inherent to probabilistic methods, namely the huge computational time associated with the numerical integration of the likelihood, and the prior analyses required to satisfy the input requirements (e.g., dating the species tree).

In practice, alternative parsimony-based approaches, a posteriori correcting gene trees inferred from sequence-only data with species tree information, are used instead. Such algorithms, although limited in some aspects when compared to probabilistic ones, have consistently produced trees with high accuracy, while being much faster. This time efficiency allows applying the correction method to a wide set of data. For example, in [60], we used ProfileNJ to correct the PhyML trees built on the whole Ensembl Compara gene families (20,519 families in total). According to several criteria, including likelihood, reconciliation score, and ancestral genome content, these corrected trees constitute an arguably better dataset than the one stored in the Ensembl database.

Another advantage of parsimony methods is that they can be easily extended to consider other sources of information. For example, gene order may provide information on gene orthology and paralogy. In fact, two synteny blocks, i.e., two chromosomal segments (in the same genome or in two different genomes) containing genes form the same gene families are likely to have a common ancestor. Depending on whether they diverged from a speciation or a duplication event, gene pairs in the two synteny blocks will either be all orthologs or all paralogs. This information has been considered for correcting a gene tree in [44, 46].

Alternatively, functional similarity between genes is also, usually, a good indicator for orthology [3, 29]. We are presently exploring ways to efficiently use scores based on Gene Ontology annotations to establish terminal preserved trees in LabelGTC.

The main difficulty remains how to integrate all the developed algorithmic tools, each handling a given type of information on genes and trees, into a single robust framework for gene tree reconstruction. In addition, rather than applying corrections in an incremental manner, with the risk of obtaining very different trees depending on the order of execution, the challenge is to consider the variety of sequence, functional, order and evolutionary information all together in a single algorithm. The LabelGTC algorithm, considering polytomy resolution and supertree reconstruction in a unifying framework is an effort in this direction. However, fitness to sequence information may still be lost after correction, unless we constraint the output to be statistically equivalent to the best maximum likelihood tree.

Therefore, approaches suitable for the resolution of Multi-Objective Optimization Problems (MOOP) have to be explored. In this context, we have developed GATC [59], a genetic algorithm minimizing a measure combining both tree likelihood (according to sequence evolution) and a reconciliation score that accounts for HGT. An advantage of this approach is its ability to improve search efficiency by exploring a population of trees at each step. Although much slower than deterministic methods for correction, GATC outperforms all these methods in terms of accuracy.

From an algorithmic point of view, a lot remains to be done. Unifying the diversity of evolutionary models and datasets is still far from being reached and raises the interesting problem of how we can simultaneously account, in the same evolutionary model, for sequence evolution as well as duplications, losses, HGTs, recombination, hybridization, and ILS. Interestingly, some of the methods developed for these events, often taken separately, might be more related than expected. For example, as we show in Fig. 5.11, the parsimony method described in [95] for reconciling a binary gene tree with a binary species tree, while accounting for duplications, losses and ILS (see Sect. 5.3.3), may be compared to the strategy using $MinSGT$ that we explored in Sect. 5.6.2. The latter consists of removing upper branches of the gene tree, keeping *speciation trees*, i.e., subtrees with only speciation nodes, and then using a supertree method to reconstruct a most parsimonious supertree containing them all. To which extent the two methods are comparable from a theoretical point of view? How can the supertree method be applied to account for ILS? Can we take advantage of the similarity between the two problems to design more efficient algorithms than the exponential dynamic programming algorithm developed for $MinSGT$? These are few questions that will be considered in future developments.

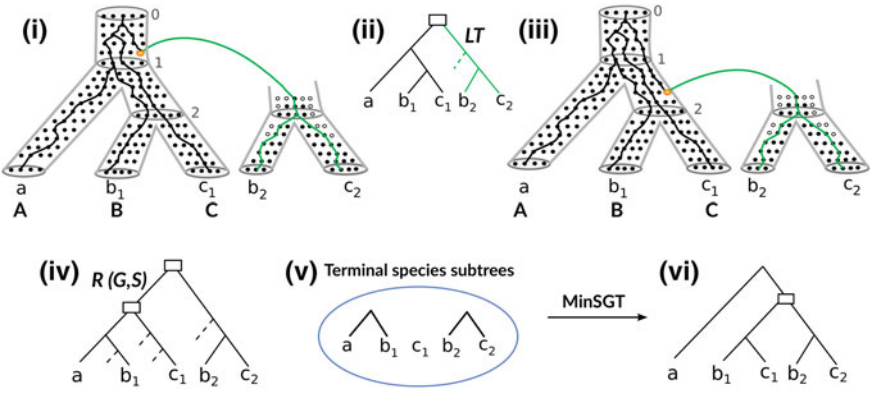


Fig. 5.11 (i) The same evolution with ILS represented in Fig. 5.4; (ii) The locus tree inferred by DLCpar [95], inducing one duplication at the root and one loss. (iii) An alternative explanation of the gene tree with ILS, the duplication occurring lower in the species tree, and no loss. This most parsimonious DL history with ILS is not inferred by DLCpar, however the hill-climbing heuristic described in the same paper did find it.; (iv) The gene tree/species tree reconciliation leading to two duplications and four losses; (v) The set of largest speciation subtrees in the gene tree; (v) The tree obtained by $MinSGT$ reflecting the most parsimonious history represented in (iii)

As we have no direct access to the past, it is difficult to objectively evaluate the accuracy of gene tree reconstruction methods. The most intuitive way is to compare inferences on simulated gene families, where the "true" evolutionary histories according to some given model of evolution with controlled parameters, are known. Aside from tree topology comparison using metrics such as the Robinson-Foulds distance [63, 68], we can also assess how close the evolutionary scenarios inferred are to the true ones. In [60], we have additionally considered metrics based on ancestral gene content inferred from reconciliation, and ancestral gene adjacencies [9]. The latter is particularly useful as measure for gene tree accuracy for linear genomes, given that at most two adjacencies per gene copy should be expected.

Since good results on simulated datasets do not guarantee the same on real ones, as they may not conform to the evolutionary model used for simulations, well-studied gene families for which good trees are available have been used to construct reference datasets. In this regard, several ongoing works, such as the SwissTree [12] project, are undertaking great efforts to provide manually curated "gold standard" gene trees. However, the number of available "gold standard" remains extremely low (19 in SwissTree) and does not allow extensive covering of the many and intricate pathways of gene evolution. Therefore, developing new sophisticated frameworks, accounting for various gene characteristics for producing good benchmarks, is still needed.

Acknowledgements The authors acknowledge the support of the Fonds de Recherche du Québec Nature et Technologie (FRQNT) and of the Natural Sciences and Engineering Research Council (NSERC) (Discovery Grant RGPIN-249834).

References

1. Aho, A., Yehoshua, S., Szymanski, T., Ullman, J.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. **10**(3), 405–421 (1981)
2. Akerborg, O., Sennblad, B., Arvestad, L., Lagergren, J.: Simultaneous bayesian gene tree reconstruction and reconciliation analysis. Proc. Nal. Acad. Sci. USA **106**(14), 5714–5719 (2009)
3. Altenhoff, A.M., Studer, R.A., Robinson-Rechavi, M., Dessimoz, C.: Resolving the ortholog conjecture: orthologs tend to be weakly, but significantly, more similar in function than paralogs. PLoS Comput. Biol. **8**(5), e1002,514 (2012)
4. Arvestad, L., Berglund, A., Lagergren, J., Sennblad, B.: Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In: RECOMB, pp. 326–335 (2004)
5. Bader, D., Moret, B., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J. Comput. Biol. **8**(5), 483–491 (2001)
6. Bansal, M., Alm, E., Kellis, M.: Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics **28**(12), i283–i291 (2012). https://doi.org/10.1093/bioinformatics/bts225
7. Bansal, M., Burleigh, J., Eulenstein, O., Fernández-Baca, D.: Robinson-foulds supertrees. Alg. Mol. Biol. **5**(18) (2010)

8. Bansal, M., Wu, Y., Alm, E., Kellis., M.: Improved gene tree error-correction in the presence of horizontal gene transfer. Bioinformatics **31**(8), 1211–1218 (2015). https://doi.org/10.1093/bioinformatics/btu806
9. Bérard, S., Gallien, C., Boussau, B., Szollosi, G., Daubin, V., Tannier, E.: Evolution of gene neighborhoods within reconciled phylogenies. Bioinformatics **28**(18), i382–i388 (2012)
10. Berglund, A., Sjolund, E., Ostlund, G., Sonnhammer, E.: InParanoid 6: eukaryotic ortholog clusters with inparalogs. Nucl. Acid Res. **36** (2008)
11. Bininda-Emonds, O. (ed.): Phylogenetic Supertrees combining information to reveal The Tree of Life. In: Computational Biology. Kluwer Academic, Dordrecht, The Netherlands (2004)
12. Boeckmann, B., Robinson-Rechavi, M., Xenarios, I., Dessimoz, C.: Conceptual framework and pilot study to benchmark phylogenomic databases based on reference gene trees. Brief. Bioinform. **12**(5), 423–435 (2011)
13. Bork, D., Cheng, R., Wang, J., Sung, J., Libeskind-Hadas, R.: On the computational complexity of the maximum parsimony reconciliation problem in the duplication-loss-coalescence model. Algorithms Mol. Biol. **12**(1), 6 (2017)
14. Boussau, B., Szöllősi, G., Duret, L., Gouy, M., Tannier, E., Daubin, V.: Genome-scale coestimation of species and gene trees. Genome Res. **23**, 323–330 (2013)
15. Chan, Y., Ranwez, V., Scornavacca, C.: Exploring the space of gene/species reconciliations with transfers. J. Math. Biol. **71**(5), 1179–1209 (2015)
16. Chan, Y., Ranwez, V., Scornavacca, C.: Inferring incomplete lineage sorting, duplications, transfers and losses with reconciliations. J. Theoret. Biol. **432**, 1–13 (2017)
17. Chang, W., Eulenstein, O.: Reconciling gene trees with apparent polytomies. In: Chen, D., Lee, D.T. (eds.) Proceedings of the 12th Conference on Computing and Combinatorics (COCOON). Lecture Notes in Computer Science, vol. 4112, pp. 235–244 (2006)
18. Chen, K., Durand, D., Farach-Colton, M.: Notung: dating gene duplications using gene family trees. J. Comput. Biol. **7**, 429–447 (2000)
19. Constantinescu, M., Sankoff, D.: An efficient algorithm for supertrees. J. Classif. **12**, 101–112 (1995)
20. Darby, C.A., Stolzer, M., Ropp, P.J., Barker, D., Durand, D.: Xenolog classification. Bioinformatics **33**(5), 640–649 (2016)
21. David, L., Alm, E.: Rapid evolutionary innovation during an Archaean genetic expansion. Nature **469** (2011)
22. Doyon, J.P., Chauve, C., Hamel, S.: Space of gene/species trees reconciliations and parsimonious models. J. Comput. Biol **16**(10), 1399–1418 (2009)
23. Doyon, J., Ranwez, V., Daubin, V., Berry, V.: Models, algorithms and programs for phylogeny reconciliation. Brief. Bioinform. **12**(5), 392–400 (2011)
24. Doyon, J.P., Scornavacca, C., Gorbunov, K.Y., Szöllősi, G.J., Ranwez, V., Berry, V.: An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In: Tannier, E. (ed.) RECOMB International Workshop on Comparative Genomics, RECOMB-CG, pp. 93–108. Springer (2010)
25. Durand, D., Halldórsson, B.V., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. J. Comput. Biol. **13**(2), 320–335 (2006)
26. El-Mabrouk, N., Ouangraoua, A.: A general framework for gene tree correction based on duplication-loss reconciliation. In: LIPIcs, Workshop on Algorithms in Bioinformatics (WABI), vol. 88, pp. 8:1–8:14 (2017)
27. Fitch, W.: Homology—a personal view on some of the problems. Trends Genet. **16**(5), 227–231 (2000)
28. Flicek, P., et al.: Ensembl 2012. Nucleic Acids Res. **40**, D84–D90 (2012)
29. Gabaldón, T., Koonin, E.V.: Functional and evolutionary implications of gene orthology. Nat. Rev. Genet. **14**(5), 360 (2013)
30. Goodman, M., Czelusniak, J., Moore, G., Romero-Herrera, A., Matsuda, G.: Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. Syst. Zool. **28**, 132–163 (1979)

31. Górecki, P., Eulenstein, O.: Algorithms: simultaneous error-correction and rooting for gene tree reconciliation and the gene duplication problem. BMC Bioinform. **13**(Supp 10), S14 (2011)
32. Gorecki, P., Eulenstein, O., Tiuryn, J.: Unrooted tree reconciliation: a unified approach. IEEE/ACM Trans. Comput. Biol. Bioinform. **10**(2), 522–536 (2013)
33. Guindon, S., Gascuel, O.: A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. **52**, 696–704 (2003)
34. Hallett, M., Lagergren, J.: Efficient algorithms for lateral gene transfer problems. In: Proceedings of the Fifth Annual International Conference on Computational Biology, RECOMB-CG, pp. 149–156 (2001)
35. Höhna, S., Drummond, A.J.: Guided tree topology proposals for bayesian phylogenetic inference. Syst. Biol. **61**(1), 1–11 (2011)
36. Jacox, E., Chauve, C., Szöllősi, G.J., Ponty, Y., Scornavacca, C.: ecceTERA: comprehensive gene tree-species tree reconciliation using parsimony. Bioinformatics **32**(13), 2056–2058 (2016). https://doi.org/10.1093/bioinformatics/btw105
37. Jacox, E., Weller, M., Tannier, E., Scornavacca, C.: Resolution and reconciliation of non-binary gene trees with transfers, duplications and losses. Bioinformatics **33**(7), 980–987 (2017)
38. Kordi, M., Bansal, M.: On the complexity of duplication-transfer-loss reconciliation with non-binary gene trees. IEEE/ACM Trans. Comput. Biol. Bioinform. (2016)
39. Kordi, M., Bansal, M.: Exact algorithms for duplication-transfer-loss reconciliation with non-binary gene trees. IEEE/ACM Trans. Comput. Biol. Bioinform. (2017)
40. Lafond, M., Chauve, C., Dondi, R., Manuel, El-Mabrouk, N.: Polytomy refinement for the correction of dubious duplications in gene trees. Bioinformatics **30**(17), i519–i526 (2014)
41. Lafond, M., Chauve, C., El-Mabrouk, N., Ouangraoua, A.: Gene tree construction and correction using supertree and reconciliation. IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB) **PP**(99), 12 pp. (2018)
42. Lafond, M., Noutahi, E., El-Mabrouk, N.: Efficient non-binary gene tree resolution with weighted reconciliation cos. In: 27th Annual Symposium on Combinatorial Pattern Matching (CPM) (2016)
43. Lafond, M., Ouangraoua, A., El-Mabrouk, N.: Reconstructing a supergenetree minimizing reconciliation. BMC Genomics **16**, S4 (2015). Special issue of RECOMB-CG 2015
44. Lafond, M., Semeria, M., Swenson, K., Tannier, E., El-Mabrouk, N.: Gene tree correction guided by orthology. BMC Bioinform. **14**(supp 15)(S5) (2013)
45. Lafond, M., Swenson, K., El-Mabrouk, N.: An optimal reconciliation algorithm for gene trees with polytomies. In: WABI. LNCS, vol. 7534, pp. 106–122 (2012)
46. Lafond, M., Swenson, K., El-Mabrouk, N.: Error detection and correction of gene trees. In: Models and Algorithms for Genome Evolution. Springer (2013)
47. Lai, H., Stolzer, M., Durand, D.: Fast heuristics for resolving weakly supported branches using duplication, transfers, and losses. In: RECOMB-CG, 22 pp. (2017)
48. Lartillot, N., Philippe, H.: A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. Mol. Biol. Evol. **21**(6), 1095–1109 (2004). http://dx.doi.org/10.1093/molbev/msh112
49. Lechner, M., Findeiß, S., Steiner, L., Manja, M., Stadler, P., Prohaska, S.: Proteinortho: Detection of co-orthologs in large-scale analysis. BMC Bioinform. **12**(1), 1 (2011)
50. Li, L., Stoeckert, C.J., Roos, D.: OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res. **13**, 2178–2189 (2003)
51. Libeskind-Hadas, R., Charleston, M.: On the computational complexity of the reticulate cophylogeny reconstruction problem. J. Comput. Biol. **16** (2009)
52. Maddison, W.P.: Gene trees in species trees. Syst. Biol. **46**(3), 523–536 (1997)
53. Massey, S., Churbanov, A., Rastogi, S., Liberles, D.: Characterizing positive and negative selection and their phylogenetic effects. Gene **418**, 22–26 (2008)
54. Moret, B., Warnow, T.: Molecular evolution: producing the biochemical data. In: Zimmer, E., Roalson, E. (eds.) Methods in Enzymology, Part B, vol. 395, pp. 673–700. Elsevier (2005)

55. Moret, B.M., Bader, D.A., Wyman, S., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Biocomputing 2001, pp. 583–594. World Scientific (2000)
56. Ng, M., Wormald, N.: Reconstruction of rooted trees from subtrees. Discrete Appl. Math. **69**, 19–31 (1996)
57. Nguyen, N., Mirarab, S., Warnow, T.: MRL and SuperFine+MRL: new supertree methods. Algorithms Mol. Biol. **7**(3) (2012)
58. Nguyen, T.H., Ranwez, V., Pointet, S., Chifolleau, A.M.A., Doyon, J.P., Berry, V.: Reconciliation and local gene tree rearrangement can be of mutual profit. Algorithms Mol. Biol. **8**(1), 12 (2013). http://dx.doi.org/10.1186/1748-7188-8-12
59. Noutahi, E., El-Mabrouk, N.: GATC: a genetic algorithm for gene tree construction under the duplication-transfer-loss model of evolution. BMC Genomics **19**(2), 102 (2018)
60. Noutahi, E., Semeria, M., Lafond, M., Seguin, J., Gueguen, L., El-Mabrouk, N., Tannier, E.: Efficient gene tree correction guided by genome evolution. PLoS One **11**(8) (2016)
61. Ovadia, Y., Fielder, D., Conow, C., Libeskind-Hadas, R.: The cophylogeny reconstruction problem is NP-complete. J. Comput. Biol. **18**(1), 59–65 (2011). https://doi.org/10.1089/cmb.2009.0240
62. Page, R.D., Cotton, J.A.: Genetree: a tool for exploring gene family evolution. In: Comparative Genomics, pp. 525–536. Springer (2000)
63. Pattengale, N., Gottlieb, E., Moret, B.: Efficiently computing the Robinson-Foulds metric. J. Comput. Biol. **14**(6), 724–735 (2007)
64. Ranwez, V., Berry, V., Criscuolo, A., Fabre, P., Guillemot, S., Scornavacca, C., Douzery, E.: PhySIC: a veto supertree method with desirable properties. Syst. Biol. **56**(5), 798–817 (2007)
65. Ranwez, V., Criscuolo, A., Douzery, E.: SuperTriplets: a triplet-based supertree approach to phylogenomics. Bioinformatics **26**(12), i115–i123 (2010)
66. Rasmussen, M., Kellis, M.: A Bayesian approach for fast and accurate gene tree reconstruction. Mol. Biol. Evol. **28**(1), 273–290 (2010)
67. Rasmussen, M.D., Kellis, M.: Unified modeling of gene duplication, loss, and coalescence using a locus tree. Genome Res. **22**(4), 755–765 (2012)
68. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Math. Biosci. **53**, 131–147 (1981)
69. Rodrìguez-Ezpeleta, N., Brinkmann, H., Roure, B., Lartillot, N., Lang, B.F., Philippe, H.: Detecting and overcoming systematic errors in genome-scale phylogenies. Syst. Biol. **56**(3), 389–399 (2007). http://dx.doi.org/10.1080/10635150701397643
70. Rogers, J., Fishberg, A., Youngs, N., Wu, Y.C.: Reconciliation feasibility in the presence of gene duplication, loss, and coalescence with multiple individuals per species. BMC Bioinform. **18**(1), 292 (2017)
71. Ronquist, F., Huelsenbeck, J.: MrBayes3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**, 1572–1574 (2003)
72. Roshan, U., Moret, B., Warnow, T., Williams, T.: Performance of supertree methods on various dataset decompositions. In: Bininda-Edmonds, O. (ed.) Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, pp. 301–328. Springer (2004)
73. Scornavacca, C., van Iersel, L., Kelk, S., Bryant, D.: The agreement problem for unrooted phylogenetic trees is FPT. J. Graph Algorithms Appl. **18**(3), 385–392 (2014)
74. Scornavacca, C., Jacox, E., Szollosi, G.: Joint amalgamation of most parsimonious reconciled gene trees. Bioinformatics **31**(6), 841–848 (2015)
75. Semple, C.: Reconstructing minimal rooted trees. Discrete Appl. Math. **127**(3) (2003)
76. Sjöstrand, J., Tofigh, A., Daubin, V., Arvestad, L., Sennblad, B., Lagergren, J.: A Bayesian method for analyzing lateral gene transfer. Sys. Biol. **63**(3), 409–420 (2014)
77. Skovgaard, M., Kodra, J., Gram, D., Knudsen, S., Madsen, D., Liberles, D.: Using evolutionary information and ancestral sequences to understand the sequence-function relationship in GLP-1 agonists. J. Mol. Biol. **363**, 977–988 (2006)
78. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analysis with thousands of taxa and mixed models. Bioinformatics **22**, 2688–2690 (2006)

79. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. J. Classif. **9**, 91–116 (1992)
80. Steel, M., Rodrigo, A.: Maximum likelihood supertrees. Syst. Biol. **57**(2), 243–250 (2008)
81. Stolzer, M., Lai, H., Xu, M., Sathaye, D., Vernot, B., Durand, D.: Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. Bioinformatics **28**(18), i409–i415 (2012)
82. Swenson, K.M., El-Mabrouk, N.: Gene trees and species trees: irreconcilable differences. BMC Bioinform. **13**(Suppl 19), S15 (2012)
83. Swenson, M., Suri, R., Linder, C., Warnow, T.: SuperFine: fast and accurate supertree estimation. Sys. Biol. **61**(2), 214–227 (2012). Special issue RECOMB-CG 2012
84. Szöllősi, G., Rosikiewicz, W., Boussau, B., Tannier, E., Daubin, V.: Efficient exploration of the space of reconciled gene trees. Syst. Biol. **62**(6), 901–912 (2013). http://dx.doi.org/10.1093/sysbio/syt054
85. Szöllősi, G., E., Tannier, Daubin, V., Boussau, B.: The inference of gene trees with species trees. Syst. Biol. **64**(1), e42–e62 (2014)
86. Szöllősi, G.J., Tannier, E., Lartillot, N., Daubin, V.: Lateral gene transfer from the dead. Syst. Biol. **62**(3), 386–397 (2013)
87. Tatusov, R., Galperin, M., Natale, D., Koonin, E.: The COG database: a tool for genome-scale analysis of protein functions. Nucleic Acids Res. **28**, 33–36 (2000)
88. Taylor, S., de la Cruz, K., Porter, M., Whiting, M.: Characterization of the long-wavelength opsin from Mecoptera and Siphonaptera: does a flea see? Mol. Biol. Evol. **22**, 1165–1174 (2005)
89. Thomas, P.: GIGA: a simple, efficient algorithm for gene tree inference in the genomic age. BMC Bioinform. **11**, 312 (2010)
90. Tofigh, A.: Using trees to capture reticulate evolution: lateral gene transfers and cancer progression. Ph.D. thesis, KTH Royal Institute of Technology, Sweden (2009)
91. Tofigh, A., Hallett, M., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(2), 517–535 (2011). https://doi.org/10.1109/TCBB.2010.14
92. Vernot, B., Stolzer, M., Goldman, A., Durand, D.: Reconciliation with non-binary species trees. J. Comput. Biol. **15**, 981–1006 (2009)
93. Wu, T., Zhang, L.: Structural properties of the reconciliation space and their applications in enumerating nearly-optimal reconciliations between a gene tree and a species tree. BMC Bioinform. **12**, S7 (2011)
94. Wu, Y., Rasmussen, M., Bansal, M., Kellis, M.: TreeFix: statistically informed gene tree error correction using species trees. Syst. Biol. **62**(1), 110–120 (2013)
95. Wu, Y., Rasmussen, M., Bansal, M., Kellis, M.: Most parsimonious reconciliation in the presence of gene duplication, loss, and deep coalescence using labeled coalescent trees. Genome Res. **24**, 475–486 (2014)
96. Zhang, L.: On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. J. Comput. Biol. **4**, 177–188 (1997)
97. Zheng, Y., Wu, T., Zhang, L.: Reconciliation of gene and species trees with polytomies (2012). arXiv:1201.3995
98. Zheng, Y., Zhang, L.: Reconciliation with non-binary gene trees revisited. In: Proceedings of RECOMB. Lecture Notes in Computer Science, vol. 8394, pp. 418–432 (2014)
99. Zmasek, C.M., Eddy, S.R.: A simple algorithm to infer gene duplication and speciiation events on a gene tree. Bioinformatics **17**, 821–828 (2001)

Chapter 6
Divide-and-Conquer Tree Estimation: Opportunities and Challenges

Tandy Warnow

Abstract Large-scale phylogeny estimation is challenging for many reasons, including heterogeneity across the Tree of Life and the difficulty in finding good solutions to NP-hard optimization problems. One of the promising ways for enabling large-scale phylogeny estimation is through divide-and-conquer: a dataset is divided into overlapping subsets, trees are estimated on the subsets, and then the subset trees are merged together into a tree on the full set of taxa. This last step is achieved through the use of a supertree method, which is popular in systematics for use in combining species trees from the scientific literature. Because most supertree methods are heuristics for NP-hard optimization problems, the use of supertree estimation on large datasets is challenging, both in terms of scalability and accuracy. In this chapter, we describe the current state of the art in supertree construction and the use of supertree methods in divide-and-conquer strategies, and we identify directions where future research could lead to improved supertree methods. Finally, we present a new type of divide-and-conquer strategy that bypasses the need for supertree estimation, in which the division into subsets produces disjoint subsets. Overall, this chapter aims to present directions for research that will potentially lead to new methods to scale phylogeny estimation methods to large datasets.

Keywords Supertrees · Phylogenetics · Species trees · Divide-and-conquer · Incomplete lineage sorting · Tree of Life

6.1 Introduction

The estimation of phylogenetic trees, whether gene trees (that is, the phylogeny relating the sequences found at a specific locus in different species) or species trees, is a precursor to many downstream analyses, including protein structure and function prediction, the detection of positive selection, coevolution, human population genetics, etc. (see [60] for just some of these applications). Indeed, as Dobzhanksy said, "Nothing in biology makes sense except in the light of evolution" [44].

T. Warnow (✉)
University of Illinois at Urbana-Champaign, Champaign, USA
e-mail: warnow@illinois.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_6

Standard approaches for estimating phylogenies from a given set of molecular sequences typically involve two steps: first, a multiple sequence alignment is computed, and then a tree is computed on the multiple sequence alignment. Both steps are computationally intensive on large datasets, but for somewhat different reasons. Multiple sequence alignment is approached using many different techniques, and generally runs in polynomial time, but when the input set of sequences is both large and highly heterogeneous (in the sense that many insertions and deletions have occurred in the history relating the sequences), then the alignments can become extremely large—much larger than the input set—presenting challenges if the available memory is insufficient for the alignment itself. Highly accurate large-scale multiple sequence alignment can be difficult to obtain with standard methods, but new approaches (largely based on divide-and-conquer) have been developed that enable multiple sequence alignment methods to scale with high accuracy to ultra-large datasets [80, 98, 100].

Tree estimation presents a different challenge: while some polynomial time methods have been developed for tree estimation (e.g., neighbor joining and its variants [53, 118, 152], FastME [78], and other distance-based methods), simulation studies suggest that computationally intensive methods, such as heuristics for maximum likelihood (ML) and Bayesian MCMC, produce more accurate phylogenies [146]. Maximum likelihood codes are more frequently used than Bayesian MCMC methods, for the simple reason that they can be used on large datasets. Advances in techniques for maximum likelihood (discussed in chapters by Stamatakis, Guindon & Gascuel, and Bader & Madduri in this volume) have led to codes with substantially improved scalability, including RAxML [129], PhyML [58], IQTree [97], and FastTree-2 [107]. Of these, FastTree-2 provides the best scalability to large datasets, but achieves this by limiting the number of operations; hence, the scalability comes at the cost of accuracy in the maximum likelihood criterion score. As shown in [79], although RAxML obtains trees with better ML scores, the trees computed using FastTree-2 and RAxML can have similar topological accuracy.

Despite these advances, the estimation of single-gene phylogenies using the best ML methods is typically limited to at most a few thousand sequences (and much fewer for multigene phylogenies), and even these can be very computationally intensive. For example, our own analyses of single-gene datasets with many thousands of sequences can take weeks for RAxML to converge to good local optima. Thus, truly large-scale maximum likelihood phylogeny estimation is still not feasible in practice.

Another challenge in large-scale gene tree estimation is "heterotachy" [83], which is where the statistical models that best characterize the evolutionary processes operating on the sequences (such as the rates of change between different nucleotides) vary across the tree, thus creating opportunities for model misspecification when standard statistical models are used in maximum likelihood or Bayesian phylogeny estimation.

Species tree estimation presents additional challenges as multiple loci (i.e., genes) are needed since gene trees can differ from species trees due to many different causes (e.g., incomplete lineage sorting), and the most accurate methods for species tree estimation are also computationally challenging [90]. Hence, phylogenetic tree esti-

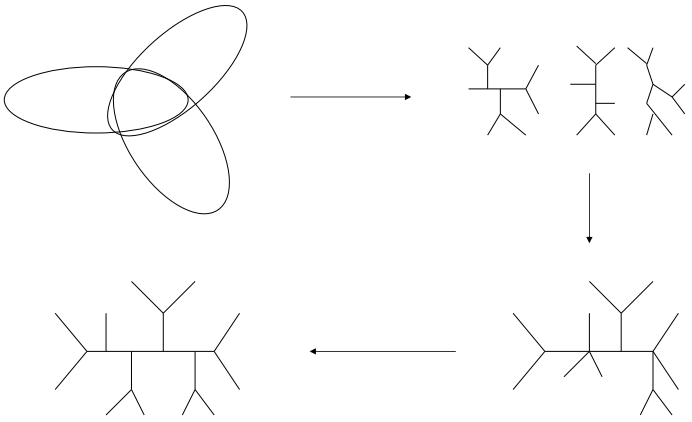


Fig. 6.1 An example of a divide-and-conquer tree estimation. In the first step, the set of species is divided into three subsets that overlap. Then, trees are computed on the subsets and the subset trees are combined into a tree on the full set of taxa using a supertree method. If the supertree method produces an unresolved tree, a final refinement step can be applied

mation, whether based on single genes or based on multiple loci, presents computational and statistical challenges, and large-scale phylogeny estimation is particularly difficult.

Divide-and-conquer strategies, in which a dataset is divided into overlapping subsets, trees are constructed on the subsets, and then the subset trees are merged together (using a supertree method), can be used to address these challenges. Figure 6.1 presents a description of one such divide-and-conquer strategy.

The disk-covering methods (DCMs) [62, 63, 93, 148] are methods that follow this paradigm, and which are designed for use in conjunction with a selected "base" phylogenetic method. For example, the first two DCMs [62, 148] were designed for use with distance-based phylogeny estimation with the goal of creating methods that were provably "fast-converging" (i.e., methods that are guaranteed to return the true tree with high probability converging to 1 from only polynomial length sequences) under standard Markov models of sequence evolution. As shown in Fig. 6.2, DCM1-NJ (i.e., the use of a DCM with the neighbor joining method of Saitou and Nei [118]) produced more accurate trees than neighbor joining alone on simulated data. Furthermore, as proven in [148], DCM-NJ is fast-converging whereas neighbor joining is not fast-converging [73]. See the chapter by Sébastien Roch, this volume, for more on fast-converging methods and sequence length requirements for phylogeny estimation methods in general.

Other DCMs have been developed for empirical performance (e.g., reductions in running time) rather than theoretical guarantees [64, 93, 116]. For example, an early DCM was used to improve the scalability of the breakpoint phylogeny search strategy in GRAPPA [56] that constructs phylogenies based on gene order rearrangements [139]. The combination of divide-and-conquer with iteration, so that the tree that is

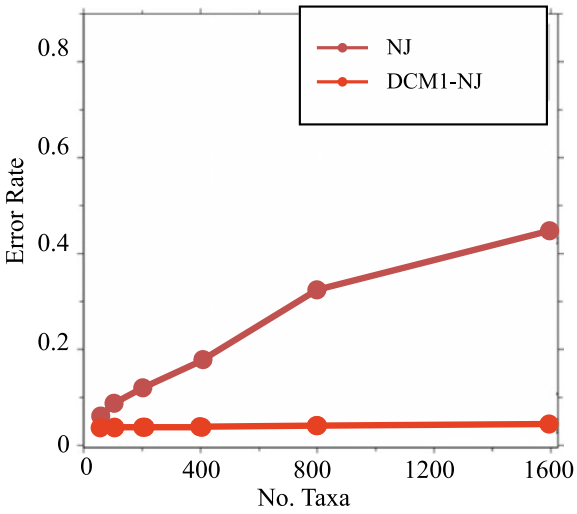


Fig. 6.2 (Adapted from [91], with permission from IEEE Computer.) The impact of the DCM1 divide-and-conquer strategy on tree estimation. Here we show a comparison of DCM1-NJ and neighbor joining (NJ) on simulated datasets with up to 1600 sequences, each with 1000 sites, that have evolved under the K2P model of sequence evolution. DCM1 is one of the disk-covering methods, and it operates by using chordal graph theory to divide the species set into overlapping subsets, constructs trees on the subsets (here using neighbor joining), and then combines the subset trees using a variant of SuperFine [135]. The y-axis shows the tree error rate (proportion of edges missing from the true tree in the estimated tree). DCM1-NJ (i.e., DCM1 used with neighbor joining to construct subset trees) has much lower error than neighbor joining used alone. Furthermore, DCM1-NJ is fast-converging, as shown in [148], whereas neighbor joining is not

constructed during one iteration can be used as a basis for a new decomposition, has been particularly powerful. As an example, DACTAL [93] was developed for constructing trees from unaligned sequences, but has since been used in other contexts, including estimating species trees using coalescent-based methods [15]. See Fig. 6.3 for how DACTAL combines divide-and-conquer with iteration.

In each case, the use of DCMs resulted in improvements in accuracy, scalability, or running time compared to the original method. Thus, DCMs are generic techniques that can be used to improve base phylogeny estimation methods more generally. In addition, as shown in [64, 93, 148], under some circumstances, the constructed trees are provably correct (i.e., topologically identical to the tree that generates the data given as input to the DCM).

The accuracy of any tree produced using a divide-and-conquer strategy depends closely on the decomposition strategy, the technique used to compute subset trees, and the supertree method used to combine the subset trees. While all of these steps impact the final accuracy (e.g., see [117] for how the decomposition strategy impacts accuracy), this chapter focuses on the last step in the pipeline: the supertree method.

The early supertree methods largely focused on the calculation of supertrees for the idealized case where the source trees could be combined into a species tree that

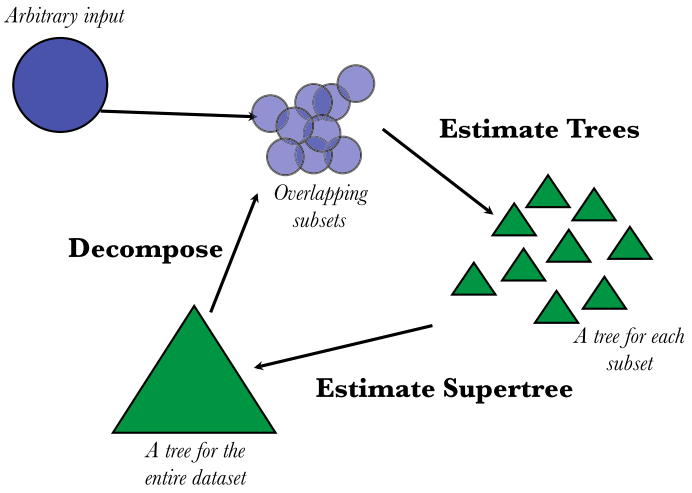


Fig. 6.3 (Figure modified from [93] with permission from Oxford University Press.) The generic DACTAL design. DACTAL was originally developed to enable tree estimation from an input set of unaligned sequences, but in its generic form, it can be used to construct trees from arbitrary inputs. The input provides data for a given set of species; the first step decomposes the set of species into overlapping subsets, then trees are computed for each subset, and the subset trees merged together using a supertree method. Then the set of species can be decomposed into overlapping subsets using specialized strategies designed for use when a tree is given (e.g., the padded recursive decomposition strategy from [93]), and the process can be repeated

agreed with them perfectly. Since phylogeny estimation nearly always has some error, this requirement on the source trees is highly restrictive. Today, there are many supertree methods that can handle conflict between the source trees and the construction of supertrees on biological datasets is a fairly common occurrence; for example, [10, 21, 42, 49, 57, 69, 70, 104, 105, 119] present species trees computed using supertree methods. Supertrees also provide insight into a surprisingly large number of biological questions, as surveyed in [22].

One of the earliest supertree methods to be developed is matrix representation with parsimony, also referred to as MRP [12, 108]. MRP is so widely used that it has become synonymous with "supertree method". Today, the development of new species tree methods is an active research area in computational phylogenetics and draws on discrete mathematics, computer science, and probability theory, and many new supertree methods have been developed, some of which are even more accurate than MRP. Yet more research is needed, as even the best supertree methods fail to be highly accurate on large datasets with many thousands (and even tens of thousands) of species, which is where supertree methods are most needed [22], and most cannot even run on datasets of these sizes.

The purpose of this chapter is to explore the challenges to scalability for existing supertree methods, especially in the context of divide-and-conquer strategies for large-scale tree estimation. We identify the major algorithmic approaches in supertree

construction and the challenges of scaling those approaches to large datasets, and identify open problems where advances by computer scientists would potentially lead to practical supertree methods with good accuracy on large datasets. We also discuss the use of supertree methods in divide-and-conquer strategies that aim to estimate phylogenetic trees on ultra-large datasets. The basic theoretical material for supertree construction is provided here in a brief form, and further details can be found in [147].

6.2 Background

6.2.1 Terminology

We introduce the basic terminology used in supertree estimation.

Definition 1 The input to a supertree problem is a set $\mathscr{T}$ of trees. The set $\mathscr{T}$ is referred to as a **profile** and the individual trees in $\mathscr{T}$ are referred to as **source trees**. We let $\mathscr{L}(t)$ denote the leafset of tree t. Then a supertree for $\mathscr{T}$ is a tree T with one leaf for every $s \in S = \cup_{t \in \mathscr{T}} \mathscr{L}(T)$.

The focus in this chapter is on unrooted source trees, since techniques for rooting phylogenies depend on careful selection of outgroup taxa and can introduce error into the supertree profile. Furthermore, although in some cases the source trees will have branch lengths or branch support values, in the classical setting (and most common use case) the source trees come without any branch lengths or branch support values and are just "tree topologies". Hence, the description of supertree methods in this chapter is for this simplest setting where the input set of source trees are unrooted binary trees without branch lengths or support values.

Finally, we assume that the source trees will differ from the true species tree (when restricted to the same taxon set) only due to *estimation error*. Thus, we do not address the case where the source trees are gene trees (either estimated or true gene trees), as these can differ from the true species tree and from each other due to processes such as gene duplication and loss, incomplete lineage sorting, and horizontal gene transfer [84]. When the input profile consists of gene trees, then methods that explicitly address gene tree discord (such as phylogenomic "summary methods", which combine estimated gene trees to produce an estimated species tree) are more suitable and are discussed in Sect. 6.5.

For a given profile $\mathscr{T}$, the best possible outcome is where there is a supertree T that agrees with all the source trees. We formalize this as follows:

Definition 2 We let $T|X$ denote the subtree of T induced by the leaves in X. If T is a supertree for $\mathscr{T}$ and $t \in \mathscr{T}$ is a binary tree, we say that T agrees with t if $T|\mathscr{L}(t)$ is isomorphic to t. We also say that T is a **compatibility tree** for $\mathscr{T}$ if T agrees with t for all $t \in \mathscr{T}$. When a compatibility tree exists for a set $\mathscr{T}$, we say that the set $\mathscr{T}$ is **compatible**.

However, usually source trees are not compatible (because tree estimation nearly always has some error), so that the objective is to find a supertree that minimizes some total distance (or maximizes some total similarity score) to the input source trees [39, 141].

6.2.2 Representations of Trees

Unrooted supertrees and input source trees can be represented in several different ways, and many of the supertree methods that have been developed are based on these representations. Here we describe three basic techniques for representing trees: sets of bipartitions, sets of quartet trees, and additive distance matrices.

Definition 3 Every edge in a tree t defines a bipartition on the leafset of t (i.e., when you delete the edge e from t, you separate the leaves into two sets, A_e and B_e, thus defining the bipartition $A_e|B_e$). We let **Bip(t)** denote the set of bipartitions of the leafset of t defined by the edges of t.

It is not hard to see that the set $Bip(t)$ defines the tree t, and that t can be reconstructed from $Bip(t)$ in polynomial time [147].

Definition 4 Every set q of four leaves in an unrooted binary tree t induces a quartet tree t_q, and hence t defines the set **Q(T)** of all its induced quartet trees.

As with bipartitions, the set $Q(t)$ defines the tree t, and t can be reconstructed from $Q(t)$ in polynomial time [147]. The last representation we present is based on distance matrices:

Definition 5 A matrix D is said to be **additive** if there is a tree T with n leaves and nonnegative lengths on the edges so that D_{ij} is the total of the edge lengths on the path in T between leaves i and j.

Given an additive matrix, there is a unique minimal edge-weighted tree corresponding to the additive matrix [28], and it can be constructed from the additive matrix in polynomial time (see [150] for the first such method, and see [147] for others).

For a given unrooted tree t, there are several ways of representing t using an additive matrix. For example, some source trees are computed using methods such as maximum likelihood or neighbor joining [118], which explicitly provide positive branch lengths on the edges of the tree; given such a tree, the matrix of leaf-to-leaf distances (summing all the branch lengths on the path) is by definition additive. Other source trees come without branch lengths, and so using unit lengths on all the edges produces an additive matrix. Therefore, whether the source trees are either unweighted or have strictly positive branch lengths, they can be used to define additive matrices, and are uniquely defined by their additive matrices.

Thus, every unrooted binary tree t can be defined using either its set $Bip(t)$ of bipartitions, its set $Q(t)$ of quartet trees, or by an additive matrix A, and given any

of these representations the tree t can be reconstructed in polynomial time. Each of these representations suggests approaches for constructing supertrees from a set $\mathscr{T}$ of source trees. For each of the following general categories of problems, the input is a set $\mathscr{T}$ of source trees and the objective is a supertree T that is close to the source trees, with the measurement of distance based on one of these representations of source trees and the supertree.

6.2.3 Bipartition-Based Supertree Methods

Three of the most well-known optimization problems for supertree construction—Matrix Representation with Parsimony, Robinson–Foulds Supertrees, and Maximum Likelihood Supertrees—are based on bipartition encodings. Furthermore, Matrix Representation with Likelihood is a new approach that also uses the bipartition encoding and has been shown to be competitive with MRP. Therefore, we will begin with this category of supertree methods.

6.2.3.1 Matrix Representation with Parsimony

We begin with the well-known NP-hard supertree problem "Matrix Representation with Parsimony" (MRP) [12, 13, 20, 108]. There are many variants on this problem, described in [10], but the basic approach is the one that is most frequently used. The input set $\mathscr{T}$ of (source) trees is represented by a matrix (called the MRP matrix) with 0s, 1s, and ?s, as follows.

Definition 6 Given a profile $\mathscr{T}$ of source trees, each edge e in each tree $t \in \mathscr{T}$ defines a bipartition $A_e|B_e$ on $\mathscr{L}(t)$. The bipartition $A_e|B_e$ can be represented as a n-tuple (where $|S| = n$) where the ith entry has 0 for the elements $s_i \in A_e$, 1 for the elements $s_i \in B_e$, and ? for the elements in $S \setminus (A_e \cup B_e)$. These n-tuples representing the bipartitions (taken across all the edges from all the source trees) form the columns in the MRP matrix. Thus, the **MRP matrix** has one row for every species in $S = \cup_t \mathscr{L}(t)$ and one column for every internal edge in any tree in $\mathscr{T}$.

Once the MRP matrix is computed, the rows of the matrix are treated as strings (all having the same length) that will represent the leaves of the tree we will seek, and a maximum parsimony tree is sought for the matrix:

Definition 7 The **cost** of a tree T in which all the nodes are labeled by strings of length k is the total of the Hamming distances on the edges of the tree, where the Hamming distance between two k-tuples $s = (s_1, s_2, \ldots, s_k)$ and $s' = (s'_1, s'_2, \ldots, s'_k)$ is the number of positions i for which $s_i \neq s'_i$. The tree T with the lowest achievable cost (among all ways of assigning sequences to its internal nodes) is said to be a **maximum parsimony tree**.

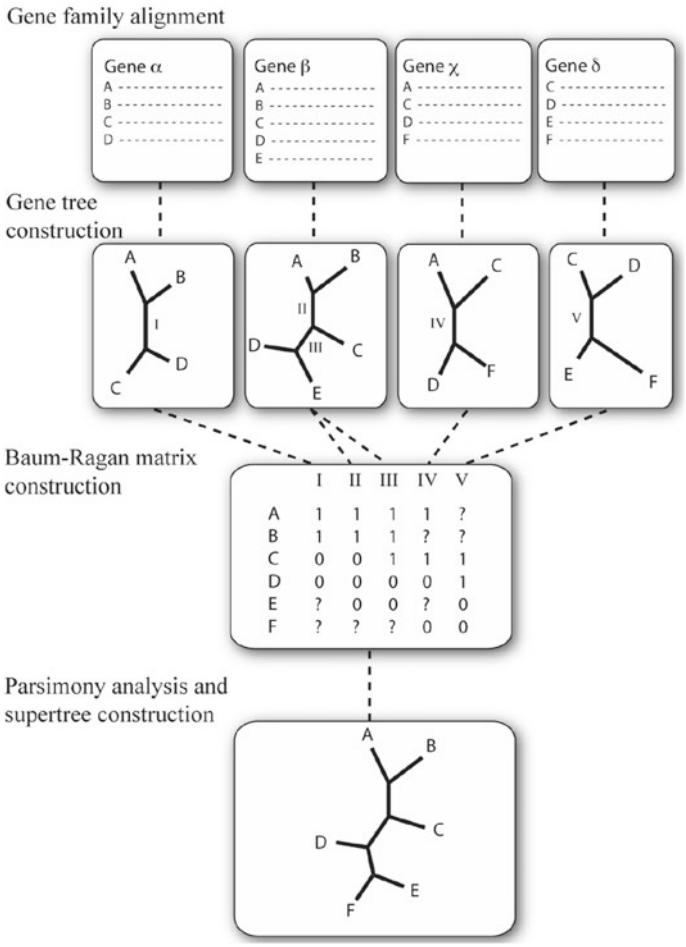


Fig. 6.4 (From [39]; reprinted with permission from Springer) The MRP pipeline: construct source trees (here, gene trees on different loci), then construct the MRP matrix from the set of source trees, and then run maximum parsimony heuristics to obtain the MRP supertree. For this particular example, note that the MRP tree induces each of the four source trees, so that they are all compatible. Note also that the characters (i.e., columns) in the MRP matrix are compatible on the MRP tree

Figure 6.4 gives an example of the MRP matrix computed on four source trees, and the maximum parsimony tree found for that matrix.

Exact solutions to maximum parsimony are unlikely to be achievable in polynomial time, since finding the best possible tree is NP-hard [52]. For this reason, heuristic searches (using standard maximum parsimony software, such as PAUP* [136] or TNT [54]) are used to find good solutions (i.e., local optima) to maximum parsimony. The running time of these heuristics is mainly impacted by n, the number of sequences in the input, as the number of trees grows exponentially in n, but

the number of sites (sequence length) also impacts the running time as the time to calculate the score of a given tree is linear in the number of sites. Note that when the source trees are compatible, then the optimal solutions to MRP are compatibility trees, and hence an exact solution MRP will return a compatibility tree in that case; this is one of the positive aspects of the MRP approach to supertree estimation. In addition, simulation studies have generally found MRP to be among the most accurate supertree methods, with MRP referred to as the "gold standard" among supertree methods developed as of 2011 [25].

The MRP approach is just one of the approaches for constructing supertrees from the MRP matrix, and other optimization problems can be considered for the same matrix representation of the input source trees. For example, Matrix Representation with Likelihood (MRL) [99], Matrix Representation with Compatibility (MRC) [115], and Matrix Representation with Flipping [32] have also been proposed. These are also NP-hard problems, and so heuristics are used to find good solutions. Of these methods, only MRL has been shown to provide accuracy comparable to (and sometimes exceeding) MRP [99]; hence, MRL deserves some additional discussion.

6.2.3.2 Matrix Representation with Likelihood

The MRL optimization problem takes the same input matrix as MRP, and randomizes the 0s and 1s to avoid bias. Then, a solution to maximum likelihood under the two-state symmetric sequence evolution model is sought using RAxML [129]. As shown in [99], supertrees computed using MRL were more accurate than supertrees computed using MRP, whether TNT or PAUP* heuristics were used, and MRL scores showed somewhat stronger correlations with tree accuracy than MRP scores. Furthermore, RAxML was reasonably fast on these datasets, finishing in less time than the parsimony heuristics that were used.

6.2.3.3 Robinson–Foulds Supertrees

We continue with the Robinson–Foulds Supertree [11, 31].

Definition 8 The **Robinson–Foulds (RF) distance** between two binary trees T_1 and T_2 on the same leafset is $|Bip(T_1) \triangle Bip(T_2)| = |Bip(T_1) \setminus Bip(T_2)| + |Bip(T_2) \setminus Bip(T_1)|$ [113]. When T_1 and T_2 have different leafsets, then we define the RF distance between T_1 and T_2 to be $RF(T_1|L, T_2|L)$, where $L = \mathscr{L}(T_1) \cap \mathscr{L}(T_2)$ [37]. Given profile $\mathscr{T}$ of source trees, the RF distance between a supertree T and $\mathscr{T}$ is $\sum_{t \in \mathscr{T}} RF(T, t)$. A **Robinson–Foulds Supertree** for $\mathscr{T}$ is a supertree T that has the minimum RF distance to $\mathscr{T}$.

The Robinson–Foulds Supertree problem is NP-hard, since determining if the profile $\mathscr{T}$ is compatible is NP-hard. Algorithms for the Robinson–Foulds Supertree problem include MulRF [29], PluMiST [72], and FastRFS [143].

6.2.3.4 Likelihood-Based Supertree Methods

Supertree estimation can be framed as a statistical inference problem that treats the estimated supertree T as the model species tree that generates the set $\mathscr{T}$ of observed source trees. In this setting, the objective is a maximum likelihood supertree, i.e., a tree T such that $Pr(\mathscr{T}|T)$ is maximized.

The initial formulation of this approach is due to Steel and Rodrigo [132], who defined the probability of a source tree, given a model species tree, as a function of the Robinson–Foulds distance between the two trees, a development that was met with great excitement in the literature [38]. Two methods that have been developed that are based on the Steel and Rodrigo model are [3], which attempts to compute the maximum likelihood supertree, and the Bayesian method in [4]. Furthermore, as noted in [27], there is a (potentially close) relationship between optimal solutions to the Robinson–Foulds Supertree and Maximum Likelihood Supertree problems, suggesting that good solutions to one problem might be pretty good solutions to the other problem. Hence, methods for the Robinson–Foulds Supertree may provide a good approximation to the maximum likelihood supertree problem under the Steel and Rodrigo model.

6.2.4 Quartet-Based Supertree Methods

Since each unrooted tree t can be defined by its set $Q(t)$ of quartet trees, quartet-based supertree optimization problems have been posed. Here we describe a very simple such optimization problem.

Definition 9 The **Maximum Quartet Support Supertree** problem is defined as follows. The input is a profile $\mathscr{T}$ of source trees, and we seek the supertree T that maximizes $qsim(T, \mathscr{T})$, where $qsim(T, \mathscr{T}) = \sum_{t \in \mathscr{T}} |Q(T) \cap Q(t)|$.

The Maximum Quartet Support Supertree problem is NP-hard, since even the decision problem of determining whether a set of unrooted source trees is compatible is NP-hard [130]. Furthermore, it remains NP-hard even for the special case where every source tree is on the full set S of taxa [74]. Note that quartet-based supertree problems can also be formulated as minimizing the quartet distance to the source trees, as these are equivalent.

One approach to solving this problem is to represent every source tree by its set of quartet trees, use those sets of quartet trees to produce a single quartet tree for every four leaves (e.g., by majority vote), and then construct a tree on the full set of species from the constructed set of quartet trees.[1]

[1]This approach has been used to construct species trees from gene trees under the multispecies coalescent; an example of such a method is the population tree in BUCKy [76], but see also [147] for others.

The last step in this process, where quartet trees are combined into a single tree on the entire dataset, is called "quartet amalgamation". The most well-known quartet amalgamation methods are Quartet Puzzling [133] and Quartets MaxCut [128], but other methods have also been developed with good theoretical and empirical performance [16, 103, 112, 156]. Some of these quartet amalgamation methods are based on weighted versions of the standard quartet amalgamation problem, where the input set of quartet trees have weights on them, perhaps reflecting the level of confidence in the quartet tree. Weighted quartet amalgamation methods include Weight Optimization [111] and Weighted Quartets MaxCut [8], and these weighted versions tend to provide better accuracy than unweighted quartet amalgamation methods. Since quartet compatibility is itself NP-complete, optimization problems for quartet amalgamation are NP-hard; however, many approximation algorithms theoretical results about quartet amalgamation have been developed [6, 17, 18, 23, 55, 68, 74].

Quartet-based supertree methods have been developed and studied, and often shown to have very good accuracy, sometimes matching or improving on heuristics for MRP, generally considered the best supertree method for various criteria [9, 103, 134]. However, the default usage of these methods typically depends on using all the quartet trees, and hence explicitly will run in $\Omega(n^4)$ time, where there are n leaves; hence, quartet-based supertree methods will typically be inapplicable to large datasets. Some quartet amalgamation methods can be run on just a subset of the quartet trees, which enables quartet-based supertree methods to be applied to sparse samples of the quartet trees. A study [134] evaluating the impact of sampling quartets on supertrees computed using Quartets MaxCut [128] showed that some sampling strategies produced supertrees that were competitive with MRP, but the most accurate supertrees were obtained when all quartet trees are used rather than a subset of the quartet trees.

6.2.5 Distance-Based Supertree Methods

Distance-based supertree optimization problems are also popular [25, 40, 75, 155, 155]. As with quartet-based methods, one approach is to seek an additive matrix that is as close as possible to the set of additive matrices defining the source trees. To make this concrete, for each source tree t an additive distance matrix D^t is computed. Then, given an additive distance matrix D^T (corresponding to an edge-weighting of a supertree T), we define the distance between D^T and each d^t by constraining D^T to the rows and columns corresponding to the species that are in t and then computing any of several natural distances between matrices (e.g., the L_2 or L_∞ metrics). The objective is an additive distance matrix that minimizes that total distance. Once such an additive distance matrix is found, then the tree corresponding to the distance matrix is returned.

Another approach for distance-based supertree optimization is to compute a single distance matrix D from the set of distance matrices $\{d^t : t \in \mathscr{T}\}$ and then seek an additive distance matrix D^* that is close (under some metric) to D [40, 75]. For

example, $D[i, j]$ can be set to the average of the $d^t[i, j]$ for all the source trees t that contain both i and j, and the objective could be an additive matrix that minimizes the L^2-distance to D. Once an additive matrix is found, the tree (and its set of edge weights) can be computed in polynomial time. Thus, this approach can also be used to compute supertrees.

Distance-based tree estimation is a very well-studied problem, and there are many methods that have been developed to construct trees from distances [149]. Fortunately, although finding the nearest additive matrix to a given distance matrix is NP-hard (see [1, 2, 48] for an entry to this literature), there are many polynomial time methods for computing trees from distance matrices, including neighbor joining [118] and FastME [78], and some interesting theory about these approaches [24, 43, 46, 47, 87, 102, 125, 131]. However, distance-based tree estimation has another challenge in that for many supertree datasets, there can be pairs of species i, j that are not found together in any source tree, and when this happens $D[i, j]$ will not have any value. This is referred to as "missing data", and computing trees from distance matrices with missing data is not well solved. A few methods are available to estimate trees from distance matrices with missing data [41, 71, 106]; however, the accuracy of trees computed from incomplete distance matrices is generally not as good as trees computed from complete matrices [143].

In summary, distance-based supertree estimation has been approached in two different (but related) ways: (1) represent each source tree by an additive distance matrix and then seek an additive distance matrix that is close to the set of additive distance matrices computed on the source trees, or (2) represent the set of source trees by a distance matrix D_0 (that may be incomplete) and then seek an additive matrix that is close to D_0. Both approaches are challenged by the fact that desirable definitions of "close" result in NP-hard optimization problems (and current approaches for finding good solutions to NP-hard optimization problems are not scalable). The second approach has the additional challenge that the matrices D_0 produced by supertree profiles are often incomplete, and current methods for constructing trees from incomplete distance matrices do not have adequate accuracy. Thus, highly accurate distance-based supertree estimation requires novel techniques in order to scale to most large supertree datasets.

6.3 Accuracy and Scalability of Existing Supertree Methods

Up to 2011, MRP has been generally found to produce more accurate supertrees than competing methods, leading [25] to refer to MRP a "gold standard" for supertree methods. Several supertree methods have been developed since then, some of which have shown to be competitive with MRP in terms of accuracy. For example, ASTRID [142], ASTRAL-2 [89] (an improved version of ASTRAL [88]), MRL [99], and FastRFS [143] all demonstrate high accuracy that is comparable to (or better than)

MRP under some conditions. However, ASTRID has poor accuracy on supertree datasets where there is no source tree containing most of the species, whereas other supertree methods can perform well under those conditions [143].

As discussed earlier, maximum likelihood and Bayesian supertree estimation methods are also very promising, and a maximum likelihood supertree method [3] and a Bayesian method [4] have been proposed for supertree estimation under the Steel and Rodrigo model. Yet both methods are computationally intensive, and have not (to our knowledge) been used on large datasets.

Thus, all the methods we have described are computationally intensive on large datasets, and most do not scale to large datasets. In particular, nearly all supertree methods explicitly search for good solutions to NP-hard problems using standard techniques for exploring tree space, which are ineffective on large datasets since the number of trees grows exponentially in the number of leaves. Some of the distance-based methods are polynomial time (i.e., the ones that first compute a distance matrix summarizing all the source trees, and then run polynomial time distance-based methods such as neighbor joining). However, the polynomial time distance-based supertree methods have not been shown to provide comparable accuracy to MRP, and furthermore have reduced accuracy when the average distance matrix they compute has missing entries. Modifications of these distance-based methods so that they also seek good solutions to NP-hard optimization problems might improve accuracy but would also make them computationally intensive. Hence, scalability to large datasets is not yet achieved using existing supertree methods.

Thus, the best *current* supertree methods are either based on encodings using the bipartitions in the source trees (i.e., the MRP matrix) or the quartet trees in the source trees, and these representations are not efficient. In particular, the MRP matrix has one row for every species and as many columns as there are edges in the input source trees, and so is larger than the input profile! For example, suppose we wish to run MRP on a total of 10,000 species from 100 source trees each with 1,000 species; the MRP matrix will have 10,000 rows and nearly 100,000 columns. This is a huge matrix, and maximum parsimony heuristics are not likely to be able to find good local optima on such a dataset. A similar issue occurs for quartet-based supertree methods, since there are $\Omega(n^4)$ quartets on an n-species dataset, and this is quickly a very large number even for modest values for n.

6.4 Improving Scalability of Supertree Methods

This section describes some of the major approaches to supertree estimation, focusing on those methods that have been designed for improved scalability to datasets with large numbers of species.

6.4.1 SuperFine: Boosting Supertree Methods

The SuperFine method [135] was developed to improve the scalability of supertree methods to large datasets. Thus, the input to SuperFine is a set of source trees and also a selected supertree method, such as MRP, MRL, Quartets MaxCut, etc.

SuperFine has two steps, where the first step produces a constraint tree, and the second step refines the constraint tree using the base supertree method. The constraint tree produced in the first step is an unrooted tree that contains all the species, called a "Strict Consensus Merger" (SCM) tree, that by design only contains bipartitions that are consistent with all the input source trees.

The SCM tree is computed by merging pairs of source trees until the entire set of source trees is merged into a single tree (Fig. 6.5). When two trees are merged, the set of shared leaves is computed. If the two trees do not induce the same subset tree on this shared leaf set, the edges of the two trees are collapsed to make the resultant trees identical on the induced subset trees. Specifically, the strict consensus tree (which is the tree that has only the bipartitions in both subset trees) of the two induced trees is computed, and the two input trees have edges collapsed to make them induce this strict consensus tree. The strict consensus tree on the shared leaf set is called the "backbone tree". This backbone tree is typically not fully resolved, since any conflict between the two trees will result in the strict consensus being unresolved. Then the remaining taxa are added into the backbone tree. However, this step can also result

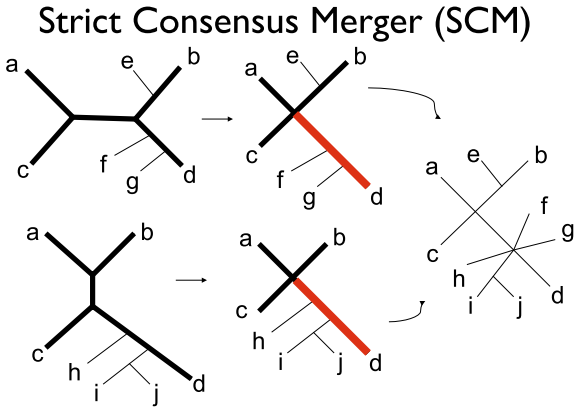


Fig. 6.5 The Strict Consensus Merger (SCM) method, which is the first step in SuperFine [135]. First, the induced trees (shown with thick black lines) on the set of shared taxa in the two trees are computed, and the conflicting edges are collapsed in each tree so that they induce the same subtree (called the "backbone tree") on the shared taxa. Then the two trees are merged into a supertree by adding the missing taxa to the backbone tree. A "collision" occurs when both trees contribute taxa to the same edge in the backbone tree (the edges colored red in this figure are backbone edges with a collision). In the event of collision, all subtrees in the two trees contributing to that edge will be attached to the same location in the merged supertree. Thus, the SCM tree will tend to have lower resolution than the source trees

in additional collapses of edges, as any time both of the trees contribute taxa to the same edge in the backbone tree a polytomy is created. Thus, the Strict Consensus Merger (SCM) tree is generally unresolved (i.e., it is likely to have nodes of degree greater than three).

The second step of SuperFine refines the SCM tree into a binary tree using the base supertree method, applied to encoded versions of the source trees. The refinement step has several nice properties. First, the polytomies (i.e., nodes in the SCM that have degree four or more) can be refined independently without changing the final supertree; thus, this step can be easily parallelized [94, 95]. The other nice property is that the refinement of a single node of degree d can be performed by applying the base method to a modified version of the source trees where each source tree is replaced by a new source tree with leaves labeled $1 \ldots d$. Therefore, if the maximum degree d_{max} of any node in the SCM is not large (say, at most 100), then the refinement step can be very fast since the base supertree method is only applied to a collection of source trees with a total of d_{max} species. This property is true for the Strict Consensus Merger, but may not be true of other ways of computing a tree from the input source trees!

As shown in [99, 135], SuperFine used with base supertree methods MRP, MRL, or Quartets MaxCut produces trees that are either as accurate or more accurate than the base supertree method run alone (i.e., without the SuperFine divide-and-conquer strategy); furthermore, when the total number of taxa is large enough, the use of SuperFine often reduces the total time to calculate the supertree. In addition, there are datasets on which the base supertree method cannot run due to computational reasons, but where the SuperFine divide-and-conquer framework enables it to run to completion. Thus, SuperFine is a generic technique for improving the accuracy, running time, and scalability of supertree methods.

However, the advantage obtained in using SuperFine depends on the SCM tree—if it is completely unresolved, then there is no benefit to using SuperFine since the refinement step is identical to just applying the base supertree method to the original set of source trees. Similarly, if the SCM tree has polytomies with very high degree (reflected by having very few internal edges), then the opportunity for improvement is greatly reduced. There are two conditions that cause the SCM to have reduced resolution: one is conflict between the source trees (i.e., two source trees are incompatible), and the other is insufficient overlap between source trees. The first condition (i.e., conflict between source trees) is likely to occur since perfectly accurate phylogeny estimation is rarely seen and hence will lead to conflict. However, the second condition is also likely to occur and will contribute to the loss of resolution.

It therefore makes sense to consider the trends exhibited on biological data. On five biological supertree datasets analyzed using SuperFine in [96, 99], the maximum degree of resolution in the SCM tree was 57%, where the resolution is the ratio between the number of internal edges in the tree and the number of internal edges in a fully resolved tree on the same set of leaves. In fact, for two of these datasets, the SCM tree had at most 10% resolution (one of which only had 1% resolution). Thus, the degree of resolution of the SCM tree was variable on biological data, but there were cases where the degree of resolution was very poor.

Since the SCM is computed by greedily combining source trees until they are all merged into a single tree, the number of source trees should also impact the degree of resolution in the SCM (with less resolution occurring for larger numbers of source trees). This is consistent with the results shown on biological datasets in [99], where the SCM trees with the lowest degree of resolution occurred for the datasets with the most source trees. Fleischauer and Böcker [50] explored techniques to improve the degree of resolution in the SCM, mainly by modifying the order in which the source trees are combined. Their new techniques show some improvement in resolution for the SCM, especially when the use of the semi-strict consensus (instead of the strict consensus) is used (which would make the result not actually a strict consensus tree but something else). However, their study shows that *all* the variants they explored have high missing branch rates and low false positive rates, indicating that even with the improvement the SCM trees were fairly unresolved.

Conversely, if the number of source trees is small enough to avoid reduction in resolution in the SCM, then datasets with high evolutionary distances and/or large numbers of species are bound to have some source trees that are large and/or span large evolutionary distances. However, these are exactly the cases where the source trees are likely to have high error, and hence lead to conflict between source trees (and hence decrease resolution in the SCM). In other words, when the full set of taxa is large and/or spans evolutionary distances, the SCM is likely to have poor resolution.

In summary, although SuperFine can improve supertree methods scalability and accuracy, these improvements only occur when the SCM is highly resolved, and biological supertree datasets with large numbers of source trees and/or large numbers of species, and especially those spanning deep evolutionary histories, would seem likely to produce SCM trees that are highly unresolved. In other words, SuperFine may not enable scalability to datasets with many thousands of species spanning deep evolutionary histories, except—perhaps—under limited conditions.

6.4.2 Explicitly Constraining the Search Space

Although SuperFine's technique for constraining the search space doesn't always help, there are other ways that the search space can be constrained. Here we describe one such type of approach where a set X of "allowed bipartitions" is constructed in some manner, and a supertree T is sought that optimizes some criterion, subject to the constraint that $Bip(T) \subseteq X$. Both FastRFS [143] and ASTRAL [88, 89] use this approach.

Phylogeny estimation using constrained optimization (where the constraints are explicitly provided by a set of allowed bipartitions or clades) was first introduced in [59] in the context of constructing species trees from a set of gene trees, under a duplication-loss model. This technique has also been used in other phylogeny estimation methods for different optimization problems [14, 26, 137, 140, 143, 145, 157]. Each of these methods seeks a tree that optimizes some criterion (typically the

distance to the input profile), but constrains the set of feasible trees by an explicitly constructed set X of allowed bipartitions (or, if the tree that is sought must be rooted, then by an explicitly constructed set X of allowed clades). Once the set X is defined, the construction of the optimal tree drawing its bipartitions from X can be performed in polynomial time using dynamic programming, although the specific subproblem formulation depends on the optimization problem.

The accuracy of these methods clearly depends on how the set X is defined, since if X is very small then the set of feasible solutions is also very small (or may even be empty), while if X is all possible bipartitions then these methods are guaranteed to find the globally optimal tree. However, because the running times of these dynamic programming methods grow (polynomially) with $|X|$, it is infeasible to make $|X|$ too large.

The simplest way to define X is to use all the bipartitions of the source trees in $\mathscr{T}$, and this is the technique used in nearly all the methods described above. However, when a source tree t does not contain all the species, its bipartition set $Bip(t)$ is not directly usable, since these bipartitions are not on S but rather on $\mathscr{L}(t)$, which is a proper subset of S. Hence, this technique is restricted to using just those source trees that contain all the species.

Most of the methods above (with the sole exception of FastRFS) were developed for species tree estimation where the input is a set of gene trees; in that setting, typically some (and perhaps most) of the gene trees will contain all the species. Furthermore, phylogenomic species tree estimation is increasingly being performed with hundreds to thousands (or tens of thousands) of genes [66, 153]. Hence, for the case of phylogenomic species tree estimation, this simple technique will produce a set X that has a large number of bipartitions. Furthermore, as shown in [88, 145, 157], highly accurate species trees can be constructed using this technique, making this approach useful in practice for species tree estimation from multigene datasets.

However, the supertree setting presents challenges to using this technique to compute the set X. As noted, when a source tree t does not contain all the species, its bipartition set $Bip(t)$ is not directly usable, and hence, this technique is restricted to just those source trees that have all the species. But large supertree datasets may have no such source trees, making this simple technique for computing X completely useless. Alternative approaches for constructing the constraint set X of bipartitions when the input profile has no complete source trees have been developed and used in ASTRAL-2 [89] (and further enhanced in ASTRAL-3 [158]). The basic approach in [89, 158] is to construct a distance matrix relating all the species, and then use that distance matrix to "complete" all the incomplete gene trees; then, the bipartitions from the completed gene trees can be used for X; this is a relatively efficient method on most datasets, but it can be slow on some large datasets with poorly supported gene trees. Similarly, [61] can be used to complete incomplete source trees, but it runs in $\Omega(n^4)$ time (where n is the number of species) and so is not scalable to large datasets. OCTAL [36] is a linear time algorithm that uses a different approach to complete source trees, but this requires that a tree containing all the species already be available. Thus, all the approaches for completing incomplete source trees seem to have inherent limitations that reduce scalability. In conclusion, defining the con-

straint set X of allowed bipartitions is challenging for many supertree datasets, and especially so when the number of species is very large.

However, if an initial set X of allowed bipartitions can be computed using some technique, adding to that set can be beneficial in terms of the resultant supertree or species tree, as shown in [143, 145]. For example, FastRFS is designed to find optimal solutions to the Robinson–Foulds Supertree problem, within the constraint set defined by the set X of allowed bipartitions. Since MulRF [29] and PluMiST [72] are also designed to find good solutions to the same optimization problems, adding the bipartitions from the trees they find to the initial set X computed by FastRFS ensures that FastRFS will be guaranteed to find solutions that are *at least* as good (with respect to the optimization problem) as those found by MulRF and PluMiST. Furthermore, adding computed supertrees or species trees, however they are computed, can enlarge the search space and lead to improved topological accuracy; this is the basis of the "enhanced" versions of FastRFS and SVDquest [145], which use bipartitions from trees computed using other methods for species tree estimation, and obtain more accurate species tree as a result.

6.5 Relationship to Phylogenomic Species Tree Estimation

The assumption throughout this chapter is that the input profile contains estimated species trees, so that the only cause for the source trees to be different from the true species tree is estimation error. Yet, supertree methods are also used to combine estimated gene trees into a species tree, which is a different type of analysis and presents different challenges.

Phylogenomic species tree estimation—i.e., the estimation of species trees using multiple loci taken from the genomes of the different species—has become a common practice in systematics [66, 153]. The traditional approach is concatenation, in which the multiple sequence alignments for the different loci are concatenated into one long alignment (called a super-matrix or a super-alignment), and then standard phylogeny estimation methods, such as maximum likelihood, are used to construct a tree on the super-matrix. Yet, it is now well established that gene trees can differ from the species tree due to multiple biological processes, including incomplete lineage sorting, gene duplication and loss, and horizontal gene transfer [84], and that concatenation analyses can be statistically inconsistent (and worse, positively misleading) when there is sufficient heterogeneity between true gene trees and species trees [114]. Furthermore, gene tree heterogeneity is commonly found in large-scale biological datasets (e.g., [66, 153]), leading many systematists to seek alternative approaches to concatenation analysis in constructing species trees.

Much of the focus in this area has been on addressing gene tree heterogeneity due to incomplete lineage sorting (ILS), as it is expected to appear in all large phylogenomic datasets to some extent [45]. Furthermore, gene tree heterogeneity due to ILS is modeled by the multispecies coalescent (MSC), and there are several methods for estimating species trees that have been proven to be statistically consistent under the

MSC. For example, "summary methods", which operate by estimating gene trees and then combining the gene trees into a species tree, form one category of species tree estimation methods that are statistically consistent under the MSC. Examples of summary methods that are statistically consistent under the MSC include ASTRAL (and its improved versions), ASTRID, MP-EST [82], NJst, the population tree in BUCKy [76], and GLASS [67, 92]. Furthermore, some of these methods (e.g., ASTRAL) provide very good accuracy in practice and are fast enough to run on large datasets.

On the face of it, a summary method is just a supertree method. Yet the two types of methods operate on different assumptions about the cause for heterogeneity in the input profile. Thus, we distinguish between the two terms to reflect the difference in the assumptions made by the methods: supertree methods assume that differences between source trees and the desired supertree are the result of source tree estimation error, while summary methods assume that biological processes also contribute to these differences.

Two of the methods we discussed in this chapter, ASTRAL and ASTRID, are actually summary methods and are developed to construct species trees from gene trees that can differ from each other and from the true species tree as the result of ILS. Furthermore, ASTRAL and ASTRID have been proven statistically consistent under the MSC, while the other supertree methods we discussed either have not been evaluated with respect to statistical consistency or are known to be inconsistent (e.g., MRP).

It is interesting to note that species tree estimation from multi-locus datasets can also be approached using distance-based supertree methods. For example, if a gene tree is computed for each locus, then a distance matrix can be computed (called the "internode" distance matrix) of average leaf-to-leaf distances across all the gene trees. As shown in [5], the internode distance matrix converges to an additive matrix for the species tree when the gene trees evolve within the species tree under the multispecies coalescent (MSC) model. Therefore, distance-based supertree methods, which use the input gene trees as source trees, will be statistically consistent under the MSC model and hence will converge to the true species tree as the number of gene trees increases. This is the basis of species tree estimation methods such as NJst [81] and ASTRID [142], which apply neighbor joining and FastME [78], respectively, to the computed internode distance matrix.

When gene tree discord is the result of gene duplication and loss, then even gene tree estimation is complicated as each species can have multiple copies of any given gene. In that case, usually the set of copies is reduced to a single representative for each species, typically based on inferring "orthology"—a difficult and still unsolved problem, as discussed in [7, 77, 126]. However, since orthology determination can be difficult to perform correctly, species trees can be estimated from the multi-copy gene trees (sometimes called "gene family trees") using summary methods that take gene duplication and loss into account; examples of such methods include iGTP [30], DupTree [151], DynaDup [14], MulRF [29], *guenomu* [101], and the method developed in [138].

One of the most interesting methods of this type is *guenomu*, a Bayesian supertree method that addresses heterogeneity between gene trees and the species trees due

to multiple biological processes. As shown in [85, 101], *guenomu* has high accuracy compared to several other species tree methods and can run on moderate-sized datasets (a few hundred species); furthermore, *guenomu* can construct species trees from gene family trees (which means that there are multiple copies of the species in each gene tree). This performance is very encouraging, especially as there are not many species tree methods that can handle gene family trees. However, *guenomu* has not been studied on datasets where the source trees are estimated species trees rather than gene trees, and so the accuracy of *guenomu* as a supertree method (in the sense that we use it in this chapter) is not yet known.

6.6 Further Reading

This chapter focused on supertree estimation in the context of divide-and-conquer methods, and mainly discussed recent methods with improved accuracy for very large datasets. Chapter 11 in [147] provides a deeper introduction to the divide-and-conquer strategies used in DCMs and the graph theory they use for accuracy guarantees. Background about the early supertree methods (and the use of supertree methods for other contexts) can be obtained from [19, 39].

This chapter focused on calculating large supertrees (with thousands of species) from unrooted source trees. Thus, we explicitly did not discuss supertree construction from rooted source trees, which are also popular. The MinCut Supertree [123], MinFlip Supertree [32–34], and PhySIC [109] are the most well-known supertree methods that work with rooted source trees. Some newer supertree methods that can handle rooted trees may have better accuracy, including Triplet MaxCut [110, 124, 127], SuperTriplets [110], and Bad Clade Deletion (BCD) supertrees [51]. Also, the Robinson–Foulds Supertree (RFS) approach [31, 143] is based on the parsimony criterion and so can be adapted for use with rooted supertrees. When the source trees are correctly rooted, supertree methods that can use rooted source trees have the potential to provide improved accuracy compared to methods that don't use this information (e.g., see [144] for a study comparing the BCD supertree method to the leading supertree methods).

There is also a substantial literature about the theoretical aspects of supertree estimation, some of which is an extension of the results for consensus methods [37, 65, 154]. When the source trees are unrooted, most optimization problems are NP-hard and the theory (which typically has to do with axiomatic approaches) is generally negative [86]; however, when the source trees are rooted some problems become polynomial time and some positive axiomatic theory can be established [22, 141].

One of the theoretical questions regarding supertree estimation has to do with when a set of source trees is "decisive", so that there is a unique tree that is compatible with all the source trees. Determining if a supertree profile is decisive is itself NP-hard, and the probability of being decisive is impacted by missing data [120]. However, there are conditions for which the supertree profile will be decisive and

where the construction of the unique supertree can be performed in polynomial time [93, 148]. There is also a relationship between decisiveness and "phylogenetic terraces", which occurs when there are multiple trees that have the same optimality score [35, 121, 122], which is interesting to explore.

Finally, the chapter by Redelings and Holder in this volume discusses taxonomy-based supertrees, which is a different type of method than the supertree methods we have discussed in this chapter.

6.7 Concluding Remarks

Given the scientific benefit of very large phylogenies, whether of genes or of species, large-scale phylogenetic tree estimation remains an important but unsolved problem. In my opinion, there are natural advantages to using divide-and-conquer to achieve these large-scale estimations, and supertree estimation provides a very useful tool for these strategies. Yet, as I have hopefully shown in this chapter, none of the current supertree methods is adequate to the task of ultra-large-scale tree estimation.

On the other hand, many of the current methods have very interesting techniques that could be improved on, and perhaps new supertree methods can be developed using these techniques that will have improved accuracy and scalability compared to current methods. My hope is that the readers of this chapter might be among the developers of these new methods.

While working on this chapter, I pursued a new type of divide-and-conquer strategy in which the taxon set is divided into *disjoint* subsets, trees are constructed on the subsets, and then the subset trees are merged into a combined tree (i.e., one with all the taxa) that maintains all the topological relationships within the subset trees. Note that because the subset trees are disjoint, none of the supertree methods discussed in this chapter can be used to merge them into a combined tree.

Two methods of this type are NJMerge [90] (which is a variant of neighbor joining) and INC [159]. Since the subset trees are disjoint, INC and NJMerge use a distance matrix computed on the taxon set to merge the disjoint subset trees. INC-NJ is INC used with neighbor joining on appropriately selected subsets, and has been proven to be fast-converging under a standard Markov model of sequence evolution [159]. NJMerge is designed to reduce the running time of computationally intensive methods and has been shown to provide that reduction for ASTRAL and SVDquartets, two coalescent-based species tree estimation methods, without any loss of accuracy. (There is no publication yet on the empirical impact of using INC in phylogeny estimation, so it is too early to comment on how INC performs.) Whether this type of method will provide the desired advantages of the best supertree methods (which have high accuracy when they can run, provided there is sufficient overlap and the subset trees are sufficiently accurate) without having their computational drawbacks remains to be seen.

Acknowledgements The author wishes to thank Pranjal Vachaspati for careful and thoughtful comments on the manuscript. We also thank the anonymous reviewers whose comments were helpful in improving the manuscript. This paper was supported in part by NSF grant CCF-1535977, but much of the work described in this book chapter was done while the author was part of the CIPRES (www.phylo.org) project, an NSF-funded multi-institutional grant that was initially led by Bernard Moret and then subsequently by the author. The first divide-and-conquer methods (DCM-NJ, DACTAL, etc.) were developed with CIPRES support, as were the supertree methods SuperFine and the Strict Consensus Merger that enabled those divide-and-conquer methods to have good performance.

References

1. Agarwala, R., Bafna, V., Farach, M., Paterson, M., Thorup, M.: On the approximability of numerical taxonomy (fitting distances by tree metrics). SIAM J. Comput. **28**(3), 1073–1085 (1998)
2. Ailon, N., Charikar, M.: Fitting tree metrics: hierarchical clustering and phylogeny. SIAM J. Comput. **40**(5), 1275–1291 (2011)
3. Akanni, W., Creevey, C., Wilkinson, M., Pisani, D.: L.U.-St: a tool for approximated maximum likelihood supertree reconstruction. BMC Bioinform. **15**, 183 (2014)
4. Akanni, W., Wilkinson, M., Creevey, C., Foster, P., Pisani, D.: Implementing and testing Bayesian and maximum-likelihood supertree methods in phylogenetics. R. Soc. Open Sci. **2**, 140,436 (2015)
5. Allman, E.S., Degnan, J.H., Rhodes, J.A.: Species tree inference from gene splits by unrooted star methods. IEEE/ACM Trans. Computat. Biol. Bioinform. (TCBB) **15**(1), 337–342 (2018)
6. Alon, N., Snir, S., Yuster, R.: On the compatibility of quartet trees. In: Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '14, pp. 535–545. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2014). http://dl.acm.org/citation.cfm?id=2634074.2634114
7. Altenhoff, A., Boeckmann, B., Capella-Gutierrez, S., Dalquen, D., et al.: Standardized benchmarking in the quest for orthologs. Nat. Methods **13**, 425–430 (2016)
8. Avni, E., Cohen, R., Snir, S.: Weighted quartets phylogenetics. Syst. Biol. **64**(2), 233–242 (2014)
9. Avni, E., Yona, Z., Cohen, R., Snir, S.: The performance of two supertree schemes compared using synthetic and real data quartet input. J. Mol. Evol. **86**(2), 150–165 (2018). https://doi.org/10.1007/s00239-018-9833-0
10. Baker, W.J., Savolainen, V., Asmussen-Lange, C.B., Chase, M.W., Dransfield, J., Forest, F., Harley, M.M., Uhl, N.W., Wilkinson, M.: Complete generic-level phylogenetic analyses of palms (arecaceae) with comparisons of supertree and supermatrix approaches. Syst. Biol. **58**(2), 240–256 (2009)
11. Bansal, M., Burleigh, J., Eulenstein, O., Fernández-Baca, D.: Robinson-Foulds supertrees. Algorithms Mol. Biol. **5**, 18 (2010)
12. Baum, B.: Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. Taxon **41**, 3–10 (1992)
13. Baum, B., Ragan, M.A.: The MRP method. In: Bininda-Emonds, O.R.P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal The Tree Of Life, pp. 17–34. Kluwer Academic, Dordrecht, The Netherlands (2004)
14. Bayzid, M., Mirarab, S., Warnow, T.: Inferring optimal species trees under gene duplication and loss. Pac. Symp. Biocomput. **18**, 250–261 (2013)
15. Bayzid, M.S., Hunt, T., Warnow, T.: Disk covering methods improve phylogenomic analyses. BMC Genomics **15**(Suppl 6), S7 (2014)

16. Ben-Dor, A., Chor, B., Graur, D., Ophir, R., Pelleg, D.: Constructing phylogenies from quartets: elucidation of eutherian superordinal relationships. J. Comput. Biol. **5**(3), 377–390 (1998)
17. Berry, V., Bryant, D., Jiang, T., Kearney, P.E., Li, M., Wareham, T., Zhang, H.: A practical algorithm for recovering the best supported edges of an evolutionary tree. In: Proceedings of the SIAM-ACM Symposium on Discrete Algorithms (SODA), pp. 287–296 (2000)
18. Berry, V., Gascuel, O.: Inferring evolutionary trees with strong combinatorial evidence. Theoret. Comput. Sci. **240**(2), 271–298 (2000). https://doi.org/10.1016/S0304-3975(99)00235-2. http://www.sciencedirect.com/science/article/pii/S0304397599002352
19. Bininda-Emonds, O. (ed.): Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. Kluwer Academic Publishers, Dordrecht (2004)
20. Bininda-Emonds, O.R.P.: MRP supertree construction in the consensus setting. In: Bioconsensus. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 61, pp. 231–242. American Mathematical Society-DIMACS, Providence, Rhode Island (2003)
21. Bininda-Emonds, O.R.P., Gittleman, J.L., Purvis, A.: Building large trees by combining phylogenetic information: a complete phylogeny of the extant Carnivora (Mammalia). Biol. Rev. Camb. Philos. Soc. **74**, 143–175 (1999)
22. Bininda-Emonds, O.R.P., Gittleman, J.L., Steel, M.A.: The (super)tree of life: procedures, problems, and prospects. Annu. Rev. Ecol. Syst. **33**, 265–289 (2002)
23. Böcker, S., Bryant, D., Dress, A.W., Steel, M.A.: Algorithmic aspects of tree amalgamation. J. Algorithms **37**(2), 522–537 (2000)
24. Bordewich, M., Mihaescu, R.: Accuracy guarantees for phylogeny reconstruction algorithms based on balanced minimum evolution. In: Moulton, V.. Singh, M. (eds.) Proceedings of the 2010 Workshop on Algorithms for Bioinformatics, pp. 250–261. Springer, Berlin, Heidelberg (2010)
25. Brinkmeyer, M., Griebel, T., Böcker, S.: Polynomial supertree methods revisited. Adv. Bioinform. **2011** (2011)
26. Bryant, D., Steel, M.: Constructing optimal trees from quartets. J. Algorithms **38**(1), 237–259 (2001)
27. Bryant, D., Steel, M.: Computing the distribution of a tree metric. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(3), 420–426 (2009)
28. Buneman, P.: The recovery of trees from measures of dissimilarity. In: Hodson, F., Kendall, D., Tautu, P. (eds.) Mathematics in the Archaeological and Historical Sciences, pp. 387–395. Edinburgh University Press, Edinburgh, Scotland (1971)
29. Chaudhary, R.: MulRF: a software package for phylogenetic analysis using multi-copy gene trees. Bioinformatics **31**, 432–433 (2015)
30. Chaudhary, R., Bansal, M.S., Wehe, A., Fernández-Baca, D., Eulenstein, O.: iGTP: a software package for large-scale gene tree parsimony analysis. BMC Bioinform. **11**, 574 (2010)
31. Chaudhary, R., Burleigh, J.G., Fernández-Baca, D.: Fast local search for unrooted Robinson-Foulds supertrees. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**, 1004–1013 (2012)
32. Chen, D., Diao, L., Eulenstein, O., Fernández-Baca, D., Sanderson, M.: Flipping: a supertree construction method. In: Bioconsensus. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 61, pp. 135–160. American Mathematical Society-DIMACS, Providence, Rhode Island (2003)
33. Chen, D., Eulenstein, O., Fernández-Baca, D., Burleigh, J.: Improved heuristics for minimum-flip supertree construction. Evol. Bioinform. **2**, 401–410 (2006)
34. Chen, D., Eulenstein, O., Fernández-Baca, D., Sanderson, M.: Minimum-flip supertrees: complexity and algorithms. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**, 165–173 (2006)
35. Chernomor, O., von Haeseler, A., Minh, B.Q.: Terrace aware data structure for phylogenomic inference from supermatrices. Syst. Biol. **65**(6), 997–1008 (2016)
36. Christensen, S., Molloy, E.K., Vachaspati, P., Warnow, T.: OCTAL: Optimal Completion of gene trees in polynomial time. Algorithms Mol. Biol. **13**(1), 6 (2018). https://doi.org/10.1186/s13015-018-0124-5
37. Cotton, J., Wilkinson, M.: Majority rule supertrees. Syst. Biol. **56**(3), 445–452 (2007)

38. Cotton, J., Wilkinson, M.: Supertrees join the mainstream of phylogenetics. Trends Ecol. Evol. **24**, 1–3 (2009)
39. Creevey, C., McInerney, J.: Trees from trees: construction of phylogenetic supertrees using CLANN. In: Bioinformatics for DNA Sequence Analysis, vol. 537, pp. 139–61. Springer, Clifton, NJ (2009)
40. Criscuolo, A., Berry, V., Douzery, E., Gascuel, O.: SDM: a fast distance-based approach for (super) tree building in phylogenomics. Syst. Biol. **55**, 740–755 (2006)
41. Criscuolo, A., Gascuel, O.: Fast NJ-like algorithms to deal with incomplete distance matrices. BMC Bioinform. **9**(166) (2008)
42. Davies, T., Barraclough, T., Chase, M., Soltis, P., Soltis, D., Savolainen, V.: Darwin's abominable mystery: insights from a supertree of the angiosperms. Proc. Natl. Acad. Sci. **101**, 1904–1909 (2004)
43. Desper, R., Gascuel, O.: Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. Mol. Biol. Evol. **21**(3), 587–598 (2004). http://dx.doi.org/10.1093/molbev/msh049
44. Dobzhansky, T.: Nothing in biology makes sense except in the light of evolution. Am. Biol. Teacher **35**, 125–129 (1973)
45. Edwards, S.: Is a new and general theory of molecular systematics emerging? Evolution **63**(1), 1–19 (2009)
46. Erdös, P., Steel, M., Székely, L., Warnow, T.: A few logs suffice to build (almost) all trees (I). Random Struct. Algorithms **14**, 153–184 (1999)
47. Erdös, P., Steel, M., Székely, L., Warnow, T.: A few logs suffice to build (almost) all trees (II). Theoret. Comput. Sci. **221**, 77–118 (1999)
48. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. J. Comput. Syst. Sci. **69**(3), 485–497 (2004)
49. Fernández, M.H., Vrba, E.S.: A complete estimate of the phylogenetic relationships in ruminantia: a dated species-level supertree of the extant ruminants. Biol. Rev. **80**(2), 269–302 (2005)
50. Fleischauer, M., Böcker, S.: Collecting reliable clades using the greedy strict consensus merger. PeerJ **4**, e2172 (2016)
51. Fleischauer, M., Böcker, S.: Bad clade deletion supertrees: a fast and accurate supertree algorithm. Mol. Biol. Evol. **34**(9), 2408–2421 (2017)
52. Foulds, L.R., Graham, R.L.: The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. **3**(43–49), 299 (1982)
53. Gascuel, O.: BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol. Biol. Evol. **14**, 685–695 (1997)
54. Goloboff, P., Farris, J., Nixon, K.: TNT, a free program for phylogenetic analysis. Cladistics **24**, 1–13 (2008)
55. Gramm, J., Niedermeier, R.: A fixed-parameter algorithm for minimum quartet inconsistency. J. Comput. Syst. Sci. **67**(4), 723–741 (2003)
56. Grappa (genome rearrangements analysis under parsimony and other phylogenetic algorithms). https://www.cs.unm.edu/~moret/GRAPPA/
57. Grotkopp, E., Rejmánek, M., Sanderson, M.J., Rost, T.L.: Evolution of genome size in pines (pinus) and its life-history correlates: supertree analyses. Evolution **58**(8), 1705–1729 (2004)
58. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. **52**, 696–704 (2003) (1063-5157 (Print))
59. Hallett, M., Lagergren, J.: New algorithms for the duplication-loss model. In: Proceedings of the ACM Symposium on Computational Biology RECOMB2000, pp. 138–146. ACM Press, New York (2000)
60. Hillis, D.M., Huelsenbeck, J.P., Cunningham, C.W.: Application and accuracy of molecular phylogenies. Science **264**, 671–677 (1994)
61. Holland, B., Conner, G., Huber, K., Moulton, V.: Imputing supertrees and supernetworks from quartets. Syst. Biol. **56**(1), 57–67 (2007). http://dx.doi.org/10.1080/10635150601167013

62. Huson, D., Nettles, S., Warnow, T.: Disk-covering, a fast converging method for phylogenetic tree reconstruction. J. Comput. Biol. **6**(3), 369–386 (1999)
63. Huson, D., Vawter, L., Warnow, T.: Solving large scale phylogenetic problems using DCM2. In: Proceedings of 7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99), pp. 118–129. AAAI Press (1999)
64. Huson, D.H., Vawter, L., Warnow, T.: Solving large scale phylogenetic problems using DCM2. In: Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology table of contents, pp. 118–129. AAAI Press (1999)
65. Janowitz, M., Lapointe, F.J., McMorris, F., Mirkin, B., Roberts, F. (eds.): Bioconsensus: DIMACS Working Group Meetings on Bioconsensus, 25–26 Oct 2000 and 2–5 Oct 2001, DIMACS Center 61. American Mathematical Society (2003)
66. Jarvis, E., Mirarab, S., Aberer, A.J., Li, B., Houde, P., Li, C., Ho, S., Faircloth, B.C., Nabholz, B., Howard, J.T., Suh, A., Weber, C.C., da Fonseca, R.R., Li, J., Zhang, F., Li, H., Zhou, L., Narula, N., Liu, L., Ganapathy, G., Boussau, B., Bayzid, M.S., Zavidovych, V., Subramanian, S., Gabaldón, T., Capella-Gutiérrez, S., Huerta-Cepas, J., Rekepalli, B., Munch, K., Schierup, M., Lindow, B., Warren, W.C., Ray, D., Green, R.E., Bruford, M.W., Zhan, X., Dixon, A., Li, S., Li, N., Huang, Y., Derryberry, E.P., Bertelsen, M.F., Sheldon, F.H., Brumfield, R.T., Mello, C.V., Lovell, P.V., Wirthlin, M., Schneider, M.P.C., Prosdocimi, F., Samaniego, J.A., Velazquez, A.M.V., Alfaro-Núnez, A., Campos, P.F., Petersen, B., Sicheritz-Ponten, T., Pas, A., Bailey, T., Scofield, P., Bunce, M., Lambert, D.M., Zhou, Q., Perelman, P., Driskell, A.C., Shapiro, B., Xiong, Z., Zeng, Y., Liu, S., Li, Z., Liu, B., Wu, K., Xiao, J., Yinqi, X., Zheng, Q., Zhang, Y., Yang, H., Wang, J., Smeds, L., Rheindt, F.E., Braun, M., Fjeldsa, J., Orlando, L., Barker, F.K., Jonsson, K.A., Johnson, W., Koepfli, K.P., O'Brien, S., Haussler, D., Ryder, O.A., Rahbek, C., Willerslev, E., Graves, G.R., Glenn, T.C., McCormack, J., Burt, D., Ellegren, H., Alstrom, P., Edwards, S.V., Stamatakis, A., Mindell, D.P., Cracraft, J., Braun, E.L., Warnow, T., Jun, W., Gilbert, M.T.P., Zhang, G.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science **346**(6215), 1320–1331 (2014)
67. Jewett, E., Rosenberg, N.A.: iGLASS: an improvement to the GLASS method for estimating species trees from gene trees. J. Comput. Biol. **19**(3), 293–315 (2012)
68. Jiang, T., Kearney, P., Li, M.: A polynomial-time approximation scheme for inferring evolutionary trees from quartet topologies and its applications. SIAM J. Comput. **30**(6), 1924–1961 (2001)
69. Jones, K.E., Purvis, A., MacLarnon, A., Bininda-Emonds, O.R.P., Simmons, N.B.: A phylogenetic supertree of the bats (Mammalia: Chiroptera). Biol. Rev. Camb. Philos. Soc. **77**, 223–259 (2002)
70. Jonsson, K.A., Fjeldsa, J.: A phylogenetic supertree of oscine passerine birds (Aves: Passeri). Zoologica Scripta **35**, 149–186 (2006)
71. Kettleborough, G., Dicks, J., Roberts, I.N., Huber, K.T.: Reconstructing (super) trees from data sets with missing distances: not all is lost. Mol. Biol. Evol. **32**(6), 1628–1642 (2015)
72. Kupczok, A.: Split-based computation of majority rule supertrees. BMC Evol. Biol. **11**, (2011)
73. Lacey, M., Chang, J.: A signal-to-noise analysis of phylogeny estimation by neighbor-joining: insufficiency of polynomial length sequences. Math. Biosci. **199**(2), 188–215 (2006)
74. Lafond, M., Scornavacca, C.: On the Weighted Quartet Consensus Problem (2016). arXiv:1610.00505
75. Lapointe, F.J., Cucumel, G.: The average consensus procedure: combination of weighted trees containing identical or overlapping sets of taxa. Syst. Biol. **46**(2), 306–312 (1997)
76. Larget, B., Kotha, S., Dewey, C., Ané, C.: BUCKy: gene tree/species tree reconciliation with the Bayesian concordance analysis. Bioinformatics **26**(22), 2910–2911 (2010)
77. Lechner, M., Hernandez-Rosales, M., Doerr, D., Wieseke, N., Thévenin, A., Stoye, J., Hartmann, R., Prohaska, S., Stadler, P.: Orthology detection combining clustering and synteny for very large datasets. PLoS ONE **9**(8), e105,015 (2014). https://doi.org/10.1371/journal.pone.0105015
78. Lefort, V., Desper, R., Gascuel, O.: FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. Mol. Biol. Evol. **32**(10), 2798–2800 (2015). https://doi.org/10.1093/molbev/msv150

79. Liu, K., Linder, C., Warnow, T.: RAxML and FastTree: comparing two methods for large-scale maximum likelihood phylogeny estimation. PLoS ONE **6**(11), e27,731 (2012)
80. Liu, K., Raghavan, S., Nelesen, S., Linder, C.R., Warnow, T.: Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science **324**(5934), 1561–1564 (2009)
81. Liu, L., Yu, L.: Estimating species trees from unrooted gene trees. Syst. Biol. **60**(5), 661–667 (2011)
82. Liu, L., Yu, L., Edwards, S.: A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evol. Biol. **10**, 302 (2010)
83. Lopez, P., Casane, D., Philippe, H.: Heterotachy, an important process of protein evolution. Mol. Biol. Evol. **19**, 1–7 (2002)
84. Maddison, W.P.: Gene trees in species trees. Syst. Biol. **46**, 523–536 (1997)
85. Martins, L., Mallo, D., Posada, D.: A Bayesian supertree model for genome-wide species tree reconstruction. Syst. Biol. **65**, 397–416 (2016)
86. McMorris, F.: Axioms for consensus functions on undirected phylogenetic trees. Math. Biosci. **74**, 17–21 (1985)
87. Mihaescu, R., Levy, D., Pachter, L.: Why neighbor-joining works. Algorithmica **54**(1), 1–24 (2009)
88. Mirarab, S., Reaz, R., Bayzid, M.S., Zimmermann, T., Swenson, M., Warnow, T.: ASTRAL: Accurate Species TRee ALgorithm. Bioinformatics **30**(17), i541–i548 (2014)
89. Mirarab, S., Warnow, T.: ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. Bioinformatics **31**(12), i44–i52 (2015)
90. Molloy, E.K., Warnow, T.: NJMerge: a generic technique for scaling phylogeny estimation methods and its application to species trees. In: Blanchette, M., Ouangraoua, A. (eds.) Comparative Genomics, pp. 260–276. Springer International Publishing, Cham (2018)
91. Moret, B.M.E., Wang, L.S., Warnow, T.: New software for computational phylogenetics. IEEE Comput.: Spec. Issue Bioinform. **35**(7), 55–64 (2002)
92. Mossel, E., Roch, S.: Incomplete lineage sorting: consistent phylogeny estimation from multiple loci. IEEE Trans. Comput. Biol. Bioinform. **7**(1), 166–171 (2011)
93. Nelesen, S., Liu, K., Wang, L.S., Linder, C.R., Warnow, T.: DACTAL: divide-and-conquer trees (almost) without alignments. Bioinformatics **28**, i274–i282 (2012)
94. Neves, D., Sobral, J.: Parallel SuperFine—a tool for fast and accurate supertree estimation: features and limitations. Future Gener. Comput. Syst. **67**, 441–454 (2017)
95. Neves, D., Warnow, T., Sobral, J., Pingali, K.: Parallelizing SuperFine. In: 27th Symposium on Applied Computing (ACM-SAC), Bioinformatics, pp. 1361–1367. ACM (2012). https://doi.org/10.1145/2231936.2231992
96. Neves, D.T., Sobral, J.L.: Parallel SuperFine—a tool for fast and accurate supertree estimation: Features and limitations. Future Gener. Comput. Syst. **67**, 441–454 (2017). https://doi.org/10.1016/j.future.2016.04.004. http://www.sciencedirect.com/science/article/pii/S0167739X16300814
97. Nguyen, L.T., Schmidt, H., von Haeseler, A., Minh, B.: IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. Mol. Biol. Evol. **32**(1), 268–274 (2015). https://doi.org/10.1093/molbev/msu300
98. Nguyen, N., Mirarab, S., Kumar, K., Warnow, T.: Ultra-large alignments using phylogeny aware profiles. Genome Biol. **16**(124) (2015). https://doi.org/10.1186/s13059-015-0688-z. A preliminary version appeared in the Proceedings RECOMB 2015
99. Nguyen, N., Mirarab, S., Warnow, T.: MRL and SuperFine+MRL: new supertree methods. J. Algorithms Mol. Biol. **7**(3) (2012)
100. Nute, M., Warnow, T.: Scaling statistical multiple sequence alignment to large datasets. BMC Genomics **17**(10), 764 (2016). https://doi.org/10.1186/s12864-016-3101-8
101. de Oliveira Martins, L., Posada, D.: Species tree estimation from genome-wide data with Guenomu. In: Bioinformatics, pp. 461–478. Springer (2017)
102. Pardi, F., Guillemot, S., Gascuel, O.: Combinatorics of distance-based tree inference. Proc. Natl. Acad. Sci. (USA) **109**(41), 16443–16448 (2012)

103. Piaggio-Talice, R., Burleigh, J.G., Eulenstein, O.: Quartet supertrees. In: Bininda-Emonds, O.R.P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal The Tree of Life, pp. 173–191. Kluwer Academic, Dordrecht, The Netherlands (2004)
104. Pisani, D.: A genus-level supertree of the Dinosauria. Proc. R. Soc. Lond. B: Biol. Sci. **269**, 915–921 (2002)
105. Pisani, D., Cotton, J.A., McInerney, J.O.: Supertrees disentangle the chimeric origin of eukaryotic genomes. Mol. Biol. Evol. (2007)
106. Popescu, A.A., Huber, K.T., Paradis, E.: ape 3.0: New tools for distance-based phylogenetics and evolutionary analysis in R. Bioinformatics **28**(11), 1536–1537 (2012)
107. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree 2—approximately maximum-likelihood trees for large alignments. PLoS ONE **5**(3), e9490 (2010)
108. Ragan, M.A.: Phylogenetic inference based on matrix representation of trees. Mol. Phylogenet. Evol. **1**, 53–58 (1992)
109. Ranwez, V., Berry, V., Criscuolo, A., Fabre, P.H., Guillemot, S., Scornavacca, C., Douzery, E.J.: PhySIC: a veto supertree method with desirable properties. Syst. Biol. **56**(5), 798–817 (2007)
110. Ranwez, V., Criscuolo, A., Douzery, E.J.: SuperTriplets: a triplet-based supertree approach to phylogenomics. Bioinformatics **26**(12), i115–i123 (2010)
111. Ranwez, V., Gascuel, O.: Quartet-based phylogenetic inference: improvements and limits. Mol. Biol. Evol. **18**(6), 1103–1116 (2001)
112. Reaz, R., Bayzid, M., Rahman, M.: Accurate phylogenetic tree reconstruction from quartets: a heuristic approach. PLoS ONE (2014). https://doi.org/10.1371/journal.pone.0104008
113. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Math. Biosci. **53**, 131–147 (1981)
114. Roch, S., Steel, M.: Likelihood-based tree reconstruction on a concatenation of aligned sequence data sets can be statistically inconsistent. Theoret. Popul. Biol. **100**, 56–62 (2015)
115. Rodrigo, A.G.: A comment on Baum's method for combining phylogenetic trees. Taxon **42**(3), 631–636 (1993)
116. Roshan, U., Moret, B.M., Williams, T.L., Warnow, T.: REC-I-DCM3: a fast algorithmic technique for reconstructing large phylogenetic trees. In: Proceedings of 3rd IEEE Computational Systems Bioinformatics Conference CSB '04, LCBB-CONF-2004-002, pp. 98–109. IEEE Press (2004)
117. Roshan, U., Moret, B.M.E., Williams, T.L., Warnow, T.: Performance of supertree methods on various dataset decompositions. In: Bininda-Emonds, O.R.P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal The Tree Of Life, pp. 301–328. Kluwer Academic, Dordrecht, The Netherlands (2004)
118. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
119. Salamin, N., Davies, J.T.: Using supertrees to investigate species richness in grasses and flowering plants. In: Bininda-Emonds, O.R.P. (ed.) Phylogenetic Supertrees: Combining Information to Reveal The Tree Of Life, pp. 461–487. Kluwer Academic, Dordrecht, The Netherlands (2004)
120. Sanderson, M., McMahon, M., Steel, M.: Phylogenomics with incomplete taxon coverage: the limits to inference. BMC Evol. Biol. **10**, 155 (2010)
121. Sanderson, M.J., McMahon, M.M., Stamatakis, A., Zwickl, D.J., Steel, M.: Impacts of terraces on phylogenetic inference. Syst. Biol. **64**(5), 709–726 (2015)
122. Sanderson, M.J., McMahon, M.M., Steel, M.: Terraces in phylogenetic tree space. Science **333**(6041), 448–450 (2011)
123. Semple, C., Steel, M.: A supertree method for rooted trees. Discrete Appl. Math. **105**(1–3), 147–158 (2000). https://doi.org/10.1016/S0166-218X(00)00202-X. http://www.sciencedirect.com/science/article/pii/S0166218X0000202X
124. Sevillya, G., Frenkel, Z., Snir, S.: Triplet MaxCut: a new toolkit for rooted supertree. Methods Ecol. Evol. **7**, 1359–1365 (2016). https://doi.org/10.1111/2041-210X.12606
125. Shigezumi, T.: Robustness of greedy type minimum evolution algorithms. In: Proceedings of International Conference on Computational Science, pp. 815–821. Springer (2006)

126. Sjölander, K., Datta, R., Shen, Y., Shoffner, G.: Ortholog identification in the presence of domain architecture rearrangement. Brief. Bioinform. **12**(5), 413–422 (2011). https://doi.org/10.1093/bib/bbr036. http://bib.oxfordjournals.org/content/12/5/413.abstract
127. Snir, S., Rao, S.: Using max cut to enhance rooted trees consistency. IEEE/ACM Trans. Comput. Biol. Bioinform. 323–333 (2006)
128. Snir, S., Rao, S.: Quartets MaxCut: a divide and conquer quartets algorithm. IEEE/ACM Trans. Comput. Biol. Bioinform. **7**(4), 704–718 (2010)
129. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**, 2688–2690 (2006)
130. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. J. Classif. **9**, 91–116 (1992)
131. Steel, M., Gascuel, O.: Neighbor-joining revealed. Mol. Biol. Evol. **23**(11), 1997–2000 (2006)
132. Steel, M., Rodrigo, A.: Maximum likelihood supertrees. Syst. Biol. **57**(2), 243–250 (2008)
133. Strimmer, K., von Haeseler, A.: Quartet puzzling: a quartet maximim-likelihood method for reconstructing tree topologies. Mol. Biol. Evol. **13**(7), 964–969 (1996)
134. Swenson, M., Suri, R., Linder, C., Warnow, T.: An experimental study of Quartets MaxCut and other supertree methods. Algorithms Mol. Biol. **6**, 7 (2011). PMID: 21504600
135. Swenson, M., Suri, R., Linder, C., Warnow, T.: SuperFine: fast and accurate supertree estimation. Syst. Biol. **61**(2), 214–227 (2012)
136. Swofford, dD.: PAUP*: Phylogenetic Analysis Using Parsimony (*d and Other Methods) Ver. 4. Sinauer Associated, Sunderland, Massachusetts (2002)
137. Szöllősi, G., Rosikiewicz, W., Boussau, B., Tannier, E., Daubin, V.: Efficient exploration of the space of reconciled gene trees. Syst. Biol. (2013). https://doi.org/10.1093/sysbio/syt054. http://sysbio.oxfordjournals.org/content/early/2013/08/06/sysbio.syt054.abstract
138. Szöllősi, G.J., Boussau, B., Abby, S.S., Tannier, E., Daubin, V.: Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. Proc. Natl. Acad. Sci. **109**(43), 17513–17518 (2012). https://doi.org/10.1073/pnas.1202997109
139. Tang, J., Moret, B.: Scaling up accurate phylogenetic reconstruction from gene-order data. Bioinformatics **19 (Suppl. 1)**, i305–i312 (2003). Proceedings of 11th International Conference on Intelligent Systems for Molecular Biology ISMB'03
140. Than, C., Nakhleh, L.: Species tree inference by minimizing deep coalescences. PLoS Comput. Biol. **5**, 31000,501 (2009)
141. Thorley, J., Wilkinson, M.: A view of supertree methods. DIMACS Ser. Discrete Math. Theoret. Comput. Sci. **61**, 185–194 (2003)
142. Vachaspati, P., Warnow, T.: ASTRID: accurate species TRees from internode distances. BMC Genomics **16**(Suppl 10), S3 (2015)
143. Vachaspati, P., Warnow, T.: FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. Bioinformatics (2016). https://doi.org/10.1093/bioinformatics/btw600
144. Vachaspati, P., Warnow, T.: SIESTA: Enhancing searches for optimal supertrees and species trees. BMC Genomics (2018) (to appear)
145. Vachaspati, P., Warnow, T.: SVDquest: Improving SVDquartets species tree estimation using exact optimization within a constrained search space. Mol. Phylogenet. Evol. **124**, 122–136 (2018). https://doi.org/10.1016/j.ympev.2018.03.006. http://www.sciencedirect.com/science/article/pii/S105579031730338X
146. Wang, L.S., Leebens-Mack, J., Wall, P.K., Beckmann, K., DePamphilis, C.W., Warnow, T.: The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**, 1108–1119 (2011)
147. Warnow, T.: Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation. Cambridge University Press, Cambridge UK (2018)
148. Warnow, T., Moret, B.M.E., St. John, K.: Absolute convergence: true trees from short sequences. In: Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA 01), pp. 186–195. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2001)

149. Waterman, M., Smith, T., Beyer, W.: Some biological sequence metrics. Adv. Math. **20**, 367–387 (1976)
150. Waterman, M., Smith, T., Singh, M., Beyer, W.: Additive evolutionary trees. J. Theoret. Biol. **64**, 199–213 (1977)
151. Wehe, A., Bansal, M., Burleigh, J., Eulenstein, O.: DupTree: a program for large-scale phylogenetic analyses using gene tree parsimony. Bioinformatics **24**(13), 1540–1541 (2008). https://doi.org/10.1093/bioinformatics/btn230. http://bioinformatics.oxfordjournals.org/content/24/13/1540.abstract
152. Wheeler, T.: Large-scale neighbor-joining with NINJA. In: Proceedings of Workshop Algorithms in Bioinformatics (WABI), vol. 5724, pp. 375–389 (2009)
153. Wickett, N., Mirarab, S., Nguyen, N., Warnow, T., Carpenter, E., Matasci, N., Ayyampalayam, S., Barker, M., Burleigh, J., Gitzendanner, M., Ruhfel, B.R., Wafula, E., Der, J.P., Graham, S.W., Mathews, S., Melkonian, M., Soltis, D.E., Soltis, P.S., Miles, N.W., Rothfels, C.J., Pokorny, L., Shaw, A.J., DeGironimo, L., Stevenson, D.W., Surek, B., Villarreal, J.C., Roure, B., Philippe, H., dePamphilis, C.W., Chen, T., Deyholos, M.K., Baucom, R.S., Kutchan, T.M., Augustin, M.M., Wang, J., Zhang, Y., Tian, Z., Yan, Z., Wu, X., Sun, X., Wong, G.K.S., Leebens-Mack, J.: Phylotranscriptomic analysis of the origin and early diversification of land plants. Proc. Natl. Acad. Sci. **111**(45), E4859–E4868 (2014)
154. Wilkinson, M., Cotton, J.A., Lapointe, F.J., Pisani, D.: Properties of supertree methods in the consensus setting. Syst. Biol. **56**(2), 330–337 (2007). https://doi.org/10.1080/10635150701245370
155. Willson, S.: Constructing rooted supertrees using distances. Bull. Math. Biol. **66**(6), 1755–1783 (2004)
156. Xin, L., Ma, B., Zhang, K.: A new quartet approach for reconstructing phylogenetic trees: quartet joining method. In: Proceedings. Computing and Combinatorics (COCOON) 2007, Lecture Notes in Computer Science, vol. 4598, pp. 40–50. Springer, Berlin, Heidelberg (2007)
157. Yu, Y., Warnow, T., Nakhleh, L.: Algorithms for MDC-based multi-locus phylogeny inference: beyond rooted binary gene trees on single alleles. J. Comput. Biol. **18**, 1543–1559 (2011). https://doi.org/10.1089/cmb.2011.0174
158. Zhang, C., Sayyari, E., Mirarab, S.: ASTRAL-III: increased scalability and impacts of contracting low support branches. In: Meidanis, J., Nakhleh, L. (eds.) Comparative Genomics, pp. 53–75. Springer International Publishing, Cham (2017)
159. Zhang, Q., Rao, S., Warnow, T.: New absolute fast converging phylogeny estimation methods with improved scalability and accuracy. In: Parida, L., Ukkonen, E. (eds.) 18th International Workshop on Algorithms in Bioinformatics (WABI 2018), pp. 8:1–8:12. LIPICS, Dagsttuhl (2018)

Chapter 7
Taxonomic Supertree Construction with *Incertae sedis* Taxa

Benjamin D. Redelings and Mark T. Holder

Abstract A supertree method is a form of meta-analysis. It combines phylogenetic tree estimates, which typically include only a subset of the species of interest, into a single tree that includes every species found in any input tree. These methods are usually applied to phylogenetic estimates that only have taxonomic labels at their leaves. Taxonomies are another source of information about phylogenetic relationships. They usually convey names for groups of species in addition to species names. If a taxonomy places every species, then its phylogenetic information can be fully expressed as a tree. Such a taxonomy could easily serve as one of the inputs to a supertree method. However, most taxonomies contain *incertae sedis* taxa, which are species or groups of species that have an uncertain placement within the taxonomic hierarchy. Here, we review some principles of building trees from splits and describe a supertree method that is able to handle *incertae sedis* taxa without losing taxon names.

Keywords Supertrees · Phylogenetics · *Incertae sedis* taxa · Taxonomy · Tree of Life

7.1 Introduction

A phylogenetic tree is a depiction of the ancestor–descendant relationships between different species. Most phylogenetic trees are estimated from DNA sequence data or morphological (anatomical) character data. Supertree methods of phylogenetic estimation combine a collection of input trees with different taxon sets into a single tree on the combined taxon set (see Fig. 7.1 for definitions of some of the terminology from the field of biological systematics that used throughout this chapter). These

B. D. Redelings
Duke University, Durham, USA
e-mail: benjamin.redelings@gmail.com

M. T. Holder (✉)
University of Kansas, Lawrence, USA
e-mail: mtholder@ku.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_7

Taxon. A taxon (pluralized *taxa*) is a group of organisms (typically species and groups of species) that is thought to be a natural group for purpose of organizing information about the diversity of life.
Taxonomy. A *taxonomy* (or a *classification*) is a formal system for organizing and assigning naming to taxa. Taxonomies are typically hierarchies with *higher taxa* being groups that contain some lower level taxa. Unlike in the case of a phylogenetic estimates, the higher level groups in taxonomies usually are given names. Taxonomic groups are often assigned *ranks* to indicate their level in the hierarchy. Thus, the ranking *family* $>$ *genus* $>$ *species* conveys the fact that a family would contain at least one genus and each genus would have at least one species in it. Note that *genera* is the plural of genus.
Clade or **monophyletic group.** A clade is a group that consists of an ancestral species and all of its descendant taxa.
Phylogenetic classification. A phylogenetic taxonomy or phylogenetic classification is one in which every higher level group is thought to be a *clade*. The vast majority of modern taxonomies are phylogenetic classifications.

Fig. 7.1 A brief glossary of some relevant terminology from biological systematics

methods are usually applied to leaf-labeled phylogenetic trees. Here, we describe a supertree approach that overcomes issues associated with using a taxonomy as one of the inputs to a supertree method.

The motivation for this work was the Open Tree of Life project [6], which seeks to build a comprehensive supertree (see [10]) covering all of life. The project combines information in published phylogenies with a comprehensive taxonomy that supplies taxon names. The supertree method has used the Open Tree Taxonomy (OTT hereafter, see [11]) as an input, but, in principle, it could make use of other taxonomies. The taxonomy is a crucial input for several reasons: it covers a wide range of species, its list of names allows for alignment of different phylogenetic estimates to each other, and it provides names for clades. Only a small proportion of known species have been included in a phylogenetic analysis; thus, a taxonomy is important for achieving comprehensive coverage of known taxa. Phylogenetic estimates collected from the published literature often use different names for the same species. Lists of synonyms in OTT (and other taxonomies) allow data curators to "align" input phylogenies to the taxonomy (see [8], for discussion of the curation process that the Open Tree of Life project uses to align phylogenetic estimates to OTT). This alignment is important for recognizing when two different estimates refer to the same taxon. Biologists are familiar with names for "higher" taxa (taxa above the species rank). Thus, naming clades in the supertree makes the tree easier to navigate and use.

Some taxonomies indicate which taxa are thought to be the result of hybridization, but the supertree approach described here does not consider any hybridization. Thus, the taxonomic hierarchy can be converted to a tree. While not all taxonomists believe that named taxa should correspond to clades (see, for example, [7]), the principles of phylogenetic classification are used by a large majority of practicing taxonomists. Thus, we have chosen to treat the taxonomic tree that mirrors the hierarchy of the taxonomy as an estimate of phylogenetic relationships. We refer to labels of taxonomy nodes as "taxon names", and assume that taxon names and taxonomy nodes have a one-to-one correspondence. Note that the taxonomy may also contain out-degree-1 nodes. These nodes are referred to as monotypic taxa. They contain a single child of lower taxonomic rank, so they do not reveal new information about the phylogenetic relationships of the species. They are only useful to the extent that having names for each expected taxonomic rank is useful.

The supertree method described by Redelings and Holder [10] is able to use a set of phylogenetic estimates and a taxonomy to produce a supertree with clades named according to the taxonomy. However, the method was not able to accommodate the fact that taxonomies frequently contain nodes with uncertain placement. These nodes are often labeled "*incertae sedis*", which means "uncertain seat" in Latin. The common interpretation of such taxa is that they may not be moved further toward the root, but may be moved into their sibling taxa. For example, a genus with a sibling that is a family may be annotated as "*incertae familia*", indicating that it is unclear which family the genus should be placed in. *Incertae sedis* taxa frequently occur when a specimen is identified down to a given taxonomic level, but no further. Extinct taxa are often *incertae sedis*. OTT is constructed from several source taxonomies, and these sources include various ways of indicating that the position of a taxon within the hierarchy is uncertain. OTT uses a series of "flags" to annotate these taxa. The supertree of Redelings and Holder [10] simply pruned these taxa from the taxonomy and input trees. Thus, the final supertree lacked taxa which were *incertae sedis* or the descendants of such taxa.

We describe a supertree method that can resolve taxonomic uncertainty by using published phylogenies to place *incertae sedis* taxa. Developing this method required deciding on a set of operational semantics to be used when interpreting the *incertae sedis* annotation. These semantics affect the interpretation of the phylogenetic statements being made by the input taxonomy and the rules for applying taxonomic names to the final supertree. In addition to discussing how to interpret the *incertae sedis* annotation, we describe the algorithmic changes to the supertree pipeline that were required to adequately represent the taxonomic uncertainty. The approach described here allows us to include thousands of new taxa in our supertree analysis that were previously filtered out. It will also enable the Open Tree of Life project to include extinct taxa, since many of these taxa are *incertae sedis*. This makes substantial progress toward our goal of comprehensive inclusion of known taxa into the Open Tree of Life summary tree.

7.1.1 Background

7.1.1.1 Core of Previous Algorithm

The supertree algorithm of Redelings and Holder [10] takes as input a ranked list $\mathcal{T} = \{T_1, \ldots, T_n\}$ of leaf-labeled rooted input phylogenies and a single rooted taxonomy tree $\mathbb{T}_p$ that is ranked below all of the input trees. As discussed in Redelings and Holder [10], the preferred tree would display as many of the highly ranked input splits as possible. At its core, the algorithm acts by first producing a ranked list of rooted splits: $\mathcal{S} = S(T_1) + S(T_2) + \cdots + S(T_n) + S(\mathbb{T})$ where $S(T_i)$ denotes a list of nontrivial rooted splits created from a post-order traversal of tree i and "+" denotes concatenation.

A split is a fundamental concept in phylogenetic theory (see, for further background, context, and applications [5]). A split is a bipartition of the set of leaf labels for the phylogenetic problem of interest. There is a bijection between every edge in a phylogenetic tree and a split. For any edge, consider removing the edge from the graph. This will convert the tree into a graph that is no longer connected, but instead consists of two trees. If we treat the leaf label set of each of these two trees as a separate subset in a partition of the total leaf label set, then the resulting bipartition is the split that corresponds to the edge. By "rooted split" we simply mean a split that is augmented by introducing a label for the root of the tree and treating that label as if it were part of the leaf set.

The supertree of Redelings and Holder [10] produces the supertree from $\mathcal{S}$ in the following greedy manner. Imagine initializing C to be the empty set, then iteratively consider each split in the list $\mathcal{S}$ and insert it into C if the resulting set remains jointly compatible. A set of rooted splits is jointly compatible if there exists a tree that can display every rooted split in the set. Aho et al. [2] developed an algorithm called BUILD that can check for joint compatibility of a set of splits and a tree that displays the splits. In the pipeline of Redelings and Holder [10], the BUILD algorithm is used to check for split compatibility and to construct many aspects of a compatible tree from the final set C.

7.1.1.2 Optimizations to Supertree Algorithm

Because of the large number (over 2.6 million) of tips in the full tree, the core supertree algorithm of Redelings and Holder [10] is not applied directly to the input trees and the taxonomy. Instead, our supertree pipeline first applies two primary approximations, which we briefly summarize here. First, taxa that do not occur in any phylogenetic estimates are pruned during preprocessing, and a supertree is constructed on this reduced subset of taxa. Pruned taxa are reattached after the supertree on the smaller taxon set is constructed. This optimization is not crucial to the current work, but it does cause some groups to attach more tipward than they would if the entire tree were constructed using BUILD algorithm mentioned above.

Second, after taxa are pruned, the resulting supertree problem is decomposed into a set of independent subproblems. The decomposition partitions the list of taxonomy splits $S(\mathbb{T}_p)$ into two lists: $\mathcal{Z}(\mathbb{T}_p)$, the list of splits from the taxonomy tree which are compatible with every rooted split in the phylogenetic inputs, and $\mathcal{Y}(\mathbb{T}_p)$, the list of taxonomic splits that conflict with at least one phylogenetic input split. Conceptually the approximate algorithm works by greedily adding all compatible splits in altered ranked list of splits: $\mathcal{S}' = \mathcal{Z}(\mathbb{T}_p) + S(T_1) + S(T_2) + \cdots + S(T_n) + \mathcal{Y}(\mathbb{T}_p)$.

In practice, the decomposition allows the compatibility of splits to be checked on smaller, relabeled trees. The core algorithm described above is then applied to these smaller relabeled trees. This can be done much more efficiently than if compatibility was checked using the BUILD algorithm [2] on the full leaf set. Thus, after pruning taxa that do not occur in phylogeny estimates, the supertree algorithm divides the supertree problem into independent pieces by bisecting trees at the uncontested taxa which correspond to Z, then solves the supertree problem on these pieces, and finally glues the pieces back together to produce the full tree.

7.1.1.3 Formal Description of the Taxonomy

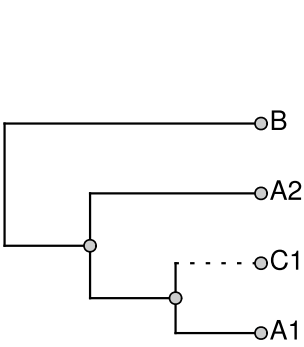
The taxonomy tree, $\mathbb{T}_p$, is derived from OTT. As mentioned above, OTT not only contains a taxonomic hierarchy but also contains taxonomic labels and annotations (referred to as "flags") on the taxa to carry extra information. The entire hierarchy encoded by OTT is a complete taxonomic tree, $\mathbb{T}^*$. Nodes in $\mathbb{T}^*$ have a one-to-one correspondence with a set of labels $\mathcal{N}$ that are called taxon names. We will decompose $\mathcal{N}$ into the taxon names for leaf taxa $\mathcal{L}$ and higher taxa $\mathcal{H}$. We will use $\mathcal{U}$ to refer to the subset ($\mathcal{U} \subseteq \mathcal{N}$) of taxa that are annotated with the *incertae sedis* property. Other annotations identify dubious taxa and taxonomic names that are unwanted artifacts of the process of building OTT. Thus, the taxonomic tree which is used as an input to the supertree algorithm is pruned version of $\mathbb{T}^*$. For our purposes, the relevant information from the taxonomy consists of the taxonomy tree $\mathbb{T}^*$, the labels $\mathcal{N}$, and set $\mathcal{U}$ of *incertae sedis* taxa. In previous work, taxa in $\mathcal{U}$ and all of their descendants were all pruned from the taxonomic hierarchy when $\mathbb{T}_p$ is extracted from OTT. In the present work, we consider supertree approaches that operate on a taxonomic tree $\mathbb{T}$, which is produced from $\mathbb{T}^*$ but does not prune *incertae sedis* taxa.

Throughout we assume that leaf labels on input phylogenetic trees are only taken from the leaf set of the taxonomic tree that is being used and that each label occurs at most once in each input tree. Redelings and Holder [10] describe a method called "exemplification" that ensures this is the case. Thus, pruning of the *incertae sedis* taxa from the taxonomy was also accompanied by pruning those taxa and all of their descendants from the input phylogenetic trees.

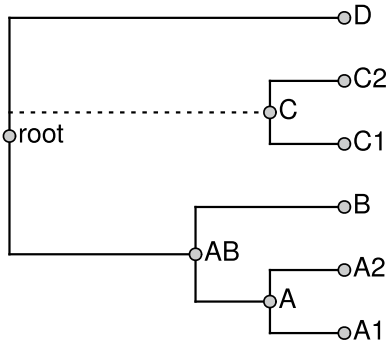
7.1.2 *A Naive Semantics: Consequences of Ignoring the Annotation*

The most straightforward approach for increasing the taxonomic completeness of the supertree is simply to no longer prune taxa from $\mathbb{T}$ because they are flagged as *incertae sedis*, and then run the rest of the pipeline of Redelings and Holder [10] unaltered. Before we propose a semantics of *incertae sedis*, it is worth pointing out what this naive semantics is undesirable.

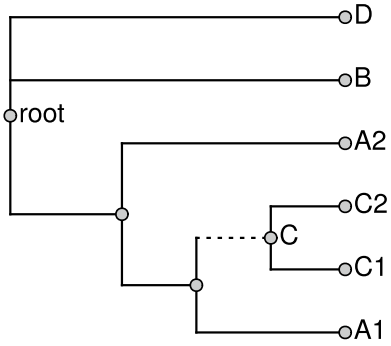
This approach incorrectly treats the intrusion of an *incertae sedis* taxon into a sibling taxon as a conflicting with monophyly of the sibling taxon. Consider, for example, the trees shown in Fig. 7.2. In this figure (and throughout this chapter)



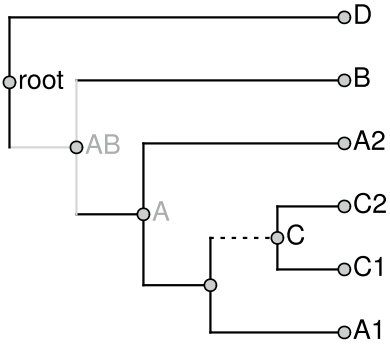
(a) Input tree #1



(b) Taxonomy



(c) naive Synthesis Tree (meaning of *incertae sedis* not handled correctly)



(d) Synthesis Tree (*incertae sedis* aware)

Fig. 7.2 Handling *incertae sedis* taxa recovers additional edges and taxon names. Panel **a** shows the only phylogenetic input to the supertree algorithm; **b** shows the taxonomy tree; the edge subtending the *incertae sedis* clade *C* is shown using a dashed line. The tree in **c** would be the output of the supertree algorithm if *incertae sedis* annotations are simply ignored. The tree shown in **d** is the supertree produced if *incertae sedis* annotations are respected. The tree in (**d**) recovers taxon names *AB* and *A*, as well as the edge to clade *AB* that groups *A* and *B*. Taxon names and edges that are conditional on handling *incertae sedis* are shaded gray

dashed lines are used for edges that connect an *incertae sedis* taxon to its parent. Note that if we were to use the naive semantics, the placement of the *incertae sedis* taxon *C* within *A* and *AB* is considered to contradict the monophyly of *A* and *AB* and thus to conflict with those taxa. This has three main effects. These effects are illustrated in the difference between the synthesis tree in Fig. 7.2c, which ignores *incertae sedis* annotations, and the synthesis tree in Fig. 7.2d, which respects them.

First, the naive approach results in a less-resolved tree. While input trees are free to place *incertae sedis* taxa within a sibling, the naive approach interprets this placement as conflict with the sibling. Since the taxonomy is ranked below all input trees, conflict is always resolved in favor of the input trees, and so the sibling in the taxonomy is removed from the synthesis tree. For example, in the synthesis tree in Fig. 7.2c, the placement of taxon *C* in group *AB* is interpreted as a rejection of the monophyly of *AB*. The edge leading to *AB* is therefore not incorporated into the synthesis tree. In contrast, the synthesis tree in Fig. 7.2d that does not reject the monophyly of *AB* retains the grouping of *A* and *B*.

Second, the naive approach results in the removal of clade names for all clades that have an *incertae sedis* taxon placed within them. For example, in Fig. 7.2c the names for the clades *A* and *AB* are lost.

Third, the naive approach cannot use information from phylogenies to resolve the location of uncertain taxa. For example, the synthesis tree in Fig. 7.2c has lost the label for *A*, and so does not allow us to note that *C* has been placed within *A*. In contrast, the synthesis tree in Fig. 7.2d retains the label for *A*, which allows us to resolve taxonomic uncertainty by inferring that *C* has been placed within *A*.

7.2 Semantics of *Incertae sedis* Taxa

7.2.1 Goals for an Incertae sedis *Semantics*

In order to incorporate *incertae sedis* taxa into a supertree analysis, we must arrive at an operational definition of the meaning of the *incertae sedis* label. The core meaning of the phrase is that the annotated taxon may actually be a member of one of the taxa that appear as siblings to it in the hierarchical representation of the taxonomy. In the most expansive interpretation, the author of the taxonomy is stating that the *incertae sedis* taxon could be placed anywhere inside one of the sibling taxa or their descendants. However, frequently only a subset of the possible placement points would be viewed as plausible.

We seek a semantics for supertrees with *incertae sedis* taxa that satisfy the following properties:

1. An *incertae sedis* node may intrude into its siblings and their descendants without this intrusion being viewed as conflicting with the existence of the sibling taxa.
2. An *incertae sedis* node has an unrestricted range within the descendants of its sibling taxa.

3. The semantics of an *incertae sedis* node does not depend on additional information such as taxon ranks, but only on the taxonomy tree.
4. The semantics is based on deriving a rooted (partial) split for each branch of the taxonomy.

7.2.2 Terminology for Rooted Splits

The rooted split associated with taxonomy node n serves to determine both when n is in conflict with an input tree, and where to place the taxon name for clade n on the synthesis tree. As is typically done, we define rooted splits for each node by noting that removing the edge connecting the node to its parent would produce two trees. The leaf label sets of these trees partition the full leaf set $\mathcal{L}$ into two groups: the include set $\mathcal{I}(n)$ which does not contain the root, and the exclude set $\mathcal{E}(n)$ which does contain the root. Such a split may be written

$$\mathcal{I}(n)| \bullet \mathcal{E}(n),$$

where the $\bullet$ indicates the root. The include set is the set of labels for leaves that are descendants of the node. The exclude set is the set of labels for leaves that are not descendants of that node. The "include set" and "exclude set" of a "rooted split" used here are equivalent to the "inside set" and "outside set" of a "n-taxon statement" (*sensu* [12]).

If no taxa are *incertae sedis*, then the exclude set for a node is just the total tip set minus the include set for the node: $\mathcal{E}(n) = \mathcal{L} - \mathcal{I}(n)$. For any two rooted splits $A = A_1| \bullet A_2$ and $B = B_1| \bullet B_2$, we say that A *displays* B if $B_1 \subseteq A_1$ and $B_2 \subseteq A_2$.

Note that when a taxon rooted at node n is not marked as *incertae sedis*, then the leaf labels that are descendants of n will be in the *include* set for n and for all ancestors of n, but will be in the *exclude* set for all other nodes.

7.2.2.1 Equivalence of a Taxonomy Tree and a List of Rooted Splits

Note that, for the purposes of the supertree method, all of the information from the taxonomic tree is contained in the set of rooted splits. Thus, one could imagine replacing that taxonomy, $\mathbb{T}_p$, with a set of rooted splits $\mathcal{R}(\mathbb{T}_p)$ such that each edge in $\mathbb{T}_p$ is encoded in one split in $\mathcal{R}(\mathbb{T}_p)$. If the order of trees is set to be: $\mathcal{R}(\mathbb{T}_p) = \mathcal{Z}(\mathbb{T}_p) + \mathcal{Y}(\mathbb{T}_p)$, then using $\mathcal{R}(\mathbb{T}_p)$ in place of $\mathbb{T}_p$ in the supertree algorithm would produce the same output.

7.2.2.2 Equivalence of Rooted Split Definitions and Phylogenetic Definitions of Names

In terms of phylogenetic nomenclature, the supertree pipeline acts as if the taxonomy defines a "conditional *node-based* name" (see [4]) for each higher taxon. So for the name labeling node n in the taxonomic tree, the operational definition would be: "the clade defined by the most recent common ancestor (MRCA) of $\mathcal{I}(n)$ as long as that MRCA is not an ancestor of any member of $\mathcal{E}(n)$." OTT is not based on explicit phylogenetic name definitions. In many contexts, similar behavior would result from the use of a *branch-based* definition for n ("the clade defined by all of the taxa more closely related to all members of $\mathcal{I}(n)$ than to any member of $\mathcal{E}(n)$, if such a clade exists") during the tree construction. However, a case where node-based and branch-based naming procedures would lead to different outcomes is shown in Fig. 7.4b and discussed in detail below.

7.2.2.3 Equivalence of Rooted Split Definitions with Character Encodings

Representing a split in a tree by a character is the basis of one of the oldest supertree methods: matrix representation with parsimony (MRP, see [3, 9]). In the rooted form of such an approach, character state 1 is used for tips that are descendants of an internal node in an input tree, and 0 is used for other tips and for the root. Partial splits, such as those caused by *incertae sedis* annotations, are naturally encoded by encoding missing labels in a split as missing data in the corresponding character (typically denoted with the "?" symbol in a matrix).

A rooted split is equivalent to the corresponding character in a matrix representation in the sense that each can be treated as a condition that each tree may or may not satisfy. A tree satisfies a split if there a branch which divides include and exclude group when cut. A tree satisfies the corresponding character in a matrix representation if the character has a parsimony score of 1 on the tree. This equivalence is illustrated in Fig. 7.3. Thus, the taxonomy shown in Fig. 7.2b could be represented in terms of the splits in Fig. 7.3a or the characters in Fig. 7.3b. In the naive treatment of ignoring the *incertae sedis* annotation, the "?" characters in the row for taxa $C1$ and $C2$ would be replaced with 0's. Equivalently, the naive interpretation would add $C1$ and $C2$ to the exclude set for the A and AB splits.

7.2.3 *Split-Based Semantics for* Incertae sedis *Taxa*

An *incertae sedis* taxon n differs from a normal taxon because it can be placed as a descendant of some set of taxa without invalidating these taxa. We will use $\mathcal{P}_n$ to denote this set of taxa which are *potential* ancestors for *incertae sedis* taxon n. Note that this is distinct from the taxa that are ancestral to n in the taxonomy; those are

$$C = C1\,C2 \quad | \bullet A1\,A2\,B\,D$$
$$A = A1\,A2 \quad | \bullet B\,D$$
$$AB = A1\,A2\,B \,| \bullet D$$

(a) Split representation

	Character/clade label		
leaves	C	AB	A
$A1$	0	1	1
$A2$	0	1	1
B	0	1	0
$C1$	1	?	?
$C2$	1	?	?
D	0	0	0
(root)	0	0	0

(b) Matrix representation

Fig. 7.3 Rooted splits have an equivalent matrix encoding. **a** Rooted splits for an *incertae sedis* interpretation of the taxonomy shown in Fig. 7.2b. **b** The equivalent matrix representation. The root is coded as a special leaf. Split C is a full split, because it mentions all labels. Splits A and AB are partial splits, and therefore have missing labels in the left panel, and "?" in the right panel

taxa which must contain all of the descendants of n. The effect of the *incertae sedis* annotation of taxon n is to simply remove $\mathcal{I}(n)$ from the exclude set for each taxon in $\mathcal{P}_n$. The include sets of these splits remain unchanged, but the exclude sets may be reduced, allowing *incertae sedis* taxa to intrude. Such a split does not mention the complete set of tip labels $\mathcal{L}$ and is therefore called a "partial split", as opposed to a "full split" that contains the entirely of $\mathcal{L}$.

When considering *incertae sedis* annotations, the taxonomy is still comprehensive if we include the degenerate grouping for the base node ρ which has $\mathcal{I}(\rho) = \mathcal{L}$ and $\mathcal{E}(\rho) = \emptyset$, but many of the taxonomic nodes will correspond to partial splits. Thus, when interpreted correctly the *incertae sedis* annotations are reducing the amount of information that the taxonomy brings to the supertree problem.

7.2.4 *Unrestricted Range and Ignoring Additional Information*

In principle, an *incertae sedis* taxon n could carry with it additional information that indicates which specific points within the taxonomic tree it could attach. Unfortunately, reliable information about the range of valid attachment points for each *incertae sedis* taxon is lacking in OTT and most other large-scale taxonomies. Constructing a range for each *incertae sedis* taxon relies on detailed readings of character argumentation (which is lacking in OTT) or rank-based information (which is frequently untrustworthy in OTT). Thus, here we pursue an approach of ascribing a meaning to the *incertae sedis* label which attempts to capture the core of the idea

that it articulates, but which can be applied automatically across the taxonomy without additional information.

Specifically, the methods described here only consider cases in which $\mathcal{P}_n$ can be determined from the structure of the tree. Our method takes $\mathcal{P}_n$ to be the set of nodes that are in the taxonomic subtrees rooted at the sibling taxa of *incertae sedis* taxon n. This allows us to encode the exclude set of some node x efficiently by sweeping over the tree from root to tip. If $\mathcal{S}(x)$ is the set of siblings of x and $p(x)$ is the parent of x, then the naive approach of ignoring the *incertae sedis* annotation amounts to setting the exclude set as

$$\mathcal{E}(x) = \mathcal{E}(p(x)) \cup \left\{ \bigcup_{s \in \mathcal{S}(x)} \mathcal{I}(s) \right\} \tag{7.1}$$

with $\mathcal{E}(\rho) = \emptyset$ for the root of the taxonomic tree.

If we treat the *incertae sedis* annotation as an indication that the *incertae sedis* taxon x can be placed in any of its sister taxa, then we let $\mathcal{Q}(x)$ denote the set of siblings of x that are not flagged as being *incertae sedis* and we define the exclude set as

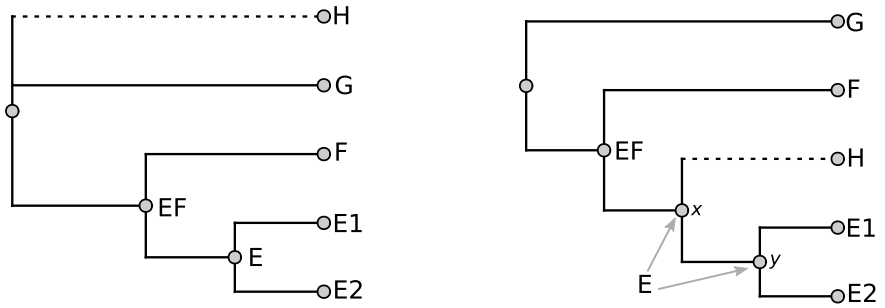
$$\mathcal{E}(x) = \mathcal{E}(p(x)) \cup \left\{ \bigcup_{s \in \mathcal{Q}(x)} \mathcal{I}(s) \right\}. \tag{7.2}$$

We will refer to this interpretation of the *incertae sedis* annotation as "unrestricted intrusion" semantics.

7.2.5 Naming

After constructing a supertree, tip nodes already have names in $\mathcal{L}$. However, we still need to assign higher taxon names to internal supertree nodes based on the taxonomy tree in the problem. Each taxon name n corresponds to a split $S(n) = S(n)_1 | \bullet S(n)_2$ on the corresponding branch of the taxonomy tree. Without *incertae sedis*, such splits are always of the form $S(n)_1 | \bullet \mathcal{L} - S(n)_1$, but with *incertae sedis* taxa $S_2(n)$ may be smaller than $\mathcal{L} - S(n)_1$.

Without *incertae sedis*, each name applies to at most one node, and each node can take at most one name, with the exception of monotypic taxa. Thus, we may simply search the solution tree for a node that has the same cluster $S(n)_1$ and apply the name n to that node.



(a) Taxonomy tree (b) Solution tree

Fig. 7.4 One name can be consistent with multiple nodes

7.2.5.1 One Name, Multiple Nodes

However, in the *incertae sedis* framework, it is possible for one name to apply to multiple nodes. For example, in Fig. 7.4b, the name E can apply to nodes x and y. Here, the name E corresponds to the split $E1\ E2 \mid \bullet\ F\ G$, leaving H out of the definition since H is *incertae sedis*. The two nodes x and y display the splits $E1\ E2 \mid \bullet\ F\ G\ H$ and $E1\ E2\ H \mid \bullet\ F\ G$, respectively, and both of these splits display the split $E1\ E2 \mid \bullet\ F\ G$, so the name E can apply to both x and y. This cannot happen when the rooted splits that correspond to a taxonomic name are generated without *incertae sedis* taxa except at monotypic nodes.

This situation is the result of the fact that it is not clear whether the names in the taxonomic tree should be treated as node-based or branch-based phylogenetic definitions. A node-based definition of E would be "the clade rooted at the MRCA of $E1$ and $E2$ as long as it is not the ancestor of F or G"; this would correspond to node y in the solution depicted in Fig. 7.4b. The branch-based definition would take the form "The clade containing all lineages more closely related to $E1$ and $E2$ than to either F or G, if such a clade exists"; this definition would include node x and the lineage connecting it to its parent. The ambiguity arises because both definitions refer to the same node in the taxonomy, so it is unclear which is a preferable way to define the taxon concept of E.

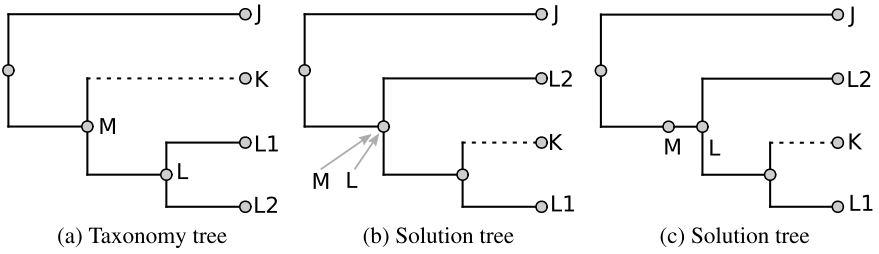
When faced with a choice about where to place a name, our solution is to find the most tipward node where the name can apply and attach the name to this node. This corresponds to using a node-based definition of the taxonomic names. The result of this naming choice is that the algorithm is conservative with respect to when the *incertae sedis* taxa should be placed within another named taxon.

Note that the intrusion of an *incertae sedis* taxon within another taxon does not always lead to ambiguity in which node should be named. For example, H is placed inside taxon EF in Fig. 7.4, but the name EF only applies to one node in the solution tree.

7.2.5.2 One Node, Multiple Names

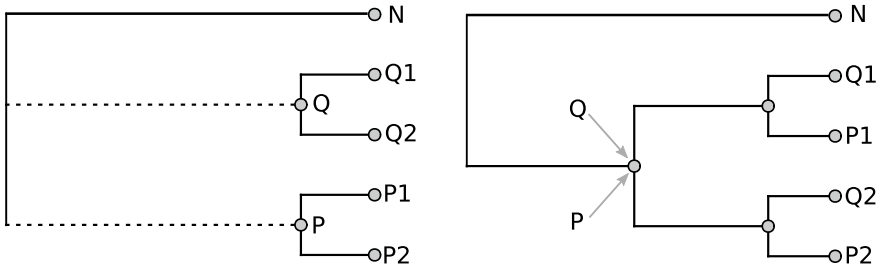
It is also possible for multiple names to apply to a single node. For example, in Fig. 7.5a, the taxon M corresponds to the split $L1\,L2\,K \mid \bullet\, J$, and its child taxon L corresponds to the split $L1\,L2 \mid \bullet\, Y$. The edge leading to M is consistent with the split for L, but the name L is applied to the node with the smallest include group. However, in the solution tree (Fig. 7.5b), there is only node for both names M and L to apply to. This kind of situation arises when a taxon becomes monotypic after its *incertae sedis* children are placed elsewhere. For such cases, we can address this synonymy by (i) selecting a taxon at the node that is ancestral to all other taxa and (ii) moving this taxon to a newly introduced out-degree-1 parent. For example, M is moved to an out-degree-1 parent in Fig. 7.5c. We repeatedly perform this procedure until no taxon at the node is the parent of all taxa.

However, there are cases where this procedure does not result in one name per node. This case can occur, for example, when two *incertae sedis* siblings are "interdigitated", or combined into a single clade in such a way that their members cannot be separated. When the members of the two clades share the same MRCA, as in Fig. 7.6, then the names for both clades apply to the same node. However, in this case, neither name is ancestral to the other, and the previous solution does not apply. In such cases, we treat all applicable names as synonyms for the node.



(a) Taxonomy tree (b) Solution tree (c) Solution tree

Fig. 7.5 Multiple names can apply to a single node



(a) Taxonomy tree (b) Solution tree

Fig. 7.6 Interdigitation of *incertae sedis* siblings. Here, the names P and Q apply to the same node on the solution tree, but neither is ancestral to the other on the taxonomy

7.2.6 Taxonomic Revision

7.2.6.1 Revision of Unbroken *Incertae sedis* Taxa

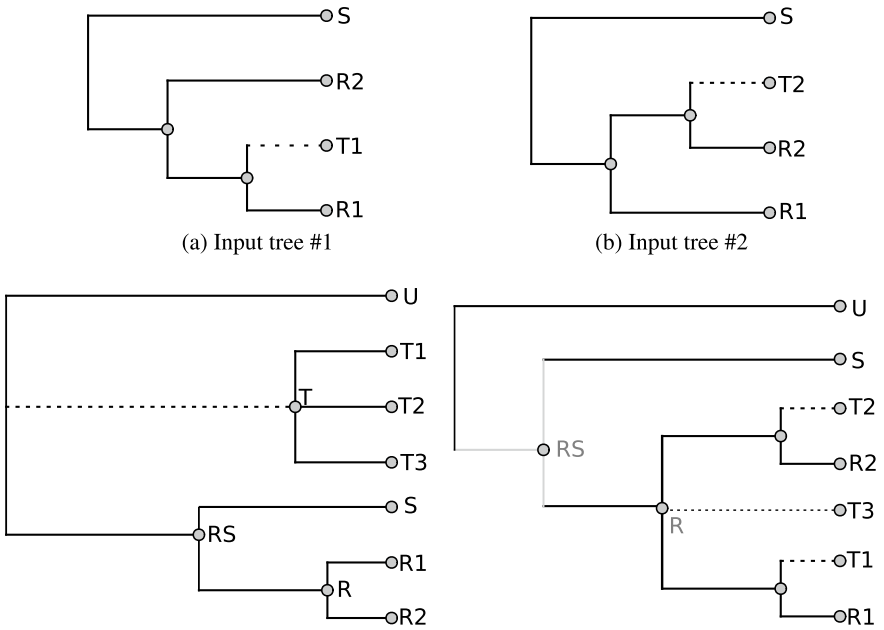
After we have attached taxon names to the synthesis tree, we would like to interpret the position of these names in terms of placing *incertae sedis* taxa in a revised taxonomy. This would allow us to interpret the synthesis tree as saying that phylogenetic information has (for example) placed genus A within family B, or perhaps outside of all named families. The simplest approach to placement involves noting whenever a taxon B is a descendant of a taxon A on the named synthesis tree but not the taxonomy tree. For example, in Fig. 7.2, taxon C is a descendant of taxon A in the synthetic tree (Fig. 7.2d), but not the taxonomy (Fig. 7.2b). Thus, we could say that the synthetic tree places C within A.

In general, after naming the clades in the supertree, the taxonomic placement of an *incertae sedis* taxon can be detected if the name of the *incertae sedis* taxon survives in the solution tree, and the most tipward named ancestor of the clade in the supertree was not an ancestor in the taxonomic tree. In complex scenarios such as Fig. 7.8c, it may be necessary to walk the solution tree in preorder and revise the taxonomy incrementally, in order to (for example) place V within WX before placing W within V.

7.2.6.2 Revision of Broken *Incertae sedis* Taxa

When an *incertae sedis* taxon is broken and its name does not occur on the solution tree, taxonomic revision becomes considerably more complex. For example, it can be ambiguous whether or not all of the members of a broken taxon should be placed in a new parent taxon. See, for example, Fig. 7.7. Here taxon $T1$ is a descendant of R in the synthesis tree, but not the taxonomy. Thus, $T1$ is placed within R; taxon $T2$ is similarly placed within R. $T3$ is not mentioned in either input tree, and the only tree that speaks to its placement is the taxonomic input. However, the taxon T cannot be monophyletic based on the two input trees. Thus, the correct placement of $T3$ is unclear. Furthermore, the taxon name for T does not occur on the solution tree.

The ambiguity of where to place $T3$ can be explored by comparing the behavior of maximum-compatibility tree estimation and parsimony-based scoring schemes such as MRP. The taxonomic tree's character representation of taxon T would include $T1$, $T2$, and $T3$ all being assigned state 1 and all of the other leaves being assigned state 0. This character is not compatible with the solution tree. The compatibility approach would be to conclude that the character (or, equivalently, the corresponding rooted split) encoding that taxon was incorrect and should be completely ignored. Since no other character speaks to the position of $T3$, it would fall to being a child of the root of the tree. This seems to contradict the intuition that $T3$ should be placed at the MRCA of $T1$ and $T2$.



(a) Input tree #1 (b) Input tree #2

(c) Taxonomy (d) Synthesis Tree #1

Fig. 7.7 *Incertae sedis* clade T broken by conflicting placement. Input trees **a** and **b** conflict in the placement of taxa in T. The synthesis supertree **d** places $T1$ and $T2$ separately within R. The BUILD algorithm would place the unsampled taxon $T3$ at the root of the tree, while our unprune operation implemented to handle taxonomy-only taxa would place $T3$ as a child of the MRCA of the other members of T, as pictured in (**d**)

Using a parsimony-based scoring scheme, the information from the character that encodes taxon T is not completely ignored, even though it is contradicted by an input tree. Placing $T3$ as either sister to $T1$ or $T2$ could explain the character representation of taxon T with only two acquisitions of the state 1. Other placements of $T3$ would require at least three 0 to 1 changes (or a reversion to state 0). Placing $T3$ at the MRCA of the other members of T corresponds to handling of "rogue" taxa in Adam's consensus tree [1] method of the two most parsimonious placements of $T3$ under an irreversible parsimony model.

Our current supertree algorithm places $T3$ at the MRCA of $T1$ and $T2$ if $T3$ is a taxonomy-only taxon, but places $T3$ at the root of the tree if it occurs in an input tree. This inconsistent behavior is not restricted to cases that contain *incertae sedis* taxa, but is a general problem with compatibility-based algorithms that do not retain partial information from incompatible splits. Such information could include compatible triplets that comprise parts of the incompatible split. The inconsistency results from the fact that our subproblem solver implements compatibility-based consensus and completely discards incompatible splits, while the unpruner implements an *ad hoc* rule to place pruned taxonomy-only taxa at the MRCA of taxa that occur in input

trees. It would be desirable to update the subproblem solver to be consistent with the unpruner, so that we could treat the pruning and unpruning steps as an optimization that has no effect on which tree is returned.

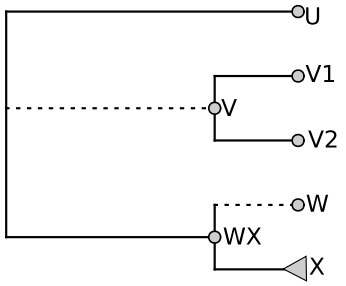
It is possible to update our compatibility-based supertree algorithm to achieve the intuitive result of placing $T3$ at the MRCA of $T1$ and $T2$ without resorting to a parsimony-based scoring scheme. We note that the split $T1\,T2\,T3 \mid \bullet\ U\,S\,R1\,R2$ implies many rooted triplets, and that we need not discard all of these triplets because some of them are contradicted by input trees. A simple method of retaining partial information from incompatible rooted splits would be to note that the MRCA of $T1$, $T2$ and must also have $R1$ and $R2$ as descendants. We would then remove these additional taxa from the exclude group, obtaining the partial split $T1\,T2\,T3 \mid \bullet\ U\,S$. Adding this split to the solution tree places $T3$ at the MRCA of $T1$ and $T2$ as desired. While this approach places all members of T inside R, it does not place a monophyletic T within R, since T is not monophyletic. Placing the name for T at the MRCA of its members would make T a synonym for R.

7.2.7 *Ambiguity of Split Representation for* Incertae sedis *Taxa*

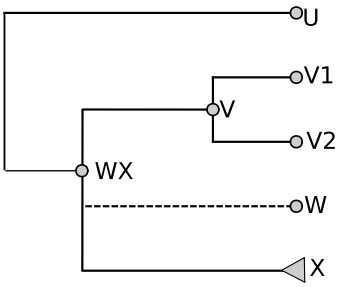
Equation (7.2) introduces a method for encoding a taxonomy with *incertae sedis* taxa into a set of partial splits. These partial splits have a unique representation as a matrix of 0-1 characters with missing data indicated by "?". Clearly, we will not be able to recover the identity, or even the number, of characters that a taxonomist might have used when classifying a set of organisms, simply by looking at the character matrix that represents a taxonomy (as Fig. 7.3 represents Fig. 7.2b). Nevertheless, one might hope that Eq. (7.2) is a unique and uncontroversial way to encode any taxonomic tree as a set of splits.

However, Eq. (7.2) is not the only set of rules for generating a split representation for a taxonomy. For example, consider the case of the taxonomy shown in Fig. 7.8a. Either of the two matrices shown in Fig. 7.9a or Fig. 7.9c could represent this taxonomy along with its *incertae sedis* interpretation. The matrix in Fig. 7.9a, which corresponds to Eq. (7.2), assumes that *incertae sedis* taxa may only intrude into taxa that are siblings in the taxonomy. The placement of W within V leads to taxon V not being named in the supertree shown in Fig. 7.8c under the encoding in Fig. 7.9a because V is not a sibling of W in Fig. 7.8a.

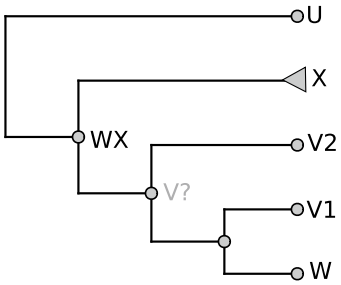
The matrix shown in Fig. 7.9c allows V to be named in the supertree shown in Fig. 7.8c because it removes the assumption that *incertae sedis* taxa can only be placed within siblings. However, as a side effect, if taxon WX is found to be non-monophyletic, the matrix from Fig. 7.9c allows W to be placed within V. In general, this approach would allow *incertae sedis* children of any taxon Γ_1 to be placed arbitrary deep into group Γ_2 if group Γ_1 is broken. This behavior can lead to surprising and nonintuitive results. Additionally, our unpruner currently assumes that *incertae*



(a) Taxonomy



(b) Possible intermediate



(c) Final supertree

Fig. 7.8 Placement of *incertae sedis* taxa within non-siblings could occur if taxa become siblings by placement. Given the taxonomy (**a**), placement of V within WX is shown in (**b**), followed by placement of W within V as shown in (**c**). Since W is not a sibling of V in (**a**), the method proposed here and illustrated in Fig. 7.9a does not preserve the taxon name for V. However, with an alternative method for computing reduced exclude sets that is shown in Fig. 7.9c, the name V would be preserved

	Character	
taxa	V	WX
U	0	0
$V1$	1	?
$V2$	1	?
W	0	1
X	0	1
(root)	0	0

(a) from original taxonomy

	Character	
taxa	V	WX
U	0	0
$V1$	1	1
$V2$	1	1
W	?	1
X	0	1
(root)	0	0

(b) from revised taxonomy

	Character	
taxa	V	WX
U	0	0
$V1$	1	?
$V2$	1	?
W	?	1
X	0	1
(root)	0	0

(c) alternative encoding

Fig. 7.9 Panels **a** and **c** show two possible encodings that would support the V and WX branches and *incertae sedis* annotation shown in the taxonomy from in Fig. 7.8a. The representations only vary in the scoring of leaf W for character V. Panel **b** shows the information encoded by the tree in Fig. 7.8b

sedis taxa can only be placed within siblings, and therefore does not know what to do in this situation. Thus, it is not clear that either Fig. 7.9a or Fig. 7.9c comprise a unique, best encoding of the taxonomy. Thus, to honor the input taxonomic content accurately, we may need more taxonomic information than simply a hierarchy and knowledge of which taxa can float more tipward.

Note that Eq. (7.2) allows *incertae sedis* taxa to intrude only into siblings in the original taxonomy, and not into taxa that become siblings through solution to the supertree problem. This can lead to cases where computing a supertree from two input trees T_1 and T_2 and an original taxonomy $\mathbb{T}$ can lead to a different outcome than if we compute a supertree from T_1 and $\mathbb{T}$, and then compute a supertree from T_2 and a revised taxonomy $\mathbb{T}'$. Consider a case in which the taxonomy $\mathbb{T}$ is given in Fig. 7.8a (with matrix representation given in Fig. 7.9a) and T_1 and T_2 each provide the single rooted splits $V\,1\,X \mid \bullet U$ and $W\ V\,1 \mid \bullet V\,2$, respectively. Running the supertree pipeline on T_1 and $\mathbb{T}$ yields the tree in Fig. 7.8b. We use this tree as the revised taxonomy $\mathbb{T}'$, and its matrix representation appears in Fig. 7.9b. Running the supertree pipeline on T_2 and $\mathbb{T}'$ then results in the tree in Fig. 7.8c.

One might therefore hope that running the supertree pipeline on T_1, T_2, and $\mathbb{T}$ simultaneously would yield the same result. While the topology for the final tree would indeed be the tree shown in Fig. 7.8c, the decision about whether or not to name V differs. As expected, the definition of WX in $\mathbb{T}'$ has been altered to reflect the fact T_1 placed V inside WX. Perhaps unexpected is the fact that storing the revised taxonomy with an *incertae sedis* annotation and then reprocessing the taxonomy using our rooted partial splits approach would also shift the definition of V from a taxon that must exclude W to one that may include W. This results from the fact that Eq. (7.2) only allows *incertae sedis* taxa to intrude into taxa that are their sisters in the taxonomy. Outcomes differ because W is a sister of V in $\mathbb{T}'$ but not $\mathbb{T}$. Use of the alternate representation of $\mathbb{T}$ in Fig. 7.9c allows V to be named in Fig. 7.8c, which solves the seeming inconsistency. However, this has some surprising side effects as mentioned above and is therefore not the approach that we focus on here.

7.3 Handling *Incertae sedis* Taxa in a Software Pipeline

In order to handle *incertae sedis* taxa in the software pipeline described by Redelings and Holder [10], we must modify some of the stages of the pipeline. Subproblem decomposition must place *incertae sedis* taxa in the correct subproblem. Subproblem files must indicate which taxa are *incertae sedis*. The subproblem solver must read this information, account for *incertae sedis* taxa when solving subproblems, and correctly name taxa that have been modified by having *incertae sedis* taxa place inside them. The unpruner must be aware of *incertae sedis* taxa. Annotations of the tree must be aware of *incertae sedis* taxa so that it does not consider taxa broken when they have an *incertae sedis* taxon placed inside them.

7.3.1 Subproblem Decomposition

The presence of *incertae sedis*taxa poses a problem to subproblem decomposition, since taxonomy edges no longer completely separate subproblems. Instead, *incertae sedis* taxa may attach on either side of a taxonomy edge. We seek to place *incertae sedis* taxa into subproblems in such a way that the subproblem solver can perform the placement inside the subproblem. This approach postpones handling of conflict in *incertae sedis* taxa to the subproblem solver, where the problem is well formulated in terms of splits. However, it does have the effect of creating larger subproblems.

Simply allowing the subproblem decomposer to recognize that intrusion of an *incertae sedis* taxon does not contest the existence of a clade would be straightforward, but problems arise if the decomposer is not further altered. Consider, for example, the inputs of the τ_1, τ_2, τ_3, and the taxonomy shown in Fig. 7.10. Because the taxon α is annotated as being *incertae sedis*, neither τ_1, τ_2, nor τ_3 contest the monophyly of the taxa Y, Z, or α. This would lead to the taxonomic portion of both the Y and Z subproblems containing the taxonomy for α, leading to the members of α occurring in two different subproblems (see Fig. 7.10f). Solving and merging these problems would result in the duplication of α in the supertree.

One could easily imagine a "pruning decomposer" that does not allow the same taxon to occur in more than one subproblem. This approach would work in a straightforward way if the phylogenetic inputs were only τ_1 and τ_3, yielding the decomposition shown in Fig. 7.10g, but if both τ_1 and τ_2 were inputs the decomposer would

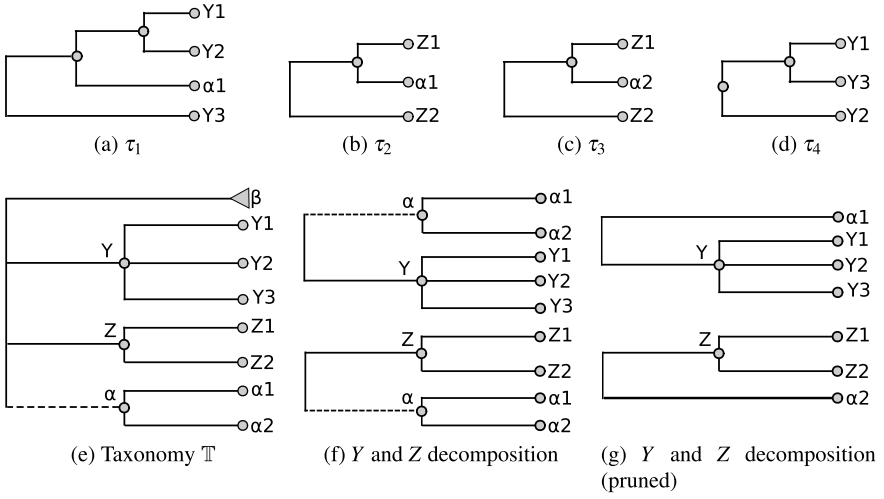


Fig. 7.10 Four phylogenetic trees **a**–**d** an example taxonomy **e** which are possible inputs to the supertree pipeline. **f** Shows the taxonomic component of the subproblem decompositions for taxa Y and Z subproblems that would result from an unmodified pipeline with used τ_1 and either τ_2 or τ_3 as the phylogenetic inputs. **g** Shows a possible decomposition from a decomposition tool that avoids duplication of taxa

have to resolve the conflict about the placement of α_1. Implementing the behavior in the subproblem decomposition step would break the separation between the "divide" and "conquer" steps which is a key aspect of the pipeline's current efficiency.

Even considering ranked phylogenetic inputs of τ_4 then τ_1 from Fig. 7.10 reveal some difficulties. In this case, the inclusion of α within the taxon Y in τ_1 would cause α to be solved in the subproblem for Y. If τ_4 were ranked higher than τ_1, then the grouping of $Y_1 + Y_2 + \alpha$ would not be permitted because they conflict with the split $Y_1\ Y_3 \mid \bullet\ Y_2$ from τ_4. Thus, α would be placed inside of Y on the basis of a rooted split that is not displayed in the final tree. While this is placement is permissible based on the *incertae sedis* annotation of the taxonomy, a major goal of the pipeline is to make it easy for biologists to understand the cause of different groupings in the supertree. Allowing splits which are not displayed to determine the placement of groups runs counter to that goal.

We choose to solve these problems by merging any subproblems that an *incertae sedis* taxon might be placed in. The simplest way to achieve this is simply to regard any taxon that has an *incertae sedis* taxon placed within it as contested. This results in marking both B and C as contested edges in the example above. In fact, this is the current behavior of the non-*incertae sedis* aware subproblem decomposer. One downside of this approach is that, when an *incertae sedis* taxon occurs in only one input tree, the clade it is placed in will be marked as contested even though none of the counterintuitive behaviors mentioned above would arise if we did not mark it as contested.

7.3.2 Subproblem Solution

Our subproblem solver naturally handles *incertae sedis* taxa. This is because we define the semantics of *incertae sedis* taxa in terms of partial splits, and our solver natively supports building trees from partial splits through its use of the BUILD algorithm. Handling *incertae sedis* taxa thus requires loading *incertae sedis* information and computing partial splits for *incertae sedis* taxa before solving a subproblem. After solving a subproblem, we must apply taxon names from the taxonomy tree to the subproblem solution tree. The solution tree is considered to be a fixed tree and not to have any *incertae sedis* nodes, or any other forms of uncertainty.

7.3.2.1 Implementation: Finding the Node for a Name

To find the node for a name n, we find the MRCA of the cluster $S_1(n)$. If the MRCA excludes the entire exclude group $S_2(n)$ then the name applies to the MRCA; otherwise the taxon does not exist on the tree.

7.3.2.2 Implementation: Handling Name Clashes

When multiple names $N = \{n_1, \ldots n_N\}$ map to the same solution node x, then these names must satisfy some tree structure on the taxonomy, such that $n_1 < n_2$ if n_1 is a descendant of n_2 in the taxonomy. If it is possible to find a name n_{max} that is the unique maximal element of N, then it is permissible to

1. create a monotypic parent $p(x)$ of x, and assign n_{max} to $p(x)$, and
2. continue handling name clashes at x with the set of possible names reduced to $N - n_{max}$.

However, it's certainly possible that there might not be any such N_{max}, in which case we could just choose a name for x from N (perhaps not an *incertae sedis* name) and then record all the other names as equivalents somewhere.

7.4 Discussion

This contribution introduces the necessary concepts and methods for handling *incertae sedis* taxa in taxonomic supertrees. We describe the goals of preserving internal edges, preserving higher taxon names, and preserving the opportunity to revise the taxonomy based on the placement of *incertae sedis* taxa. In addition to explaining why the naive approach to taxonomic supertrees that ignores *incertae sedis* taxa sacrifices these three goals, we introduce a simple and satisfying way to achieve these goals through defining rooted splits for taxonomy edges with reduced exclude sets. We also provide a simple and efficient recursion for computing these splits.

Despite the simplicity of our approach, some complexities and conceptual difficulties remain. While the correspondence of nodes and names is a simple one-to-one function without *incertae sedis* taxa, when *incertae sedis* taxa are introduced, the relationship might be neither one-to-one (Figs. 7.5 and 7.6) nor a function (Fig. 7.4). This necessitates putting more thought into the semantics of taxon names, including whether names might become synonyms. This may be considered a "complexity".

In addition, our attempt to define reduced exclude sets for taxonomy splits yields a semantics that is not the only conceivable semantics for *incertae sedis* taxa (Fig. 7.8). It allows *incertae sedis* siblings to inter-digitate (Fig. 7.6), which might be a surprising result to the taxonomist. Furthermore, in what might seem a lack of consistency, it does *not* allow *incertae sedis* taxa to intrude into each other if they become siblings through the result of placement, as is the case for taxa W and V in Fig. 7.8c. While each of these problems can be solved separately by adjusting the definition of the reduced exclude sets, no solution seems to be best in all respects. This may be a result of the fact that split-based approaches to supertrees seek a simultaneous solution to a set of constraints. This simultaneity can be contrasted with a solution that is phrased as a set of sequential operations to the taxonomy, as would be the case when (for example) (i) placing V within WX so that it is sister to W, (ii) marking V as no longer

incertae sedis, and then (iii) placing W within V (which is no longer *incertae sedis*). This maybe considered a "conceptual difficulty".

Another issue that we have postponed for future work is how to more narrowly delimit the ranges in which *incertae sedis* may attach. The current paper allows *incertae sedis* to attach within any of their siblings, but the taxonomist may only consider it plausible for the taxon to intrude only within specific siblings. We also assume that a taxon labeled as *incertae sedis* may attach at any depth within its siblings. In practice, an *incertae sedis* genus with sister families (for example) is unlikely to attach so deeply within a sister family that it becomes nested within another genus.

Our method also assumes that the *incertae sedis* status of a taxon consists solely of a flag on that taxon. In order to consider more precise attachment ranges for *incertae sedis* taxa, we would have to obtain machine-readable specifications for attachment ranges for groupings that a taxonomist has labeled as *incertae sedis*. Obtaining these specifications would be difficult or impossible in many cases. However, not all *incertae sedis* taxa in our taxonomy are directly labeled *incertae sedis* by taxonomists. *Incertae sedis* taxa can also result from the automatic merging of taxonomies to create the Open Tree taxonomy. For example, in Fig. 3 of [11], cases #4 and #6 illustrate examples where merging of two taxonomies leads to a taxonomy with a taxon of uncertain placement.

Taxonomy merging generates an automated *incertae sedis* annotation in a number of ways. Perhaps, the easiest cause to contemplate is if one input taxonomy, $\mathbb{T}_1$, contains more levels of hierarchy than a second input taxonomy, $\mathbb{T}_2$. When the merger adds internal nodes that are unique to $\mathbb{T}_2$, it is unclear how these extra leaves should be nested or placed in the richer hierarchy of $\mathbb{T}_1$. In such a case, it is feasible to imagine the taxonomy-merging procedure outputting a list of plausible attachment points for each generated *incertae sedis* label. For the case in which the *incertae sedis* label is present in an input taxonomy, restricting the range of attachment points would need to be treated as manually curated statements.

Acknowledgements The authors thank Dr. Karen Cranston for comments and the NSF (awards 1759838 and 1208393) for funding. MTH would like to thank Dr. Bernard Moret. Dr. Moret's leadership of the CIPRES project was crucial to MTH's early career, and Dr. Moret's career consistently served as an inspiring example of integrating the discipline of computer science with evolutionary biology.

References

1. Adams III, E.N.: Consensus techniques and the comparison of taxonomic trees. Syst. Biol. **21**(4), 390–397 (1972). https://doi.org/10.1093/sysbio/21.4.390.
2. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comput. **10**(3), 405–421 (1981)
3. Baum, B.R.: Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. Taxon **41**, 3–10 (1992)

4. de Queiroz, K.: Nodes, branches, and phylogenetic definitions. Syst. Biol. **62**(4), 625–632 (2013). https://doi.org/10.1093/sysbio/syt027. https://doi.org/10.1093/sysbio/syt027
5. Dress, A., Huber, K.T., Koolen, J., Moulton, V., Spillner, A.: Basic Phylogenetic Combinatorics. Cambridge University Press (2012)
6. Hinchliff, C.E., Smith, S.A., Allman, J.F., Burleigh, J.G., Chaudhary, R., Coghill, L.M., Crandall, K.A., Deng, J., Drew, B.T., Gazis, R., Gude, K., Hibbett, D.S., Katz, L.A., Laughinghouse, H.D., McTavish, E.J., Midford, P.E., Owen, C.L., Ree, R.H., Rees, J.A., Soltis, D.E., Williams, T., Cranston, K.A.: Synthesis of phylogeny and taxonomy into a comprehensive tree of life. Proc. Natl. Acad. Sci. **112**(41), 12,764–12,769 (2015). https://doi.org/10.1073/pnas.1423041112. http://www.pnas.org/content/112/41/12764.abstract
7. Hörandl, E., Stuessy, T.F.: Paraphyletic groups as natural units of biological classification. Taxon **59**(6), 1641–1653 (2010). http://www.jstor.org/stable/41059863
8. McTavish, E.J., Hinchliff, C.E., Allman, J.F., Brown, J.W., Cranston, K.A., Holder, M.T., Rees, J.A., Smith, S.A.: Phylesystem: a git-based data store for community-curated phylogenetic estimates. Bioinformatics btv276 (2015)
9. Ragan, M.A.: Phylogenetic inference based on matrix representation of trees. Mol. Phylogenet. Evol. **1**(1), 53–58 (1992)
10. Redelings, B.D., Holder, M.T.: A supertree pipeline for summarizing phylogenetic and taxonomic information for millions of species. PeerJ **5**, e3058 (2017)
11. Rees, J.A., Cranston, K.: Automated assembly of a reference taxonomy for phylogenetic data synthesis. Biodivers. Data J. **5**(5), e12581 (2017)
12. Wilkinson, M.: Common cladistic information and its consensus representation: reduced adams and reduced cladistic consensus trees and profiles. Syst. Biol. **43**(3), 343–368 (1994)

Chapter 8
Evolutionary Rate Change and the Transformation from Additive to Ultrametric: Modal Similarity of Orthologs in Fish and Flower Phylogenomics

Daniella Santos Muñoz, Eric Lam and David Sankoff

Abstract Branch lengths in a phylogeny may be in units of elapsed time, so that the nodes have dates associated with them, or in units of evolutionary change, such as the number of mutations that have accrued between the two endpoints of a branch. Methods to account for the mutational change in terms of an additive tree are generally incompatible with the ultrametric requirement of time-based tree representations because of changes in mutation rate. There are some principled ways of converting additive trees to ultrametric form, and these suggest which branches have seen increased or decreased rates. We spell these methods out and apply them to the ray-finned fishes and the plant families Solanaceae and Malvaceae. The methods based on nonparametric rate smoothing prove to be more revealing than the Farris transform methods.

Keywords Tree metrics · Peaks tree · Fish phylogeny · Solanaceae · Malvaceae

8.1 Introduction

Phylogenomics, like phylogenetics, attempts to reconstruct evolutionary history by converting data of various kinds on a set of genomes into a rooted tree whose nodes represent speciation events, historical points at which one existing lineage is split into two or more lineages, each of which continued to evolve independently. Accompanying the tree, which is basically a special combinatorial object consisting of vertices

D. Santos Muñoz · E. Lam · D. Sankoff (✉)
University of Ottawa, Ottawa, Canada
e-mail: sankoff@uottawa.ca

D. Santos Muñoz
e-mail: dsant041@uottawa.ca

E. Lam
e-mail: elam041@uottawa.ca

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_8

(nodes) and edges (branches), we usually associate positive branch lengths. These lengths may be in units of elapsed time, so that the nodes have dates associated with them, or in units of evolutionary change, such as the number of mutations that have accrued between the two endpoints of a branch.

Were evolution clock-like, so that the number of mutations during a period of time was strictly proportional to the duration of that period, the two types of tree would be congruent. They would both be consistent with the ultrametric inequalities, with the given genomes assigned the present time. The number of mutations along each lineage, from the root to each of the present-day genomes, would be the same. Evolution, however, proceeds at an uneven pace, so that the rate of mutation may be elevated in one branch and depressed in another. A phylogenetic representation faithful to the amount of evolution on each branch, such as an additive tree, satisfying only the weaker four-point metric, will not generally be consistent with the ultrametric inequalities; we cannot assign dates to the nodes in any direct way so that the mutational change from the root to each of the present-day genomes is the same. Methods such as neighbor joining (NJ) [29], which produce edge-weighted trees (and hence additive trees) from a matrix of distances between pairs of genomes, will not generally output an ultrametric tree. Methods like UPGMA [24], which produce an ultrametric tree from the same data, will not generally give a good fit to data that are consistent with a lopsided additive tree and are liable to output a wrong tree topology in forcing an ultrametric structure on the data.

Inherent in the non-ultrametric nature of additive data is the existence of branches in the "true" phylogeny that generated the data with very high or very low mutational rates. This suggests that we could compare the length of branches in an additive tree and an ultrametric tree for the same set of data. The problem with this is that methods designed to produce an ultrametric and those generating an additive tree would generally produce different topologies, so that some of the branches in one would not exist in the other, so no comparison would be possible. An alternative would be to adapt an additive tree by stretching or compressing branches in some principled way so that it assumed ultrametric shape. This is the approach we will take here, exploring three alternative published techniques for converting an additive tree to an ultrametric, in order to detect evolution speeding up or slowing down on specific branches.

Recently a new, uniquely phylogenomic, way of deriving evolutionary distances was introduced [33, 36, 37], something that has no counterpart in traditional gene-based phylogeny or even concatenation-of-many-genes extensions of classical methods. This is based on the distribution of *all* syntenically validated ortholog similarities between two genomes. Syntenic validation is assured by software such as SynMap on the CoGe platform [20, 21]. Whole Genome Doubling events and speciation events can be separated out by locating "peaks" (local modes) in this distribution. The most recent (highest similarity) peak is always due to speciation. Although there may be relatively few pairs of orthologous genes with peak similarity, its accuracy is buttressed by the hundreds, thousands, or tens of thousands of ortholog pairs distributed in both sides of it on the x-axis. (Our description is phrased in terms of similarity rather than K_s, but either is feasible.) The peak similarities p can be trans-

formed to estimates of evolutionary divergence using a negative log transform, or simply by $1 - p$.

Since the peaks method depends on no one gene family, and does not require any single-copy constraint, it is not susceptible to the rapid expansion of gene families or paralog-based confusion; it is uniquely placed to measure whether entire genomes have adopted a faster or slower pace of evolution. This motivates the application of additive to ultrametric transformations to peaks phylogenies in situations where WGD may muddy the waters when using traditional kinds of distance. The goals of this paper are thus to apply three transformation methods to distance-based phylogenies based on peaks, and to use these techniques to detect increased or diminished rates of evolution in three phylogenetic domains.

8.2 Approaches to Transformation

8.2.1 Farris Transform

Originally suggested in [10], the Farris transform is among the early methods for converting an additive tree metric into an ultrametric. Its original implementation, however, was to be used in conjunction with UPGMA [24]. The idea was to correct for the constant evolution rate across all lineages assumption, which may result in incorrect topologies given an additive tree, through the use of an outgroup [17].

Let $X = \{1, 2, \ldots, n\}$ represent the set of present-day taxa, and T a rooted and dated phylogenetic tree on X with vertex set $V_T \supset X$. Assume that for any two present-day taxa $i, j \in X$, the genetic distance between them is represented by $D(i, j)$. Now, consider any vertex $v \in V_T$, and any two distinct taxa i, j of v, in the set $X(v) = \{k \in X : v \text{ is an ancestor of } k\}$. The set $X(v)$ contains all present-day descendants of v in X. Finally, if we let $a \notin X(v)$ be a known outgroup and $D(i, j)$ be an additive distance matrix for the set of species X, then we have the following transformation:

$$D'(i, j) = \overline{D(a)} + \frac{1}{2}\left(D(i, j) - D(i, a) - D(j, a)\right) \tag{8.1}$$

where $\overline{D(a)} = \frac{1}{n}\sum_{i=1}^{n} D(i, a)$, which is the average distance between the outgroup and all ingroups [17]. An alternative constant that can be used is $D = \max_i D(i, a)$, the maximum distance value between the outgroup and any ingroup from the additive distance matrix [12].

Theorem 1 ([3]) *Let D be an additive distance matrix. If D' is the Farris transform of D, then D' is ultrametric.*

For more details on the mathematical context of the Farris transform, see [9].

8.2.2 Nonparametric Rate Smoothing (NPRS)

NPRS [31] is a method that estimates divergence times without assuming evolutionary rates are constant across lineages. The method relaxes the assumption of the molecular clock by using a least squares smoothing of local estimates of substitution rates. NPRS is dependent on the minimization of ancestor-descendent local rate changes and is motivated by the likelihood that evolutionary rates are autocorrelated in time. It also estimates divergence times for all unfixed nodes. Sanderson's paper [31] demonstrates that NPRS-produced divergence time estimates are more consistent with paleobotanical evidence than tests using clock-based estimates.

The main idea of Sanderson's method was to construct an estimate of the evolutionary local rate of each branch in a tree while minimizing the difference between that estimate and the local rate estimates of their immediate descendants.

A simple local estimate of rate is

$$\hat{r} = \frac{b}{t} \tag{8.2}$$

where b is the length of branch and t is its temporal duration.

All branches on the path between the root and a terminal node are directed away from the root. We denote by $\hat{r}_k$ the rate on the unique branch directed to node k and $\mathscr{D}(k)$ to the set of nodes immediately descended from node k. Now, for each internal node k of the tree, we define

$$w_k = \sum_{j \in \mathscr{D}(k)} \left| \hat{r}_k - \hat{r}_j \right|^2 \tag{8.3}$$

Thus, an overall function to be minimized is then attained by summing the terms over all internal nodes as defined by

$$W = \sum_{k \in \text{internal nodes}} w_k \tag{8.4}$$

A reasonable estimator of local rate will take into account the unknown time endpoints of the branches and some function of the distance matrix, **D**. Hence, we can rewrite the function W as the function $W(t_1, t_2, \ldots, t_m|\mathbf{D})$, where m is the number of internal nodes and $\{t_i\}$ are the unknown times of internal nodes. The minimization of W over these unknown times can then give estimates of those divergence times (comparable to nonparametric regression techniques). The objective is to smooth the local transformations in rate as the rates change over the tree. Equation (8.3) is a minimization of changes in rate from an ancestral lineage to its descendant lineages.

Sanderson [31] estimates the root branch local rate as the mean of all the estimated descendant rates throughout the tree and then constructs an expression similar to Eq. (8.3) for the root node. The mean estimated rate is defined by

$$\hat{r}_{\text{root}} = \frac{1}{n}\sum_{k}\hat{r}_k \tag{8.5}$$

where n is the number of branches. Thus the objective function to be minimized, W, should also include the term

$$w_{\text{root}} = \sum_{j\in\mathscr{D}(root)} \left|\hat{r}_{root} - \hat{r}_j\right|^2. \tag{8.6}$$

8.2.3 Penalized Likelihood

Penalized likelihood [32] is a semiparametric smoothing method. It features a trade-off between a parametric model having a different substitution rate on every branch with a nonparametric model which penalizes the model more for rapid rate changes on a tree. This is controlled by an optimality criterion, namely the log likelihood minus λ times a roughness penalty, where λ is the "smoothing parameter." As smoothing values increase, the variation in rates are smoother and the model is reasonably clock-like; whereas for small values of smoothing, the parametric component dominates and rates heavily vary among branches. Optimal values of the smoothing parameter can be determined by performing cross-validation, a resampling procedure that successively removes each terminal branch and estimates the remaining parameters of the reduced tree using penalized likelihood [11].

Consider a rooted phylogenetic tree with n taxa and m internal nodes, where the root node is labeled by a, the remaining internal nodes are labeled by $\{1, \ldots, m-1\}$ and the leaf nodes are labeled by $\{m, \ldots, m+n-1\}$. Branches are labeled by the node they are directed away from, as in the NPRS method. Let node k have an age t_k and its ancestral node, called $anc(k)$, have an age $t_{anc(k)}$. The branch defined by these two nodes has a duration in time given by $t_{anc(k)} - t_k$.

In a clock-like (CL) model, the rate parameters are the same for every branch, $\hat{r}_k = \hat{r}$. In a saturated model (SAT), each branch can have a unique rate, $\hat{r}_k$. The known parameters of each model can be written respectively as $\theta_{\text{CL}} = \left\{t_a, \ldots, t_{m-1}; \hat{r}\right\}$ and $\theta_{\text{SAT}} = \left\{t_a, \ldots, t_{m-1}; \hat{r}_1, \ldots, \hat{r}_{m+n-1}\right\}$, for $m+1$ or $2m+n-1$ free parameters. Next, let $P(x|\xi) = \frac{\xi^x \exp(-\xi)}{x!}$ be the usual probability that an observation x is taken from a Poisson distribution with parameter ξ. Then, the log likelihood of θ for the saturated model is

$$\log L\left(\theta_{\text{SAT}}|x_1, \ldots, x_{m+n-1}\right) = \sum_{k=1}^{m+n-1} \log P\left(x_k|\hat{r}_k\left[t_{anc(k)} - t_k\right]\right) \tag{8.7}$$

The following penalized likelihood to be maximized is given by:

$$\Psi\left(\theta_{\mathrm{SAT}}|x_1,\ldots,x_{m+n-1}\right)=\log L\left(\theta_{\mathrm{SAT}}|x_1,\ldots,x_{m+n-1}\right)-\lambda\Phi\left(\hat{r}_1,\ldots,\hat{r}_{m+n-1}\right) \tag{8.8}$$

where Φ is a roughness penalty and λ is the smoothing parameter previously discussed above. The roughness penalty Φ should be chosen to reflect change in rate between neighboring branches of the tree. Following the NPRS method above, this penalty was chosen to penalize squared difference in rates between ancestral and descendant branches and the variance in rate between the branches descended from the root node:

$$\Phi\left(\hat{r}_1,\ldots,\hat{r}_{m+n-1}\right)=\sum_{k\notin a\bigcup\mathscr{D}(a)}\left(\hat{r}_k-\hat{r}_{anc(k)}\right)^2+\mathrm{Var}\left(\hat{r}_k:k\in\mathscr{D}(a)\right) \tag{8.9}$$

where $\mathscr{D}(k)$ is the set consisting of the descendants of node k. The summation extends to all internal nodes except the root node and the descendants of the root. The second term compares the branches descended from the root node and minimizes the variances of their rates [32].

The choice of λ will affect the estimated rates and times. We use cross-validation [11] to assist in choosing the value for this parameter.

We drop each leaf successively, leaving its ancestral node in place and repeat the analysis with the remaining tree, using the **ape** package [26] in R [27]. For the ith leaf, the following is calculated:

$$\sum_{j=1}^{m-1}\frac{\left(t_j-t_j^{-i}\right)^2}{t_j} \tag{8.10}$$

where t_j is the estimated date for the jth node with the full phylogeny, t_j^{-i} is the estimated date for the jth node after removing leaf i from the tree, and lastly, m is the number of internal nodes.

8.3 Pipeline

The data for this study were collected from the genomes stored on the CoGe platform [20] for the fish genomes and for some of the others, and from the NCBI genome website for many of the flower genomes; the latter were stored in a private account in CoGe. The SynMap [22] tool in CoGe was used to compare the CDS versions of each of the pairs of the genomes included in the fish, Solanaceae and Malvaceae data sets. Histograms of measures of dissimilarity $(1-p)$, K_s (also written as D_s, the rate of synonymous substitutions in the coding sequence) and $\log_{10} K_s$ were compiled for the syntenically validated orthologous gene pairs detected by SynMap, and examined to obtain the location of "peak" frequencies of these measures. These locations were arrayed as a genome distance matrix for each data set, and was used as input to

UPGMA to derive an ultrametric tree and to the NJ algorithm to create an additive tree.

To apply the transformations, we first derived the branch matrix from the NJ results, containing the sum of the branch lengths on the path through the tree for each pair of species. We applied each of the transformations: Farris, NPRS and penalized likelihood, to the branch matrix. Then, NJ was repeated on the transformed data to produce an ultrametric tree for display purposes.

8.4 Applications

8.4.1 Fish

Fish phylogeny has long been controversial, particularly in the classification of the perciforme genomes due to the numerous speciation events that have occurred within a relatively short time span. Previously, some groups of perciformes were thought to be monophyletic, but are now considered polyphyletic [5, 13]. To help resolve these issues, we set out to construct a peaks-based phylogeny of 15 fish species, distributed among 10 orders of ray-finned fishes including five perciforme orders, as well as a shark (a cartilaginous fish) and a coelacanth (a lobe-finned fish) (Table 8.1).

Figure 8.1 illustrates the use of histograms derived from SYNMAP analyses. The use of a log scale for the K_s value helps separate the various peaks. In the salmon-

Table 8.1 Species included in the study

Order	Species	Common name
Chimaeriformes	*Callorhinchus milii* [35]	Elephant shark
Coelacanthiformes	*Latimeria chalumnae* [1]	African coelacanth
Semionotiformes	*Lepisosteus oculatus* [6]	Spotted gar
Characiformes	*Astyanax mexicanus* [23]	Mexican tetra
Cypriniformes	*Danio rerio* [14]	Zebrafish
Esociformes	*Esox lucius* [28]	Northern pike
Salmoniformes	*Salmo salar* [19]	Atlantic salmon
Pleuronectiformes	*Cynoglossus semilaevis* [8]	Tongue sole
Perciformes	*Oreochromis niloticus* [7]	Nile tilapia
Gasterosteiformes	*Gasterosteus aculeatus* [16]	Three-spined stickleback
Tetraodontiformes	*Takifugu rubripes* [2]	Torafugu
	Tetraodon nigroviridis [15]	Spotted green pufferfish
	Mola mola [25]	Ocean sunfish
Cyprinodontiformes	*Poecilia reticulata* [18]	Guppy
	Xiphophorus maculatus [34]	Southern platyfish

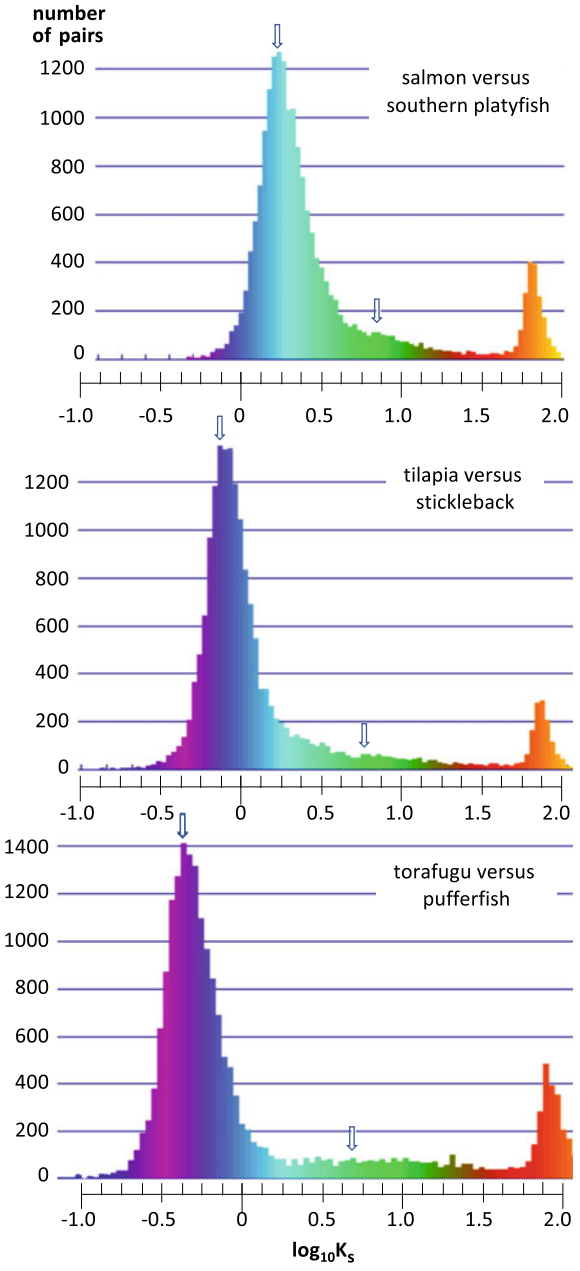


Fig. 8.1 Distribution of log K_s of duplicate gene pairs in salmon-platyfish, tilapia-stickleback and torafugu-pufferfish comparisons, showing a common WGD before speciation. Arrows indicate speciation peaks and WGD peaks. Peaks at far right represent noise (gene fragments, common domains in unrelated genes, etc.) accumulating in exponentially larger intervals, as well as earlier vertebrate WGDs

platyfish comparison, we can identify the speciation peak at $\log_{10} K_s = 0.25$ and a WGD that, being pre-speciation, is shared by both genomes, with $\log_{10} K_s$ in the range of 0.76–0.81. The salmonid WGD, occurs later than this speciation, and so does not show up as a peak. The visible peak is the "teleost WGD" at the root of the ray-finned fish radiation, and is confirmed in the other two comparisons, with tilapia versus stickleback ($\log_{10} K_s$ range 0.75–81) and torafugu versus pufferfish (diffuse peak for $\log_{10} K_s$ between 0.68 and 1.0), respectively. More important for our purposes are the more precisely defined speciation peaks: salmon-platyfish, tilapia-stickleback, and torafugu-pufferfish at $\log_{10} K_s = 0.25, -0.15$, and -0.4, respectively.

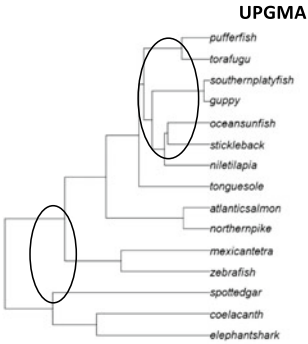
We use the matrix of speciation peak locations (in K_s terms) in all $\binom{15}{2} = 105$ comparisons as the input to UPGMA and a Java-implemented NJ algorithm. We filled in the one missing data point, for the northern pike-spotted gar comparison (due to very sparse K_s information) with the salmon-spotted gar value, since salmon and northern pike paralleled each other in all the other comparisons. The ultrametric tree produced by UPGMA and the additive tree output by NJ, as well as two of its transforms, are depicted in Fig. 8.2. The Farris transformation analysis failed, for reasons explained in Sect. 8.5 below.

The tree produced by UPGMA is a good illustration of why we avoid dating evolutionary events by directly producing an ultrametric from a distance matrix. The topology of the tree should first be determined by methods less sensitive to violations of the constant rate condition. Adjusting the branch length while retaining the topology produces a more meaningful timing of event history. The fish UPGMA produces a biologically thoroughly implausible sister group to the ray-finned fishes consisting of the elephant shark and the coelacanth, while it also scrambles aspects of perciform evolution.

The NJ tree and its transforms, on the other hand, are in complete accord with the current understanding of fish phylogeny. This is true of every branching in the tree, although the tongue sole should probably be grouped with tilapia, guppy and platyfish, rather than as an outgroup to these and the tetradontiform/stickleback clade [4, 5, 30].

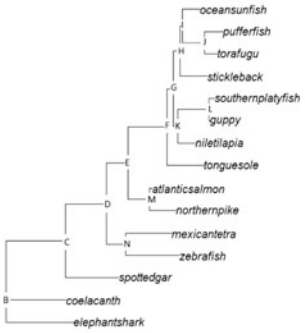
The ratios of transformed branch lengths suggest conservative evolutionary tendencies in the earliest branching fish species (elephant shark, coelacanth and spotted gar), since these branches need to be multiplied by a relatively large factor in the transition to ultrametric, although the lineage represented by the internal branches leading through the ray-finned fish (including spotted gar) to the teleosts (including the remaining ten species) underwent rapid evolution, since these branches need to be multiplied by a very small factor in the transformation to an ultrametric representation. Salmon and guppy both seem much more conservative than their sister species, northern pike and platyfish, respectively, although the common ancestor of salmon and pike also appears to have been conservative.

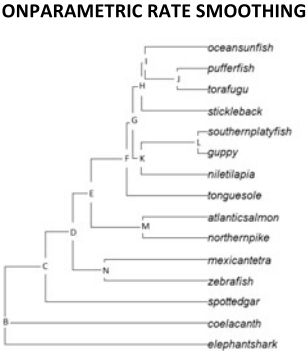
Compared to nonparametric rate smoothing, the roughness penalty in the penalized likelihood method attenuates somewhat the discordance between salmon and pike and between guppy and platyfish.



RATIOS

	NPRS	PL
Elephant shark	1.02	1.02
Coelacanth	1.14	1.14
Spotted gar	1.03	1.02
Zebrafish	0.65	0.64
Mexican tetra	0.76	0.75
Salmon	7.61	5.33
Northern pike	0.81	0.57
Tongue sole	0.74	0.70
Tilapia	1.28	1.19
Guppy	3.40	2.45
Platyfish	0.61	0.44
Stickleback	0.81	0.77
Torafugu	0.80	0.76
Pufferfish	0.55	0.53
Sunfish	1.19	1.13
B-C	0.23	0.24
C-D	0.23	0.24
D-N	0.54	0.52
D-E	0.28	0.30
E-M	0.81	1.05
E-F	0.30	0.30
F-G	0.34	0.34
G-K	0.61	0.65
K-L	0.61	0.59
G-H	0.44	0.42
H-I	0.49	0.48
I-J	0.49	0.45





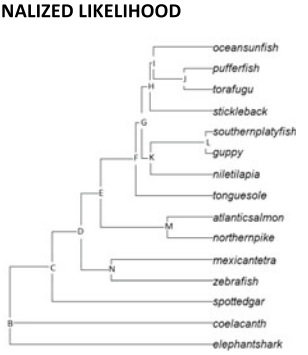


Fig. 8.2 UPGMA tree, NJ tree and its transformations by nonparametric smoothing and penalized likelihood methods, for the fish data. "Ratios" summarize transformed branch lengths versus original NJ values. Ovals surround regions of UPGMA tree that contradict accepted biological knowledge

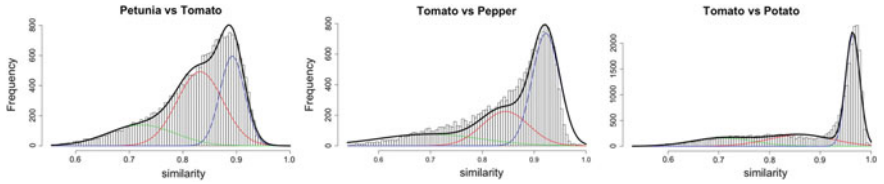


Fig. 8.3 Solanaceae Whole Genome Doubling (WGD) and speciation peaks

8.4.2 Solanaceae

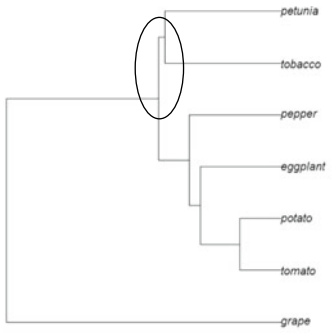
The Solanaceae family, here represented by tomato, potato, eggplant, pepper, tobacco, and petunia, descends from two whole genome triplications events, the first one, 120 million years ago affecting all core eudicots, and the second one, specific to this family [37], dating from around 40 ± 10 million years ago. The distributions of gene pair similarities thus have three peaks, two dating from the ancient polyploidization events, and one from speciation, as seen in Fig. 8.3. As seen in the previous section, more recent peaks tend to be less dispersed. This holds not only within each comparison, but also for the speciation comparisons, where potato is the most closely related to tomato, and petunia is the most distantly related.

As with the fish data, Fig. 8.4 shows an error in the UPGMA tree, this time for our sample of plants from the Solanaceae family; petunia and tobacco are grouped together separately from pepper and the *Solanum* species: tomato, potato, and eggplant. The NJ tree and its transforms correctly place tobacco as branching from the other species *after* petunia does.

The branch length ratios indicate that evolutionary change has accelerated in the *Solanum*, though not dramatically. This trend emerges most clearly in the nonparametric rate smoothing and penalized likelihood methods. All three methods show that after the speciation event leading to petunia, the ancestor of the other genomes underwent rapid evolutionary change.

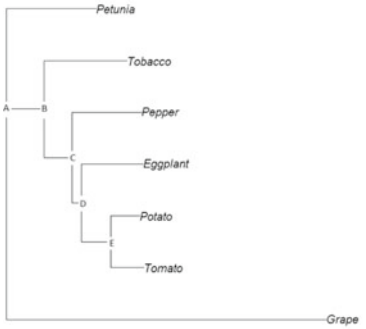
8.4.3 Malvaceae

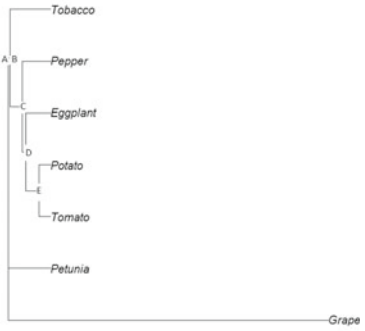
Genomes that have been sequenced in the Malvaceae include cacao, monkey cacao, jute (two species), durian, hibiscus and cotton (two species). Once again, there is a discrepancy between the UPGMA tree and the NJ tree. Figure 8.5 shows that durian branches after jute in the lineage toward cocoa in the UPGMA tree, and before jute in that lineage in the NJ tree. In this case, however, it may very well be that the UPGMA tree is correct, and that the NJ tree results from statistical fluctuations involving the cotton genome [36]. In any case, there is as yet no consensus in the literature about the placement of durian.

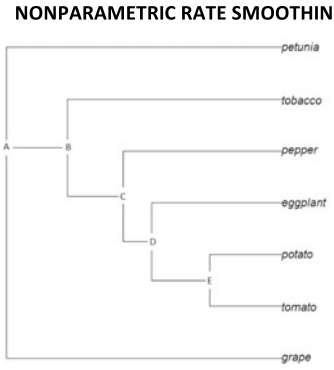


RATIOS

	Farris	NPRS	PL
Grape	1.15	0.57	0.57
Tomato	0.40	1.13	1.33
Potato	0.46	1.28	1.51
Eggplant	0.46	1.23	1.45
Pepper	0.47	1.56	1.54
Tobacco	0.57	1.81	1.99
Petunia	0.54	2.04	2.04
A-B	0.05	0.90	0.52
D-E	0.50	1.32	1.54
C-D	0.50	3.55	1.86
B-C	0.50	1.49	2.05







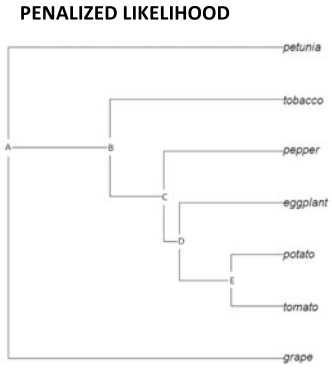
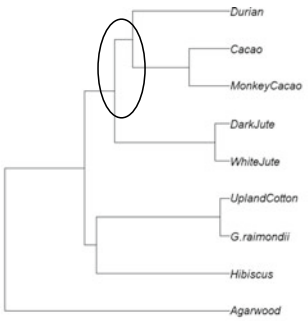


Fig. 8.4 Ultrametric, NJ tree and its transformations by the Farris, nonparametric rate smoothing and penalized likelihood methods, for the family Solanaceae and outgroup grape. "Ratios" summarize transformed branch lengths versus original NJ values

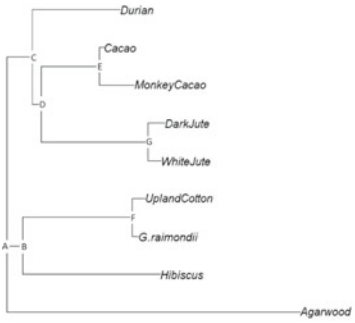
UPGMA



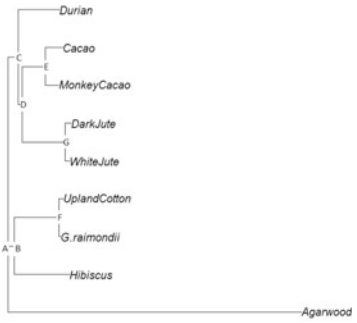
RATIOS

	Farris	NPRS	PL
Agarwood	1.24	0.67	0.67
Hibiscus	0.49	1.13	1.25
G. raimondii	0.35	5.68	1.96
Upland Cotton	0.43	2.79	0.96
White Jute	0.42	2.94	1.31
Dark Jute	0.44	2.20	0.98
Monkey Cacao	0.47	2.03	1.46
Cacao	0.48	1.58	1.13
Durian	0.58	1.82	1.65
A-B	0.50	2.66	1.61
B-F	0.50	1.08	1.46
A-C	0.50	1.44	2.05
C-D	0.50	1.86	1.34
D-G	0.50	1.00	1.10
D-E	0.50	1.26	1.43

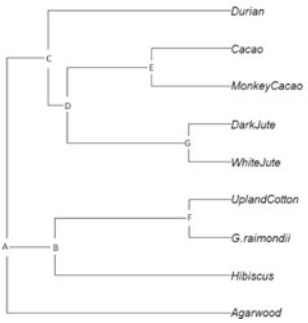
NEIGHBOUR JOINING



FARRIS



NONPARAMETRIC RATE SMOOTHING



PENALIZED LIKELIHOOD

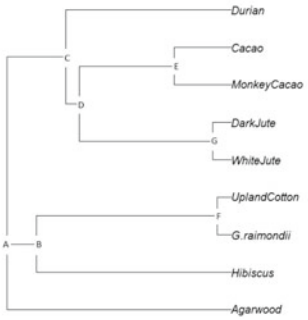


Fig. 8.5 Ultrametric, NJ tree and its transformations by the Farris, nonparametric smoothing and penalized likelihood methods, for the family Malvaceae and outgroup agarwood. "Ratios" summarize transformed branch lengths versus original NJ values

In examining the branch length ratios, there is little that emerges that is consistent across all three methods, except that *G. raimondii* is somewhat conservative. Nonparametric rate smoothing and penalized likelihood both have differential rates of evolution at the origin of the two main clades, but they differ on which side is more conservative, with nonparametric rate smoothing pointing to the cacao side, and with penalized likelihood suggesting the cotton/hibiscus clade. In addition, the ancestor of the jute species appears to have undergone accelerated evolution.

8.5 Discussion

Preliminary to the evaluation of the transformation protocols, we demonstrated in all three evolutionary domains that the NJ with the peaks approach recovers the correct topology for the phylogenetic tree, and that the UPGMA was not able to. (Although the correct topology is not known for the Malvaceae, we do know that the UPGMA result was not consistent with the additive results.)

In applying the three transformations, the two that incorporate smoothing between coincident branches were clearly preferable to the Farris method. In addition, penalized likelihood stood out as being able to achieve ultrametric status with the least stretching or contracting of branches.

The Farris transform displaces much of the rate variation to the branch leading to the outgroup. In the case of the fish phylogeny in Fig. 8.2, this could not be achieved because the outgroup was too conservative, and other branches could not be shortened enough by the transformation without becoming negative.

Finally, we have demonstrated that the non-ultrametric status of additive trees, at least on these data sets, may be resolved by identifying one or two branches on which evolutionary rate has clearly slowed or accelerated. It would be useful to be able to confirm these cases with data on individual gene trees.

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

References

1. Amemiya, C.T., Alföldi, J., Lee, A.P., Fan, S., Philippe, H., MacCallum, I., Braasch, I., Manousaki, T., Schneider, I., Rohner, N., Organ, C., Chalopin, D., Smith, J.J., Robinson, M., Dorrington, R.A., Gerdol, M., Aken, B., Biscotti, M.A., Barucca, M., Baurain, D., Berlin, A.M., Blatch, G.L., Buonocore, F., Burmester, T., Campbell, M.S., Canapa, A., Cannon, J.P., Christoffels, A., De Moro, G., Edkins, A.L., Fan, L., Fausto, A.M., Feiner, N., Forconi, M., Gamieldien, J., Gnerre, S., Gnirke, A., Goldstone, J.V., Haerty, W., Hahn, M.E., Hesse, U., Hoffmann, S., Johnson, J., Karchner, S.I., Kuraku, S., Lara, M., Levin, J.Z., Litman, G.W., Mauceli, E., Miyake, T., Mueller, M.G., Nelson, D.R., Nitsche, A., Olmo, E., Ota, T., Pallavicini, A., Panji, S., Picone, B., Ponting, C.P., Prohaska, S.J., Przybylski, D., Saha, N.R., Ravi, V., Ribeiro,

F.J., Sauka-Spengler, T., Scapigliati, G., Searle, S.M.J., Sharpe, T., Simakov, O., Stadler, P.F., Stegeman, J.J., Sumiyama, K., Tabbaa, D., Tafer, H., Turner-Maier, J., van Heusden, P., White, S., Williams, L., Yandell, M., Brinkmann, H., Volff, J.N., Tabin, C.J., Shubin, N., Schartl, M., Jaffe, D.B., Postlethwait, J.H., Venkatesh, B., Di Palma, F., Lander, E.S., Meyer, A., Lindblad-Toh, K.: The African coelacanth genome provides insights into tetrapod evolution. Nature **496**, 311 (2013)

2. Aparicio, S., Chapman, J., Stupka, E., Putnam, N., Chia, J.m., Dehal, P., Christoffels, A., Rash, S., Hoon, S., Smit, A., Gelpke, M.D.S., Roach, J., Oh, T., Ho, I.Y., Wong, M., Detter, C., Verhoef, F., Predki, P., Tay, A., Lucas, S., Richardson, P., Smith, S.F., Clark, M.S., Edwards, Y.J.K., Doggett, N., Zharkikh, A., Tavtigian, S.V., Pruss, D., Barnstead, M., Evans, C., Baden, H., Powell, J., Glusman, G., Rowen, L., Hood, L., Tan, Y.H., Elgar, G., Hawkins, T., Venkatesh, B., Rokhsar, D., Brenner, S.: Whole-genome shotgun assembly and analysis of the genome of *Fugu rubripes*. Science **297**(5585), 1301–1310 (2002). https://doi.org/10.1126/science.1072104
3. Bandelt, H.J.: Recognition of tree metrics. SIAM J. Discrete Math. **3**(1), 1–6 (1990)
4. Betancur-Rodriguez, R., Ortí, G., Pyron, R.A., Morlon, H.: Fossil-based comparative analyses reveal ancient marine ancestry erased by extinction in ray-finned fishes. Ecol. Lett. **18**(5), 441–450 (2015). https://doi.org/10.1111/ele.12423
5. Betancur-Rodriguez, R., Wiley, E.O., Arratia, G., Acero, A., Bailly, N., Miya, M., Lecointre, G., Ortí, G.: Phylogenetic classification of bony fishes. BMC Evol. Biol. **17**(1), 162 (2017). https://doi.org/10.1186/s12862-017-0958-3
6. Braasch, I., Gehrke, A.R., Smith, J.J., Kawasaki, K., Manousaki, T., Pasquier, J., Amores, A., Desvignes, T., Batzel, P., Catchen, J., Berlin, A.M., Campbell, M.S., Barrell, D., Martin, K.J., Mulley, J.F., Ravi, V., Lee, A.P., Nakamura, T., Chalopin, D., Fan, S., Wcisel, D., Cañestro, C., Sydes, J., Beaudry, F.E.G., Sun, Y., Hertel, J., Beam, M.J., Fasold, M., Ishiyama, M., Johnson, J., Kehr, S., Lara, M., Letaw, J.H., Litman, G.W., Litman, R.T., Mikami, M., Ota, T., Saha, N.R., Williams, L., Stadler, P.F., Wang, H., Taylor, J.S., Fontenot, Q., Ferrara, A., Searle, S.M.J., Aken, B., Yandell, M., Schneider, I., Yoder, J.A., Volff, J.N., Meyer, A., Amemiya, C.T., Venkatesh, B., Holland, P.W.H., Guiguen, Y., Bobe, J., Shubin, N.H., Di Palma, F., Alföldi, J., Lindblad-Toh, K., Postlethwait, J.H.: The spotted gar genome illuminates vertebrate evolution and facilitates human-teleost comparisons. Nat. Genet. **48**, 427 (2016)
7. Brawand, D., Wagner, C.E., Li, Y.I., Malinsky, M., Keller, I., Fan, S., Simakov, O., Ng, A.Y., Lim, Z.W., Bezault, E., Turner-Maier, J., Johnson, J., Alcazar, R., Noh, H.J., Russell, P., Aken, B., Alföldi, J., Amemiya, C., Azzouzi, N., Baroiller, J.F., Barloy-Hubler, F., Berlin, A., Bloomquist, R., Carleton, K.L., Conte, M.A., D'Cotta, H., Eshel, O., Gaffney, L., Galibert, F., Gante, H.F., Gnerre, S., Greuter, L., Guyon, R., Haddad, N.S., Haerty, W., Harris, R.M., Hofmann, H.A., Hourlier, T., Hulata, G., Jaffe, D.B., Lara, M., Lee, A.P., MacCallum, I., Mwaiko, S., Nikaido, M., Nishihara, H., Ozouf-Costaz, C., Penman, D.J., Przybylski, D., Rakotomanga, M., Renn, S.C.P., Ribeiro, F.J., Ron, M., Salzburger, W., Sanchez-Pulido, L., Santos, M.E., Searle, S., Sharpe, T., Swofford, R., Tan, F.J., Williams, L., Young, S., Yin, S., Okada, N., Kocher, T.D., Miska, E.A., Lander, E.S., Venkatesh, B., Fernald, R.D., Meyer, A., Ponting, C.P., Streelman, J.T., Lindblad-Toh, K., Seehausen, O., Di Palma, F.: The genomic substrate for adaptive radiation in African cichlid fish. Nature **513**, 375 (2014)
8. Chen, S., Zhang, G., Shao, C., Huang, Q., Liu, G., Zhang, P., Song, W., An, N., Chalopin, D., Volff, J.N., Hong, Y., Li, Q., Sha, Z., Zhou, H., Xie, M., Yu, Q., Liu, Y., Xiang, H., Wang, N., Wu, K., Yang, C., Zhou, Q., Liao, X., Yang, L., Hu, Q., Zhang, J., Meng, L., Jin, L., Tian, Y., Lian, J., Yang, J., Miao, G., Liu, S., Liang, Z., Yan, F., Li, Y., Sun, B., Zhang, H., Zhang, J., Zhu, Y., Du, M., Zhao, Y., Schartl, M., Tang, Q., Wang, J.: Whole-genome sequence of a flatfish provides insights into ZW sex chromosome evolution and adaptation to a benthic lifestyle. Nat. Genet. **46**, 253 (2014)
9. Dress, A., Huber, K.T., Moulton, V.: Some uses of the Farris transform in mathematics and phylogenetics—a review. Ann. Comb. **11**, 1–37 (2007)
10. Farris, J.: On the phenetic approach to vertebrate classification. In: Major Patterns in Vertebrate Evolution, pp. 823–850 (1997)

11. Green, P., Silverman, B.: Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach. Chapman and Hall (1994)
12. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press (1997)
13. Harrington, R.C., Faircloth, B.C., Eytan, R.I., Smith, W.L., Near, T.J., Alfaro, M.E., Friedman, M.: Phylogenomic analysis of crangimorph fishes reveals flatfish asymmetry arose in a blink of the evolutionary eye. BMC Evol. Biol. **16**(1), 224 (2016). https://doi.org/10.1186/s12862-016-0786-x
14. Howe, K., Clark, M.D., Torroja, C.F., Torrance, J., Berthelot, C., Muffato, M., Collins, J.E., Humphray, S., McLaren, K., Matthews, L., McLaren, S., Sealy, I., Caccamo, M., Churcher, C., Scott, C., Barrett, J.C., Koch, R., Rauch, G.J., White, S., Chow, W., Kilian, B., Quintais, L.T., Guerra-Assunção, J., Zhou, Y., Gu, Y., Yen, J., Vogel, J.H., Eyre, T., Redmond, S., Banerjee, R., Chi, J., Fu, B., Langley, E., Maguire, S.F., Laird, G.K., Lloyd, D., Kenyon, E., Donaldson, S., Sehra, H., Almeida-King, J., Loveland, J., Trevanion, S., Jones, M., Quail, M., Willey, D., Hunt, A., Burton, J., Sims, S., McLay, K., Plumb, B., Davis, J., Clee, C., Oliver, K., Clark, R., Riddle, C., Elliott, D., Threadgold, G., Harden, G., Ware, D., Begum, S., Mortimore, B., Kerry, G., Heath, P., Phillimore, B., Tracey, A., Corby, N., Dunn, M., Johnson, C., Wood, J., Clark, S., Pelan, S., Griffiths, G., Smith, M., Glithero, R., Howden, P., Barker, N., Lloyd, C., Stevens, C., Harley, J., Holt, K., Panagiotidis, G., Lovell, J., Beasley, H., Henderson, C., Gordon, D., Auger, K., Wright, D., Collins, J., Raisen, C., Dyer, L., Leung, K., Robertson, L., Ambridge, K., Leongamornlert, D., McGuire, S., Gilderthorp, R., Griffiths, C., Manthravadi, D., Nichol, S., Barker, G., Whitehead, S., Kay, M., Brown, J., Murnane, C., Gray, E., Humphries, M., Sycamore, N., Barker, D., Saunders, D., Wallis, J., Babbage, A., Hammond, S., Mashreghi-Mohammadi, M., Barr, L., Martin, S., Wray, P., Ellington, A., Matthews, N., Ellwood, M., Woodmansey, R., Clark, G., Cooper, J.D., Tromans, A., Grafham, D., Skuce, C., Pandian, R., Andrews, R., Harrison, E., Kimberley, A., Garnett, J., Fosker, N., Hall, R., Garner, P., Kelly, D., Bird, C., Palmer, S., Gehring, I., Berger, A., Dooley, C.M., Ersan-Ürün, Z., Eser, C., Geiger, H., Geisler, M., Karotki, L., Kirn, A., Konantz, J., Konantz, M., Oberländer, M., Rudolph-Geiger, S., Teucke, M., Lanz, C., Raddatz, G., Osoegawa, K., Zhu, B., Rapp, A., Widaa, S., Langford, C., Yang, F., Schuster, S.C., Carter, N.P., Harrow, J., Ning, Z., Herrero, J., Searle, S.M.J., Enright, A., Geisler, R., Plasterk, R.H.A., Lee, C., Westerfield, M., de Jong, P.J., Zon, L.I., Postlethwait, J.H., Nüsslein-Volhard, C., Hubbard, T.J.P., Crollius, H.R., Rogers, J., Stemple, D.L.: The zebrafish reference genome sequence and its relationship to the human genome. Nature **496**, 498 (2013)
15. Jaillon, O., Aury, J.M., Brunet, F., Petit, J.L., Stange-Thomann, N., Mauceli, E., Bouneau, L., Fischer, C., Ozouf-Costaz, C., Bernot, A., Nicaud, S., Jaffe, D., Fisher, S., Lutfalla, G., Dossat, C., Segurens, B., Dasilva, C., Salanoubat, M., Levy, M., Boudet, N., Castellano, S., Anthouard, V., Jubin, C., Castelli, V., Katinka, M., Vacherie, B., Biémont, C., Skalli, Z., Cattolico, L., Poulain, J., de Berardinis, V., Cruaud, C., Duprat, S., Brottier, P., Coutanceau, J.P., Gouzy, J., Parra, G., Lardier, G., Chapple, C., McKernan, K.J., McEwan, P., Bosak, S., Kellis, M., Volff, J.N., Guigó, R., Zody, M.C., Mesirov, J., Lindblad-Toh, K., Birren, B., Nusbaum, C., Kahn, D., Robinson-Rechavi, M., Laudet, V., Schachter, V., Quétier, F., Saurin, W., Scarpelli, C., Wincker, P., Lander, E.S., Weissenbach, J., Roest Crollius, H.: Genome duplication in the teleost fish Tetraodon nigroviridis reveals the early vertebrate proto-karyotype. Nature **431**, 946 (2004)
16. Jones, F.C., Grabherr, M.G., Chan, Y.F., Russell, P., Mauceli, E., Johnson, J., Swofford, R., Pirun, M., Zody, M.C., White, S., Birney, E., Searle, S., Schmutz, J., Grimwood, J., Dickson, M.C., Myers, R.M., Miller, C.T., Summers, B.R., Knecht, A.K., Brady, S.D., Zhang, H., Pollen, A.A., Howes, T., Amemiya, C., Team, B.I.G.S.P..W.G.A., Lander, E.S., Di Palma, F., Lindblad-Toh, K., Kingsley, D.M.: The genomic basis of adaptive evolution in threespine sticklebacks. Nature **484**, 55 (2012)
17. Krane, D., Raymer, M.: Fundamental Concepts of Bioinformatics. The Genetics Place Series. Benjamin Cummings (2003)

18. Künstner, A., Hoffmann, M., Fraser, B.A., Kottler, V.A., Sharma, E., Weigel, D., Dreyer, C.: The genome of the Trinidadian guppy, Poecilia reticulata, and variation in the Guanapo population. PLoS ONE **11**(12), 1–25 (2016). https://doi.org/10.1371/journal.pone.0169087
19. Lien, S., Koop, B.F., Sandve, S.R., Miller, J.R., Kent, M.P., Nome, T., Hvidsten, T.R., Leong, J.S., Minkley, D.R., Zimin, A., Grammes, F., Grove, H., Gjuvsland, A., Walenz, B., Hermansen, R.A., von Schalburg, K., Rondeau, E.B., Di Genova, A., Samy, J.K.A., Olav Vik, J., Vigeland, M.D., Caler, L., Grimholt, U., Jentoft, S., Inge Våge, D., de Jong, P., Moen, T., Baranski, M., Palti, Y., Smith, D.R., Yorke, J.A., Nederbragt, A.J., Tooming-Klunderud, A., Jakobsen, K.S., Jiang, X., Fan, D., Hu, Y., Liberles, D.A., Vidal, R., Iturra, P., Jones, S.J.M., Jonassen, I., Maass, A., Omholt, S.W., Davidson, W.S.: The Atlantic salmon genome provides insights into rediploidization. Nature **533**, 200 (2016)
20. Lyons, E., Freeling, M.: How to usefully compare homologous plant genes and chromosomes as DNA sequences. Plant J. **53**(4), 661–673 (2008). https://doi.org/10.1111/j.1365-313X.2007.03326.x
21. Lyons, E., Pedersen, B., Kane, J., Alam, M., Ming, R., Tang, H., Wang, X., Bowers, J., Paterson, A., Lisch, D.M., Freeling, M.: Finding and comparing syntenic regions among *arabidopsis* and the outgroups papaya, poplar and grape: CoGe with rosids. Plant Physiol. **148**, 1772–1781 (2008)
22. Lyons, E., Pedersen, B., Kane, J., Freeling, M.: The value of nonmodel genomes and an example using synmap within coge to dissect the hexaploidy that predates the rosids. Trop. Plant Biol. **1**(3), 181–190 (2008). https://doi.org/10.1007/s12042-008-9017-y
23. McGaugh, S.E., Gross, J.B., Aken, B., Blin, M., Borowsky, R., Chalopin, D., Hinaux, H., Jeffery, W.R., Keene, A., Ma, L., Minx, P., Murphy, D., O'Quin, K.E., Rétaux, S., Rohner, N., Searle, S.M.J., Stahl, B.A., Tabin, C., Volff, J.N., Yoshizawa, M., Warren, W.C.: The cavefish genome reveals candidate genes for eye loss. Nat. Commun. **5**, 5307 (2014)
24. Michener, C.D., Sokal, R.R.: A quantitative approach to a problem in classification. Evolution **11**(2), 130–162 (1957). https://doi.org/10.1111/j.1558-5646.1957.tb02884.x
25. Pan, H., Yu, H., Ravi, V., Li, C., Lee, A.P., Lian, M.M., Tay, B.H., Brenner, S., Wang, J., Yang, H., Zhang, G., Venkatesh, B.: The genome of the largest bony fish, ocean sunfish (Mola mola), provides insights into its fast growth rate. GigaScience **5**(1), 36 (2016). https://doi.org/10.1186/s13742-016-0144-3
26. Paradis, E., Claude, J., Strimmer, K.: APE: analyses of phylogenetics and evolution in R language. Bioinformatics **20**, 289–290 (2004)
27. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013). http://www.R-project.org/
28. Rondeau, E.B., Minkley, D.R., Leong, J.S., Messmer, A.M., Jantzen, J.R., von Schalburg, K.R., Lemon, C., Bird, N.H., Koop, B.F.: The genome and linkage map of the northern pike (Esox lucius): conserved synteny revealed between the salmonid sister group and the neoteleostei. PLoS ONE **9**(7), 1–18 (2014). https://doi.org/10.1371/journal.pone.0102089
29. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4), 406–425 (1987)
30. Sanciangco, M.D., Carpenter, K.E., Betancur-Rodriguez, R.: Phylogenetic placement of enigmatic Percomorph families (teleostei: Percomorphaceae). Mol. Phylogenet. Evol. **94**, 565–576 (2016). https://doi.org/10.1016/j.ympev.2015.10.006
31. Sanderson, M.J.: A nonparametric approach to estimating divergence times in the absence of rate constancy. Mol. Biol. Evol. **14**(12), 1218–1231 (1997)
32. Sanderson, M.J.: Estimating absolute rates of molecular evolution and divergence times: a penalized likelihood approach. Mol. Biol. Evol. **19**(1), 101–109 (2002). https://doi.org/10.1093/oxfordjournals.molbev.a003974
33. Sankoff, D., Zheng, C., Zhang, Y., Meidanis, J., Lyons, E., Tang, H.: Models for similarity distributions of syntenic homologs and applications to phylogenomics. IEEE/ACM Trans. Comput. Biol. Bioinform. (2018). https://doi.org/10.1109/TCBB.2018.2849377
34. Schartl, M., Walter, R.B., Shen, Y., Garcia, T., Catchen, J., Amores, A., Braasch, I., Chalopin, D., Volff, J.N., Lesch, K.P., Bisazza, A., Minx, P., Hillier, L., Wilson, R.K., Fuerstenberg, S.,

Boore, J., Searle, S., Postlethwait, J.H., Warren, W.C.: The genome of the platyfish, Xiphophorus maculatus, provides insights into evolutionary adaptation and several complex traits. Nat. Genet. **45**, 567 (2013)
35. Venkatesh, B., Lee, A.P., Ravi, V., Maurya, A.K., Lian, M.M., Swann, J.B., Ohta, Y., Flajnik, M.F., Sutoh, Y., Kasahara, M., Hoon, S., Gangu, V., Roy, S.W., Irimia, M., Korzh, V., Kondrychyn, I., Lim, Z.W., Tay, B.H., Tohari, S., Kong, K.W., Ho, S., Lorente-Galdos, B., Quilez, J., Marques-Bonet, T., Raney, B.J., Ingham, P.W., Tay, A., Hillier, L.W., Minx, P., Boehm, T., Wilson, R.K., Brenner, S., Warren, W.C.: Elephant shark genome provides unique insights into Gnathostome evolution. Nature **505**, 174 (2014)
36. Zhang, Y., Zheng, C., Islam, S., Kim, Y.M., Sankoff, D.: Branching out to speciation with a birth-and-death model of fractionation: the Malvaceae. IEEE/ACM Trans. Comput. Biol. Bioinform. (2019)
37. Zhang, Y., Zheng, C., Sankoff, D.: Speciation and rate variation in a birth-and-death account of WGD and fractionation; the case of Solanaceae. In: Proceedings of the 16th RECOMB Comparative Genomics Satellite Workshop. Lecture Notes in Computer Science 11183 (2018)

Chapter 9
Ancestral Genome Reconstruction

Jijun Tang

Abstract Reconstruction of extinct ancestral genomes is an important topic in comparative genomics and has a wide range of applications. By comparing a current-day species against its ancestor, we can deduce how it differs from the ancestor and infer detailed information about the evolution of species. With more and more fully sequenced genomes becoming available, we are able to reconstruct ancestors at the whole genome level by using evolutionary events such as genome rearrangements, gene insertions, deletions and duplications. In this chapter, we will present the concepts related to whole genome evolution and ancestral reconstruction. We will review evolutionary models and algorithms in pairwise comparison of genomes, computing of the median problem and optimizations in inferring phylogenies and ancestors from multiple genomes.

Keywords Ancestral reconstruction · Phylogenetics · Genome rearrangement · Median problem · Double-cut-and-joining

9.1 Introduction

Given a phylogeny with modern species as leaves, the problem of ancestral reconstruction is to infer genomes on all internal nodes of the tree. These reconstructions can be achieved at different levels: we can infer ancestral DNA sequences based on base-pair mutations, or we can infer ancestral genomic structures (both gene content and orderings) based on events such as inversions and duplications, which is the focus of this chapter. Ancestral genome reconstruction is important because by comparing a current-day species with its ancestor, we can deduce how it differs from the ancestor and infer detailed information about the evolution of species.

With the advent of new sequencing and computing technologies, more and more whole genome sequences have become available. After gene homologs can be identi-

J. Tang (✉)
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA
e-mail: jtang@cse.sc.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_9

fied, we can assign each gene family a unique integer and represent each chromosome with an ordering of signed integers, where the sign indicates the strand. The orderings and contents of genomes can be changed under rearrangement of genes (such as reversal and transposition), as well as other operations such as duplications, deletions and insertions.

Reconstructing phylogeny and ancestors at the whole genome level is computationally much harder than computing with sequence data. For example, the classical maximum parsimony problem is solvable in polynomial time when the phylogeny is given (i.e., given a phylogeny with leaves labeled with DNA or protein sequences, find the optimal ancestral sequences on all internal nodes to minimize the sum of the edit distances along all edges) [14]. However, even the simplest corresponding problems are NP-hard for gene-order data when rearrangements are allowed, even when the input tree has only three leaves [1, 35].

Existing ancestral reconstruction methods generally assume there already exists a phylogeny for the given species, which can be used as a guide tree. This approach is known as the small phylogeny problem (SPP) [11, 15, 21, 26]. On the other hand, the big phylogeny problem (BPP) builds the most appropriate phylogeny first and then uses it to reconstruct the ancestral genomes [19, 34, 46].

One of the main research areas of Dr. Moret is to develop new algorithms and models for phylogenetic reconstruction and ancestral inference from genome rearrangement data [2, 10, 23, 24, 28, 32–34, 39, 40, 43, 44, 46]. Over the past few years, pushed by Dr. Moret and other researchers, many methods have been developed [13, 30], ranging from pairwise genome comparison to large-scale computation on multiple genomes [5, 15, 17, 21, 23, 27, 34, 36, 43, 46]. In this chapter, we will present the fundamental concepts of genome rearrangements and review some of the most important algorithms.

9.2 Definitions

A genome can be represented by the signed ordering of a set of n genes $\{1, 2, \ldots, n\}$ on one or more chromosomes, where signs are used to indicate the strandedness of genes. Each gene is assigned with an orientation that is either positive or negative. The gene order of a genome can be changed through genome rearrangement events such as reversal, transposition, and translocations. There are other events, such as gene insertions, deletions and duplications, that change the gene content of a genome as well. The following are some of the important definitions.

Adjacency: Two genes i and j are *adjacent* if i is followed by j, or, $-j$ is followed by $-i$. Each gene can be split into two parts, the head (denote i^h) and the tail (denote i^t). Depending on the signs of two consecutive genes i and j, there are a total of four different scenarios in forming an *adjacency*: $\{i^t, j^t\}$, $\{i^h, i^t\}$, $\{i^t, i^h\}$, and $\{i^h, i^h\}$. Gene i at the end of a linear chromosome forms a *telomere* that is either $\{i^t\}$ or $\{i^h\}$.

Reversal (inversion): Let G be a genome with ordering $\{1, 2, \ldots, n\}$. A *reversal* (also called *inversion*) between indices i and j $(i \leq j)$ produces a new genome G'

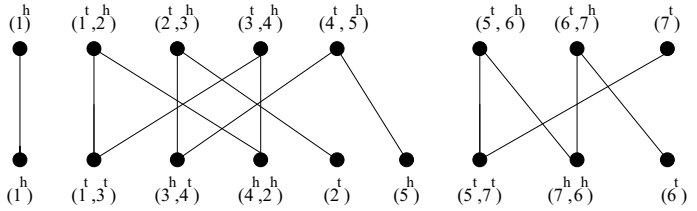


Fig. 9.1 G_1 has one linear chromosome (1, 2, 3, 4, 5, 6, 7) and G_2 has two linear chromosomes (1, −3, −4, 2) and (−5, −7, 6). In $AG(G_1, G_2), n = 7, C = 1, I = 2$, the DCJ distance of G_1 and G_2 is 5

with linear ordering

$$1, 2, \ldots, i-1, -j, -(j-1), \ldots, -i, j+1, \ldots, n.$$

Transposition: Given genome G and three indices i, j, k, with $i \leq j$ and $k \notin [i, j]$, a *transposition* picks up genes in the interval $i, \ldots, j$ and inserts them immediately after k, produces a new genome (assume $k > j$) with linear ordering

$$1, \ldots, i-1, j+1, \ldots, k, i, i+1, \ldots, j, k+1, \ldots, n.$$

Insertion, deletion, and duplication: An *insertion* is the acquisition of one or more genes and a *deletion* is the loss of these genes. A *duplication* copies a section of the chromosome and inserts it immediately after the original segment (*tandem duplication*) or somewhere else (*transposed duplication)*.

Translocation, fission and fusion: For multi-chromosomal genomes, there are additional operations that involve two chromosomes: a *translocation* exchanges one end segment of a chromosome with one end segment in the other chromosome, a *fission* splits one chromosome into two, and a *fusion* combines two chromosomes into one).

Breakpoint: Given two genomes G_1 and G_2, a breakpoint is defined as a pair of genes (i, j) such that i and j are adjacent in G_1 but not in G_2.

Genomic distance and sorting: Given two genomes G_1 and G_2, the *edit distance* is defined as the minimum number of events required to transform G_1 into G_2. The genomic *sorting problem* is to find the shortest sequence of events that transform one genome into the other.

Adjacency graph: Given two genomes G_1 and G_2, the *adjacency graph* $AG(G_1, G_2)$ has vertices from all adjacencies and telomeres of the two genomes [4]. An edge is created between a vertex $u \in G_1$ and a vertex $v \in G_2$ if they share the same part (head or tail) of a gene (Fig. 9.1).

Median problem: Given three genomes, the *median problem* is to find a single (median) genome that minimizes the sum of pairwise distances between itself and each of the three input genomes (Fig. 9.2 Left). The median problem is NP-hard for most distance models [8, 35, 47].

9.3 Pairwise Distance and Sorting

Hannenhalli and Pevzner developed the first polynomial-time algorithm to compute the *reversal distance* [16]. Dr. Moret and colleagues developed the first linear algorithm to compute the reversal distance [2]. Yancopoulos et al. [49] proposed a universal double-cut-and-join (DCJ) operation that accounts for all rearrangement events including reversals, transpositions, translocations, fissions, and fusions. The reversal and DCJ models have been extended to handle other events such as insertions, deletions, and duplications, including models from Dr. Moret and colleagues [3, 9, 24, 25, 28, 38, 39, 41].

The most important data structure for dealing with DCJ events is the adjacency graph $AG(G_1, G_2)$ between genomes G_1 and G_2, whose vertices are the adjacencies and telomeres. An example of an adjacency graph is shown in Fig. 9.1, which consists of *paths* and *cycles*. There are two types of paths: paths starting from one genome and ending at the other are named as *odd path* because they always have an odd length, while paths starting from and ending at the same genome are named as *even paths* because they always have an even length.

Given two vertices u and v of the adjacency graph, a DCJ operation cuts the graph at the two vertices and rejoins the (new) four ends in one of the four ways [4]

- If both vertices are telomeres, then can be replaced by a new adjacency; i.e., if $u = \{q\}$ and $v = \{r\}$, they can be replaced by $\{q, r\}$.
- An adjacency $\{q, r\}$ can be replaced by two new telomeres $\{q\}$ and $v = \{r\}$.
- If one vertex is an adjacency and the other is a telomere, they will be replaced by a new adjacency and new telomere. For example, if $u = \{p, q\}$ (an adjacency) and $v = \{r\}$ (a telomere), they will be replaced either by $\{p, r\}$ and $\{q\}$, or by $\{q, r\}$ and $\{p\}$.
- If both are adjacencies, they will be replaced by two new adjacencies. For example, if $u = \{p, q\}$ and $v = \{r, s\}$ the two new adjacencies are $\{p, r\}$ and $\{q, s\}$, or $\{p, s\}$ and $\{r, q\}$.

The DCJ distance can be computed based on the number of cycles and odd paths [4].

Theorem 1 *The DCJ distance for two genomes G_1 and G_2 is $d_{\mathrm{DCJ}}(G_1, G_2) = n - (C + I/2)$ where C is the number of cycles and I is the number of odd paths in the adjacency graph $AG(G_1, G_2)$.*

Compared to reversals, sorting by DCJs is much simpler. A single DCJ operation on the adjacency graph changes its number of odd paths by -2, 0 or 2, or changes its number of cycles by -1, 0 or 1 [4, 7]. As a result, a DCJ operation can bring G_2 one step closer to G_1, keep them at the same distance or make it one step away. A DCJ operation is defined as *optimal* if it decreases the number of cycle or the number of odd paths to decrease the distance, *neutral* if it does not change these numbers, and *bad* if it increases these numbers and increases the distance.

There are several algorithms that provably find a shortest DCJ sorting path between two genomes [4, 7]. Bergeron et al. [4] proposed a simple greedy algorithm for sorting by DCJ, which iteratively identifies an adjacency in one genome but not in the other and applies a DCJ operation that creates the missing adjacency in the first genome. As a result, this greedy algorithm always increases the number of cycles by 1. Obviously, this greedy approach may miss many sorting scenarios. Braga and Stoye [7] proposed a more general algorithm that can identify all sorting scenarios by splitting large cycles into two smaller cycles, splitting a long path into a shorter path and a new cycle, or combining two even paths that do not start from the same genome.

9.4 Solving the Median Problem

The median problem deals with the smallest unrooted binary tree, using the three input genomes as current-day species and the inferred median genome as an estimation of their common (extinct) ancestor. For the example shown in Fig. 9.2 (Left), the gene order (−5 4 1 2 3) gives the minimum median score, and thus can be treated as the ancestral gene order of the three input genomes.

Given an unrooted tree T with three or more leaves, let $w(T)$ be the score of the tree, defined to be the sum of the total edge lengths given the best possible assignment of genomes to the internal nodes of T. If $1, 2, \ldots, n$ is a circular-ordering of the leaves of T (Fig. 9.2), then we have $w(T) \geq \frac{d_{1,2}+d_{2,3}+\cdots+d_{n,1}}{2}$. This lower bound on the median score is called the *perfect median score* (Fig. 9.2 Left).

In the past years, many median solvers have been developed. For example, the one proposed by Caprara [8] uses a branch-and-bound approach that enumerates all possible gene orders and eliminates bad branches based on some lower bounds. The solver proposed by Xu and Sankoff [48] is based on the decomposition of multiple breakpoint graphs into adequate sub-graphs, enabling efficient and exact algorithms for the DCJ median problem. Dr. Moret and colleagues proposed several median solvers, including a solver based on sorting reversals and several heuristics [31, 37, 40].

Here we briefly review the new median solver by Xia et al. [45], which uses the concept of simulated annealing [29], which has been widely used in optimization and is inspired from the annealing technique in metallurgy to simulate the heating and controlled cooling of a material to reduce defects.

The major challenge in applying the concept of simulated annealing in solving the DCJ median problem is to deal with the large search space: given n genes, the possible number of gene orders is $n!2^n$–it will be much larger if other events such as deletion and insertion are considered. Xia et al. [45] utilized genomic sorting to control the search so that it can converge faster toward good genomic structures.

The simulated annealing algorithm contains three phases: (1) it first initializes the median genome; (2) it then generates the next possible genome and evaluates

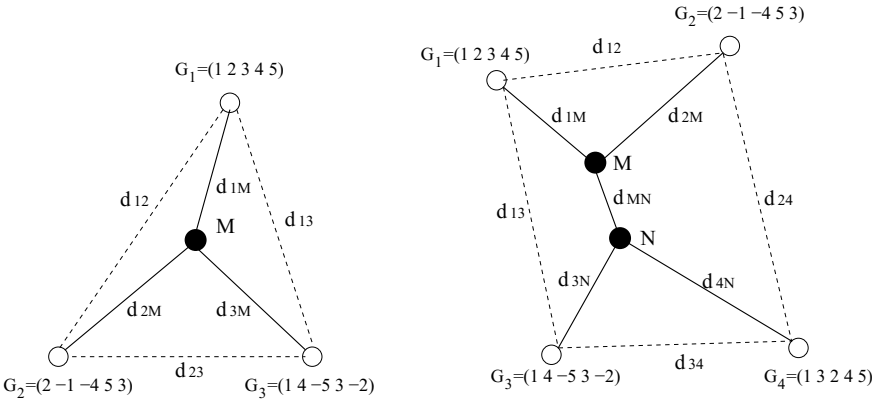


Fig. 9.2 Left: the median problem on three genomes G_1, G_2, G_3. The median score is $d_{1M} + d_{2M} + d_{3M}$ and the perfect median score is $\frac{d_{12}+d_{23}+d_{13}}{2}$. The resulted median genome is $G_M = (-5\ 4\ 1\ 2\ 3)$. Right: scoring a tree t on four genomes G_1, G_2, G_3, G_4. The tree score is $w(t) = d_{1M} + d_{2M} + d_{MN} + d_{3N} + d_{4N}$, where M and N are internal nodes. The circular-ordering lower bound is $w(t) \geq \frac{d_{13}+d_{34}+d_{24}+d_{12}}{2}$

whether to accept or reject the new candidate; (3) it will iterate over the previous two steps until converging or exceeding the maximum number of iterations.

The initial state (median candidate) and temperature can be chosen based on the observation that the median genome may be on a sorting path from one leaf genome to another. There are six pairs of leaf genomes, we randomly choose one of them (G_i and G_j) and apply m ($m \leq d_{ij}$) optimal DCJ operations on G_i to obtain G_i^k, which is used as the initial median for the next generation.

The neighboring state can be produced similarly: from the second generation, let M be the current median genome, we randomly pick a leaf genome and apply m (randomly chosen) optimal DCJ operations from M to it and use the resulted genome for the next generation.

The new potential candidate of the median genome is compared against the current median. The candidate will be accepted it if has a lower median score than the current solution. If the candidate has a worse median score, it may still be accepted, but the chance is based on the current temperature: higher temperature means there is a higher chance that a worse solution will be accepted. Let T be the current temperature and ΔE be the difference from the new (worse) median score and the current (better) median score, the possibility of acceptance is defined as

$$Acceptance = \begin{cases} \exp -\Delta E/T & \text{if } \Delta E \geq 0 \\ 1 & \text{if } \Delta E < 0 \end{cases} \tag{9.1}$$

The cooling schedule directly impacts the performance. After experiments on simulations, Xia et al. adopted the Exponential Multiplicative Cooling scheme [22]: letting T_0 be the initial temperature, T_k be the temperature after k iterations, and α

be the cooling rate, we have

$$T_k = T_0 \cdot \alpha^k \qquad (0.8 \leq \alpha \leq 0.9)$$

During each iteration, the algorithm proposes a new state, accepts or rejects it based on the current temperature, and cools down for the next step. The maximum number of iterations for the simulated annealing solver is set as K at the start, but the process could be terminated early if the perfect median score is achieved (Fig. 9.2).

Xia et al. [45] conducted extensive experiments on simulated datasets and found that the new algorithm is very efficient and has high accuracy. This framework is very flexible and can be easily extended to handle other events as many algorithms for sorting have been proposed.

9.4.1 Computing with Multiple Genomes

Given more than three genomes and a phylogeny, we can estimate the ancestors by iteratively solving the median problem defined on the internal nodes. Figure 9.2 (Right) illustrates an example, where the two internal nodes M and N each defines a median problem with their neighboring nodes. For example, genomes G_1, G_2, and N define the median problem on node M, while genomes G_3, G_4, and M define the median problem on node N.

Blanchette and Sankoff formulated the breakpoint distance phylogeny problem: given a set of genomes, find a phylogeny relating them (with internal nodes labeled by other genomes) to minimize the total number of breakpoints across the edges of the tree. They also developed BPAnalysis [6], which provided heuristics for the breakpoint phylogeny problem. BPAnalysis needs to compute all possible binary trees and picks the one that produces the minimum score as the phylogeny of the input genomes. Blanchette and Sankoff showed that the breakpoint median could be reduced to a traveling salesperson problem and uses an iterative approach to search for an optimal solution to the breakpoint phylogeny problem: for each internal node v, with neighbors A, B, and C, solve the median problem defined by these three genomes to yield m; if using m improves the tree score, then update v with the new genome m. This procedure will repeat until no change occurs.

However, since ancestral genomes on the internal nodes are unknown, we must initialize them first. The simplest approach is to generate random gene orders on each of the internal nodes. The more complex approach is to initialize through solving median problems: for each internal node, we can initialize it with the solution of the median problem defined on its three closest leaves. For example, in Fig. 9.2 (Right), the internal node M can be initialized with the solution from G_1, G_2, and G_3 and N can be initialized by G_3, G_4, and G_2 respectively. Although this approach is more complex and more expensive, it is more efficient since the number of iterations can be greatly reduced.

Dr. Moret and colleagues developed GRAPPA as an improvement and new implementation of BPAnalysis [34]. GRAPPA applied algorithmic engineering techniques pioneered by Dr. Moret to enhance its efficiency. GRAPPA is more cache-sensitive and has a smaller footprint and is easily extended with new events and median solvers. Another one major improvement in GRAPPA is to test the circular-ordering lower bound of a tree to determine whether it is worthy of scoring: if a tree has a lower bound exceeding the current best tree score, GRAPPA will discard it to avoid the expensive computation of tree scoring. As a result, GRAPPA has a speed up of two to three orders of magnitude over BPAnalysis and can handle a much larger dataset that could not be solved by BPAnalysis [34]. Over the years, GRAPPA has been extended to cope with more complex events by using various median solvers.

Nevertheless, since the number of trees grows exponentially with the increase of number of genomes, it will take years for GRAPPA to finish datasets with even just a dozen genomes. When the genomes are distant from each other, finding a good median is very difficult to do, which makes obtaining tree scores also very challenging. Dr. Moret and colleagues developed several other methods to overcome these obstacles, including a method to combine GRAPPA with a DCM2 [20], a disk-covering method (see chapter by Warnow, this book, for more about disk-covering methods), to scale it to handle hundreds of genomes [43], distance-based methods for rearrangements and indels [24, 42], as well as a series of probabilistic methods based on gene adjacencies [18, 23, 43]. These methods have been integrated into a web service (http://www.geneorder.org) called MLGO [17], which handles the big phylogeny problems and reconstructs ancestral genomes even though the phylogeny is unknown.

9.5 Conclusions

With the continuous efforts from Dr. Moret and other researchers, great progress has been made in computing phylogenies and ancestral genomes at whole genome level. We are now able to analyze within a few hours of computation dozens of nuclear genomes [12, 23], compared to just a few small chloroplast or mitochondrial genomes in early 2000. The new methods, especially those based on probabilistic models have achieved scale and accuracy that is comparable to those developed for sequence analysis [17]. In recent years, more and more new types of data have emerged in areas such as cancer evolution, population genetics, and epigenomics, where genome rearrangements play important roles. Integrating genome rearrangements with such data is very challenging and requires new models, theories, and algorithms.

Acknowledgements JT was supported by the National Science Foundation of US (grant number IIS 1161586).

References

1. Avdeyev, P., Jiang, S., Aganezov, S., Hu, F., Alekseyev, M.A.: Reconstruction of ancestral genomes in presence of gene gain and loss. J. Comput. Biol. **23**(3), 150–164 (2016)
2. Bader, D., Moret, B., Yan, M.: A fast linear-time algorithm for inversion distance with an experimental comparison. J. Comput. Biol. **8**(5), 483–491 (2001)
3. Bader, M.: Genome rearrangements with duplications. BMC Bioinform. **11**(Supple 1), S27 (2010)
4. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proceedings of the 6th Workshop on Algorithms in Bioinformatics (WABI'06). Lecture Notes in Computer Science, vol. 4175, pp. 163–173. Springer Verlag, Berlin (2006)
5. Biller, P., Feijao, P., Meidanis, J.: Rearrangement-based phylogeny using the single-cut-or-join operation. IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB) **10**(1), 122–134 (2013)
6. Blanchette, M., Bourque, G., Sankoff, D.: Breakpoint phylogenies. Genome Inform. **8**, 25–34 (1997)
7. Braga, M., Stoye, J.: Counting all DCJ sorting scenarios. In: Proceedings of RECOMB International Workshop on Comparative Genomics (RECOMB-CG'09), pp. 36–47. Springer Verlag, Berlin (2009)
8. Caprara, A.: On the practical solution of the reversal median problem. In: Proceedings of the 1st International Workshop Algorithms in Bioinformatics (WABI'01). Lecture Notes in Computer Science, vol. 2149, pp. 238–251 (2001)
9. Compeau, P.: A simplified view of DCJ-Indel distance. In: Proceedings of the 12th Workshop Algorithms in Bioinformatics (WABI'12). Lecture Notes in Computer Science, vol. 7534, pp. 365–377. Springer Verlag, Berlin (2012)
10. Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L.S., Warnow, T., Wyman, S.: A new fast heuristic for computing the breakpoint phylogeny and a phylogenetic analysis of a group of highly rearranged chloroplast genomes. In: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00), pp. 104–115 (2000)
11. Feijão, P., Meidanis, J.: SCJ: a variant of breakpoint distance for which sorting, genome median and genome halving problems are easy. In: International Workshop on Algorithms in Bioinformatics, pp. 85–96. Springer (2009)
12. Feng, B., Lin, Y., Zhou, L., Guo, Y., Friedman, R., Xia, R., Hu, F., Liu, C., Tang, J.: Reconstructing yeasts phylogenies and ancestors from whole genome data. Sci. Rep. **7**(1), 15,209 (2017)
13. Fertin, G., Labarre, A., Rusu, I., Vialette, S., Tannier, E.: Combinatorics of Genome Rearrangements. MIT press (2009)
14. Fitch, W.M.: Toward defining the course of evolution: minimum change for a specific tree topology. Syst. Biol. **20**(4), 406–416 (1971)
15. Gagnon, Y., Blanchette, M., El-Mabrouk, N.: A flexible ancestral genome reconstruction method based on gapped adjacencies. BMC Bioinform. **13**(Suppl 19), S4 (2012)
16. Hannenhalli, S., Pevzner, P.: Transforming mice into men (polynomial algorithm for genomic distance problems). In: Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS'95), pp. 581–592 (1995)
17. Hu, F., Lin, Y., Tang, J.: MLGO: phylogeny reconstruction and ancestral inference from gene-order data. BMC Bioinform. **15**(1), 1 (2014)
18. Hu, F., Zhou, J., Zhou, L., Tang, J.: Probabilistic reconstruction of ancestral gene orders with insertions and deletions. IEEE/ACM Trans. Comput. Biol. Bioinform. **11**(4), 667–672 (2014)
19. Hu, F., Zhou, L., Tang, J.: Reconstructing ancestral genomic orders using binary encoding and probabilistic models. In: Bioinformatics Research and Applications, pp. 17–27. Springer (2013)
20. Huson, D., Vawter, L., Warnow, T.: Solving large scale phylogenetic problems using DCM-2. In: Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB'99) (1999)

21. Jones, B.R., Rajaraman, A., Tannier, E., Chauve, C.: ANGES: reconstructing ANcestral GEnomeS maps. Bioinformatics **28**(18), 2388–2390 (2012)
22. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
23. Lin, Y., Hu, F., Tang, J., Moret, B.: Maximum likelihood phylogenetic reconstruction from high-resolution whole-genome data and a tree of 68 eukaryotes. In: Pacific Symposium on Biocomputing, pp. 357–366. World Scientific (2013)
24. Lin, Y., Moret, B.: Estimating true evolutionary distances under the DCJ model. In: Proceedings of the 16th International Conference on Intelligent Systems for Molecular Biology (ISMB'08) (2008)
25. Lin, Y., Rajan, V., Swenson, K., Moret, B.: Estimating true evolutionary distances under rearrangements, duplications, and losses. In: Proceedings of the 8th Asia Pacific Bioinformatics Conference (APBC'10). BMC Bioinform. **11**(Suppl 1), S54 (2010)
26. Ma, J.: A probabilistic framework for inferring ancestral genomic orders. In: 2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 179–184. IEEE (2010)
27. Ma, J., Zhang, L., Suh, B.B., Raney, B.J., Burhans, R.C., Kent, W.J., Blanchette, M., Haussler, D., Miller, W.: Reconstructing contiguous regions of an ancestral genome. Genome Res. **16**(12), 1557–1565 (2006)
28. Marron, M., Swenson, K., Moret, B.: Genomic distances under deletions and insertions. In: Proceedings of the 9th International Conference Computing and Combinatorics (COCOON'03). Lecture Notes in Computer Science, vol. 2697, pp. 537–547. Springer Verlag, Berlin (2003)
29. Metropolis, N., Rosenbluth, A., Rosenbluth, M.: Equation of state calculations by fast computing machines. J. Chem. Phys. **21**(6), 1087–1092 (1953)
30. Moret, B., Lin, Y., Tang, J.: Rearrangements in phylogenetic inference: compare, model, or encode? In: Models and Algorithms for Genome Evolution, pp. 147–171. Springer (2013)
31. Moret, B., Siepel, A., Tang, J., Liu, T.: Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In: Proceedings of the 2nd Workshop on Algorithms in Bioinformatics (WABI'02). Lecture Notes in Computer Science, vol. 2452, pp. 521–536. Springer Verlag, Berlin (2002)
32. Moret, B., Tang, J., Wang, L.S., Warnow, T.: Steps toward accurate reconstructions of phylogenies from gene-order data. J. Comput. Syst. Sci. **65**(3), 508–525 (2002)
33. Moret, B., Warnow, T.: Reconstructing optimal phylogenetic trees: a challenge in experimental algorithmics. In: Fleischer, R., Moret, B., Schmidt, E. (eds.) Experimental Algorithmics. Lecture Notes in Computer Science, vol. 2547, pp. 163–180. Springer Verlag, Berlin (2002)
34. Moret, B., Wyman, S., Bader, D., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Proceedings of the 6th Pacific Symposium on Biocomputing (PSB'01), pp. 583–594. World Scientific Pub. (2001)
35. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Elec. Colloq. Comput. Complex. **71** (1998)
36. Perrin, A., Varré, J.S., Blanquart, S., Ouangraoua, A.: Procars: progressive reconstruction of ancestral gene orders. BMC Genomics **16**(Suppl 5), S6 (2015)
37. Rajan, V., Xu, W., Lin, Y., Swenson, K., Moret, B.: Heuristics for the inversion median problem. In: Proceedings of the 8th Asia Pacific Bioinformatics Conference (APBC'10). BMC Bioinform. **11**(Suppl 1), S30 (2010)
38. Shao, M., Lin, Y.: Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. BMC Bioinform. **13**(Suppl 19), S13 (2012)
39. Shao, M., Lin, Y., Moret, B.: An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. J. Comput. Biol. **22**(5), 425–435 (2015)
40. Siepel, A., Moret, B.: Finding an optimal inversion median: experimental results. In: Proceedings of the 1st International Workshop Algorithms Bioinformatics (WABI'01). Lecture Notes in Computer Science, vol. 2149, pp. 189–203. Springer Verlag, Berlin (2001)
41. Silva, P., Braga, M., Machado, R., Dantas, S.: DCJ-indel distance with distinct operation costs. In: Proceedings of the 12th Workshop on Algorithms in Bioinformatics (WABI'12). Lecture Notes in Computer Science, vol. 7534, pp. 378–390. Springer Verlag, Berlin (2012)

42. Swenson, K., Arndt, W., Tang, J., Moret, B.: Phylogenetic reconstruction from complete gene orders of whole genomes. In: Proceedings of the 6th Asia Pacific Bioinformatics Conference (APBC'08), pp. 241–250 (2008)
43. Tang, J., Moret, B.: Scaling up accurate phylogenetic reconstruction from gene-order data. In: Proceedings of the 11th International Conference on Intelligent Systems for Molecular Biology (ISMB'03). Bioinformatics, vol. 19, pp. i305–i312. Oxford U. Press (2003)
44. Tang, J., Moret, B., Cui, L., dePamphilis, C.: Phylogenetic reconstruction from arbitrary gene-order data. In: Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering BIBE'04, pp. 592–599. IEEE Press, Piscataway, NJ (2004)
45. Xia, R., Lin, Y., Zhou, J., Feng, B., Tang, J.: A median solver and phylogenetic inference based on double-cut-and-join sorting. J. Comput. Biol. **25**(3), 302–312 (2018)
46. Xu, A., Moret, B.: GASTS: parsimony scoring under rearrangements. In: Proceedings of the 11th Workshop on Algorithms in Bioinformatics (WABI'11). Lecture Notes in Computer Science, vol. 6833, pp. 351–363. Springer Verlag, Berlin (2011)
47. Xu, W.: DCJ median problems on linear multichromosomal genomes: graph representation and fast exact solutions. In: RECOMB International Workshop on Comparative Genomics, pp. 70–83. Springer (2009)
48. Xu, W., Sankoff, D.: Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Proceedings of the 8th International Workshop Algorithms in Bioinformatics (WABI'08). Lecture Notes in Computer Science, vol. 5251, pp. 25–37 (2008)
49. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)

Chapter 10
Genome Rearrangement Problems with Single and Multiple Gene Copies: A Review

Ron Zeira and Ron Shamir

Abstract Genome rearrangement problems arise in both species evolution and cancer research. Basic genome rearrangement models assume that the genome contains a single copy of each gene and the only changes in the genome are structural, i.e., reordering of segments. In contrast, numerical changes such as deletions and duplications, which change the number of copies of genes, have been observed in species evolution and prominently in tumorigenesis. Here, we review various computational models of evolution by rearrangements designed for the analysis of species or cancer genomes, focusing mainly on genomes with multiple gene copies. Models differ in the assumptions taken on the genome structure and in the type of rearrangements allowed during their evolution. Most problems regarding genomes with multiple gene copies are computationally hard, and practical methods for their analysis are reviewed. As more high-resolution genomes become available, especially in cancer, better models and efficient algorithms will be needed.

Keywords Genome rearrangements · Cancer · Tumor · Structural aberrations · Numerical aberrations · Genome sorting problem · Complexity · Algorithms · Evolution

Prologue

The computational study of genome rearrangements is a subarea of computational biology born about 25 years ago [85, 87]. Over that period, it has flourished and developed into a fascinating research area, combining beautiful combinatorial models, elegant theory, and applications. Models of the first generation, motivated by species evolution, were simple (though their analysis was sometimes quite sophisticated) and assumed that genomes contain only one copy of each gene. With the

R. Zeira · R. Shamir (✉)
Tel Aviv Universirty, Tel Aviv, Israel
e-mail: rshamir@tau.ac.il

R. Zeira
e-mail: ronzeira@post.tau.ac.il

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_10

explosion of biological data, new analysis opportunities arose, necessitating more complex models and theory.

This manuscript describes some of the problems and results related to rearrangement models allowing multiple gene copies. This research area is motivated by evolution of species and of cancer genomes. Studies of cancer genome evolution are stimulated by the recent large-scale deep sequencing of thousands of tumor genomes, which has brought about a plethora of novel challenges. Our main focus is on multi-copy models, but key single-copy models are also reviewed briefly for context.

This review is by no means exhaustive. The field of modeling genome rearrangements is vast and cannot be covered in one paper. The selection of topics reflects our knowledge (or lack thereof) and taste, and we apologize to the many researchers whose work is not mentioned. For further reading see, e.g., [37, 40, 126].

This review is intended for researchers and graduate students in Computer Science and Bioinformatics. Experts in computational genome rearrangements can use it as a reference source. For newcomers to the field, it can be a roadmap of key models and problems in the rich literature on genome rearrangements. As the motivation to the field comes from biology and medicine, we describe very briefly the biological context. However, by and large, the review can be read and understood without that context.

10.1 Introduction

In this section, we give biological introduction and motivation to genome rearrangements (*GR*) in both species evolution and cancer.[1] Sect. 10.2 gives computational background and some fundamental results in the analysis of single-copy genomes. In Sects. 10.3 and 10.4, we review GR models that handle genomes with multiple gene copies in the context of species and cancer evolution.

10.1.1 Genomes and Rearrangements

The *genome*[2] encodes instructions used in the development and functioning of all living organisms (bacteria, plants, animals, etc.). Genomes are built of *DNA*, a double-stranded molecule in which each *strand* is a long sequence of *nucleotides* (or *bases*). Each base can be of four types *A*, *C*, *G*, and *T*. The two strands are *complementary* such that an *A* on one strand is coupled with a *T* on the other strand, and similarly *C* is

[1] See the Box 3 for a list of abbreviations.

[2] Since this review concentrates mainly on the computational aspects of GR, we only give a brief biological introduction. We italicize terms that actually require definitions. For concise biological definitions see, e.g., [63]. Box 1 defines some biological terms that are mentioned in the text.

coupled with G. Because of this complementarity, one strand completely determines the other, and DNA molecules are usually represented by the sequence of one strand.

The *genome* is the total DNA material in the cell. It is partitioned into physically disjoint subsequences called *chromosomes*. Chromosomes can be either *linear* and contain two ends called *telomeres* and *circular*. A *gene* is a segment along the chromosome containing information for the construction of a *protein*. Proteins are molecules that form the "machines" and building blocks of most cellular functions. The direction in which a gene is transcribed into a protein on a given strand determines its *orientation*. Genes are a basic unit of heredity passed from one generation to the other.

The causes of diversity of organisms are changes in the DNA between generations. Such changes, which arise due to inaccurate replication and also due to environmental effects on the DNA, open the possibilities for modified genes, new genes, and eventually new species.

Genomes can evolve in a local and global manner. *Local* alterations refer to *point mutations* in the DNA sequence that can either *substitute* a single base (or a very short subsequence) with a different one, *insert* a single base into the sequence, or *delete* a base from the sequence. Such local alterations can also involve very short sequence segments. On the other hand, a sequence can also evolve by modifying its organization on a large scale. These *global* mutations, called *genome evolution* or *structural variations*, relocate, duplicate, or delete large fragments of the DNA. The main rearrangement types include the following (compare Fig. 10.1):

- *Deletion*. A segment of DNA is lost. A *chromosome deletion* is a deletion of an entire chromosome.
- *Inversion* or *reversal*. A segment is cut and reinserted in the opposite orientation. Since the insertion reverses the two strands, the result is an inverted and reverse complemented DNA sequence.
- *Transposition*. A DNA segment is moved to a different location.
- *Duplication*. A genomic segment is copied and reinserted into the genome. In a *tandem duplication*, the copy is inserted right after the original one. An arbitrary (non-tandem) duplication inserts the new copy at an arbitrary position (one particular type of such is *retrotransposition*). A *whole chromosome duplication* makes another copy of an entire chromosome. A *whole genome duplication* duplicates all the genome's chromosomes.
- *Translocation*. Two linear chromosomes exchange their end segments.
- *Fusion*. Two chromosomes are joined into one.
- *Fission*. A chromosome splits into two chromosomes.

The above rearrangement operations affect DNA segments rather than nucleotides and thus genomes are often represented by sequences of segments in this context. Two segments are called *homologous* if they derive from a common ancestor either by speciation (in that case the segments appear in the genomes of different species) or by duplication (where they occur on the same genome).

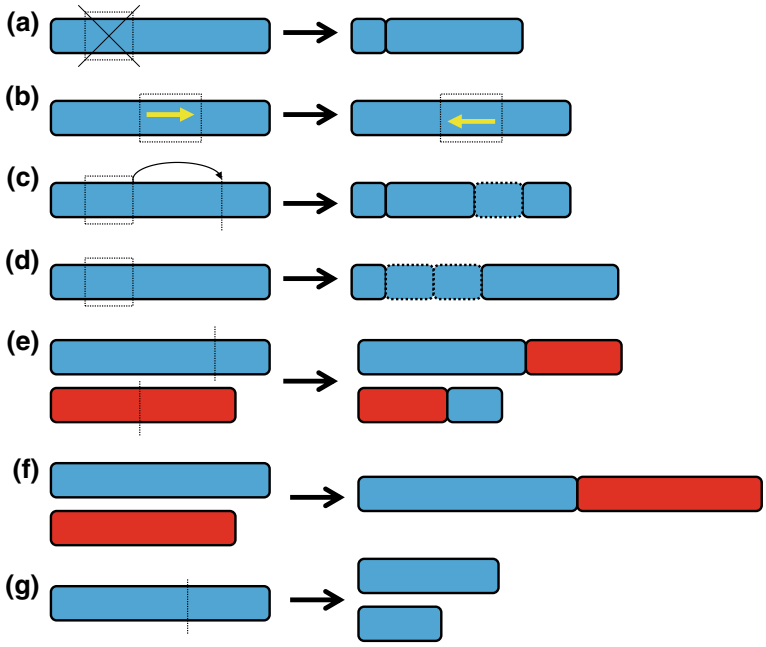


Fig. 10.1 *Genome evolution*. **a** Deletion. **b** Reversal. **c** Transposition. **d** Tandem duplication. **e** Translocation. **f** Fusion. **g** Fission

10.1.2 Genome Rearrangements in Species Evolution

The genomes of related species are very similar. For instance, most of the mouse and human genomes can be divided into segments in which gene content is conserved [27]. However, the order of these segments along the human and mouse genomes is different. This difference is attributed to rearrangement events occurring after the divergence of the two lineages.

The phenomenon of GR in evolution was discovered by Sturtevant and Dobzhansky who demonstrated inversions between genomes of drosophila species [100]. Palmer and colleagues observed that mitochondrial DNA of related plant species has similar gene content but different segment orderings (Fig. 10.2) [75, 99]. This immediately raises the question of how this change came about, the fundamental problem that underlies the GR field.

The detection of GRs in the studies mentioned was largely based on molecular cytogenetics techniques such as *chromosome banding* and *in-situ hybridization* [80]. These studies mostly focused on relatively close species and a small number of rearrangements between them [88]. With the advent of sequencing technologies, bioinformatic methods enabled locating homologous segments in different genome sequences, thus creating finer comparative maps based on genome sequences [77]. See Box 2 for details on the technologies for rearrangement detection.

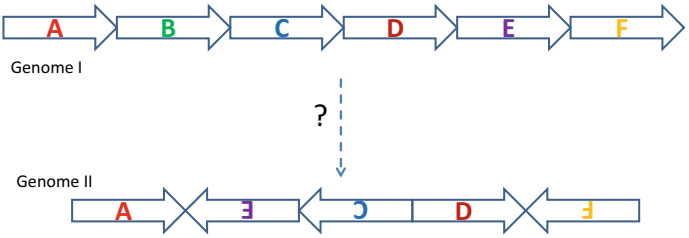


Fig. 10.2 The basic sorting problem. Given Genome I and Genome II, and a set of allowed operations, we wish to find a shortest sequence of operations transforming Genome I into Genome II. The sequence is called a sorting scenario and the number of operations in it is called the sorting distance. See Fig. 10.3 for a sorting scenario

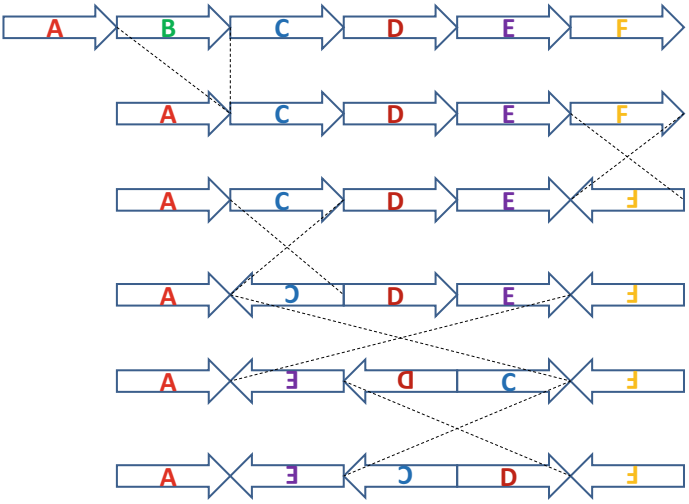


Fig. 10.3 A sorting scenario for the chloroplast genome evolution between two conifers. The genome at the top is transformed to the one at the bottom in five steps. The first is a deletion and the next four are inversions of genomic segments. The ends of the involved segments are indicated by the broken lines. Adapted and simplified from Strauss et al. [99]

David Sankoff pioneered the computational study of GR in species evolution [85, 87]. The basic assumption of most mathematical models is that evolution is parsimonious and prefers a shortest or most likely sequence of events. In their seminal works, Hannenhalli and Pevzner gave the first polynomial algorithm for the problem of transforming one genome into the other by the minimum number of reversals and of reversals and translocations, respectively [47, 48]. They used their algorithm to give a shortest event sequence between men and mice, and between cabbage and turnip.

Classical computational rearrangement models assume that each gene in the two genomes under study appears only once and that 1–1 homology between the genes

of the genomes has been established. While this assumption may hold for closely related genomes, it is unwarranted for divergent species with several copies of the same genes or highly similar genes. Duplications are an important source of new gene functions since new gene copies tend to diverge through mutations and develop new functions. For instance, evidence of whole genome duplication events has been observed in most angiosperm genomes [20].

Box 1 Some biological jargon

Angiosperms—the flowering plants
Chloroplasts—specialized compartments in plant cells responsible for photosynthesis
Conifers—cone-bearing seed plants
Drosophila—fruit fly
Metaphase—a stage in cell division. During metaphase chromosomes can be distinguished under the microscope after appropriate painting
Orthologs—descendant copies of the same gene sequence in different species. Orthologs can usually be identified by their sequence similarity
Somatic cell—any cell forming the organism body other than the reproductive cells. The genome in sperm and egg cells is inherited in sexual reproduction, along with any mutations in it. In contrast, the genome of somatic cells is not inherited, but mutations in cancer genomes are inherited in cell division.
Somatic mutation—a mutation occurring in somatic cells.

10.1.3 Genome Rearrangements in Cancer

Cancer is a complex disease driven by the accumulation of somatic DNA mutations over generations of cell divisions. Such mutations affect tumor growth, clinical progression, immune escape, and drug resistance [30].

Mutations in cancer cells can be local, affecting single-DNA base pairs. These mutations, called *single-nucleotide variants* (*SNV*), can number in the thousands per cancer cell. On the other hand, large-scale mutations, i.e., GRs, can relocate fragments of the DNA. Aberrations that change the amount of genomic content, called *copy number alterations* (*CNAs*), include duplications and deletions of genomic regions. The *karyotype* of a cell is its complete set of chromosomes, consisting of the number and structure of the chromosomes in it. Large-scale aberrations can have a dramatic effect on the cancer karyotype (see Fig. 10.4).

Somatic mutations may amplify genes that promote cancer (*oncogenes*) or harm genes that inhibit cancer development (*tumor suppressor* genes). In addition, rearrangements such as translocations and inversions may change gene structure and regulation and create novel fusion genes, with or without additional changes in copy number (CN).

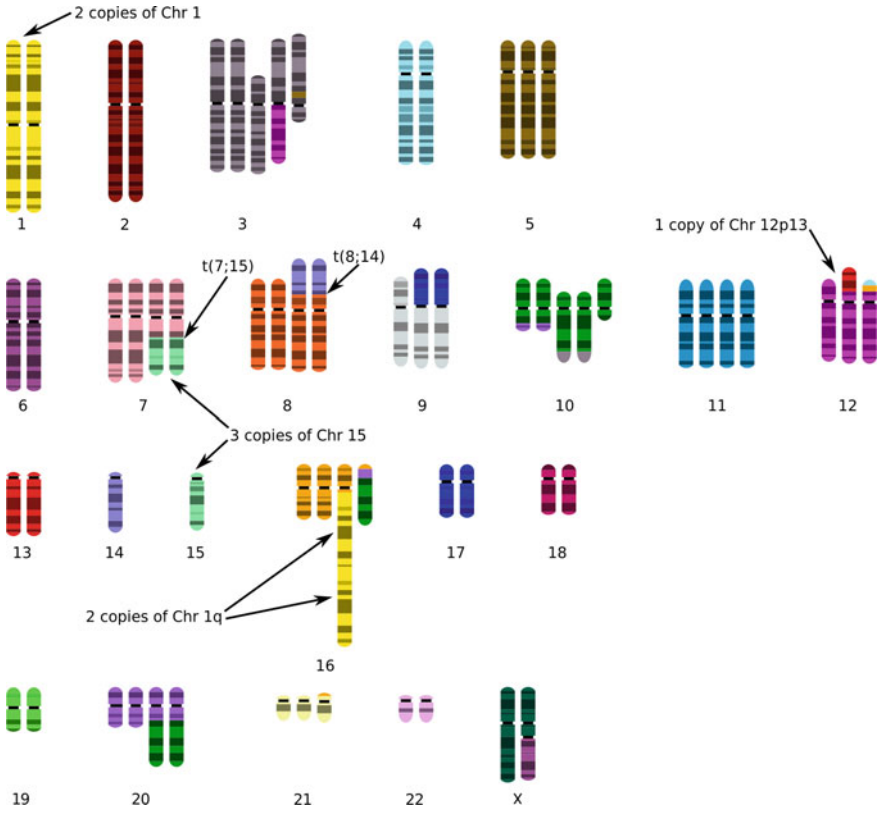


Fig. 10.4 A schematic of the karyotype of the T47D breast cancer cell line. The chromosome numbers in the normal diploid are indicated below each subfigure. In a normal karyotype, each chromosome has two copies, as for Chr. 4, 13, 17, and 18. Among the GRs in this cancer genome, we see chromosomal duplications (e.g., four copies of Chr. 11), translocations (between Chr. 8 and Chr. 14), and more complex events (e.g., tandem duplication of one arm of Chr. 1 and fusion with an extra arm of Chr. 16). Image source: [10] and Wikimedia Commons [50]. This image is used under license CC BY-SA 3.0

Cancer is an evolutionary process in which a normal genome accumulates mutations that eventually transform it into a cancerous one [11]. The gain of advantageous mutations leads to a *clonal expansion*, forming a larger population of the mutated cells. Subsequent clonal expansions occur as additional advantageous mutations accumulate in descendant cells. A single tumor biopsy will often contain a mixture of several competing tumor clones. These tumor clones frequently differ in their genomic content and structure. When sequencing the tumor, one actually obtains a mixture of several tumor clones and normal cells. Recent research suggests that this heterogeneity has profound clinical implications [30].

Box 2 Detection of genome evolution

The classical ways to detect chromosomal abnormalities in cytogenetics are *G-banding* and *fluorescence in situ hybridization* (*FISH*), which allow viewing the chromosome in metaphase at low resolution [79]. FISH measures the CN of tens to hundreds of targeted genes [29]. *Array comparative genomic hybridization* (*array CGH*) gives a higher resolution of CN estimation for a cell population [107].
Today, next generation sequencing technologies are the main data source for cancer mutation analyses [31]. Whole genome sequencing provides tens to hundreds of millions of DNA reads that enable the detection of variants. These short reads are assembled into longer DNA sequences and alignment to a reference genome can determine sequence similarity and structural changes. This reference genome can be of a related species for evolutionary studies or of a normal tissue in the case of cancer.
Paired-end read technologies generate pairs of short reads such that the approximate distance between them and their relative orientations in the target genome are known. Read pairs in which the location or orientation in the reference genome is not as expected are called *discordant*. These reads give evidence of structural rearrangement operations [71]. The *read depth* data, i.e. the number of concordant reads mapped to each region in the reference genome, can also be used to assess CN and CNAs [71].

10.2 Single-Gene Models, Operation Types, and Distance Measures

In this section, we give a brief introduction to GR models. We start by giving the definitions and terminology used in computational GR analysis. We then review several classical single-gene models.

10.2.1 Genome Representation

Here, we describe simple mathematical representations of genomes for GR analysis. A genome representation should preserve the information about the order, orientation, and homology between segments (see Fig. 10.3). In some representations, different copies of similar segments can be distinguished while in other representations they cannot. For instance, the two copies of chromosome 1 in Fig. 10.4 are indistinguishable. On the other hand, in some cases gene copies can be distinguished from one another, for example, due to gene sequence changes since its speciation. Different GR models may use different representations depending on the model assumptions or data used.

Consider a set $\mathscr{G}$ of n segments in the genome. For convenience, we call the segments in $\mathscr{G}$ *genes*, though they do not necessarily represent biological gene entities.

A gene g is an oriented sequence of DNA that starts with a *tail* and ends with a *head*, denoted as g_t and g_h, respectively. The default orientation of a gene, and thus its head and tail, can be determined arbitrarily or according to some reference genome. The set of *extremities* of the genes is $\mathscr{E} = \{g_t | g \in \mathscr{G}\} \cup \{g_h | g \in \mathscr{G}\}$.

An *adjacency* between two consecutive genes in a genome is an unordered pair of extremities. Thus, an adjacency between two genes $a, b \in \mathscr{G}$ can take one of four forms, depending on their orientation: $\{a_h, b_t\}$, $\{a_t, b_h\}$, $\{a_t, b_t\}$, $\{a_h, b_h\}$. An extremity that is not adjacent to another extremity is called a *telomere* and is represented by a singleton set, e.g., $\{a_h\}$.

In some formulations, a gene may have multiple copies corresponding, for example, to homologous yet distinguishable genes. The copies of such a gene $g \in \mathscr{G}$ are identified by a superscript. For example, g^1, g^2, g^3 are three distinct copies of gene g. Such a gene with multiple distinguishable copies is called a *labeled* gene. A gene that has a single copy or has multiple indistinguishable copies is called *unlabeled*. For a gene g, we call the number of copies it has its *copy number* and denotes it by $cn(g)$. A gene set $\mathscr{G}$ with one copy for each gene is called an *ordinary gene set*. A *labeled gene set* is a set $\mathscr{G}^L = \{g^i | g \in \mathscr{G}, 1 \leq i \leq cn(g)\}$ and an *unlabeled gene set* $\mathscr{G}^U$ is a multiset $\mathscr{G}^U = \cup_{g \in \mathscr{G}} \cup_{1 \leq i \leq cn(g)} \{g\}$. For instance, $\mathscr{G}^L = \{a^1, a^2, b^1, c^1, c^2, c^3\}$ and $\mathscr{G}^U = \{a, a, b, c, c, c\}$ are labeled and unlabeled gene sets, respectively, that have two copies of gene a, one of b and three of c. Similar to genes, extremities belonging to labeled genes are distinguishable (e.g., $a_h^1 \neq a_h^2$), while extremities of unlabeled gene are indistinguishable. Furthermore, unlabeled heads and tails of the same unlabeled gene cannot be matched, i.e., we do not know which tail and head come from the same gene copy.

A *labeled genome* Π over a labeled gene set $\mathscr{G}^L$ is a set of adjacencies and telomeres such that every labeled extremity $e^i \in \mathscr{E}^L$ appears exactly once in an adjacency or telomere of Π. Similarly, an *unlabeled genome* Π over an unlabeled gene set $\mathscr{G}^U$ is a multiset of adjacencies and telomeres such that every unlabeled extremity $e \in \mathscr{E}^U$ of gene g appears exactly $cn(g)$ times in adjacencies or telomeres of Π. $\Pi = \{\{a_t^1\}, \{a_h^1, b_h^1\}, \{b_t^1, c_t^1\}, \{c_h^1\}, \{a_t^2\}, \{a_h^2, b_t^2\}, \{b_h^2\}, \{b_t^3\}, \{b_h^3, c_h^2\}, \{c_t^2\}\}$ and $\Gamma = \{\{a_t\}, \{a_h, b_h\}, \{b_t, c_t\}, \{c_h\}, \{a_t\}, \{a_h, b_t\}, \{b_h\}, \{b_t\}, \{b_h, c_h\}, \{c_t\}\}$ are examples of labeled and unlabeled genomes. If the gene set of a genome is ordinary, we call it an *ordinary genome* (or a *single-copy genome*).

The *graph representation* of a genome Π is an undirected graph $G_\Pi = (\mathscr{E}, E)$. Its nodes are the extremities of Π (either labeled or unlabeled) and E consists of interval edges and adjacency edges. An *interval edge* connects the head and tail of a gene. For unlabeled genomes, there are $cn(g)$ parallel interval edges of the edge (g_h, g_t) for every gene g. For labeled genomes, each labeled gene copy g^i has a single interval edge (g_h^i, g_t^i). *Adjacency edges* connect the extremities x and y where $\{x, y\}$ is an adjacency of Π. We call G_Π the *genome graph* of Π. The representations Π and G_Π are equivalent and thus we use them interchangeably. Notice that for each node (extremity), its number of interval edges (*interval degree*) equals its number of adjacency edges (*adjacency degree*) plus the number of telomeres it belongs to. Figures 10.5, 10.6, and 10.7 show genome graphs for ordinary, labeled and unlabeled genomes, respectively.

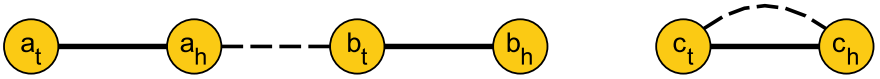


Fig. 10.5 A genome graph G_Π of an ordinary genome $\Pi = \{\{a_t\}, \{a_h, b_t\}, \{b_h\}, \{c_t, c_h\}\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. Since Π is an ordinary genome, it has a unique decomposition D_Π whose string representation is $\{(a\ b), < c > \}$

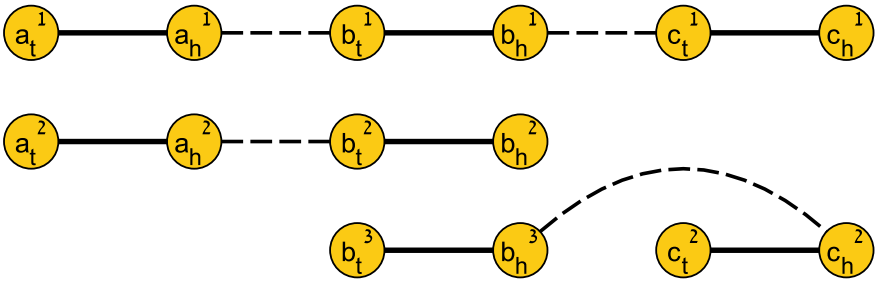


Fig. 10.6 A genome graph G_Δ of a labeled genome $\Delta = \{\{a_t^1\}, \{a_h^1, b_t^1\}, \{b_h^1, c_t^1\}, \{c_h^1\}, \{a_t^2\}, \{a_h^2, b_t^2\}, \{b_h^2\}, \{b_t^3\}, \{b_h^3, c_h^2\}, \{c_t^2\}\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. Since Δ is an ordinary genome, it has a unique decomposition D_Δ whose string representation is $\{(a^1\ b^1\ c^1), (a^2\ b^2), (b^3\ -c^2)\}$

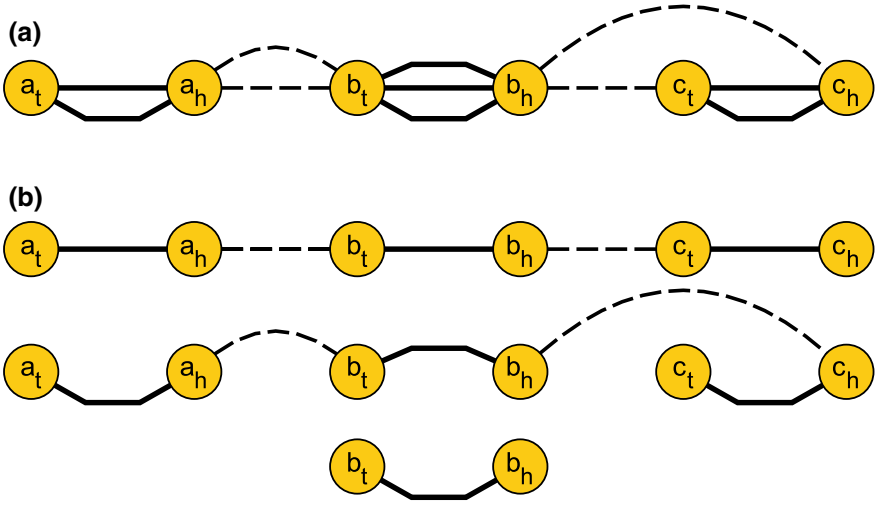


Fig. 10.7 **a** A genome graph G_Γ of an unlabeled genome $\Gamma = \{\{a_t\}, \{a_h, b_t\}, \{b_h, c_t\}, \{c_h\}, \{a_t\}, \{a_h, b_t\}, \{b_h\}, \{b_t\}, \{b_h, c_h\}, \{c_t\}\}$. Bold edges correspond to interval edges; dashed edges correspond to adjacencies. **b** One possible decomposition D_Γ^1 of Γ, whose string representation is $\{(a\ b\ c), (a\ b\ -c), (b)\}$. A different decomposition D_Γ^2 corresponding to $\{(a\ b\ c), (a\ b), (b\ -c)\}$ can be seen in Fig. 10.6 by suppressing the superscripts

An *alternating route* in G_Π is either a path or a cycle in which no two consecutive edges are of the same type (interval/adjacency). A *chromosome decomposition* D_Π of the genome Π is a decomposition of G_Π into a set of edge-disjoint maximal alternating cycles and alternating paths that cover all edges. Note that a chromosome decomposition is always possible since the interval degree is equal to the adjacency degree for every node that is not in a telomere, and that maximal paths must start and end with telomeres. Labeled and ordinary genomes have a unique chromosome decomposition by simply taking the set of connected components, since the interval degree and the adjacency degree of every non-telomere node is 1 (see Figs. 10.5 and 10.6). There may be several decompositions for a multi-copy unlabeled genome (see Fig. 10.7). Each alternating route in a decomposition is called a *chromosome*. A chromosome is called *circular* if the corresponding route is a cycle, and *linear* otherwise. A decomposition is called *linear* if all its chromosomes are linear, *circular* if all its chromosomes are circular, and otherwise *mixed*. Figure 10.5 shows an ordinary genome with one linear and one circular chromosome. An ordinary genome composed of a single linear chromosome is called a *signed permutation*.

A *signed genomic string* is a sequence of oriented genes, e.g., $1\ {-2}\ 3$. For a chromosome $C \in D_\Pi$, we define the *chromosome string* of C as follows. Start at one of the ends of a linear chromosome with the string "(". Traverse the route until all edges along the route are covered. For each traversal of an interval edge from a tail g_t to a head g_h append g to the string. For traversal from g_h to g_t append $-g$ to the string. After finishing the traversal, append the string with ")". For a chromosome string C, let $-C$ be the chromosome string in which the order and orientation of all gene are in inverted, e.g., if $C = (1\ 2\ 3)$ then $-C = (-3\ -2\ -1)$. C and $-C$ are *equivalent* as they correspond to the same set of adjacencies. For a circular chromosome, do the same starting from an arbitrary extremity interval edge without appending brackets. The resulting sequence is cyclic, and all shifts and inversions of it are equivalent. We use $<>$ to denote circular genomes (Figs. 10.5, 10.6 and 10.7).

A *string representation of a genome decomposition* D_Π is the multiset of chromosome strings for each chromosome in the decomposition (Figs. 10.5, 10.6, and 10.7). Two string representations are equivalent if there is a bijective mapping between equivalent chromosome strings in them. For labeled and ordinary genomes, the string representation is unique (up to equivalence) and therefore we sometime use this representation.

Given an unlabeled genome Π over the gene set $\mathcal{G}^U$, a *labeling* of Π produces a labeled genome Γ over the gene set $\mathcal{G}^L$ such that distinct gene copies of a gene g are mapped to distinct labeled genes $g^1, \ldots, g^{cn(g)}$ in $\mathcal{G}^L$. For example, the labeled genomes in Figs. 10.6 and 10.7b are two possible labelings of the unlabeled genome in Fig. 10.7a. We denote $L(\Pi)$ to be the set of all possible labelings of Π.

Given a genome Π_1 over the gene set $\mathcal{G}_1$, an *operation* creates a new genome $\Pi_2 \neq \Pi_1$ over a new gene set $\mathcal{G}_2$. An operation is said to be *structural* if $\mathcal{G}_1 = \mathcal{G}_2$. An operation is said to be *numerical* if the CN of some gene is different under $\mathcal{G}_1$ and $\mathcal{G}_2$. Notice that a structural operation only changes the structure, i.e., $\Pi_2 \neq \Pi_1$, whereas a numerical operation also changes the gene set, i.e., $\mathcal{G}_1 \neq \mathcal{G}_2$.

A *genome rearrangement model* is composed of a set of allowed operations $\mathcal{O}$ and additional constraints on genomes. A *sorting scenario* of length d from Π into Γ is a series of genomes $\Pi_0, \ldots, \Pi_d$ such that $\Pi_0 = \Pi$, $\Pi_d = \Gamma$ and for each i, Π_{i+1} is a legal genome (under the model constraints) that is a result of an allowed operation on Π_i. The *sorting distance* is the length of a shortest sorting scenario from Π into Γ. We call Π the *source* genome and Γ the *target* genome. The *sorting problem* under model $\mathcal{O}$ receives as input Π and Γ, and looks for a sorting scenario of minimum length from Π to Γ. Figure 10.3 shows a sorting scenario of length 5 from $(A\ B\ C\ D\ E\ F)$ to $(A\ -E\ -C\ D\ -F)$ in a model allowing deletions and inversions.

10.2.1.1 Operations

Reversal. An *inversion* of a signed genomic string reverses the string and multiplies all elements by -1. Hence, the inversion of $(2\ -3\ 5\ -1)$ denoted as $-(2\ -3\ 5\ -1)$, is $(1\ -5\ 3\ -2)$. For a string $S = s_1, \ldots, s_n$, $S[i, j]$ is the substring $s_i, \ldots, s_j$. Let C be a chromosome string. A *reversal* $\rho(i, j)$ inverts $C[i, j]$, resulting in a new chromosome $C' = C[1, \ldots, i-1] \cdot -C[i, \ldots, j] \cdot C[j+1, \ldots, m]$, where $\cdot$ is the concatenation operator. For example, $\rho(3, 5)$ of $C = (1\ 3\ 2\ 4\ 5\ 6)$ is $C' = (1\ 3\ -5\ -4\ -2\ 6)$. Reversals can be similarly defined on a single chromosome in the genome graph, by cutting two adjacencies and reconnecting the loose extremities such that the result is a linear chromosome. A reversal on a labeled or ordinary genome is a reversal on one of its chromosomes. Reversals for general (not ordinary) unlabeled genomes are not defined as they may have several chromosome decompositions. See Figs. 10.1b and 10.8a, b.

Translocation. Let $C = c_1, \ldots, c_m$ and $D = d_1, \ldots, d_n$ be two linear chromosomes in string representation of an ordinary or labeled genome. A *translocation* $tr(C, D, i, j)$ transforms C and D into two new chromosomes, either $C[1, \ldots, i] \cdot D[j+1, \ldots, n]$ and $D[1, \ldots, j] \cdot C[i+1, \ldots, m]$, or $C[1, \ldots, i] \cdot -D[1, \ldots, j]$ and $-C[i+1, \ldots, m] \cdot D[j+1, \ldots, n]$. That is, the adjacencies C_i, C_{i+1} and D_j, D_{j+1} are cut, and the four loose ends are reconnected in a new way. An equivalent definition can be made on chromosome graphs, i.e., breaking an adjacency on each chromosome and reconnecting the nodes (see Figs. 10.1e and 10.9). Again notice that translocations are not uniquely defined for general unlabeled genomes.

DCJ. A *double-cut-and-join* (*DCJ*) is an operation that cuts two adjacencies and reconnects the four loose ends in a new way into two adjacencies. It can be applied on labeled and unlabeled genomes. A DCJ can take one of the following forms:

1. If adjacencies $\{p, q\}, \{r, s\} \in \Pi$ are cut, replace them with either $\{p, r\}, \{q, s\}$ or $\{p, s\}, \{r, q\}$ (Figs. 10.8 and 10.9).
2. If adjacency $\{p, q\} \in \Pi$ is cut and telomere $\{r\} \in \Pi$ is involved, replace them with either $\{p, r\}, \{q\}$ or $\{r, q\}, \{p\}$ (Fig. 10.10).

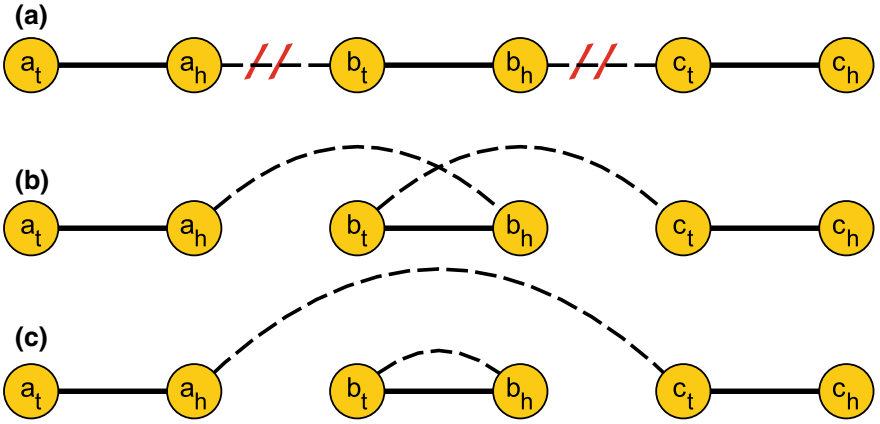


Fig. 10.8 Reversal and DCJ. **a** The genome graph of $(a\ b\ c)$; the two diagonal stripes correspond to the cut adjacencies. **b** The genome $(a\ -b\ c)$ is a result of a reversal or a DCJ. **c** The genome $\{(a\ c), <b>\}$ corresponds to the other DCJ option

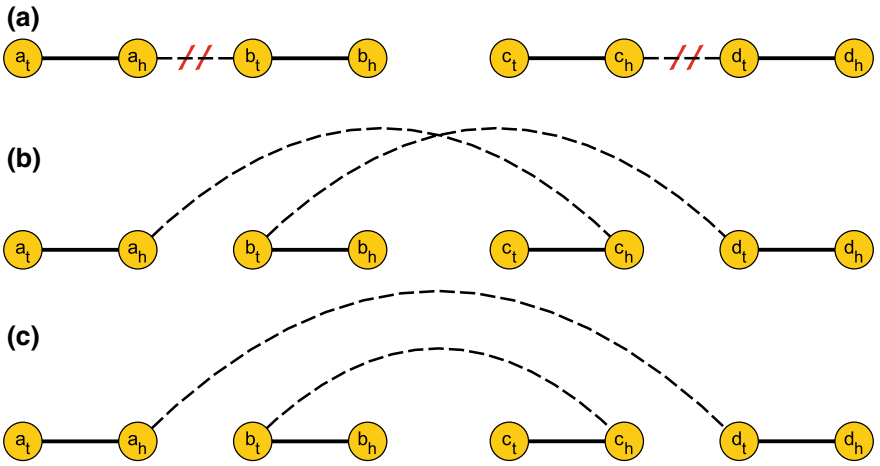


Fig. 10.9 Translocation. **a** Two chromosomes $\{(a\ b), (c\ d)\}$; the two diagonal stripes correspond to the cut adjacencies. **b**, **c** Two possible translocations (or DCJs) corresponding to $\{(a\ -c), (-b\ d)\}$ (**b**) and $\{(a\ d), (c\ b)\}$ (**c**)

3. If telomeres $\{q\}, \{r\} \in \Pi$ are involved, replace them with an adjacency $\{r, q\}$ thereby joining the two chromosomes (Fig. 10.11). This operation is referred as a *fusion* or a *join*.
4. If adjacency $\{p, q\} \in \Pi$ is cut and an empty adjacency is involved, replace them with two telomeres $\{p\}, \{q\}$ (Fig. 10.11). Hence, a linear chromosome containing the adjacency is cut into two chromosomes, or becomes linear if it was circular. This operation is referred as a *fission* or a *cut*.

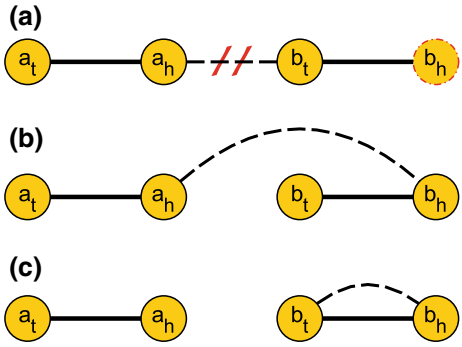


Fig. 10.10 DCJs on telomeres. **a** Chromosome $(a\ b)$; the diagonal stripes and the dotted circle show the cut adjacency and telomere involved. **b**, **c**: Two possible DCJs corresponding to $(a\ -b)$ (**b**), i.e., reversal, and $\{(a), < b >\}$ (**c**)

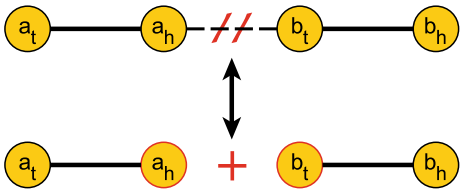


Fig. 10.11 Single-cut-or-join (SCJ). A cut breaks an adjacency into two telomeres corresponding to the transition from the top to the bottom genome. A join is the reverse operation corresponding to the transition from the bottom to the top genome

Note that a DCJ realizes both reversals (when the two adjacencies come from the same chromosome) and translocations (when they are from different chromosomes). When the adjacencies that are cut are from the same chromosome, the result of a DCJ can also be splicing out of a segment between the cuts into a separate cyclic chromosome.

SCoJ. A *single-cut-or-join* (*SCoJ*) operation either cuts an adjacency or joins two telomeres, respectively (Fig. 10.11).

In the next section, we briefly review basic results on ordinary genome models. As our focus is primarily on multiple-copy problems, we only skim selected results. The interested reader can find much more information on this topic in [40, 126].

10.2.2 Breakpoint Distance

The *breakpoint* (*BP*) *distance* is a simple measure of dissimilarity between two genomes that are not related to a specific type of operation. Generally speaking, the breakpoint distance measures the number of adjacencies and telomeres that are

in one genome but not in the other. The breakpoint distance has several definitions depending on the different weights of common adjacencies and telomeres [78, 103].

For two ordinary genomes Π and Γ over the same n genes, Tannier et al. [103] give the following formula for the breakpoint distance:

$$d_{BP} = n - (A + E/2) \tag{10.1}$$

where A is the number of common adjacencies and E is the number of common telomeres of Π and Γ. Clearly, the distance is computable in linear time.

10.2.3 Reversal and Translocation Distances

Given signed permutations Π and Γ over the same n genes, we seek a shortest sequence of reversals from Π into Γ. We can assume w.l.o.g. that Γ is the identity permutation $(1, \ldots, n)$.

Sorting signed permutations by reversals is undoubtedly the most famous GR problem [9]. In their seminal work, Hannenhalli and Pevzner gave the first polynomial-time algorithm for the problem [49]. Since then, the theory was greatly simplified [6, 13, 17, 57]. Bader, Moret, and Yan have shown that finding the reversal distance can be done in linear time [6], whereas computing a shortest sorting scenario can be done in $O(n^{3/2})$ [44, 102]. Interestingly, sorting *unsigned* permutations (i.e., without gene orientations) by reversals is NP-hard [25].

The problem of sorting multi-chromosomal genomes by translocations was first introduced by Kececioglu and Ravi [59]. Hannenhalli [46] gave the first polynomial-time algorithm for the problem and an improved, linear time algorithm was introduced by Bergeron et al. [15].

Sorting by reversals and translocations was proved to be polynomial by Hannenhalli and Pevzner [47], who reduced the problem of sorting by reversals. The theory and algorithm were later slightly corrected and revised [16, 53, 72, 73, 105]. The algorithm was used to compute for the first time a sorting scenario and distance between the mouse and human genome [47]. Interestingly, the distance achieved closely matched a prediction by Nadeau and Taylor from the 1980s [68]. Efficient implementations of the algorithms for sorting by reversals and translocations are available as part of the *GRAPPA* [6] and *GRIMM* [106] tools. Those tools also use the ability to compute exact pairwise distances efficiently in order to compute a tree of evolution by reversals and translocations among multiple species, albeit heuristically.

The main representation used for the analysis of this problem (and other rearrangement models) is the Breakpoint Graph (*BG*). Given two genomes Π and Γ, the *breakpoint graph* $BG(\Pi, \Gamma)$ is an undirected graph whose nodes are the extremities of both genomes, and whose edges are the adjacencies of both genomes distinguished by color. Edges corresponding to Π (Γ) adjacencies are called red or Π-edges (blue or Γ-edges, respectively). See an example in Fig. 10.12.

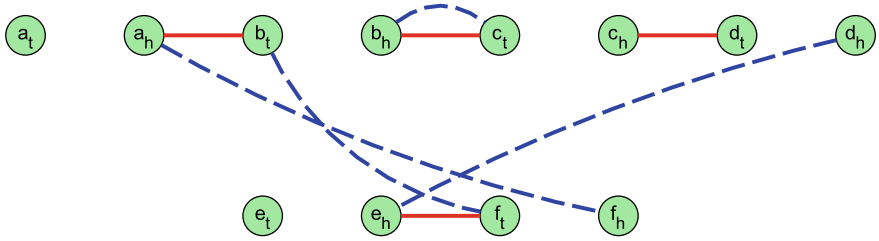


Fig. 10.12 A breakpoint graph for $\Pi = \{(a\ b\ c\ d), (e\ f)\}$ and $\Gamma = \{(a\ -f\ b\ c), (d\ -e)\}$. Π-edges are solid; Γ-edges are dashed

Hannenhalli and Pevzner [49] gave a formula for the reversal distance between signed permutations based on the number of cycles in the BG and certain structures in it called "hurdles" and "fortresses". The distance formula for sorting by reversals and translocations has been devised over the years and depends on more complex structures in the BG [16, 47, 53, 72, 105]. The definitions of these structures are beyond the scope of this review, so the exact distance formulas are omitted. Bergeron [13] and Jean and Nikolski [53] give fairly elementary presentations for sorting by reversals and sorting by reversals and translocations, respectively, including good expositions of structures and the distance formulas.

10.2.4 DCJ Distance

The inputs for this model are two ordinary genomes Π and Γ over the same set of n genes. The operations allowed in this model are DCJs. The DCJ operation, introduced by Yancopoulos et al. [115], has gained much attention in GR models in the last decade. The reason is that DCJs capture both reversals and translocations (but also splicing out a circular sub-chromosome) while allowing much simpler algorithms. Both the distance and an optimal sorting scenario can be computed in linear time [14].

In the analysis of this problem, a new graph representation was introduced. The *adjacency graph* $AG(\Pi, \Gamma)$ of genomes Π and Γ is a bipartite undirected multigraph whose set of nodes are the adjacencies and telomeres of Π and Γ. Therefore, each node is a set of one or two extremities. Nodes belonging to $\Pi(\Gamma)$ are called red- or Π-nodes (blue- or Γ-nodes, respectively). For every Π-node u and Γ-node v, there are $|u \cap v|$ edges between u and v, i.e., there is an edge for each common extremity between the two nodes. Note that $BG(\Pi, \Gamma)$ is the line graph of $AG(\Pi, \Gamma)$ and vice versa. (The line graph of $G = (V, E)$ is the graph on E in which $x, y \in E$ are adjacent as vertices iff they are adjacent as edges in G). See Fig. 10.13.

Bergeron et al. [14] prove that for ordinary genomes Π and Γ defined over the same set of n genes:

$$d_{DCJ} = n - (C + I/2) \tag{10.2}$$

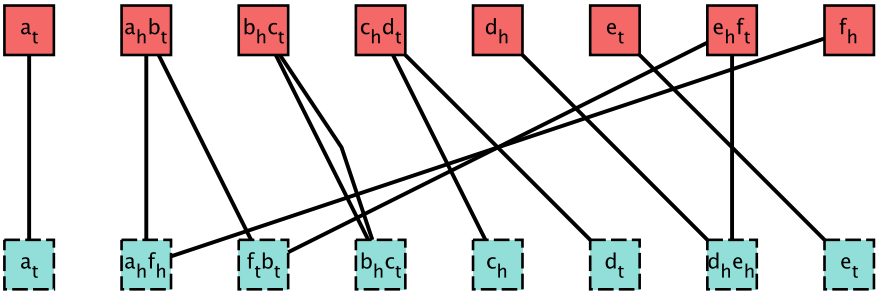


Fig. 10.13 An adjacency graph for $\Pi = \{(a\ b\ c\ d), (e\ f)\}$ and $\Gamma = \{(a\ -f\ b\ c), (d\ -e)\}$. Π-nodes are solid; Γ-nodes are dashed

where C is the number of cycles and I is the number of odd length paths starting and ending in telomeres in $AG(\Pi, \Gamma)$. For example, the AG in Fig. 10.13 has one cycle and two odd length paths. Thus, since there are six genes, the DCJ distance between the two genomes in this case is 4. Notice that there are two additional even length paths in the graph but they do not affect the distance formula.

10.2.5 SCoJ Distance

The inputs for this model are two ordinary genomes Π and Γ over the same set of n genes. The operations allowed in this model are SCoJs. Similar to DCJ, the SCoJ distance and scenario can be found in linear time [39]. Some rearrangement problems for which no polynomial solution is known for DCJ and other operations are known to be tractable for SCoJ distance. We give examples of such problems in Sect. 10.3.1.

For two ordinary genomes Π and Γ over the same n genes, let $A_\Pi(A_\Gamma)$ be the set of adjacencies of Π (Γ, respectively). The SCoJ distance is given by [39]

$$d_{SCoJ} = |A_\Pi| + |A_\Gamma| - 2|A_\Pi \cap A_\Gamma| \tag{10.3}$$

10.3 Multi-copy Models in Species Evolution

This section discusses multi-copy GR models inspired by species evolution. In Sect. 10.3.1, we present models allowing whole genome duplication events, but no other copy number changes. The models in Sect. 10.3.2 allow for the insertion and deletion of new genomic segments but do not account for multiple copies of segments. Models in Sect. 10.3.3 handle genomes with multiple copies of each gene but do not allow numeric operations. Section 10.3.4 describes a few models that can

handle genomes with multiple gene copies and allow numerical operations such as deletions or duplications.

We limit our discussion here to distance problems between two genomes. We refer the reader to the review by El-Mabrouk and Sankoff on the analysis of gene order evolution beyond single-copy genes [37], which discusses in depth the phylogenetic aspects of GR models in the context of species evolution.

10.3.1 Polyploidy

We discuss here problems motivated by *whole genome duplication* (*WGD*) events in species evolution. WGD is viewed as a fundamental step in evolution, as doubling of the gene contents allows great diversification of gene functions. For example, strong evidence for WGD events was reported for yeast [114] and for plant genomes [20]. The basic question tackled by these formulations is finding a shortest sorting scenario between a given ancestral genome (before or right after WGD) and a given extant genome under GD models allowing only structural operations.

A *duplicated genome* (either labeled or unlabeled) is a genome in which every gene has CN = 2. For an ordinary genome Π over $\mathcal{G}$, a *doubled genome* $2\Pi = \Pi \cup \Pi$ is an unlabeled duplicated genome over $2\mathcal{G} = \mathcal{G} \cup \mathcal{G}$ in which every gene, adjacency, and telomere has two copies. For example, if $\Pi = \{\{a_t\}, \{a_h, b_h\}, \{b_h\}\}$ then $2\Pi = \{\{a_t\}, \{a_h, b_h\}, \{b_h\}, \{a_t\}, \{a_h, b_h\}, \{b_h\}\}$.

The *double distance problem* [2] is defined as follows. Given an ordinary genome Π over $\mathcal{G}$, a labeled duplicated genome Θ and an operation distance measure d, find the minimum distance of Θ to some labeling of 2Π. Formally, the *double distance* between Π and Θ is

$$dd(\Pi, \Theta) = \min_{\Gamma \in L(2\Pi)} d(\Gamma, \Theta) \tag{10.4}$$

where $L(2\Pi)$ is the set of all possible labelings of 2Π. The double distance problem can be solved in linear time for the BP [60] and the SCoJ measures [39]. However, it is NP-hard under the DCJ distance [103].

Given a labeled duplicated genome Θ and an operation distance measure d, the *genome halving problem* seeks to find an ordinary genome Π that minimizes the double distance to Θ [36]. Formally, the *halving distance* of Θ is defined as

$$hd(\Theta) = \min_{\Pi} dd(\Pi, \Theta) \tag{10.5}$$

The halving distance can be solved in linear time for the BP measure, but if we restrict the genome Π to be linear or unichromsomal it becomes NP-hard [60]. For the SCoJ distance, the problem is solvable in linear time even when Π is restricted to be a linear or circular genome [39]. Under the DCJ distance, the halving problem can be solved in linear time [66, 111] even with Π restricted to a unichromsomal genome [1].

A generalization of the halving problem for finding an ordinary pre-WGD genome given an extant genome with exactly $m > 2$ copies is called *genome aliquoting* [111]. Aliquoting is polynomially solvable for the BP [112] and SCoJ [39] distances, while a 2-approximation algorithm is known for the problem under the DCJ distance [112]. Recently, efficient ILP formulations were suggested for genome halving and aliquoting under the DCJ distance [5].

The *guided genome halving problem* tries to combine both the genome halving and double distance [125]. Given an ordinary genome Δ, a labeled duplicated genome Θ, and an operation distance measure d, find an ordinary genome Π that minimizes the sum of the double distance between Π and Θ, plus the distance between Π and Δ. Formally, the guided halving distance is

$$ghd(\Delta, \Theta) = \min_{\Pi} \left[dd(\Pi, \Theta) + d(\Pi, \Delta) \right] \tag{10.6}$$

The problem can be solved in $O(n^{1.5})$ time for the BP distance, but it becomes NP-hard with additional restrictions [60]. For the SCoJ distance, the problem has linear solutions even with restrictions to linear or circular genomes [39]. It is NP-hard for the DCJ distance [103].

10.3.2 Single-Copy Models with Indels

Models presented in this section allow new numerical operations while maintaining the assumptions of ordinary genomes. The input is two ordinary genomes Π and Γ over potentially different gene sets $\mathcal{G}_1$ and $\mathcal{G}_2$. The goal is to transform Π into Γ with structural operations and additional operations that introduce new genes or remove genes. All genomes in the sorting scenario must be ordinary.

Given a chromosome string $C = c_1, \ldots, c_n$, a *deletion* $del(i, j)$ produces a new chromosome $C[1, i-1] \cdot C[j+1, n]$. An *insertion* $ins(S, i)$ of a sequence $S = s_1, \ldots, s_m$ into a chromosome C at position i results in $C[1, i] \cdot S \cdot C[i+1, n]$ (see Fig. 10.1). Insertions and deletions are commonly referred to as *indel* operations [23]. Since these models assume that all genomes are ordinary, insertions cannot introduce new copies of genes. Instead, indels are used to add and remove genes that appear in one genome but not in the other.

El-Mabrouk was the first to address sorting permutations by reversals and indels and gave exact an algorithm and a heuristic for specific cases [35]. Improved bounds for this problem were later devised [113]. Yancopoulos and Friedberg [116] analyzed the problem of sorting ordinary genomes with DCJs and indels. Their model allowed to insert and delete genes that appear in the source or target genomes, and thus a possible sorting scenario can delete all the chromosomes of the source genome and insert the chromosomes of the target genome. Braga et al. [23] gave a linear time algorithm for finding a minimum sorting scenario with DCJs and indels, restricting indels to affect genes that are not common to the source and target genomes. The

problem is solvable in linear time even when DCJs and indels have different weights [97].

Braga et al. [22] introduced a new operation that generalizes both insertions and deletions. A *substitution* is an operation that replaces a sequence of consecutive genes with another sequence. This operation can be thought of as a deletion of the sequence to be replaced followed by an insertion of the new sequence in the same place. Notice that this operation can implement both deletions and insertions by taking an empty sequence as the new or old sequence, respectively. Sorting ordinary genomes with DCJs and substitutions can be solved in linear time [22], even when substitutions have different weights than DCJs [98].

10.3.3 Multi-copy Models Without Duplications/Deletions

In this section, we focus on comparing genomes with multiple gene copies but without explicit deletion or duplication operations. The comparison can be used to assign orthology relationship between gene copies in the source and target genomes [26]. Given a source genome and a target genome with multiple gene copies, the general approach is to find a matching of the gene copies that minimizes some structural operation distance. Gene copies that are not matched are ignored, so they are implicitly deleted and do not incur the cost of a true deletion operation. Most formulations result in NP-hard problems.

There are three main formulation strategies depending on the cardinality of the matching of multi-copy genes:

- *Exemplar* strategy [86], in which in each genome, exactly one copy of each gene is selected and all other copies are ignored.
- *Intermediate* strategy [4], in which the same number of copies (at least one) for each gene is selected and matched between genomes, and all other copies are ignored.
- *Maximum matching* strategy [19], in which for each gene, the maximum possible gene copies (the smaller of the gene's CNs in the two genomes) are selected and matched between genomes, and the remaining copies are ignored.

Although most formulations are NP-hard, several exhaustive and heuristic algorithms have been suggested. In recent years, Integer Linear Programming (ILP) formulations presented by Shao and Moret were used to solve such problems, and have shown good results and scalability [92–94]. Table 10.1 summarizes selected results for different operations and different formulations.

The majority of hardness results, as well as exact and heuristic algorithms for these problems, originate from the *breakpoint graph decomposition* problem [25, 58]. The goal in this problem is to find a decomposition of a breakpoint graph into a maximum number of edge-disjoint alternating red/blue cycles. A similar maximum cycle decomposition can also be defined for the adjacency graph [91, 92]. Note that

Table 10.1 Multi-copy model results

Operations	Exemplar	Intermediate	Matching
BP	NP-hard [24] Branch and bound [69, 86] ILP [3, 93]	NP-hard [18] ILP [4, 94] Heuristics [4]	NP-hard [18] Branch and bound [19] ILP [4, 94] Heuristics [4]
Reversals and translocations	NP-hard [24]	NP-hard [26] ILP [101] Heuristics [26, 41]	NP-hard [26] ILP [101] Heuristics [26, 41]
DCJ	Branch and bound [117]	NP-hard [92]	NP-hard [92] ILP [92] Branch and bound [117] Approximation [83, 91]

such a decomposition induces a matching between genes and the maximum number of cycles minimizes an operation distance measure [91, 92].

10.3.4 Models with Duplications or Deletions

We now describe several models that include deletions or duplications as explicit numerical operations. The goal of all these models is to transform one genome representation into the other with minimum number of structural and numerical operations. Unlike the classical structural operations, numerical operations such as deletions and duplications have no standard definitions.

Chen et al. [26] analyzed a model for sorting unlabeled genomes with multiple gene copies using only reversals. Their heuristic, called *SOAR*, was the first method to assign orthology relationship between genes based on not only sequence similarity but also GRs. In a follow-up paper [41], the authors studied a model that allows reversals and single-gene duplications. The latter can insert new gene copies at arbitrary positions in the genome. They developed a heuristic called *MSOAR* for matching gene copies between the two input genomes such that the number of reversals plus gene duplications would be minimal. While SOAR requires every gene to have an equal number of copies in the two input genomes, MSOAR alleviates this assumption. In *MSOAR 2.0* [96], only tandem single-gene duplications are allowed, and again, an efficient heuristic for this sorting problem is given.

Kahn and Raphael [56] introduced a measure called the *string duplication distance* that models building a target string by repeatedly copying substrings of a fixed source string. The *string duplication* operation, $\delta_{s,t,p}(X)$, copies a substring $x_s, \ldots, x_t$ of string X and pastes it into another string Z at position p. Given a source string X without duplicate genes and a target string Y, the goal is to find a minimum length sequence of string duplications needed to build the string Y. The authors described a polynomial dynamic programming algorithm for computing the distance [56]. In a

follow-up work, they enhanced the model to allow substring deletions and inversions. A polynomial dynamic programming algorithm is given for computing the sorting problem [55]. The string duplication model was used for the analysis of repetitive segments in the human genome [54].

A model introduced by Bader [7] allows tandem duplications, segmental deletions and DCJs. Given a labeled chromosome C in string representation, a *tandem duplication* $td(i, j)$ inserts a new copy of the segment $C[i, \ldots, j]$ after the j'th position, i.e., the new chromosome is $C' = C[1, \ldots, i-1] \cdot C[i, \ldots, j] \cdot C[i, \ldots, j] \cdot C[j+1, \ldots, n]$ (Fig. 10.1). A *deletion* $del(i, j)$ removes the segment C[i,…,j] and produces $C' = C[1, \ldots, i-1] \cdot C[j+1, \ldots, n]$ (Fig. 10.1). The goal in the model is to find a minimum sorting scenario of the identity chromosome into the input multi-copy labeled chromosome. The author gave a lower bound and heuristic for the problem based on the structure of the breakpoint graph.

In a model presented by Shao and Moret [95], labeled genomes are sorted using DCJs and segmental duplications. A *segmental duplication* copies a segment of labeled genes $g_1, \ldots, g_m$ of a genome Σ and inserts the new labeled copy in Σ in a spot outside the original segment. The model allows different costs for different duplications and unit cost for DCJs. However, the optimization problem implicitly assumes that all segmental duplications either precede or follow all DCJ events. Given two labeled genomes Π, Γ, the goal is to find segmental duplications in Π and Γ, remove them, and then find a bijection between the remaining genes such that the cost of segmental duplications plus the DCJ distance is minimized. The authors analyzed this problem and gave an ILP formulation based on the adjacency graph cycle decomposition formulation proposed in [92], applied to a problem instance simplified by detection of optimal substructures.

Paten et al. [76] presented a model for genome evolution that does not fit entirely into the standard GR terminology. This model can represent both single-base substitutions and structural/numerical rearrangements such as DCJs, deletions, and duplications. They defined a data structure called *history graph*, which holds partial order information on the sequence of events. The goal is to find a full sequence of events consistent with the input history graph that minimizes the cost of substitutions and DCJs, while gene deletions and WGDs are free. The authors analyzed this problem and gave polynomially tractable bounds for the cost. In a follow-up paper, Zerbino et al. [124] further analyzed the history graph model and showed that the space of possible evolutionary histories can be sampled ergodically.

10.4 Multi-copy Models in Cancer

Cancer genomes are known to undergo structural and numerical changes [45]. These include inversions, chromosomal translocations, tandem duplications, segmental deletions, whole chromosome amplifications or losses, and more [109]. Figure 10.4 shows an example of a real cancer karyotype, and Fig. 10.14 shows a hypothetical sorting scenario for cancer evolution. A large research effort has focused on

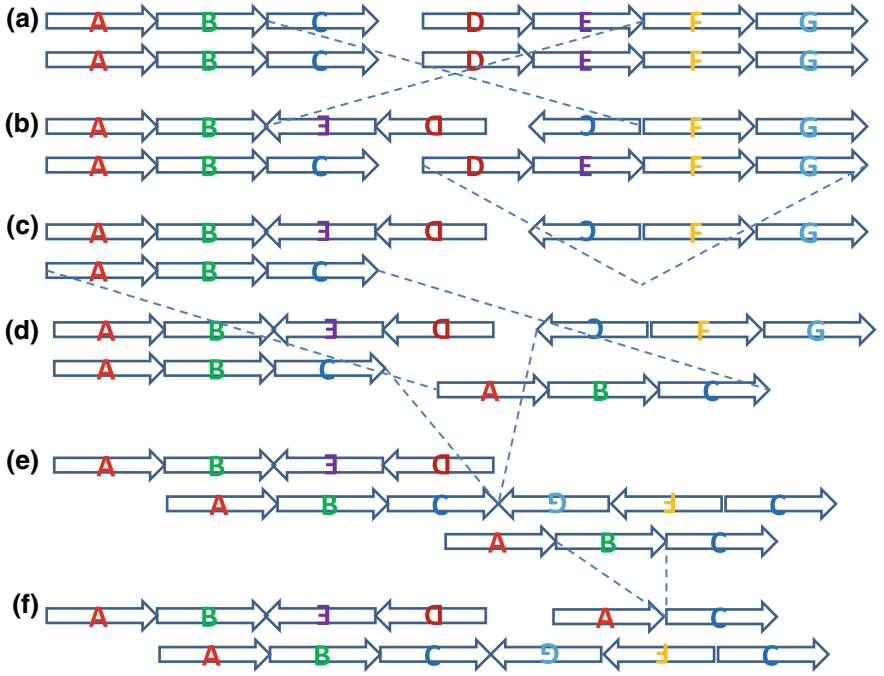


Fig. 10.14 A hypothetical sorting scenario for cancer evolution. **a** Normal diploid karyotype with two chromosomes. **a**–**b** Translocation. **b**–**c** Chromosome deletion. **c**–**d** Chromosome duplication. **d**–**e** Fusion. **e**–**f** Internal deletion. **f** The cancer karyotype. The breakpoints and telomeres involved in each operation are indicated by the broken lines

detecting signatures of these events in tumor genomic data. Currently, the effort uses mainly deep sequencing data [31], though traditional methods such as FISH and aCGH are still used to assess the CN of genomic regions. Accurate reconstruction of the numerical and structural variations remains a challenge, and a myriad of computational methods has been devised for this task [31, 104]. Some evolutionary GR models such as those presented in Sect. 10.3 could also be applied to cancer genomes. Nevertheless, the complexity of tumor karyotypes and their unique characteristics necessitate development of dedicated cancer GR models.

In Sect. 10.4.1, we discuss several classical GR models that were applied to cancer data. Section 10.4.2 describes CN edit distance problems in cancer. Section 10.4.3 presents a few other cancer models involving GRs.

10.4.1 Models with Duplications/Deletions

Here, we present several GR models with both structural and numerical operations that were designed to cancer data analysis. All models aim to find a sorting scenario from one genome representation into the other. The source genome is usually the normal genome from a healthy tissue, and the target genome is the tumor.

Ozery-Flato and Shamir [74] proposed a GR model designed specifically to analyze chromosomal aberrations in cancer. The inputs for the model are a normal unlabeled source genome with two identical copies of each chromosome and a tumor (target) genome. Both genomes are described as sets of chromosomes, each consisting of a sequence of segments. The goal is to sort the normal genome into the tumor with the fewest cuts, joins, chromosome duplications, and chromosome deletions. The authors proved a lower bound for the distance and presented a polynomial-time 3-approximation algorithm for the problem. They applied the algorithm to over 50,000 low-resolution karyotypes from the Mitelman database [65], which records cancer karyotypes reported in the scientific literature. Interestingly, the approximation algorithm gave an optimal solution in all but 30 karyotypes.

Bader [8] extended his previous model [7] in order to cope with cancer alterations. The revised model accepts multi-chromosomal genomes and allows chromosome deletions and duplications, tandem duplications, segmental deletions, and DCJs. A lower bound and a heuristic algorithm were devised, and applied to the Mitelman database [65]. The average calculated distance was 4.08, while the average lower bound was 2.72.

Zeira and Shamir [121] analyzed a model for genome sorting using cuts, joins, and whole chromosome duplications. In this model, an ordinary linear genome is to be transformed into a duplicated linear genome such that all intermediate genomes are linear. The authors gave a linear time algorithm for the sorting problem and showed that finding such a sequence with fewest duplications is NP-hard.

A more comprehensive model presented by Zeira and Shamir [122] accounted for the evolution of unlabeled genomes via DCJs, tandem duplications, segmental deletions, and chromosomal amplifications and deletions. They showed that the sorting problem is NP-hard and gave an ILP formulation that solves the problem exactly under some mild assumptions. The algorithm was applied to sort complex ovarian cancer genomes taken from TCGA sequencing data [12]. Figure 10.15 gives an example of a sorting scenario inferred by the ILP.

10.4.2 Copy Number Profile Distances

In this section, we discuss several models for edit distance between CN profiles. Unlike the genome representations in Sect. 10.2.1, these profiles give the number of copies of each segment (gene) but do not hold information about their order along the genome. A *copy number profile* (*CNP*) of a chromosome is a vector mapping

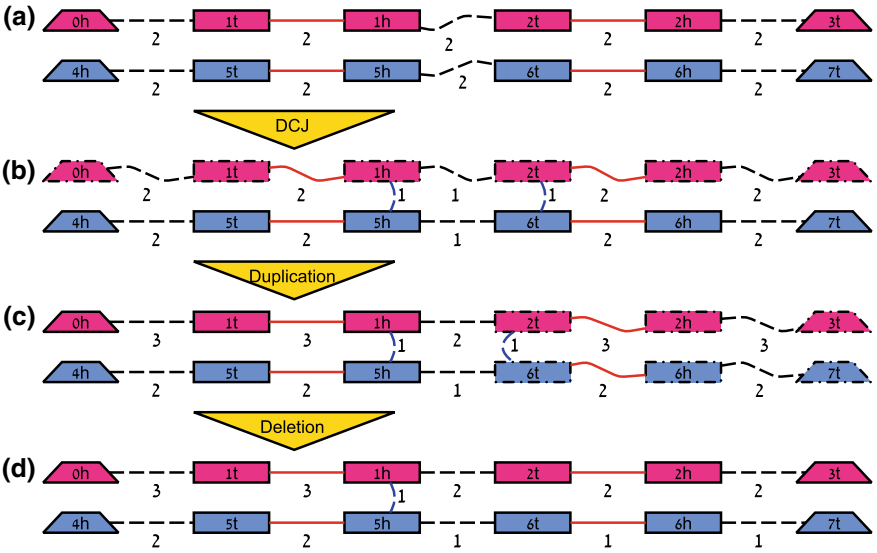


Fig. 10.15 Inferred GR scenario in ovarian cancer sample (TCGA-13-1411). A sequence of operations transforming a genome graph of chromosomes 1 (upper) and 3 (lower) from a diploid genome (**a**) to tumor genome (**d**). Square nodes represent segment extremities and trapezoid nodes represent telomeres. Dashed edges are adjacency edges, full straight (red) edges are interval edges, and dashed arcs (purple) are novel adjacencies caused by the tumor process. The number next to each edge is its CN. One operation transforms each genome graph into the one below. The operation type is listed in the triangle and the affected genes or adjacencies appear as dashed nodes and wavy edges, respectively, in the predecessor genome. The scenario was inferred using the ILP formulation of [122]

each gene to a nonnegative integer corresponding to the number of copies of the gene in the chromosome. As the order of the genes in a CNP is unknown, it is assumed to be some predefined order (typically the normal genome order). A *genome CNP* is a collection of its chromosome CNPs. We now define operations that transform CNPs and present several models for finding a sorting distance between CNPs.

Let $V = (v_1, \ldots, v_n)$ where $v_i \in \mathbb{N} \cup \{0\}$ be a CNP of a chromosome with n genes. A *copy number operation* (*CNO*) is a triple $c = (\ell, h, w)$ where $1 \leq \ell \leq h \leq n$ and $w \in \{1, -1\}$. We say that the operation is a *deletion* if $w = -1$ and an *amplification* if $w = 1$. Applying an operation c to a CNP V results in a new CNP $c(V) = (c(v_1), \ldots, c(v_n))$ such that for every $\ell \leq i \leq h$, $v_i > 0$ we have $c(v_i) = v_i + w$, and otherwise $c(v_i) = v_i$. In other words, the operation increases or decreases the CN of the genes in the interval $[\ell, h]$ if they have a positive CN, while the values of genes outside the interval and zero values are unchanged (see Fig. 10.16).

Chowdhury et al. [29] defined edit distance between CNPs obtained from FISH, where the edit operations are amplification or deletion of single gene, single chromosome, or the whole genome, and presented an algorithm for calculating the distance. The algorithm was exponential in the number of genes and therefore is limited to

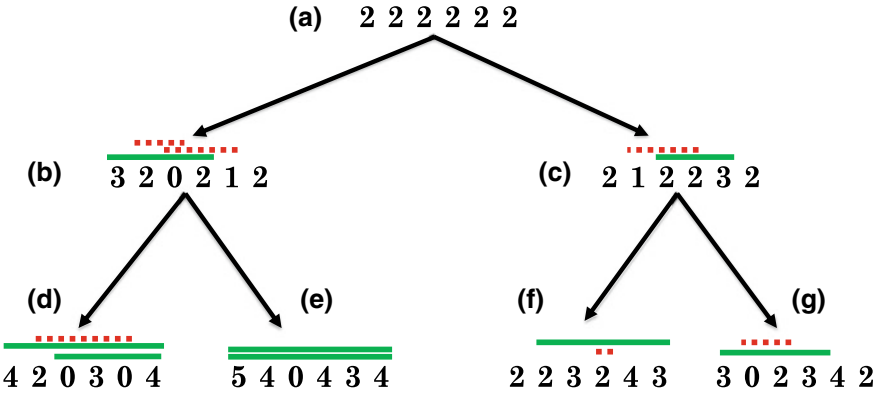


Fig. 10.16 Copy number profile evolution. A diploid CNP (**a**) evolves via CNOs into four extant CNPs (**d**, **e**, **f**, **g**). Dotted lines represent deletions and bold lines represent amplifications. The order of operations is from top to bottom. For instance, CNP **a** evolves into CNP **b** by a deletion of positions 2–3, a deletion of positions 3–5, and an amplification of positions 1–4 (in this order). The corresponding sequence of profiles is $2\,2\,2\,2\,2\,2 \rightarrow 2\,1\,1\,2\,2\,2 \rightarrow 2\,1\,0\,1\,1\,2 \rightarrow 3\,2\,0\,2\,1\,2$. The entire tree has six deletions and eight amplifications

low-resolution FISH data. An algorithm based on the pairwise distance matrix was used to heuristically infer tumor phylogenies from FISH single-cell data. A follow-up paper [28] accounted for different weights for different types of operations, again providing an exponential time algorithm.

Schwartz et al. [90] introduced a model that admits amplifications and deletions of general contiguous segments in a chromosome CNP. The edit distance between two CNPs is the minimum number of CNOs over all possible separations of the profiles into two alleles. The authors developed an algorithm called *MEDICC* for computing the edit distance, which uses finite-state transducers [67] and is exponential in the maximum CN. MEDICC was used to infer tumor phylogenies from CGH arrays of high-grade serous ovarian cancer samples [89].

Zeira et al. [123] analyzed the problem of sorting one CNP into another using a minimum number of CNOs. They showed that this problem is solvable in linear time and constant space. Notice that this edit distance is not symmetric and in fact there may not be any sequence of CNOs from one given CNP to another since genes with zero copies cannot reappear later in the sequence. To cope with this drawback, El-Kebir et al. [34] analyzed a symmetric version that given two CNPs aims to find a common ancestor profile that minimizes the sum of distances to these CNPs. They gave a pseudo-polynomial dynamic programming algorithm that is linear in the profile length and an ILP formulation.

In the more general cancer context, El-Kebir et al. [34] showed that it is NP-hard to build a phylogenetic tree whose leaves are the input CNPs that minimizes the total number of CNOs along edges in the tree (see Fig. 10.16) and gave a practical ILP formulation for this problem. Extending the CNP tree model, Zaccaria et al. [118] considered a model in which a fractional (non-integer) CNP is allowed, due to

the superposition of several CNPs of different subclones. The goal in this case is to deconvolve the fractional CNPs into a weighted sum of integer CNPs such that the phylogenetic tree built over them has minimum CNOs. A heuristic algorithm was given for the problem.

10.4.3 Other Cancer Models

Reconstruction of the exact cancer chromosomes based on short paired-end deep sequencing read data remains a hard challenge. There is a plethora of methods for detection of local rearrangement events and breakpoints [31], but only a few methods try to reconstruct the entire genome. Here, we describe a few methods designed for reconstructing cancer genomes. The output genome representation of such methods can be used as input to genome rearrangement models described earlier.

Oesper et al. [71] expanded the genome graph into a structure called the *interval adjacency graph*, which represents breakpoints, discordant reads, and CN information. Their method, called *PREGO*, uses the number of reads supporting each edge to resolve the CN of genomic segments and identify discordant adjacencies in the tumor genome, and maps this information to the graph. PREGO was shown to efficiently identify complex rearrangements in ovarian cancer data. Eitan and Shamir [33] expanded this model and tested it in extensive simulations and on real cancer data. Their analysis shows that perfect reconstruction of a complete karyotype based on short read data is very hard, but that by several measures, reasonably good reconstructions are obtainable.

Weaver, developed by Li et al. [62], is a different probabilistic graph model proposed in order to estimate both the CNs and interconnectivity of SVs. Weaver detects and quantifies CNs and SVs specific for each allele, and was also used for predicting partial timing of SVs relative to chromosome amplifications. A recent expansion of Weaver based on ILP formulation-enabled improved prediction of SV phasing and interconnectivity [81].

A probabilistic framework based on breakpoint graphs was presented by Greenman et al. [43] for the analysis of mutations and karyotypes from sequencing data. This work tries to reconstruct both the temporal sequence of rearrangements and assemble genomic segments into karyotypes. It uses allelic integer CNs for each segment, the adjacencies between segments, and the multiplicity distribution of somatic SNVs. Taking into consideration SNVs can disambiguate some sorting scenarios, since duplicated segments carry the SNVs of the original one. The method can derive partial order of accumulating numerical and single-nucleotide mutations. The framework, called *GRAFT*, was demonstrated to work well with a breast cancer sample and cancer cell lines, albeit with limitations imposed by the data quality and the genome complexities.

Epilogue

GR models and theory have developed significantly in the last couple of decades. Earlier models focused on species evolution and accounted for simple genomes with a single copy of each gene. These models concentrated on different operations transforming one genome into the other. The elegant theory and algorithms underlying the elementary models served as a basis to more complex models to come. Despite its oversimplification of biology, the research of genomic sorting has been fruitful, both computationally and biologically.

Later studies started addressing more complex genome models where each segment may have two copies. These studies were motivated by whole genome duplication events, which double the gene content of a genome. Most formulated problems were shown to be NP-hard, but heuristics based on the theory developed were utilized to derive ancestral genomes of several species.

More complex evolutionary models allow for arbitrary number of copies and numerical operations such as insertions, deletions, and duplications. Models vary in their assumptions and in the operations they allow. Most of the problems are NP-hard with several heuristic and exact algorithms proposed.

Family-free genome comparison was recently proposed as an alternative multiple gene copy model [21, 32, 64, 84]. In this setting, each gene is unique but we are additionally provided with a pairwise similarity score between every pair of genes, for instance, based on their sequence similarity. This generalizes the multi-copy models as one can assign similarity of 1 to copies of the same gene and 0 between all others. In one formulation of the family-free DCJ distance, the goal is to find a matching between genes such that the DCJ distance minus the weight of the matching is minimal [64]. Studies showed that this problem is NP-hard and even hard to approximate, and gave heuristics and ILP formulations [64, 84].

Cancer now provides a key motivation for the development of GR models handling multiple copies. During tumor progression, the genome accumulates both structural and numerical changes, thus resulting in a complex genome with varying number of gene copies. The various models trying to represent tumor evolution differ in the type of data they rely on, types of events they allow, and other assumptions. Even determining a tumor's genome and identifying structural and numerical variations (i.e., reconstructing the tumor karyotype) remains a tough problem due to the data and genome complexity as well as tumor heterogeneity. Therefore, sorting cancer genomes remains a challenging task, and better models and algorithms are needed.

Some cancer genomes were explained by complex structural and numerical events that are beyond the models discussed here. For example, a *breakage-fusion-bridge* (*BFB*) is an event in which a loss of a chromosome's end is followed by "doubling-up" and fusion of the surviving part (i.e., a chromosome (a, b) is replaced by $(a, -a)$). In a BFB cycle, this process is repeated several times. Detection of BFB cycles can be done using sequencing and CN data [119, 120]. Dramatic rearrangement events also include *chromothripsis* and *chromoplexy*, in which one or more chromosomes are shattered into many pieces and some of the pieces are assembled in random order. Identifying these events in cancer genomes from sequencing data is still a

hard challenge [70]. Computational models are in need to account for such events in the analysis of cancer evolution.

Advanced sequencing technologies could help in tackling GR problems in cancer. Long-read sequencing techniques such as those of Pacific Biosciences and Oxford Nanopore can link distant DNA segments providing additional information on the relative location of different copies and simplify breakpoint identification [52, 82]. The linked short reads sequencing technology of 10X Genomics was recently shown to help in identifying structural variations in cancer genomes [38]. We expect these technologies and others to play a prominent role in GR analysis in cancer in the years to come.

Single-cell sequencing technologies open new opportunities and challenges in computational cancer analysis [110]. Specifically, variations between individually sequenced cells taken from a tumor have been used to identify its evolutionary history [51, 61]. Detection of SVs and CNAs in single-cell sequencing is still a tough challenge due to the noise and biases in the data [42, 108]. The use of single-cell SVs or CNAs for clonal reconstruction has not been addressed yet, to the best of our knowledge. Additionally, one might use the heterogeneity among cells and their abundance in order to guide the rearrangement scenario. Alternatively, given a rearrangements scenario, we can try to map cells to stages along this sequence.

Box 3 List of acronyms

aCGH—array comparative genomic hybridization
AG—adjacency graph
BFB—breakage-fusion-bridge
BG—breakpoint graph
BP—breakpoint
CN—copy number
CNA—copy number alteration
CNO—copy number operation
CNP—copy number profile
DCJ—double cut and join
DNA—Deoxyribonucleic Acid
FISH—fluorescence in situ hybridization
GR—genome rearrangement
ILP—integer linear programming
SCoJ—single cut or join
SNV—single nucleotide variation
SV—structural variation
WGD—whole genome duplication

Acknowledgements We thank Nimrod Rappoport for helpful comments. Study supported in part by the Naomi Prawer Kadar Foundation, the Bella Walter Memorial Fund of the Israel Cancer Association and by Len Blavatnik and the Blavatnik Family Foundation. R.Z. was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University.

References

1. Alekseyev, M.A., Pevzner, P.A.: Colored de Bruijn graphs and the genome halving problem. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**(1), 98–107 (2007). https://doi.org/10.1109/TCBB.2007.1002. URL http://dl.acm.org/citation.cfm?id=1229968.1229980
2. Alekseyev, M.A., Pevzner, P.A.: Whole genome duplications and contracted breakpoint graphs. SIAM J. Comput. **36**(6), 1748–1763 (2007)
3. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: A pseudo-boolean programming approach for computing the breakpoint distance between two genomes with duplicate genes. In: Proceedings of the RECOMB-CG, pp. 16–29. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74960-8_2
4. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. J. Comput. Biol. **15**(8), 1093–115 (2008). https://doi.org/10.1089/cmb.2008.0061. http://online.liebertpub.com/doi/abs/10.1089/cmb.2008.0061
5. Avdeyev, P., Alexeev, N., Rong, Y., Alekseyev, M.A.: A unified ILP framework for genome median, halving, and aliquoting problems under DCJ. In: Meidanis, J., Nakhleh, L. (eds.) Proceedings of the RECOMB-CG, pp. 156–178. Springer International Publishing, Cham (2017)
6. Bader, D.A., Moret, B.M., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J. Comput. Biol. **8**(5), 483–491 (2001). https://doi.org/10.1089/106652701753216503. http://www.liebertonline.com/doi/abs/10.1089/106652701753216503
7. Bader, M.: Sorting by reversals, block interchanges, tandem duplications, and deletions. BMC Bioinform. **10**(Suppl 1), S9 (2009). https://doi.org/10.1186/1471-2105-10-S1-S9. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2648760&tool=pmcentrez&rendertype=abstract
8. Bader, M.: Genome rearrangements with duplications. BMC Bioinform. **11**(Suppl 1), S27 (2010). https://doi.org/10.1186/1471-2105-11-S1-S27. http://www.biomedcentral.com/1471-2105/11/S1/S27
9. Bafna, V., Pevzner, P.: Genome rearrangements and sorting by reversals. SIAM J. Comput. **25**(2), 272–289 (1996). https://doi.org/10.1137/S0097539793250627
10. Barillot, E., Calzone, L., Hupé, P., Vert, J.P., Zinovyev, A.: Computational Systems Biology of Cancer. CRC Press (2012)
11. Beerenwinkel, N., Greenman, C.D., Lagergren, J.: Computational cancer biology: an evolutionary perspective. PLoS Comput. Biol. **12**(2), e1004717 (2016). https://doi.org/10.1371/journal.pcbi.1004717. http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004717
12. Bell, D., et al.: Integrated genomic analyses of ovarian carcinoma. Nature **474**(7353), 609–15 (2011). https://doi.org/10.1038/nature10166. http://www.nature.com/doifinder/10.1038/nature10166
13. Bergeron, A.: A very elementary presentation of the Hannenhalli-Pevzner theory. Discrete Appl. Math. **146**(2), 134–145 (2005). https://doi.org/10.1016/j.dam.2004.04.010. http://linkinghub.elsevier.com/retrieve/pii/S0166218X04003440
14. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M. (eds.) Proceedings of the Workshop Algorithms in Bioinformatics (WABI).

Lecture Notes in Computer Science, vol. 4175, pp. 163–173. Springer (2006). https://doi.org/10.1007/11851561_16
15. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. J. Comput. Biol. **13**(2), 567–578 (2006)
16. Bergeron, A., Mixtacki, J., Stoye, J.: A new linear time algorithm to compute the genomic distance via the double cut and join distance. Theor. Comput. Sci. **410**(51), 5300–5316 (2009). https://doi.org/10.1016/j.tcs.2009.09.008
17. Berman, P., Hannenhalli, S.: Fast sorting by reversal. In: Proceedings of the Combinatorial Pattern Matching, pp. 168–185. Springer, Berlin, Heidelberg (1996). https://doi.org/10.1007/3-540-61258-0_14. http://link.springer.com/10.1007/3-540-61258-0_14
18. Blin, G., Chauve, C., Fertin, G., Rizzi, R., Vialette, S.: Comparing genomes with duplications: a computational complexity point of view. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**(4), 523–534 (2007). https://doi.org/10.1109/TCBB.2007.1069. http://www.ncbi.nlm.nih.gov/pubmed/17975264, http://ieeexplore.ieee.org/document/4359864/
19. Blin, G., Fertin, G., Chauve, C.: The breakpoint distance for signed sequences. In: Proceedings of the 1st Conference on Algorithms and Computational Methods for Biochemical and Evolutionary Networks, vol. 3, pp. 3–16. King's College London Publications (2004)
20. Bowers, J.E., Chapman, B.A., Rong, J., Paterson, A.H.: Unravelling angiosperm genome evolution by phylogenetic analysis of chromosomal duplication events. Nature **422**(6930), 433 (2003)
21. Braga, M.D.V., Chauve, C., Doerr, D., Jahn, K., Stoye, J., Thévenin, A., Wittler, R.: The potential of family-free genome comparison. In: Models and Algorithms for Genome Evolution, pp. 287–307. Springer (2013)
22. Braga, M.D.V., Machado, R., Ribeiro, L.C., Stoye, J.: Genomic distance under gene substitutions. BMC Bioinform. **12**(Suppl 9), S8 (2011). https://doi.org/10.1186/1471-2105-12-S9-S8. http://www.biomedcentral.com/1471-2105/12/S9/S8
23. Braga, M.D.V., Willing, E., Stoye, J.: Double cut and join with insertions and deletions. J. Comput. Biol. **18**(9), 1167–84 (2011). https://doi.org/10.1089/cmb.2011.0118. http://www.ncbi.nlm.nih.gov/pubmed/21899423
24. Bryant, D.: The complexity of calculating exemplar distances. In: Sankoff, D., Nadeau, J.H. (eds.) Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families, pp. 207–211. Springer Netherlands, Dordrecht (2000). https://doi.org/10.1007/978-94-011-4309-7_19
25. Caprara, A.: Sorting by reversals is difficult. In: Proceedings of the Annual Conference on Research in Computational Molecular Biology, pp. 75–83. New York, NY, USA (1997). https://doi.org/10.1145/267521.267531. http://dl.acm.org/citation.cfm?id=267521.267531
26. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., Jiang, T.: Assignment of orthologous genes via genome rearrangement. IEEE/ACM Trans. Comput. Biol. Bioinform. **2**(4), 302–15 (2005). https://doi.org/10.1109/TCBB.2005.48. http://dl.acm.org/citation.cfm?id=1100863.1100950
27. Chinwalla, A.T., et al.: Initial sequencing and comparative analysis of the mouse genome. Nature **420**(6915), 520–562 (2002). https://doi.org/10.1038/nature01262. http://www.nature.com/doifinder/10.1038/nature01262
28. Chowdhury, S.A., Gertz, E.M., Wangsa, D., Heselmeyer-Haddad, K., Ried, T., Schäffer, A.A., Schwartz, R.: Inferring models of multiscale copy number evolution for single-tumor phylogenetics. Bioinformatics **31**(12), i258–67 (2015). https://doi.org/10.1093/bioinformatics/btv233. http://bioinformatics.oxfordjournals.org/content/31/12/i258.full
29. Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., Ried, T., Schäffer, A.A., Schwartz, R.: Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. PLoS Comput. Biol. **10**(7), e1003740 (2014). https://doi.org/10.1371/journal.pcbi.1003740. http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003740
30. Ding, L., et al.: Clonal evolution in relapsed acute myeloid Leukaemia revealed by whole-genome sequencing. Nature **481**(7382), 506–10 (2012). https://doi.org/10.1038/nature10738

31. Ding, L., Wendl, M.C., McMichael, J.F., Raphael, B.J.: Expanding the computational toolbox for mining cancer genomes. Nat. Rev. Genet. **15**(8), 556–570 (2014). https://doi.org/10.1038/nrg3767
32. Doerr, D., Feijão, P., Stoye, J.: Family-free genome comparison. In: Setubal, J.C., Stoye, J., Stadler, P.F. (eds.) Comparative Genomics: Methods and Protocols, pp. 331–342. Springer New York, NY, USA (2018). https://doi.org/10.1007/978-1-4939-7463-4_12
33. Eitan, R., Shamir, R.: Reconstructing cancer karyotypes from short read data: the half empty and half full glass. BMC Bioinform. **18**(1), 488 (2017). https://doi.org/10.1186/s12859-017-1929-9. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1929-9
34. El-Kebir, M., Raphael, B.J., Shamir, R., Sharan, R., Zaccaria, S., Zehavi, M., Zeira, R.: Complexity and algorithms for copy-number evolution problems. Algorithm Mol. Biol **12**(1), 13 (2017). https://doi.org/10.1186/s13015-017-0103-2. http://almob.biomedcentral.com/articles/10.1186/s13015-017-0103-2
35. El-mabrouk, N.: Sorting signed permutations by reversals and insertions/deletions of contiguous segments. J. Discrete Algs. **1**(1), 105–122 (2001)
36. El-Mabrouk, N., Nadeau, J.H., Sankoff, D.: Genome halving. In: Farach-Colton, M. (ed.) Proceedings of Combinatorial Pattern Matching, pp. 235–250. Springer, Berlin, Heidelberg (1998)
37. El-Mabrouk, N., Sankoff, D.: Analysis of gene order evolution beyond single-copy genes. Methods Mol. Biol. **855**, 397–429 (2012). https://doi.org/10.1007/978-1-61779-582-4_15. http://www.ncbi.nlm.nih.gov/pubmed/22407718
38. Elyanow, R., Wu, H.T., Raphael, B.J.: Identifying structural variants using linked-read sequencing data. Bioinformatics **34**(2), 353–360 (2018). https://doi.org/10.1093/bioinformatics/btx712. https://academic.oup.com/bioinformatics/article/34/2/353/4590027
39. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–29 (2011). https://doi.org/10.1109/TCBB.2011.34. http://www.ncbi.nlm.nih.gov/pubmed/21339538
40. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press (2009). http://www.google.co.il/books?hl=en&lr=&id=_caK_vcdstwC&pgis=1
41. Fu, Z., Chen, X., Vacic, V., Nan, P., Zhong, Y., Jiang, T.: MSOAR: a high-throughput ortholog assignment system based on genome rearrangement. J. Comput. Biol. **14**(9), 1160–1175 (2007). https://doi.org/10.1089/cmb.2007.0048. http://www.liebertonline.com/doi/abs/10.1089/cmb.2007.0048
42. Garvin, T., Aboukhalil, R., Kendall, J., Baslan, T., Atwal, G.S., Hicks, J., Wigler, M., Schatz, M.C.: Interactive analysis and assessment of single-cell copy-number variations. Nat. Methods **12**, 1058 (2015). https://doi.org/10.1038/nmeth.3578
43. Greenman, C.D., Pleasance, E.D., Newman, S., Yang, F., Fu, B., Nik-Zainal, S., Jones, D., Lau, K.W., Carter, N., Edwards, P.A.W., Futreal, P.A., Stratton, M.R., Campbell, P.J.: Estimation of rearrangement phylogeny for cancer genomes. Genome Res. **22**(2), 346–61 (2012). https://doi.org/10.1101/gr.118414.110. http://genome.cshlp.org/content/early/2011/10/12/gr.118414.110
44. Han, Y.: Improving the efficiency of sorting by reversals. In: Proceedings of the 2006 International Conference on Bioinformatics and Computational Biology, vol. 6, pp. 406–409. Citeseer (2006)
45. Hanahan, D., Weinberg, R.A.: Hallmarks of cancer: the next generation. Cell **144**(5), 646–74 (2011). https://doi.org/10.1016/j.cell.2011.02.013. http://www.ncbi.nlm.nih.gov/pubmed/21376230
46. Hannenhalli, S.: Polynomial-time algorithm for computing translocation distance between genomes. Discrete Appl. Math. **71**(1), 137 – 151 (1996). https://doi.org/10.1016/S0166-218X(96)00061-3. http://www.sciencedirect.com/science/article/pii/S0166218X96000613
47. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science, vol. 36, pp. 581–592 (1995). https://doi.org/10.1109/SFCS.1995.492588. http://doi.ieeecomputersociety.org/10.110910.1109/SFCS.1995.492588

48. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM (JACM) **46**(1), 1–27 (1999)
49. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM **46**(1), 1–27 (1999). https://doi.org/10.1145/300515.300516. http://portal.acm.org/citation.cfm?doid=300515.300516
50. Hupé, P.: Karyotype of the T47D breast cancer cell line. Wikimedia Commons file. https://commons.wikimedia.org/wiki/File:Karyotype_of_the_T47D_breast_cancer_cell_line.svg
51. Jahn, K., Kuipers, J., Beerenwinkel, N.: Tree inference for single-cell data. Genome Biol. **17**(1), 86 (2016). https://doi.org/10.1186/s13059-016-0936-x. http://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0936-x
52. Jain, M., et al.: Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat. Biotechnol. **36**(4), 338–345 (2018). https://doi.org/10.1038/nbt.4060. http://www.nature.com/doifinder/10.1038/nbt.4060
53. Jean, G., Nikolski, M.: Genome rearrangements: a correct algorithm for optimal capping. Inf. Process. Lett. **104**(1), 14–20 (2007). https://doi.org/10.1016/j.ipl.2007.04.011. http://www.sciencedirect.com/science/article/pii/S0020019007001147
54. Kahn, C.L., Hristov, B.H., Raphael, B.J.: Parsimony and likelihood reconstruction of human segmental duplications. Bioinformatics **26**(18), i446–52 (2010). https://doi.org/10.1093/bioinformatics/btq368. http://bioinformatics.oxfordjournals.org/content/26/18/i446.short
55. Kahn, C.L., Mozes, S., Raphael, B.J.: Efficient algorithms for analyzing segmental duplications with deletions and inversions in genomes. Algorithm Mol. Biol **5**(1), 11 (2010). https://doi.org/10.1186/1748-7188-5-11. http://www.almob.org/content/5/1/11
56. Kahn, C.L., Raphael, B.J.: Analysis of segmental duplications via duplication distance. Bioinformatics **24**(16), i133–8 (2008). https://doi.org/10.1093/bioinformatics/btn292. http://bioinformatics.oxfordjournals.org/content/24/16/i133.short
57. Kaplan, H., Shamir, R., Tarjan, R.E.: Faster and simpler algorithm for sorting signed permutations by reversals. SIAM J. Comput. **29**(3), 880 (1997). https://doi.org/10.1137/S0097539798334207. http://link.aip.org/link/SMJCAT/v29/i3/p880/s1&Agg=doi
58. Kececioglu, J., Sankoff, D.: Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. Algorithmica **13**(1–2), 180–210 (1995). https://doi.org/10.1007/BF01188586. http://link.springer.com/10.1007/BF01188586
59. Kececioglu, J.D., Ravi, R.: Of mice and men: algorithms for evolutionary distances between genomes with translocation. In: Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 604–613. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1995). http://dl.acm.org/citation.cfm?id=313651.313825
60. Kováč, J.: On the complexity of rearrangement problems under the breakpoint distance. J. Comput. Biol. **21**(1), 1–15 (2014). https://doi.org/10.1089/cmb.2013.0004. http://online.liebertpub.com/doi/full/10.1089/cmb.2013.0004
61. Kuipers, J., Jahn, K., Raphael, B.J., Beerenwinkel, N.: Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. Genome Res. (2017). https://doi.org/10.1101/gr.220707.117. http://www.ncbi.nlm.nih.gov/pubmed/29030470
62. Li, Y., Zhou, S., Schwartz, D.C., Ma, J.: Allele-specific quantification of structural variations in cancer genomes. Cell Syst. **3**(1), 21 – 34 (2016). https://doi.org/10.1016/j.cels.2016.05.007. http://www.sciencedirect.com/science/article/pii/S240547121630182X
63. Lodish, H., Berk, A., Darnell, J.E., Kaiser, C.A., Krieger, M., Scott, M.P., Bretscher, A., Ploegh, H., Matsudaira, P., et al.: Molecular Cell Biology. Macmillan (2008)
64. Martinez, F.V., Feijão, P., Braga, M.D., Stoye, J.: On the family-free DCJ distance and similarity. Algorithm Mol. Biol **10**(1), 13 (2015). https://doi.org/10.1186/s13015-015-0041-9. http://www.almob.org/content/10/1/13
65. Mitelman, F., Johansson, B., Mertens, F.: Mitelman database of chromosome aberrations and gene fusions in cancer (2018). http://cgap.nci.nih.gov/Chromosomes/Mitelman
66. Mixtacki, J.: Genome halving under DCJ revisited. In: Hu, X., Wang, J. (eds.) Proceedings of the Computational and Combinatorics. Lecture Notes in Computer Science, vol. 5092, pp.

276–286. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69733-6. http://dl.acm.org/citation.cfm?id=1426120.1426155
67. Mohri, M.: Edit-distance of weighted automata: general definitions and algorithms. Int. J. Found. Comput. Sci. **14**(06), 957–982 (2003)
68. Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. USA **81**(3), 814–818 (1984). https://doi.org/10.1073/pnas.81.3.814. http://www.pnas.org/content/81/3/814
69. Nguyen, C.T., Tay, Y.C., Zhang, L.: Divide-and-conquer approach for the exemplar breakpoint distance. Bioinformatics **21**(10), 2171–2176 (2005). https://doi.org/10.1093/bioinformatics/bti327. http://www.ncbi.nlm.nih.gov/pubmed/15713729, https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti327
70. Oesper, L., Dantas, S., Raphael, B.J.: Identifying simultaneous rearrangements in cancer genomes. Bioinformatics **34**(2), 346–352 (2018). https://doi.org/10.1093/bioinformatics/btx745. http://academic.oup.com/bioinformatics/article/34/2/346/4665417
71. Oesper, L., Ritz, A., Aerni, S.J., Drebin, R., Raphael, B.J.: Reconstructing cancer genomes from paired-end sequencing data. BMC Bioinform. **13**(Suppl 6), S10 (2012). https://doi.org/10.1186/1471-2105-13-S6-S10. http://www.biomedcentral.com/1471-2105/13/S6/S10
72. Ozery-Flato, M., Shamir, R.: Two notes on genome rearrangement. J. Bioinform. Comput. Biol. **1**(1), 71–94 (2003). http://www.ncbi.nlm.nih.gov/pubmed/15290782
73. Ozery-Flato, M., Shamir, R.: Sorting by translocations via reversals theory. In: Bourque, G., El-Mabrouk, N. (eds.) Proceedings of the RECOMB-CG, pp. 87–98. Springer, Berlin, Heidelberg (2006)
74. Ozery-Flato, M., Shamir, R.: Sorting cancer karyotypes by elementary operations. J. Comput. Biol. **16**(10), 1445–60 (2009). https://doi.org/10.1089/cmb.2009.0083. http://online.liebertpub.com/doi/abs/10.1089/cmb.2009.0083
75. Palmer, J.D., Herbon, L.A.: Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. J. Mol. Evol. **28**(1–2), 87–97 (1988). http://www.ncbi.nlm.nih.gov/pubmed/3148746
76. Paten, B., Zerbino, D.R., Hickey, G., Haussler, D.: A unifying model of genome evolution under parsimony. BMC Bioinform. **15**(1), 206 (2014)
77. Pevzner, P., Tesler, G.: Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. Genome Res. **13**(1), 37–45 (2003). https://doi.org/10.1101/gr.757503. http://www.ncbi.nlm.nih.gov/pubmed/12529304, http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC430962
78. Pevzner, P., Tesler, G.: Transforming men into mice. In: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology, pp. 247–256. ACM Press, New York, NY, USA (2003). https://doi.org/10.1145/640075.640108. http://portal.acm.org/citation.cfm?doid=640075.640108
79. Pinkel, D., Straume, T., Gray, J.W.: Cytogenetic analysis using quantitative, high-sensitivity, fluorescence hybridization. Proc. Natl. Acad. Sci. USA **83**(9), 2934–2938 (1986). https://doi.org/10.1073/pnas.83.9.2934. http://www.pnas.org/content/83/9/2934.short
80. Popescu, P., Hayes, H.: Techniques in Animal Cytogenetics. Springer Science & Business Media (2000)
81. Rajaraman, A., Ma, J.: Toward recovering Allele-specific cancer genome graphs. J. Comput. Biol. **25**(7), 624–636 (2018). https://doi.org/10.1089/cmb.2018.0022. http://www.liebertpub.com/doi/10.1089/cmb.2018.0022
82. Rhoads, A., Au, K.F.: PacBio sequencing and its applications. Genomics Proteomics Bioinform. **13**(5), 278–289 (2015). https://doi.org/10.1016/J.GPB.2015.08.002
83. Rubert, D.P., Feijão, P., Braga, M.D.V., Stoye, J., Martinez, F.H.V.: Approximating the DCJ distance of balanced genomes in linear time. Algorithm Mol. Biol **12**(1), 3 (2017). https://doi.org/10.1186/s13015-017-0095-y. http://almob.biomedcentral.com/articles/10.1186/s13015-017-0095-y

84. Rubert, D.P., Hoshino, E.A., Braga, M.D.V., Stoye, J., Martinez, F.V.: Computing the family-free DCJ similarity. BMC Bioinform. **19**(S6), 152 (2018). https://doi.org/10.1186/s12859-018-2130-5. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2130-5
85. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) Proceedings of the Combinatorial Pattern Matching, pp. 121–135. Springer, Berlin, Heidelberg (1992)
86. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics **15**(11), 909–917 (1999). https://doi.org/10.1093/bioinformatics/15.11.909. http://bioinformatics.oxfordjournals.org/content/15/11/909.short, http://bioinformatics.oxfordjournals.org/content/15/11/909
87. Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B.F., Cedergren, R.: Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. Proc. Natl. Acad. Sci. USA **89**(14), 6575–6579 (1992)
88. Scherthan, H., Cremer, T., Arnason, U., Weier, H.U., Lima-de Faria, A., Frönicke, L.: Comparative chromosome painting discloses homologous segments in distantly related mammals. Nat. Genet. **6**(4), 342 (1994)
89. Schwarz, R.F., Ng, C.K.Y., Cooke, S.L., Newman, S., Temple, J., Piskorz, A.M., Gale, D., Sayal, K., Murtaza, M., Baldwin, P.J., Rosenfeld, N., Earl, H.M., Sala, E., Jimenez-Linan, M., Parkinson, C.A., Markowetz, F., Brenton, J.D.: Spatial and temporal heterogeneity in high-grade serous ovarian cancer: a phylogenetic analysis. PLoS Med. **12**(2), e1001789 (2015). https://doi.org/10.1371/journal.pmed.1001789. http://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1001789
90. Schwarz, R.F., Trinh, A., Sipos, B., Brenton, J.D., Goldman, N., Markowetz, F.: Phylogenetic quantification of intra-tumour heterogeneity. PLoS Comput. Biol. **10**(4), e1003535 (2014). https://doi.org/10.1371/journal.pcbi.1003535. http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003535
91. Shao, M., Lin, Y.: Approximating the edit distance for genomes with duplicate genes under DCJ, insertion and deletion. BMC Bioinform. **13**(Suppl 19), S13 (2012). https://doi.org/10.1186/1471-2105-13-S19-S13. http://www.biomedcentral.com/1471-2105/13/S19/S13
92. Shao, M., Lin, Y., Moret, B.M.: An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. J. Comput. Biol. **22**(5), 425–435 (2015). https://doi.org/10.1089/cmb.2014.0096. http://online.liebertpub.com/doi/10.1089/cmb.2014.0096
93. Shao, M., Moret, B.M.: A fast and exact algorithm for the exemplar breakpoint distance. J. Comput. Biol. **23**(5), 337–346 (2016). https://doi.org/10.1089/cmb.2015.0193. http://online.liebertpub.com/doi/10.1089/cmb.2015.0193
94. Shao, M., Moret, B.M.: On computing breakpoint distances for genomes with duplicate genes. J. Comput. Biol. **24**(6), 571–580 (2017). https://doi.org/10.1089/cmb.2016.0149. http://online.liebertpub.com/doi/10.1089/cmb.2016.0149
95. Shao, M., Moret, B.M.E.: Comparing genomes with rearrangements and segmental duplications. Bioinformatics **31**(12), i329–i338 (2015). https://doi.org/10.1093/bioinformatics/btv229. http://bioinformatics.oxfordjournals.org/content/31/12/i329.short
96. Shi, G., Zhang, L., Jiang, T.: MSOAR 2.0: incorporating tandem duplications into ortholog assignment based on genome rearrangement. BMC Bioinform. **11**(1), 10 (2010). https://doi.org/10.1186/1471-2105-11-10. http://www.biomedcentral.com/1471-2105/11/10
97. da Silva, P.H., Machado, R., Dantas, S., Braga, M.D.V.: Restricted DCJ-indel model: sorting linear genomes with DCJ and indels. BMC Bioinform. **13**(Suppl 19), S14 (2012). https://doi.org/10.1186/1471-2105-13-S19-S14. http://www.biomedcentral.com/1471-2105/13/S19/S14
98. da Silva, P.H., Machado, R., Dantas, S., Braga, M.D.V.: DCJ-indel and DCJ-substitution distances with distinct operation costs. Algorithm Mol. Biol **8**(1), 21 (2013). https://doi.org/10.1186/1748-7188-8-21. http://www.almob.org/content/8/1/21
99. Strauss, S.H., Palmer, J.D., Howe, G.T., Doerksen, A.H.: Chloroplast genomes of two conifers lack a large inverted repeat and are extensively rearranged. Proc. Natl. Acad. Sci. USA **85**(11), 3898–3902 (1988)

100. Sturtevant, A.H., Dobzhansky, T.: Inversions in the third chromosome of wild races of drosophila pseudoobscura, and their use in the study of the history of the species. Proc. Natl. Acad. Sci. USA **22**(7), 448–450 (1936)
101. Suksawatchon, J., Lursinsap, C., Bodén, M.: Computing the reversal distance between genomes in the presence of multi-gene families via binary integer programming. J. Bioinform. Comput. Biol. **5**(1), 117–33 (2007). http://www.ncbi.nlm.nih.gov/pubmed/17477494
102. Tannier, E., Bergeron, A., Sagot, M.F.: Advances on sorting by reversals. Discrete Appl. Math. **155**(6–7), 881–888 (2007). https://doi.org/10.1016/J.DAM.2005.02.033. https://www.sciencedirect.com/science/article/pii/S0166218X06003751
103. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinform. **10**(1), 120 (2009). https://doi.org/10.1186/1471-2105-10-120. http://www.biomedcentral.com/1471-2105/10/120
104. Tattini, L., D'Aurizio, R., Magi, A.: Detection of genomic structural variants from next-generation sequencing data. Frontiers Bioeng. Biotechnol. **3**, 92 (2015). https://doi.org/10.3389/fbioe.2015.00092. http://www.ncbi.nlm.nih.gov/pubmed/26161383, http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4479793
105. Tesler, G.: Efficient algorithms for multichromosomal genome rearrangements. J. Comput. Syst. Sci. **65**(3), 587–609 (2002). https://doi.org/10.1016/S0022-0000(02)00011-9
106. Tesler, G.: GRIMM: genome rearrangements web server. Bioinformatics **18**(3), 492–493 (2002). https://doi.org/10.1093/bioinformatics/18.3.492. http://bioinformatics.oxfordjournals.org/content/18/3/492.abstract
107. Urban, A.E., Korbel, J.O., Selzer, R., Richmond, T., Hacker, A., Popescu, G.V., Cubells, J.F., Green, R., Emanuel, B.S., Gerstein, M.B., Weissman, S.M., Snyder, M.: High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays. Proc. Natl. Acad. Sci. USA **103**(12), 4534–9 (2006). https://doi.org/10.1073/pnas.0511340103. http://www.pnas.org/content/103/12/4534.full
108. Voet, T., Kumar, P., Van Loo, P., Cooke, S.L., Marshall, J., Lin, M.L., Zamani Esteki, M., Van der Aa, N., Mateiu, L., McBride, D.J., Bignell, G.R., McLaren, S., Teague, J., Butler, A., Raine, K., Stebbings, L.A., Quail, M.A., DHooghe, T., Moreau, Y., Futreal, P.A., Stratton, M.R., Vermeesch, J.R., Campbell, P.J.: Single-cell paired-end genome sequencing reveals structural variation per cell cycle. Nucleic Acids Res. **41**(12), 6119–6138 (2013). https://doi.org/10.1093/nar/gkt345
109. Vogelstein, B., Papadopoulos, N., Velculescu, V.E., Zhou, S., Diaz, L.A., Kinzler, K.W.: Cancer genome landscapes. Science **339**(6127), 1546–58 (2013). https://doi.org/10.1126/science.1235122. http://www.ncbi.nlm.nih.gov/pubmed/23539594, http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3749880
110. Wang, Y., Waters, J., Leung, M.L., Unruh, A., Roh, W., Shi, X., Chen, K., Scheet, P., Vattathil, S., Liang, H., Multani, A., Zhang, H., Zhao, R., Michor, F., Meric-Bernstam, F., Navin, N.E.: Clonal evolution in breast cancer revealed by single nucleus genome sequencing. Nature **512**, 155 (2014). https://doi.org/10.1038/nature13600
111. Warren, R., Sankoff, D.: Genome halving with double cut and join. J. Comput. Biol. **7**(2), 357–371 (2009)
112. Warren, R., Sankoff, D.: Genome aliquoting revisited. J. Comput. Biol. **18**(9), 1065–1075 (2011). http://online.liebertpub.com/doi/abs/10.1089/cmb.2011.0087
113. Willing, E., Zaccaria, S., Braga, M.D., Stoye, J.: On the inversion-indel distance. BMC Bioinform. **14**(Suppl 15), S3 (2013)
114. Wolfe, K.H., Shields, D.C.: Molecular evidence for an ancient duplication of the entire yeast genome. Nature **387**, 708 (1997). https://doi.org/10.1038/42711
115. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005). https://doi.org/10.1093/bioinformatics/bti535
116. Yancopoulos, S., Friedberg, R.: DCJ path formulation for genome transformations which include insertions, deletions, and duplications. J. Comput. Biol. **16**(10), 1311–38 (2009). https://doi.org/10.1089/cmb.2009.0092. http://www.ncbi.nlm.nih.gov/pubmed/19803734

117. Yin, Z., Tang, J., Schaeffer, S.W., Bader, D.A.: Exemplar or matching: modeling DCJ problems with unequal content genome data. J. Comb. Optim. (2015). https://doi.org/10.1007/s10878-015-9940-4. http://link.springer.com/10.1007/s10878-015-9940-4
118. Zaccaria, S., El-Kebir, M., Klau, G.W., Raphael, B.J.: Phylogenetic copy-number factorization of multiple tumor samples. J. Comput. Biol. (2018). https://doi.org/10.1089/cmb.2017.0253. http://www.liebertpub.com/doi/10.1089/cmb.2017.0253
119. Zakov, S., Bafna, V.: Reconstructing breakage fusion bridge architectures using noisy copy numbers. J. Comput. Biol. **22**(6), 577–594 (2015). https://doi.org/10.1089/cmb.2014.0166. http://online.liebertpub.com/doi/10.1089/cmb.2014.0166
120. Zakov, S., Kinsella, M., Bafna, V.: An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. Proc. Natl. Acad. Sci. USA **110**(14), 5546–51 (2013). https://doi.org/10.1073/pnas.1220977110. http://www.pnas.org/content/110/14/5546.full
121. Zeira, R., Shamir, R.: Sorting by cuts, joins, and whole chromosome duplications. J. Comput. Biol. **24**(2), 127–137 (2017). https://doi.org/10.1089/cmb.2016.0045. http://online.liebertpub.com/doi/10.1089/cmb.2016.0045, http://www.ncbi.nlm.nih.gov/pubmed/27704866
122. Zeira, R., Shamir, R.: Sorting cancer karyotypes using double-cut-and-joins, duplications and deletions. Bioinformatics (2018). https://doi.org/10.1093/bioinformatics/bty381. https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/bty381/4992148
123. Zeira, R., Zehavi, M., Shamir, R.: A linear-time algorithm for the copy number transformation problem. J. Comput. Biol. **24**(12), 1179–1194 (2017). https://doi.org/10.1089/cmb.2017.0060. http://online.liebertpub.com/doi/10.1089/cmb.2017.0060
124. Zerbino, D.R., Ballinger, T., Paten, B., Hickey, G., Haussler, D.: Representing and decomposing genomic structural variants as balanced integer flows on sequence graphs. BMC Bioinform. **17**(1), 400 (2016). https://doi.org/10.1186/s12859-016-1258-4. http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016-1258-4
125. Zheng, C., Zhu, Q., Sankoff, D.: Genome halving with an outgroup. Evol. Bioinform. Online **2**, 295–302 (2007). http://www.ncbi.nlm.nih.gov/pubmed/19455223, http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2674651
126. Zimao, L., Lusheng, W., Kaizhong, Z.: Algorithmic approaches for genome rearrangement: a review. IEEE Trans. Sys. Man Cybern., Part C (Applications and Reviews) **36**(5), 636–648 (2006). https://doi.org/10.1109/TSMCC.2005.855522. http://ieeexplore.ieee.org/document/1678038/

Chapter 11
Computational Models for Cancer Phylogenetics

Russell Schwartz

Abstract Cancer development has long been recognized as a product of aberrant evolution of cell populations, inspiring the idea that phylogenetic algorithms could be a powerful tool for reconstructing progression processes in cancer. Translating that intuition into practice, however, has required extensive work on adapting phylogenetic models and algorithms to represent more accurately the many ways tumor evolution is distinct from classic species evolution. The result has been a large and growing body of problems and theory on phylogenetics as it applies specifically to cancers. This chapter surveys some of the key ideas and variants on phylogeny problem that have arisen in the development of tumor phylogeny methods. Its purpose is to introduce readers to some of the space of approaches in current practice in phylogenetics and help them appreciate how models have developed, and continue to develop, to better capture the peculiar nature of evolution in cancers.

Keywords Cancer progression · Tumor · Phylogenetics · Algorithms

11.1 Introduction

Tumor phylogenetics—the use of phylogenetic algorithms to reconstruct the process of progression in cancers—has emerged as a key tool for interpreting cancer genomic data and uncovering the mechanisms by which cancer develops, progresses, and responds to treatment. Experimental cancer research has been fueled by exciting advances in technologies for gathering data on the genetics and genomics of cancers at ever-greater scales and ever-finer resolutions. At the same time, the field has struggled with the resulting data deluge: dealing with vast volumes of data for an immensely complicated process that exhibits enormous heterogeneity patient to patient and even cell to cell in single patients. Methods from computational biology have become essential to modern cancer research in managing these problems, with phylogenetic methods in particular providing a crucial set of tools for extracting

R. Schwartz (✉)
Carnegie Mellon University, Pittsburgh, USA
e-mail: russells@andrew.cmu.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_11

meaningful information from cancer genomic data. As a result, the tumor phylogeny problem has received a great deal of attention in the computational biology community, where it has raised a host of interesting challenges for modeling, algorithm development, and practical software implementation that must be solved to provide the tools needed for tackling one of the hardest challenges in medicine.

Tumor phylogenetics also exemplifies some of the key themes of Bernard Moret's work in helping to guide the development of the modern fields of computational and evolutionary biology. Dr. Moret's work has long shown the importance of fundamental theory that is sometimes needed to solve hard computational problems arising in biology (e.g., [4, 56, 73]). Tumor phylogenetics is indeed an area that has inspired hard problems whose solution has required deep theoretical advances. At the same time, Dr. Moret's work has been guided by the notion that algorithms need to work in the real world (e.g., [48–50, 78]). Much of the innovation in tumor phylogenetics has been centered on questions not so much of solving well-posed computational problems but rather of finding the right models that will solve the real-world problems of the field given the complexity of the system and the challenges of the data used to study it. Finally, Dr. Moret has been a major innovator in finding novel uses of phylogenetics beyond classic organismal evolution [16, 51, 80]. Cancer phylogenetics has proven a spectacular example of this theme, where an exotic but extremely important form of evolutionary system has required new computational thinking and where ideas from phylogenetics have proven crucial to solving those real-world problems in practice.

Tumor phylogenetics is not just important science but also a valuable topic for pedagogy for a diverse community of scientists. For the cancer biologist, and experimental biologists more generally, the history of tumor phylogenetics is a great case study for the importance of computational thinking in solving hard problems in biology. It is a field where getting the right answers depends on thinking carefully about one's models and about the tradeoffs in modeling between what reflects the biology faithfully and what is feasible computationally. For the evolutionary biologist, tumor evolution is an exciting area because it is a rare case in which one can observe evolution repeatedly navigating the same selective landscape, giving a unique window into the interplay of distinct solutions to the same evolutionary problem in a background of extensive stochastic variation. For the computer scientist, it is similarly an instructive field for the challenges of working in biomedical domains, where good computer science is needed but the biological details are important and need to inform one's models and algorithms. For the lay person, tumor phylogenetics is a wonderful illustration of how evolution impacts our day-to-day life and why it matters today that scientists understand and continue to study it.

One chapter is not going to serve all of those audiences well, and we aim here primarily to serve the more computationally sophisticated readers, either computer scientists or computational biologists new to this area, with a focus on computational models and problems that have emerged, or might emerge, as important areas of study. We will continue this introduction by presenting some of the motivation and biological background for the field of tumor phylogenetics. The bulk of the chapter is

devoted to exploring a few major themes of computational problems in the domain. We then conclude with some discussion and consideration of future directions.

11.1.1 Background

Understanding the computational basis of tumor phylogenetics requires an appreciation for the biology it models and the tools we have for studying that biology. This chapter cannot provide a comprehensive summary of the biology and biotechnology of tumor evolution, but it does aim to summarize some of the key facts and ideas needed to understand the current state of the field. Readers interested in seriously pursuing the area would likely benefit from reading further treatments of tumor evolution aimed at biologically sophisticated readers (e.g., [33, 45, 64]) as well as more comprehensive resources on cancer in general (e.g., [76]).

The key idea behind tumor phylogenetics is the observation that cancer is an evolutionary system. This idea long predates phylogenetic approaches to reconstructing tumor evolution, generally being traced to [54], and inspired the clonal theory of cancer evolution: that tumors develop and progress through a series of random mutations that periodically produce a fitter, more aggressive cell population (clone) that undergoes an evolutionary sweep to dominate the tumor. A variety of more nuanced variants of this idea have since arisen [26]. Today, we can appreciate that while these early models beautifully articulated crucial ideas in understanding tumor progression, they also greatly oversimplify the true complexity of tumor evolution. Nonetheless, they inspired the idea that we could understand cancer by understanding the evolution of tumor cell populations. It was first suggested by [75] that if cancer development is essentially an evolutionary process then we ought to be able to learn about it by reconstructing that process computationally, an idea put into practice by Desper et al. [19] in the first variants of tumor phylogenetics.

The field of tumor phylogenetics has since seen enormous attention, resulting in a diversity of methods. We can roughly organize the space of tumor phylogeny methods by a few basic axes of variation [64]:

- Study type: cross-sectional, regional, single-cell, deconvolutional, hybrid
- Phylogeny type: distance-based, character-based combinatorial, character-based probabilistic (maximum likelihood or Bayesian), other
- Marker type: single nucleotide variant (SNV), copy number variant (CNV), structural variant (SV), other (expression, methylation, or more exotic alternatives), hybrid.

We illustrate this organization of methods in Fig. 11.1, highlighting examples of work in different parts of this space. While there are many methods that do not fit neatly into these categories, they nonetheless provide a broad classification of methodology that we will use to help categorize methods and organize this chapter.

The first axis describes the kind of data-generation study used to gather data from which one will draw a phylogeny. Tumor phylogenetics began with cross-sectional

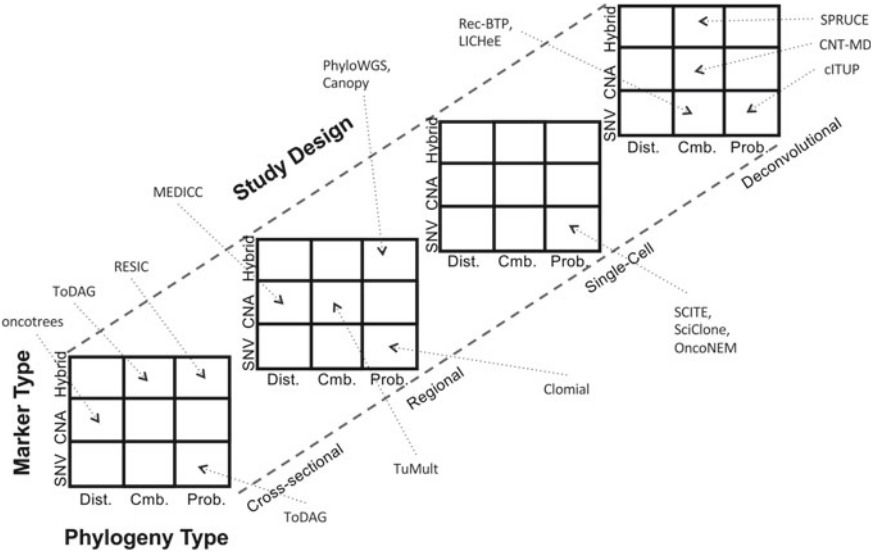


Fig. 11.1 A categorization of tumor phylogeny methods. The figures shows axes of variation corresponding to some of the major distinguishing features of tumor phylogeny methods with citations to a few notable examples of different categories of method. The figure is not comprehensive with respect to categories of method on each axis or methods specifically cited, and is only meant to illustrate some of the range of methods available

studies [19], which modeled tumor evolution by treating distinct tumors as species and an evolutionary tree as a description of decision points separating subsets of tumors. As it became possible to examine genomic data at finer resolutions, this was displaced by a combination of regional studies [31, 53], in which phylogenies describe evolution across distinct tumor sites or sub-regions, and single-cell studies [52], which modeled evolution within populations of single cells often within a single-tumor site. Much work in this domain now falls into a special class of "deconvolutional" study [7, 8, 65], which seek to infer single-cell variation from regional or cross-sectional data. A few methods are now beginning also to combine different data types, such as bulk and single-cell [42].

The second axis describes the kind of data used to profile genetic variations in tumors. While the earliest methods focused primarily on limited kinds of copy number aberrations (CNAs) that could be profiled with pre-genomic technologies [19, 57], much work later shifted towards more mathematically tractable single nucleotide variants (SNVs) (e.g., [35, 43, 59]). Nonetheless, work on CNAs specifically has continued (e.g., [14, 66]). State-of-the-art methods often use hybrid approaches taking advantage of both SNV and CNA data [18, 25, 38]. Limited work in tumor phylogenetics has involved other variant types, such as gene expression [21, 55, 61], methylation [68] or microsatellite variation [29, 67]. Recent work is now just begin-

ning to attempt to capture broader classes of structural variations (SVs) in tumor phylogenetics [22].

The third axis refers to classes of computational models. The earliest methods used specialized combinatorial models [19], which generally pose phylogenetics as a problem of optimizing for some discrete objective function. The difficulty of solving the resulting computational problems, particularly for large marker sets or numbers of taxa, prompted a move towards distance-based methods [20], which typically make use of classic efficient phylogeny algorithms but with novel cancer-specific ways of measuring evolutionary distance. Off-the-shelf distance-based tools remain a popular option particularly for those initially generating data, as they offer fast, generic algorithms that can be run with almost any kind of data. However, the advantages of generic algorithms can make them poor at modeling tumor evolution specifically [64]. Given the complexity of the biology and the data, much of the field has moved towards probabilistic tree inference methods [46, 47, 81], which are more computationally intensive but have considerable advantages over the alternatives in accommodating complex error models and objective functions and in explicitly modeling uncertainty in inference.

11.1.2 Scope and Organization

In this chapter, we focus on one specific subset of the classification of Sect. 11.1.1: well-posed combinatorial problems. Our reasoning for focusing on this space is pedagogical rather than scientific. There are important methods that fall into this class and important methods that do not. However, the space of combinatorial problems is useful for developing an understanding of the field because they tend to involve relatively simple and "clean" formulations in which the reasoning behind the model and its connection to the biology is explicit. Furthermore, these problems occupy an important area for modeling work—where the field is first thinking rigorously about how properly to model a problem and what the tradeoffs are between the fidelity of the model to biology, the computational tractability of the model, and in some cases the feasibility of learning model parameters from data—before theory gives way to often more detailed but less elegant probabilistic formulations.

We largely confine our discussion here to the modeling, i.e., formulating and reasoning about problem statements. Algorithms to solve for those problem statements are often quite involved or heuristic, and we provide brief descriptions and references to primary literature rather than dwell on the details of the algorithms. We note that in many cases the problems, as presented below, are simplifications of the computational problems as they are published and used in practice; the biology of cancer and the data used to study it are complicated and methods with real-world value almost always involve various forms of preprocessing, error-handling, or heuristic improvements beyond the idealized models. We also note that for consistency of exposition here, we will use our own naming scheme for problems, in many cases

renaming problems that are posed in other literature with different names, and changing nomenclature and variable names where needed for consistency.

For the most part, we will assume the reader is already familiar with general topics in phylogenetics that might be covered well in a general phylogeny text (e.g., [28]) or elsewhere in this volume. We must also necessarily omit a broad range of related topics in computational cancer research, such as variant discovery and typing (c.f., [44]), simulation of tumor evolution (e.g., [17]), or emerging topics such as study design for tumor phylogenetics. Finally, our focus specifically on computational problems means that we omit much interesting work on the biology informing or arising from these methods, novel studies applying them, or other issues that are important but beyond our scope. We refer the interested reader to any of a number of informative review articles that cover issues beyond those in this chapter on either the biology or the computation [9, 33, 45, 64].

Consistent with this scope, the remainder of this chapter will present a variety of problems in tumor phylogenetics largely from a computer scientist's point of view: examining the computational models and related algorithmic problems that have arisen from different conceptions of tumor phylogenetics and briefly discussing their motivations. We start with the fundamental problem of estimating evolutionary distance, which itself can be quite different for tumor cells than for organisms, and gradually work towards more complicated problem statements involved in inferring phylogenies and eventually mixtures of phylogenies. At each step, we will cover a few examples meant to show some of the diversity of problem statements that can arise from different study designs, data types, and/or model assumptions. While this coverage is far from exhaustive, we hope it is sufficient to give an understanding of some of the main challenges in bringing phylogenetics to cancer research and how those working in the field have reasoned through those challenges. We then close with a consideration of potential future directions for work in this space.

11.2 Estimating Evolutionary Distance

One of the most basic questions in phylogenetics is how to estimate an evolutionary distance between two pairs of taxa (samples), which we will dub the Pairwise Distance Problem (PDP). Estimates of pairwise distances directly form the input for distance-based phylogeny algorithms (see, for example, [28]). Much seminal work in tumor phylogenetics has used standard distance-based phylogeny algorithms with novel tumor-specific distance measures rather than fully customized approaches (e.g., [20, 52, 67]). Even where specialized phylogeny algorithms are needed, they generally require some way of measuring distances between taxa as a subroutine, making it perhaps the most fundamental question in phylogenetics. It is also one of the first places we begin the see how tumor phylogenetics differs from standard species phylogenetics, as tumors often evolve by very different mechanisms, and are profiled by very different mutation classes, than are in standard use in species phylogenetics.

Note that PDP is closely related to but distinct from the Mutational Ordering Problem (MOP) [3, 26], which seeks to find an order of mutations along an evolutionary trajectory as opposed to simply a weight of a trajectory. Methods for PDP and MOP often overlap, however, as solutions to PDP frequently involve solving a variant of MOP and solving MOP usually implicitly solves a variant of PDP. Rather than discuss these as separate problems, we will present these as examples of PDP but point out where a PDP formulation is also relevant to MOP.

11.2.1 Single Nucleotide Variants (SNVs)

Perhaps the simplest way to begin measuring evolutionary distance is via SNVs, i.e., point mutations. SNVs are arguably the primary way evolutionary distance is measured in more general species phylogenetics and especially in intraspecies phylogenetics, as SNVs accumulate rapidly on evolutionary timescales and can be represented by comparatively simple evolutionary models. Since a great deal of practical work in tumor phylogenetics has used off-the-shelf programs designed for species or intraspecies phylogenetics (e.g., [10, 52, 53, 79]), it is a useful starting point for thinking about modeling tumor phylogenies. We will begin with some computationally trivial formulations of PDP for which there is no defined objective function.

We will assume here that our input consists of two taxa, $V_1, V_2 \in \Sigma^m$ where Σ is an alphabet of allowed genetic characters and $m \in \mathcal{N}_+$ is the number of variant sites profiled. We will further denote $v_{ij} \in \Sigma$ to be character j of V_i. On the assumption that variants are relatively rare, Σ may simply be $\{0, 1\}$ with 0 standing for a presumed ancestral variant and 1 a novel somatic mutant variant. It is a simple generalization to allow for four nucleotides ($\Sigma = \{A, C, G, T\}$) or further generalization to allow for insertion/deletion (indel) variants ($\Sigma = \{A, C, G, T, -\}$).

For any such problem statement, the simplest beginning is the following:

SNV-PDP
Input: Taxa $V_1, V_2 \in \Sigma^m$
Output: A distance $d(V_1, V_2) = \sum_{i=1}^{m} I(v_{1i} \neq v_{2i})$, where I is the indicator function defined by

$$I(b) = \begin{cases} 1 : & b \\ 0 : & \neg b \end{cases}$$

More informally, **SNV-PDP** is equivalent to the Hamming distance, i.e., the number of base changes between the two input sequences. This measure has a simple interpretation as an evolutionary distance: it is the minimum number of mutations needed to transform one sequence into the other.

A straightforward generalization of SNV-PDP is a weighted variant, wSNV-PDP:

wSNV-PDP
Input: Taxa $V_1, V_2 \in \Sigma^m$, weight function $w : [1, m] \rightarrow \mathscr{R}$
Output: A distance $d(V_1, V_2) = \sum_{i=1}^{m} w_i I(v_{1i} \neq v_{2i})$

One can also easily generalize to different models for the weight function, e.g., capturing preferences of *mutation signatures* [1, 2] that model rate variation by nucleotide and possibly surrounding sequence context (e.g., AAA → AGA might have a different rate than CAC → CGC).

There are other variants on SNV distance from the noncancer literature that one might in principle consider for tumor phylogenetics. For example, the general phylogenetics literature offers a number of methods for phylogenetics from genotyped data, in which one can count variants at each site but cannot always distinguish their chromosomes of origin [34, 36, 69]. The same issue in principle applies to tumor genomics, and is in fact even more complicated due to the frequent gain and loss of genetic materials in cancers. Since we are unaware of cancer work in this space, however, we omit an explicit formulation.

11.2.2 Copy Number Aberrations (CNAs)

One of the important ways tumor phylogenetics differs from conventional phylogenetics is in that it tends predominantly to involve different mechanisms of mutation. One major form this takes is in the importance of copy number aberrations (CNAs) in tumor evolution [37]. (Note that CNAs are sometimes called copy number variations (CNVs) and the terms may be used interchangeably in the algorithmic literature, but CNAs have become the preferred nomenclature for somatic variants in cancer to distinguish them from germline variants in a population that are also called CNVs.) CNAs are the primary way tumors undergo functional adaptation [84], commonly through loss of tumor suppressor genes or amplification of oncogenes. This makes them useful as markers for progression in cancer development. While CNVs are also known to be important in more general evolutionary adaptation [41], they remain a relatively specialized measure of evolutionary diversification for noncancer applications but a crucial one for cancers. We will therefore consider here some problems arising from tumor evolution when profiled for CNA variations.

CNAs have had a complicated history with respect to tumor phylogenetics due to the changing technologies for profiling variants over the course of the field's development. The earliest work in tumor phylogenetics in fact relied on a specialized form of coarse-grained CNA profiling technology, comparative genomic hybridization (CGH), used to detect large regions of gain or loss in tumors [19]. Likewise, the earliest efforts at single-cell tumor phylogenetics were based on a pre-genomic technology, fluorescence in situ hybridization (FISH), for profiling localized copy number variations at the single-cell level [57, 58]. As tumor phylogenetics moved

into the sequencing age, some of the earliest efforts at whole-genome regional bulk sequencing also relied on copy number data from an array variant of CGH [53]. The first work in single-cell sequencing for cancers likewise relied on estimated copy numbers of small genomic regions as a way to deal with high sequencing error rates and uncertain degrees of amplification in early single-nucleus sequencing technology, which made it impossible to profile SNVs or other small variants reliably [52]. Follow-on work in all of these domains later shifted largely to SNVs (see, for example, [85] for a review of recent methods in phylogenetics from single-cell sequence data) although CNA or hybrid data sets are again increasingly being applied. We will consider here models of a few variants of CNA distance that have appeared in the recent literature.

We will now assume here that our input consists of two taxa, $V_1, V_2 \in \mathscr{N}_+^m$, where we interpret each taxon as a vector of integer copy numbers at discrete loci on a genome. In the simplest version, we can model the problem using generalizations of the computationally trivial distance measures used for SNV evolution. Here, though, we need to be a bit more precise about the cost of marker changes at a single site:

L1-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$
Output: A distance $d(V_1, V_2) = \sum_{i=1}^{m} |v_{1i} - v_{2i}|$

L1-CNA-PDP implicitly encodes a model of copy number evolution in which copy numbers evolve in a cell by a series of discrete events, in which each event increments or decrements the copy number by one at a given site. This model has been effectively assumed by some early works in this domain (e.g., [14, 57]).

One might alternatively measure distance at single site by an L2 (sum-of-squares) measure, yielding

L2-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$
Output: A distance $d(V_1, V_2) = \sqrt{\sum_{i=1}^{m} (v_{1i} - v_{2i})^2}$

L2 distance does not have as straightforward an evolutionary interpretation but may make more sense if we wish to assume we are dealing with noisy data (a fair assumption) or allowing for an evolutionary event to change the copy number by more than one at a given step. L2-CNA-PDP was effectively assumed also by seminal work in CNA tumor phylogenetics [52]. Either L1-CNA-PDP or L2-CNA-PDP could be easily generalized to weighted variants, as with the SNV-PDP problems, although we are unaware of any work doing so.

It is with CNAs that tumor phylogenetics starts to get "interesting" algorithmically, as we move towards model variants that more realistically capture the biological mechanisms of copy number mutation. One simple way this occurs, dating back to some of the first work in the field [58], is through the phenomenon of whole-genome duplication. Genome duplication, which is believed to occur in cancers when a cell passes partially through mitosis but fails to divide after doubling its chromosome

complement, is a rare event in evolution of animals but a common one in cancers with important implications for tumor prognosis [72]. To add genome doubling as a possible event in tumor evolution, we need a more complicated formulation of PDP as an optimization problem to account for the fact that we lack a simple closed form expression for whole-genome doubling-aware phylogenetics. Rather, we have to pose the problem as an *edit distance* problem, in which we seek to minimize the number of mutational edit operations we perform on one taxon to transform it into the other. We essentially need to solve a loose version of MOP, since we seek to identify a minimum-cost sequence of mutations explaining the transition between two cells. (It is a loose version, because there will typically be many equally parsimonious mutation sequences for any given pair of copy number profiles.) We might begin by posing the problem as follows, in which we account for both localized gene-scale gain and loss (G) and whole-genome duplication (W) evolutionary events:

GW-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$
Output: A sequence of mutations $\mu = (\mu_1, \ldots, \mu_k)$ s.t.
$\mu_i \in \{G_{1+}, G_{1-}, \ldots, G_{m+}, G_{m-}, W\}\ \forall i$ (see below) and s.t.
$\mu_k(\mu_{k-1}(\ldots \mu_1(V_1)\ldots)) = V_2$
Objective: $\min_\mu k$

Here, we model a mutation as an operation on a copy number vector, i.e., a function $M : N_0^m \rightarrow N_0^m$. Specifically, we define the following mutation operations:

$$G_{i+}(v_1, \ldots, v_i, \ldots, v_m) = (v_1, \ldots, v_i + 1, \ldots, v_m)$$

$$G_{i-}(v_1, \ldots, v_i, \ldots, v_m) = (v_1, \ldots, v_i - 1, \ldots, v_m) \qquad (v_i > 0)$$

$$W(v_1, \ldots, v_i, \ldots, v_m) = (2v_1, \ldots, 2v_i, \ldots, 2v_m)$$

Note that properly defined, these are not necessarily symmetric operations, which may be problematic for some common uses of distance measures. For example, a cancer genome can drop from one to zero copies of a locus by losing its only copy, but it cannot move from zero to one copy by any mechanism known to us. Likewise, a genome can double its entire complement in a single cell division but to our knowledge there is no mechanism acting in cancer that halves the entire genome complement in a single step. What we are solving here is in fact μ, a solution to the MOP, although $|\mu|$, i.e., k, is the actual distance measure. We might be interested in the MOP itself, although one should be careful about over-interpreting it. Even assuming the model is correct and that the true mutation order optimizes for the objective function (both questionable assertions) the MOP will generally be degenerate as one can arbitrarily rearrange G events between W events. The resulting GW-CNA-PDP model is a simplification of one of the earliest single-cell tumor phylogeny models [58], which introduced the W events although with a somewhat more complex set of G events that we omit in discussion here.

Algorithmically, the GW-CNA-PDP problem has not been proven intractable nor has an efficient algorithm for it been found. In practice, it is solved with a method that is fixed-parameter tractable (FPT) in the number of whole-genome duplications, meaning it is efficient for any fixed number n of W events but has exponential runtime in n. Since n is generally small (on the order of 0–3 for a typical tumor), this is an acceptable solution in practice.

A next step in that line of work was to allow for a broader class of mutations, incorporating chromosome C events that change copy numbers of all genes on a given chromosome. Assume that we can partition our phylogenetic markers into chromosomes $c_1 = (1, \ldots, r_1), c_2 = (r_1 + 1, \ldots, r_1 + r_2) \ldots, c_r = (\sum_{i=1}^{r-1} r_i + 1, \ldots, \sum_{i=1}^{r} r_i)$. Then we can add chromosome events as follows:

$$R_{i+}(v_1, \ldots, v_{\sum_{j=1}^{i-1} r_j+1}, \ldots, v_{\sum_{j=1}^{i} r_j}, \ldots, v_m) =$$

$$(v_1, \ldots, v_{\sum_{j=1}^{i-1} r_j+1} + 1, \ldots, v_{\sum_{j=1}^{i} r_j} + 1, \ldots, v_m)$$

$$R_{i-}(v_1, \ldots, v_{\sum_{j=1}^{i-1} r_j+1}, \ldots, v_{\sum_{j=1}^{i} r_j}, \ldots, v_m) =$$

$$(v_1, \ldots, v_{\sum_{j=1}^{i-1} r_j+1} - 1, \ldots, v_{\sum_{j=1}^{i} r_j} - 1, \ldots, v_m)$$

We can then pose the following problem seeking a minimum-length mutation ordering accommodating G, C, and W events [15]:

GCW-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$
Output: A sequence of mutations $\mu = (\mu_1, \ldots, \mu_k)$ s.t.
$\mu_i \in \{G_{1+}, G_{1-}, \ldots, G_{m+}, G_{m-}, C_{1+}, C_{1-}, \ldots, C_{r+}, C_{r-}, W\}\ \forall i$ and s.t.
$\mu_k(\mu_{k-1}(\ldots \mu_1(V_1)\ldots)) = V_2$
Objective: $\min_\mu k$

The above model has a straightforward generalization to a weighted model, developed in [13] with the goal of inferring mutation-specific rates on a single-tumor basis

wGCW-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$, weight function
$w : \{G_{1+}, G_{1-}, \ldots, G_{m+}, G_{m-}, C_{1+}, C_{1-}, \ldots, C_{r+}, C_{r-}, W\} \to \mathscr{R}$
Output: A sequence of mutations $\mu = (\mu_1, \ldots, \mu_k)$ s.t.
$\mu_i \in \{G_{1+}, G_{1-}, \ldots, G_{m+}, G_{m-}, C_{1+}, C_{1-}, \ldots, C_{r+}, C_{r-}, W\}\ \forall i$ and s.t.
$\mu_k(\mu_{k-1}(\ldots \mu_1(V_1)\ldots)) = V_2$
Objective: $\min_\mu \sum_{i=1}^{k} w(\mu_i)$

GCW-CNA-PDP and wGCW-CNA-PDP are also not known to be hard but lack efficient algorithms. Both are also solved in practice either by heuristic methods or by algorithms FPT in the number n of whole-genome duplications needed. These algorithms, too, are effectively solving the MOP problem, but again with the caveat that optimal solutions may be highly degenerate, as both C and G events may be reordered between W events with the exception of events that drop copy numbers to zero.

An alternative that has seen considerable attention in the literature is known as the MEDICC model [66] based on the tool that first used a variant of it. The MEDICC model assumes that we can evolve a genome by gaining or losing one copy of any arbitrary contiguous segment of a chromosome. We can express the MEDICC model also as a form of edit distance model by establishing a pair of segmental gain/loss edit operations. Given vertex $v = (v_1, \ldots, v_n)$, we define two operations:

$$S_{+,a,b}(v) = (v_1, \ldots, v_{a-1}, v_a + 1, \ldots, v_b + 1, v_{b+1}, \ldots, v_n)$$

$$S_{-,a,b}(v) = (v_1, \ldots, v_{a-1}, v_a - 1, \ldots, v_b - 1, v_{b+1}, \ldots, v_n)$$

where $S_{+,a,b}$ and $S_{-,a,b}$ are undefined if $v_i = 0$ for any $i \in [a, b]$. Note that these are not symmetric operations, since we can subtract from one copy to get zero but not vice versa. We can then formulate the MEDICC CNA distance problem as follows:

MEDICC-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$
Output: A sequence of mutations $\mu = (\mu_1, \ldots, \mu_k)$ s.t. $\mu_i \in \{S_{+,i,j}, S_{-,a,b}\}\ \forall i$ and s.t. $[a, b] \subseteq [1, n]$ and $\mu_k(\mu_{k-1}(\ldots \mu_1(V_1)\ldots)) = V_2$
Objective: $\min_\mu \sum_{i=1}^{k} w(\mu_i)$

The MEDICC CNA distance, like the operations producing it, is not symmetric. That is, the minimum distance from V_1 to V_2 may be different from the minimum distance from V_2 to V_1.

The MEDICC distance has been shown to be solvable by a fast (linear time) algorithm [87]. We could also pose a weighted MEDICC distance model in a variety of ways. A natural formulation might be the following, which would follow from the assumption that the cost of a MEDICC operation might itself be a function of the length of the region gained or lost:

wMEDICC-CNA-PDP
Input: Taxa $V_1, V_2 \in \{N_0\}^m$, weight function $w : \mathscr{N}_+ \to \mathscr{R}$
Output: A sequence of mutations $\mu = (\mu_1, \ldots, \mu_k)$ s.t. $\mu_i \in \{S_{+,i,j}, S_{-,a,b}\}\ \forall i$ and s.t. $[a, b] \subseteq [1, n]$ and $\mu_k(\mu_{k-1}(\ldots \mu_1(V_1)\ldots)) = V_2$
Objective: $\min_\mu \sum_{i=1}^{k} w(b - a + 1)$

To our knowledge, no weighted MEDICC variant has been proposed in the literature yet and there is no formal theory generalizing the MEDICC algorithms to it, although it would appear to be a straightforward extension.

There are other event types one might incorporate into CNA edit distance measures that have not yet made their way into CNA phylogeny models to our knowledge. For example, there is a body of theory on reconstructing a particular kind of structural variation common in tumors, called a breakage-fusion-bridge (BFB) event, that leads to a characteristic pattern of copy number changes [86] that one might in principle bring into the kinds of phylogeny methods considered here. We are still discovering mechanisms of structural variation in cancers and it is likely other mechanisms of copy number variation are yet to be discovered.

Figure 11.2 illustrates a few of the methods covered above. For convenience in subsequent description, we will create a shorthand to refer to the distance measures computed for these various models. We will define d_{L1} and d_{L2} to be the CNA L1 and L2 distances. We will define $d_{GW}(u, v)$, $d_{GCW}(u, v)$, $d_{wGW}(u, v)$ and $d_{wGCW}(u, v)$ as the minimum edit distances between copy number vectors u and v under the GW, GCW, weighted GW, and weighted GCW models respectively. We will define $d_{MEDICC}(u, v)$ as the edit distance computed by the MEDICC segmental gain/loss model and $d_{wMEDICC}(u, v)$ as the hypothetical weighted version of that distance.

11.2.3 Other Data Types

While the great majority of work in tumor phylogenetics has focused on either SNV or CNA data, there are alternatives, both established and emerging. Most either follow similar principles to SNV data or are new enough to the literature not to have established methods. We therefore treat them briefly here. Desper et al. [21] introduced the idea of using expression data to measure distance in oncogenetic trees. They considered three classes of measures for expression data, each assuming that taxa are represented by vectors of normalized (Z-scored) expression values across a set of genes, which we can model as $V = (v_1, \ldots, v_m), v_i \in \mathscr{R}$. The simplest measure used Mikowski norms, a generalization of the L1 and L2 norms described for SNV data:

LP-EXP-PDP
Input: Taxa $V_1, V_2 \in \mathscr{R}^m$
Output: A distance $d(V_1, V_2) = \left(\sum_{i=1}^{m} (v_{1i} - v_{2i})^p\right)^{\frac{1}{p}}$

Here, we would most commonly choose $p = 1$ to yield L1 norm, $p = 2$ to yield L2 norm, or $p = \infty$ to yield the infinity norm, equivalent to picking the largest value in each vector. They also considered two other simple measures, the mean-square distance given by $d(V_1, V_2) = 2\left(1 - \sum_{i=1}^{m} v_{1i}v_{2i}\right)$, and the mean-square absolute distance given by $d(V_1, V_2) = 2\left(1 - \left|\sum_{i=1}^{m} v_{1i}v_{2i}\right|\right)$, which we omit listing as separate problems.

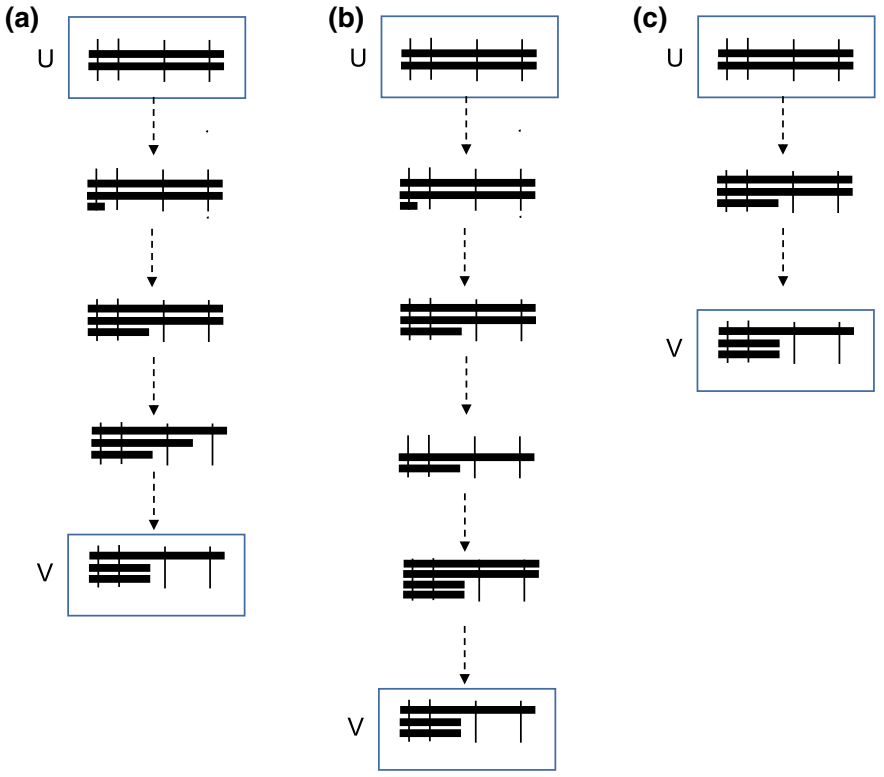


Fig. 11.2 Illustration of CNA distance and various measures by which it is estimated through an example of a single chromosome (thick black bars) probed at four CNA loci (thin black bars) as it is transformed from a putative diploid form U to a form V that is partially haploid and partially triploid. **a** Transformation by L1 distance, in which we model CNA evolution by isolated changes in copy number at single loci corresponding to our probes. Four single locus changes are required to complete the transformation. **b** Transformation by GCW distance, in which the transformation is completed in five steps by two localized genes-scale gains, a chromosome loss, a whole-genome duplication, and another chromosome loss. **c** Transformation by MEDICC distance, in which one segmental gain and one segmental loss are sufficient to perform the transformation

There are a variety of other marker types one might use for measuring evolutionary distance that we do not explicitly present as problems. Microsatellites can be treated as a special case of CNA, and although the mechanisms are quite different from the other CNAs considered here similar methods can be applied to them [67]. Epigenetic modifications (methylation) are a natural marker type for cancer progression and can be treated by similar methods to SNVs, as has been done for some more sophisticated probabilistic phylogeny methods [81]. One complication of expression and methylation data is high correlation between sites acting under common regulation, although one can correct for this bias through network models that estimate changes by co-regulated modules of markers rather than individual variant sites [55].

11.3 Median Nodes

We next consider an intermediate step on the path to full phylogenetic trees: solving a version of Median Node Problem (MNP), which we can define as the problem of identifying an ancestral node that minimizes the sum of evolutionary distances between itself and some set of input vertices. Some classic phylogeny methods depend on our ability to identify either a common ancestor of two nodes or an intermediate between three nodes, which may itself be a challenging problem. Distance-based methods may depend on a variant of this problem for inferring unobserved ancestral cell states. Some character-based methods depend on a different kind of median node as a core part of the tree inference algorithm for discovering unobserved states. We will therefore consider a few variants of MNP.

More abstractly, we will consider versions of two MNP problem variants, which we will call 2-MNP and 3-MNP. 2-MNP can be defined for a given distance measure $d : \Sigma^m \times \Sigma^m \to \mathscr{R}$ as follows:

2-MNP
Input: $V_1, V_2 \in \Sigma^m$
Output: $V_m \in \Sigma^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2)$

3-MNP can be defined for a given distance measure $d : \Sigma^m \times \Sigma^m \times \Sigma^m \to \mathscr{R}$ as follows:

3-MNP
Input: $V_1, V_2, V_3 \in \Sigma^m$
Output: $V_m \in \Sigma^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2) + d(V_m, V_3)$

We will consider below how these approaches have been applied in a few cancer-specific contexts for specific alphabets Σ and distance measures d.

11.3.1 SNV Median Nodes

For a standard (unweighted) SNV distance, both 2-MNP and 3-MNP are core routines of classic phylogenetic methods. We can first state the biallelic 2-MNP, i.e., one in which we assume there are two possible allele values at each site (which we canonically denote 0 and 1) and define distance by the Hamming distance $d(V_1, V_2) = \sum_{i=1}^{m} I(V_1 \neq V_2)$:

0,1-SNV-2-MNP
Input: $V_1, V_2 \in \{0, 1\}^m$

Output: $V_m \in \{0, 1\}^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2)$

and a variant for more general alphabets Σ, e.g., $\Sigma = \{A, C, G, T, -\}$:

Σ-SNV-2-MNP
Input: $V_1, V_2 \in \Sigma^m$
Output: $V_m \in \Sigma^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2)$

We state these for completeness, but both are trivially solvable.

The biallelic version of 3-MNP with Hamming distance is also important in practice as a subroutine of a standard method for intraspecies phylogenetics, known as the reduced median network [6]:

0,1-SNV-3-MNP
Input: $V_1, V_2, V_3 \in \{0, 1\}^m$
Output: $V_m \in \{0, 1\}^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2) + d(V_m, V_3)$

This variant is trivially solvable by taking a majority consensus at each site, i.e., taking the more common of the two variants among the V_1, V_2, and V_3. We can easily generalize to a weighted version by using weighted voting at each site rather than a strict consensus.

For more general alphabets Σ, e.g., $\Sigma = \{A, C, G, T, -\}$, we need a more involved approach, part of a generalization of the median network called the median joining method [5]

Σ-SNV-3-MNP
Input: $V_1, V_2, V_3 \in \Sigma^m$
Output: $V_m \in \Sigma^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2) + d(V_m, V_3)$

We are unaware of any work directly using either of these measures or their weighted versions for SNV tumor phylogenetics. Variants have been used for CNA tumor phylogenetics, though, and one might consider them reasonable models of SNV tumor phylogenies provided the number of mutations is not excessively large.

11.3.2 CNA Median Nodes

Just as with distance measures, median problems become more computationally challenging and interesting when one considers CNA data as inputs. In these cases, good algorithms are not obvious. We can pose CNA versions of MNP for any of the variants

of evolutionary distance measure we considered in the previous section, although not all have known efficient solutions. It is perhaps an accident of how different CNA measures have been used for phylogenetics which of these problem statements have been investigated in the literature. The MEDICC distance was originally used with the distance-based neighbor-joining method [66] which provides one motivation for median node problems in inferring ancestral states. The 2-MNP problem was first formally posed and theoretically analyzed for the MEDICC distance by El-Kebir et al. [24]. They referred to this as the copy number triplet (CNT) problem, but for consistency with our other notation we use an alternate name

MEDICC-CNA-2-MNP
Input: $V_1, V_2 \in N_0^m$
Output: $V_m \in N_0^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2)$, where $d(V_i, V_j)$ is the MEDICC distance d_{MEDICC} defined in Sect. 11.2.2

This problem is formally intractable but can be solved efficiently in practice by an FPT dynamic programming algorithm [24].

The L1 CNA distance has been used with both character and distance-based phylogeny methods, although the GW and GCW distances only with character-based methods so far. The 3-MNP variant, which is used in heuristics for character-based maximum parsimony phylogenetics, has thus been more relevant in practice for these distance measures. For L1 3-MNP, used in [14], we get the following problem statement:

L1-CNA-3-MNP
Input: $V_1, V_2, V_3 \in N_0^m$
Output: $V_m \in N_0^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2) + d(V_m, V_3)$, where $d(V_1, V_2) = \sum_{i=1}^{m} |v_{1i} - v_{2i}|$

This definition has a computationally trivial solution, consisting for each locus of the median of the three copy numbers of V_1, V_2, and V_3 at that locus. Allowing for chromosome and genome-scale events, as in [15], gives us the following:

GCW-CNA-3-MNP
Input: $V_1, V_2, V_3 \in N_0^m$
Output: $V_m \in N_0^m$
Objective: $\min d(V_m, V_1) + d(V_m, V_2) + d(V_m, V_3)$

We know of no efficient solution for this problem statement, and it is solved heuristically in practice [15], as is its weighted version [13]. Note that for asymmetric distance functions such as this, we may need to consider a more complicated problem statement that allows for the possibility that the median node may be either ancestral to or descendent from the three input nodes.

One might reasonably pose the 3-MNP versions for MEDICC distance or 2-MNP for GCW distance. This has not been done to our knowledge, though, and we might consider solutions to those statements open problems. While 3-MNP has not been explicitly posed for MEDICC distance to our knowledge, the harder problem of solving maximum parsimony phylogenies under the MEDICC distance, for which 3-MNP would normally be solved as a subproblem, has been posed and solved by other methods [24]. GCW distance is designed for data with few markers but many cells, where character-based methods are more appropriate. They are therefore less likely to be interesting problems in practice. Accounting for more complicated CNA edit operations, such as BFB [86], is to our knowledge a genuinely open problem that might be of some practical value.

11.3.3 Median Nodes for Other Data Types

We are unaware of any literature specifically on solving for median nodes for any other natural tumor phylogeny marker type. Some may have trivial solutions. For example, methylation data might be modeled identically to the 0,1-SNV-MNP problem and solved by the same methods. Real-valued markers, such as in expression-based phylogenetics, can be trivially solved by the arithmetic mean $v_{mi} = (v_{1i} + v_{2i} + v_{3i})/3$ for L1 or L2 distances. Median nodes for more complex SVs (e.g., translocations, inversions, tandem duplications) represent a more interesting challenge, where the solutions are not obvious and we know of no existing theory. These might be considered interesting open problems as of this writing.

11.4 Steiner Tree Problems

The next step in our analysis of increasingly challenging tumor phylogeny problems is the inference of complete phylogenies from tumor marker data. Variants of this problem are solved for cross-sectional study designs, where we assume we have a consensus marker vector for each tumor; regional studies, where we again assume a consensus marker vector for each region; or single-cell studies, where we assume we have typed markers for each of a set of cells in a single tumor. If we are using distance-based methods, then the problems of the preceding two sections will typically give us the tools we need to generalize classic phylogeny methods to tumors. For character-based methods, though, we typically need new algorithms appropriate to tumor marker types. For the combinatorial approaches we consider in this chapter, tree inference usually comes down to solving a form of what is called a Steiner tree problem. Generically, Steiner tree problems can be defined as follows [40]:

STEINER-TREE
Input: set V of vertices, set $S \subseteq V$ of *terminal* vertices, distance measure $d : V \times V \to \mathscr{R}$
Output: tree T with node set V' and edge set E, where $S \subseteq V' \subseteq V$
Objective: $\min_{V',E} \sum_{u,v \in E} d(u, v)$

In practice, with genomic data one often solves for an implicit variant of the Steiner Tree problem, where the Steiner nodes are not explicitly given as input but rather implied as the set of all possible unobserved nodes. Algorithmically, the details for solving such problems can vary quite a bit depending on the assumed input and objective, though. Most reasonable variants of Steiner tree inference will end up formally intractable [40], so such problems are usually solved heuristically.

11.4.1 SNV Steiner Trees

Just as with the MNP, the Steiner tree problem with SNVs is a classic problem in phylogenetics, most often used in the context of intraspecies phylogenetics. We would generally distinguish between two variants

0,1-SNV-STEINER-TREE
Input: set S of terminal vertices in $\{0, 1\}^m$
Output: tree T with node set $V' \subseteq \{0, 1\}^m$ and edge set E, where $S \subseteq V'$
Objective: $\min_{V',E} \sum_{u,v \in E} d(u, v)$, where $d(u, v) = \sum_{i=1}^{m} I(u_i \neq v_i)$

0,1-SNV-STEINER-TREE, also known as the bitstring Steiner tree problem, is a basic version of what is commonly known in the phylogenetics literature as maximum parsimony (MP) character-based phylogenetics [12]. We can also consider generalizing the problem to a generic variant alphabet Σ, which would still be considered a version of MP phylogenetics

Σ-SNV-STEINER-TREE
Input: set S of terminal vertices in Σ^m
Output: tree T with node set $V' \subseteq \Sigma^m$ and edge set E, where $S \subseteq V'$
Objective: $\min_{V',E} \sum_{u,v \in E} d(u, v)$, where $d(u, v) = \sum_{i=1}^{m} I(u_i \neq v_i)$

We can also consider weighted versions for either of these problems, although we do not explicitly state them. While all of these variants yield formally intractable problems, there is a lot of theory on solving them and excellent tools are available for solving them in practice (e.g., PAUP [77], Phylip pars [27]). These and similar off-the-shelf tools have been previously used in the cancer context for solving variants of this problem for tumor phylogenetics (e.g., [60]). There are also generic optimal solvers based on integer linear programs (ILPs) [11, 71] or on fixed-parameter tractable (FPT) methods [70] that can give provably optimal solutions for moderately hard

problem instances. We are not aware of these generic optimal solvers being used for tumor phylogenetics specifically. Those interested in developing new methods to solve these problems or close variants would be well advised to read more on generic phylogenetic inference (e.g., [28]) as there is a great deal of theory already known that is relevant to solving them in practice.

11.4.2 CNA Phylogenetics

Steiner tree inference of CNAs is a much less-studied problem than Steiner tree inference of SNVs, although there is some literature available. We can begin with a simple version of the problem, copy number variation with rectilinear (also known as Manhattan or Taxicab) distance:

L1-CNA-STEINER-TREE
Input: set S of terminal vertices in $\mathcal{N}_0^m$
Output: tree T with node set $V' \subseteq \mathcal{N}_0^m$ and edge set E, where $S \subseteq V'$
Objective: $\min_{V',E} \sum_{u,v \in E} d_{L1}(u, v)$

While this is also an intractable problem [30], it has been well studied even before its use in the cancer context as a model for a mathematically similar problem in integrated circuit design. In some cancer applications, it is common to alter the classic problem statement slightly to force an all-diploid node to be the root node of the tree even if it is not observed. In practice, some normal data will almost always be observed for small marker sets and single-cell data due to contamination of tumors by putatively healthy stromal cells. With genome-scale data, heterogeneous samples, or cell lines, though, a diploid root may need to be explicitly enforced, leading to a somewhat different problem specification than the simple version above.

We can also define the Steiner tree problem for a variety of versions of the GW, wGW, GCW, or wGCW distances. All have been solved by heuristic methods in the prior literature, and provably optimal branch-and-bound variants exist for GCW [15] and wGCW [13] that might be applied with appropriate weights to all of these variants. We present only the most general form here, as studied in [13]:

wGCW-CNA-STEINER-TREE
Input: set S of terminal vertices in $\mathcal{N}_0^m$
Output: tree T with node set $V' \subseteq \mathcal{N}^m$ and edge set E, where $S \subseteq V'$
Objective: $\min_{V',E} \sum_{u,v \in E} d_{wGCW}(u, v)$

We can also pose the MEDICC distance variant of the CNA Steiner tree problem

MEDICC-CNA-STEINER-TREE
Input: set S of terminal vertices in $\mathcal{N}_0^m$

Output: tree T with node set $V' \subseteq \mathcal{N}^m$ and edge set E, where $S \subseteq V'$
Objective: $\min_{V',E} \sum_{u,v \in E} d_{MEDICC}(u, v)$

El-Kebir et al. [24] have developed an ILP method to solve exactly for this problem variant for modest sized trees and numbers of markers. Extending to *wMEDICC* distance would be straightforward for both the model and the ILP, although to our knowledge this has not been done in the literature.

11.4.3 Hybrid and Alternative Distance Measures

Work in tumor phylogenetics has increasingly depended on combining distinct marker types. The most productive area of such work to date has been combining SNV and CNA data (e.g., [18]), although most such work does not lend itself clearly to an explicit combinatorial optimization model. To our knowledge, expression data has not been considered as a Steiner tree marker, as expression measures lend themselves more naturally to distance-based methods. Network-module-based methods that transform expression changes into discrete activation/inactivation of expression modules [55] might plausibly provide a path to a reasonable character-based version of expression phylogenetics or for the similar problem of Steiner trees from methylation data.

11.5 Phylogenetics and Deconvolution

One of the major complications of tumor phylogenetics is that most data so far comes from bulk samples (cross-sectional or regional sequencing) in which we likely have a heterogeneous mixture of cells in each sample. Because of this, the tumor phylogeny field has developed a specialized literature of methods that are based on simultaneously inferring a phylogeny and explaining each measured sample as a mixture of nodes in the phylogeny [8]. The problem of explaining genomic data as a mixture of signals is known by various names, most commonly genomic deconvolution. In understanding the space of tumor phylogeny methods, then, it is important to consider those that involve joint phylogenetics and deconvolution. There are a variety of ways one might pose this problem, which largely come down to asking whether one can explain a set of heterogeneous samples as a mixture of nodes from a phylogeny. Figure 11.3 illustrates the general problem. We consider a few basic variants here.

11.5.1 Deconvolutional Phylogenies on SNVs

Deconvolutional phylogenetics introduces a number of challenges in practice to our earlier representations of phylogeny inference. In deconvolutional phylogenetics, it

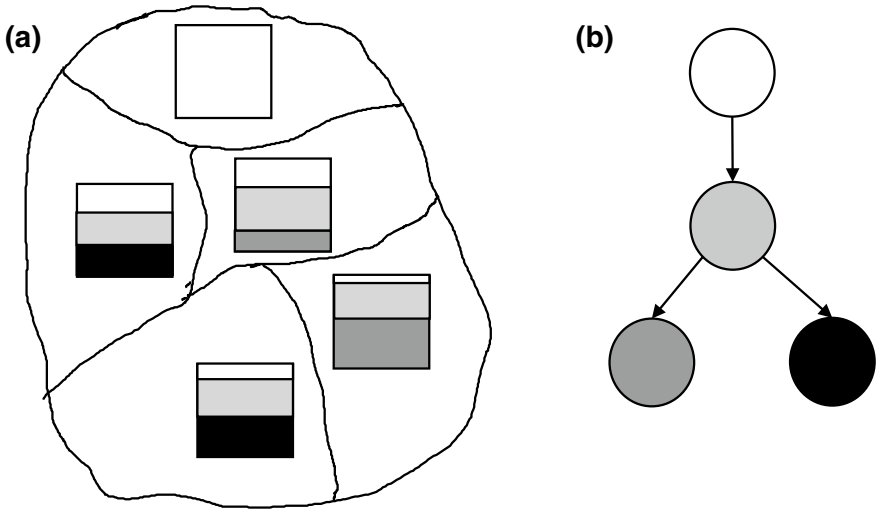


Fig. 11.3 Illustration of deconvolutional phylogenetics. **a** In a deconvolutional phylogeny problem, we assume we have a set of samples (perhaps from different regions of a single tumor, different tumor sites in an individual, or different tumors in an oncogenetic tree variant). We assume our measurements reflect a mixture of all of the clones present. **b** A deconvolutional method seeks to derive a phylogeny describing all clones present in the sample as well as a model describing how much of each clones in the tree is present in each sample

is common to assume that one has not just presence or absence of variants but also a variant allele frequency (VAF) or cancer cell fraction (CCF) describing what fraction of measured alleles or tumor cells have a particular variant at each site. This introduces some important practical complications. One is that these real values are never going to be exact, implying that reasonable models must include some notion of error tolerance. Another is that distinction between VAF and CCF is subtle but important, as VAFs can be directly estimated from sequence read counts but CCFs, which are usually what one really wants, are ambiguous without more knowledge about how alleles are distributed among cells. For example, a population of cells in which half are homozygous for one allele and half homozygous for another will have the same VAF but a different CCF than a population in which every cell is heterogyzous. This problem becomes much more complex when we allow for varying copy number of the locus containing the allele. Much work in the field initially proceeded by adding some potentially restrictive assumptions to remove these ambiguities. One common assumption is that VAFs are derived only from diploid regions of the genome, which may be problematic given widespread copy number variation in cancers. Another common assumption is that mutations accumulate according to a perfect phylogeny, meaning they never recur or revert. The perfect phylogeny assumption is a consequence of a model called the infinite sites assumption (ISA). A final practical issue is that such models have sometimes been posed in the literature in terms of decision or enumeration problems—i.e., the problem of finding the set of trees, if any, that satisfy

a set of output conditions within some error tolerance—as opposed to optimization problems.

In describing these problems, we will begin simplified variants and then consider how to extend them. Note that we necessarily simplify somewhat the actual formulation of these problems in the literature for pedagogical purposes, as most such methods involve fairly complicated probabilistic models for error-handling and modeling raw data in the form of sequence read counts. We will begin with a decision (or enumeration) variant of the most restrictive SNV model, assuming perfect phylogeny and diploid genomes. Here, we will assume data is provided as a matrix V of VAFs for n distinct tumor sites at m loci and that our goal is to infer a matrix of k clonal alleles C, a matrix describing the fraction of each clone at each tumor site, and a tree T describing evolution of the k inferred clones.

One other complication introduced by deconvolutional phylogenetics is that it involves potentially competing measures of solution quality. One can judge quality of the solution by an objective on the quality of the deconvolution, as is typical of pure deconvolution methods, or by the quality of the phylogenetic tree, as is typical of pure phylogeny methods. One way to finesse this issue is to restrict the class of trees allowed, commonly by assuming a perfect phylogeny, and then find the best deconvolution among allowed trees. That is a simplification of the approach taken by some early work in this space [39, 43]:

MIN-PP-SNV-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred
Output: $k \times m$ matrix C where $c_{ij} \in \{0, 1\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$ where T is a perfect phylogeny
Objective: $\min_{C,F} ||V - CF||_2$

The perfect phylogeny constraint says that if we interpret each row of C as a character vector for one node of a tree, then a phylogeny on these nodes can be constructed such that no column (variant nucleotide) changes more than once in the tree. While this might be simply stated, solving the problem in practice requires some significant theoretical tools defined in [35].

We can also define an existence/enumeration variant of this problem, in which we ask whether there exists any solution to the problem, similar to the approach taken by other important work in this domain [23, 35]:

ENUM-PP-SNV-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred, error tolerance $\varepsilon > 0$
Output: $k \times m$ matrix C where $c_{ij} \in \{0, 1\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$ where T is a perfect phylogeny, where for each entry v_{ij}, the corresponding entry x_{ij} for $X = CF$ satisfies $|v_{ij} - x_{ij}| < \varepsilon$
Objective: none

One might alternatively repose the existence variant of the problem to drop the perfect phylogeny constraint and optimize for phylogeny cost, essentially giving a maximum parsimony solution consistent with the mixture constraints, although this solution has not been used in practice to our knowledge

MP-SNV-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred, error tolerance $\varepsilon > 0$
Output: $k \times m$ matrix C where $c_{ij} \in \{0, 1\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$, where for each entry v_{ij}, the corresponding entry x_{ij} for $X = CF$ satisfies $|v_{ij} - x_{ij}| < \varepsilon$
Objective: min $\sum_{(u,v)\in E} d(u, v)$ where d is the Hamming distance $\sum_{i=1}^{m} I(u_i \neq v_i)$

One might also alternatively choose a multi-criterion objective optimizing for some balance between fit to the mixture data and phylogeny cost

MC-SNV-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred, regularization parameter λ
Output: $k \times m$ matrix C where $c_{ij} \in \{0, 1\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$
Objective: min $(||V - CF||_2) + \lambda \left(\sum_{(u,v)\in E} d(u, v)\right)$ where d is the Hamming distance $\sum_{i=1}^{m} I(u_i \neq v_i)$

Again, we are unaware of a method using quite this formulation, although some similar approaches exist in the CNA space, as discussed below.

11.5.2 Deconvolutional Phylogenies on CNAs

The space of methods performing deconvolution and phylogenetics solely on CNAs is limited. This is largely because of an inherent mathematical limitation: changes in copy number of a locus in a single clone are indistinguishable from changes in frequency of the clone, making the problem ill-posed without some additional constraints.

Some work sidesteps this by a variant of the multi-criterion approach, using a cost on the phylogeny to disambiguate equally good deconvolution solutions [62, 63], building on ideas from work for the purely deconvolutional variant of the problem [74].

MC-CNA-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred, regularization parameter λ
Output: $k \times m$ matrix C where $c_{ij} \in \mathcal{N}_0$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} =$

1, tree $T = (C, E)$
Objective: min $(||V - CF||_2) + \lambda \left(\sum_{(u,v)\in E} d_{L1}(u, v)\right)$

An alternative approach to this multi-objective optimization problem, offered by Zaccarria et al. [82, 83], is to optimize for one objective subject to a cap on the other. Rather than balancing deconvolution and phylogeny cost in a single objective, they optimize for deconvolution quality given a maximum phylogeny cost, specified in terms of the MEDICC distance. They further seek to limit degeneracy of solutions by exploiting the fact that copy numbers are discrete and by capping the maximum copy number one can limit possibilities for distinct but equivalent solutions. The result is a problem variant they call the copy number tree mixture deconvolution (CNTMD) problem [82, 83]. In our notation, the problem can be posed as follows:

MEDICC-CNA-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, number of clones k to be inferred, and a maximum copy number $c_{\max}$, and a maximum phylogeny cost $\Lambda_{\max}$
Output: $k \times m$ matrix C where $c_{ij} \in \{0, c_{\max}\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$, such that $\left(\sum_{(u,v)\in E} d_{MEDICC}(u, v)\right) \leq \Lambda_{\max}$
Objective: min $(||V - CF||_2)$

While there are few CNA-specific approaches in the literature, CNAs are widely used in recent deconvolution approaches in conjunction with SNV markers, as discussed in the next section.

11.5.3 Hybrid Deconvolutional Methods

As we allude to above, the VAF/CCF issue made the issue of copy numbers important even if we are only seeking to infer a phylogeny on SNVs. In particular, SNV-only methods generally need to accept the fairly restrictive assumption of working only in diploid genome regions. As a result, a number of deconvolutional tumor phylogeny methods have come to use a combination of SNVs and CNAs in inference as an alternative way to resolve the ambiguity while allowing for copy number variant DNA (e.g., [25, 38]). While the details in real practice are again complicated, we can approximately think of these as generalizations of the SNV methods considered above.

We can begin by considering a generalization of the perfect phylogeny methods to regions of variable copy number, similar to what was done in [25]. One way to pose this problem would be to optimize deconvolution quality subject to the assumption of exact SNV and CNA data and nonrecurrent mutation, similar to purely SNV optimization variants

MIN-PP-SNV + CNA-DP
Input: $n \times m$ SNV VAF matrix V where $v_{ij} \in [0, 1]$, $n \times m$ copy number matrix M

where $m_{ij} \in \mathcal{N}_0$, number of clones k to be inferred
Output: $k \times m$ SNV matrix C where $c_{ij} \in \{0, m_{ij}\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$ where T is a perfect phylogeny
Objective: $\min_{C,F} ||V - CF||_2$

We can similarly consider existence or enumeration variants and noise tolerance, as with the purely SNV methods. This alternative is closer to the problem as actually solved by El-Kebir et al. [25], although also still a simplification:

ENUM-PP-SNV + CNA-DP
Input: $n \times m$ SNV VAF matrix V where $v_{ij} \in [0, 1]$, $n \times m$ copy number matrix M where $m_{ij} \in \mathcal{N}_0$, number of clones k to be inferred, error tolerance $\varepsilon > 0$
Output: $k \times m$ SNV matrix C where $c_{ij} \in \{0, m_{ij}\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$ where T is a perfect phylogeny, where for each entry v_{ij}, the corresponding entry x_{ij} for $X = CF$ satisfies $|v_{ij} - x_{ij}| < \varepsilon$
Objective: none

We can likewise seek to drop the perfect phylogeny assumption. While there are various heuristics in practice to do this, we might pose it more formally as a problem by generalizing the multi-criterion variants of the SNV or CNA problem

MC-SNV + CNA-DP
Input: $n \times m$ matrix V where $v_{ij} \in [0, 1]$, $n \times m$ copy number matrix M where $m_{ij} \in \mathcal{N}_0$, number of clones k to be inferred, regularization parameter λ
Output: $k \times m$ matrix C where $c_{ij} \in \{0, 1\}$, $n \times k$ matrix F where $f_{ij} > 0$ and $\sum_{j=1}^{k} f_{ij} = 1$, tree $T = (C, E)$
Objective: $\min\left(||V - CF||_2\right) + \lambda \left(\sum_{(u,v)\in E} d(u, v)\right)$ where d is the Hamming distance $\sum_{i=1}^{m} I(u_i \neq v_i)$

We might alternatively adopt a three-criterion method, allowing for balancing tree cost with deconvolution quality of both SNV and CNA data. We know of no such method currently in the literature, though, and would expect one might favor moving to a true likelihood model.

11.5.4 Other Marker Types

As with other problem classes, one might in principle pose the deconvolution phylogeny problem for other variant types. These directions are sufficiently immature that we restrict ourselves here to citing some relevant work without delving into specific problem variants that are often still ill-formed as formal computational problems. There has also been some tumor phylogeny work involving deconvolution of methylation data [81], although to our knowledge only using Bayesian models beyond

the scope of the present chapter. Some of the first work in deconvolutional phylogenetics used expression data [65], although via a crude distance-based phylogeny method, and there has been some limited work combining expression and CNA data for joint deconvolution and phylogeny inference [63]. Formulations of the problem have begun to appear for more varied kinds of SV not captured by CNA methods [22], such as inversions and translocations.

11.6 Emerging Directions

One of the challenges of writing a treatment of tumor phylogenetics for a computational audience is that much of the work to be done is in defining the right computational problems, more so than in figuring out how to solve them. As this chapter should already hint, the right problem definitions are a moving target, shifting with our understanding of the biology, our ever-changing technologies for profiling tumor genetic variations, and our evolving appreciation for what tumor phylogenetics can teach us and how. It is important for anyone working in this area, or thinking of entering it, to understand that research in this field requires innovation in problem formulations, which itself depends on an appreciation for the complexity of the biological system and the biological and medical questions we are asking about it, as well as for the computational tools one might apply to them. While we cannot know for certain where the field will go, we can look at some current trends and suggest where new computational problems are likely to arise in the not-too-distant future.

One of the surest bets in tumor phylogenetics is that there will be far more data available in the future than there are today. Advances in single-cell genomic technologies have been dramatic and we can reasonably infer that they will continue to advance, making data acquisition faster and cheaper. Given the substantial advantages of single-cell over bulk sequence, one might reasonably infer that future tumor evolutions studies will be based largely on single-cell data, and in much larger volumes than is typical today. Large data sets have important implications for models and algorithms, helping drive more comprehensive and detailed models but also creating challenges for methods that depend on solving intractable computations in practice. There will likely be difficult tradeoffs between realism and tractability, and a need for new theory to better balance the two. A second trend in the field is the increasing use of long-read technologies, which can be expected to provide better data than prior sequencing technologies, particularly for structural variants, and in turn create a need for computational methods better able to make use of those data. A final conjecture is that the field will need to deal with much larger scales of study populations than has been the standard so far. One of the earliest lessons of single-tumor phylogenetics was that tumor growth is highly stochastic and it is only by averaging across large numbers of samples that one can separate robust trends from random variation [58]. This lesson has needed to be relearned as tumor phylogenetics moved from the computational biology community to mainstream cancer biology, leading to a great deal of contradictory and nonreproducible results [64]. Future tumor phylogeny methods

will need to be able to deal with large study populations. It is likely that there will be a need for a new sub-discipline of consensus tree methods specialized for tumor phylogenetics, a topic that has so far seen only heuristic treatment [32, 58], as well as a new body of theory on relevant statistical methods for optimal design of future tumor phylogeny studies.

A further important area of future work is likely to be bringing tumor phylogenetics to the clinic. There is limited work to date showing that tumor phylogenies contain predictive information that might be used in clinical decision-making. There are, however, likely to be a host of challenges in making that vision a reality. These include figuring out how best to quantify that information and apply it to decision-making tasks and applying it broadly to distinct cancers and decision tasks. These are likely to be hard problems, but crucial ones for tumor phylogenetics to realize its real potential as a window into how tumors evolve on a personalized, patient-specific basis.

11.7 Conclusions

This chapter covered a few examples of computational models that arise in the study of tumor phylogenetics. They are provided here in the hope that they will be instructive in understanding how one thinks about the differences between tumor evolution and classic species evolution, as well as for understanding how one begins to tackle the hard computational challenges in tumor phylogenetics. While each of these problems derives from a greatly simplified model of real tumor biology, many have proven useful starting points for work in developing practical phylogeny methods. They are also hopefully instructive for computer scientists or computational biologists seeking to learn more about tumor phylogenetics and perhaps work on the field themselves. Much of the work in tumor phylogenetics to date has arisen from computational biologists thinking carefully about how to pose formal problems like these that approximate the complexity of the biology and then gradually move them closer to the real system.

We hope also that these examples will prove a good didactic model for some of the lessons Bernard Moret has taught to the phylogenetics field and computational biology in general. Tumor phylogenetics has depended on thinking about evolution and the role of phylogenetics in understanding evolutionary systems beyond the conventional contexts. It has also depended, and will likely still depend, on deep thinking about algorithms for solving some challenging problems in biology. More than that, though, the field has depended on practical thinking about the real-world problem, where methods need to work with large, messy, complicated data sets and give answers that tell us something about an important problem that affects millions of lives.

Acknowledgements Russell Schwartz was supported in part by U.S. National Institutes of Health award R21CA216452 and Pennsylvania Dept. of Health award 4100070287. The Pennsylvania

Department of Health specifically disclaims responsibility for any analyses, interpretations or conclusions.

References

1. Alexandrov, L., Nik-Zainal, S., Wedge, D.C., Aparicio, S.A.J.R., Behjati, S., Biankin, A.V., Bignell, G.R., Bolli, N., Å. Borg, Børresen-Dale, A., Boyault, S., Burkhardt, B., Butler, A.P., Caldas, C., Davies, H.R., Desmedt, C., Eils, R., Eyfjörd, J.E., Foekens, J.A., Greaves, M., Hosoda, F., Hutter, B., Ilicic, T., Imbeaud, S., Imielinski, M., Jger, N., Jones, D.T.W., Jones, D., Knappskog, S., Kool, M., Lakhani, S.R., López-Otín, C., Martin, S., Munsh, N.C., Nakamura, H., Northcott, P.A., Pajic, M., Papaemmanuil, E., Paradiso, A., Pearson, J.V., Puente, X.S., Raine, K., Ramakrishna, M., Richardson, A.L., Richter, J., Rosenstiel, P., Schlesner, M., Schumacher, T.N., Spa, P.N., Teague, J.W., Totoki, Y., Tutt, A.N.J., Valdés-Mas, R., van Buuren, M.M., van 't Veer, L., Vincent-Salomon, A., Waddell, N., Yates, L.R., Australian Pancreatic Cancer Genome Initiative, ICGC Breast Cancer Consortium, ICGC MMML-Seq Consortium, ICGC PedBrain, Zucman-Rossi, J., Futreal, P.A., McDermott, U., Lichter, P., Meyerson, M., Grimmond, S.M., Siebert, R., Campo, E., Tatsuhiro: Signatures of mutation processes in human cancers. Nature **500**(7463), 415–421 (2013)
2. Alexandrov, L.B., Stratton, M.R.: Mutational signatures: the patterns of somatic mutations hidden in cancer genomes. Curr. Opin. Genet. Devel. **24**, 52–60 (2014)
3. Attolini, C.S., Cheng, Y.K., Beroukhim, R., Getz, G., Abdel-Wahab, O., Levine, R.L., Mellinghoff, I.K., Michor, F.: A mathematical framework to determine the temporal sequence of somatic genetic events in cancer. Proc. Natl. Acad. Sci. USA **107**(41), 17604–17609 (2010)
4. Bader, D.A., Moret, B.M., Yan, M.: A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. J. Comput. Biol. **8**(5), 483–491 (2001)
5. Bandelt, H., Forster, P., Röhl, A.: Median-joining networks for inferring intraspecific phylogenies. Mol. Biol. Evol. **16**(1), 37–48 (1999)
6. Bandelt, H.J., Forster, P., Sykes, B.C., Richards, M.B.: Mitochondrial portraits of human populations using median networks. Genetics **141**(2), 743–753 (1995)
7. Beerenwinkel, N., Rahnenführer, J., Däumer, M., Hoffmann, D., Kaiser, R., Selbig, J., Lengauer, T.: Learning multiple evolutionary pathways from cross-sectional data. J. Comput. Biol. **12**(6), 584–598 (2005)
8. Beerenwinkel, N., Rahnenführer, J., Kaiser, R., Hoffmann, D., Selbig, J., Lengauer, T.: Mtreemix: a software package for learning and using mixture models of mutagenetic trees. Bioinformatics **21**(9), 2106–2107 (2005)
9. Beerenwinkel, N., Schwarz, R.F., Gerstung, M., Markowetz, F.: Cancer evolution: mathematical models and computational inference. Syst. Biol. **64**(1), e1–e25 (2015)
10. Campbell, P.J., Pleasance, E.D., Stephens, P.J., Dicks, E., Rance, R., Goodhead, I., Follows, G.A., Green, A.R., Futreal, P.A., Stratton, M.R.: Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing. Proc. Natl. Acad. Sci. USA **105**(35), 13081–13086 (2008)
11. Catanzaro, D., Ravi, R., Schwartz, R.: A mixed integer linear programming model to reconstruct phylogenies from single nucleotide polymorphism haplotypes under the maximum parsimony criterion. Algorithms Mol. Biol. **8**(1), 3 (2013)
12. Cavalli-Sforza, L.L., Edwards, A.W.: Phylogenetic analysis: models and estimation procedures. Evolution **21**(3), 550–570 (1967)
13. Chowdhury, S.A., Gertz, E.M., Wangsa, D., Heselmeyer-Haddad, K., Ried, T., Schäffer, A.A., Schwartz, R.: Inferring models of multiscale copy number evolution for single-tumor phylogenetics. Bioinformatics **31**(12), i258–i267 (2015)
14. Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., Ried, T., Schäffer, A.A., Schwartz, R.: Phylogenetic analysis of multiprobe fluorescence in situ hybridization data from tumor cell populations. Bioinformatics **29**(13), i189–i198 (2013)

15. Chowdhury, S.A., Shackney, S.E., Heselmeyer-Haddad, K., Ried, T., Schäffer, A.A., Schwartz, R.: Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. PLoS Comp. Biol. **10**(7), e1003740 (2014)
16. Christinat, Y., Moret, B.M.: Inferring transcript phylogenies. In: BMC Bioinform. (BioMed Central) **13**(9), S1 (2012)
17. Deisboeck, T.S., Wang, Z., Macklin, P., Cristini, V.: Multiscale cancer modeling. Annu. Rev. Biomed. Eng. **13**, 127–155 (2011)
18. Deshwar, A.G., Vembu, S., Yung, C.K., Jang, G.H., Stein, L., Morris, Q.: PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. Genome Biol. **16**, 35 (2015)
19. Desper, R., Jiang, F., Kallioniemi, O.P., Moch, H., Papadimitriou, C.H., Schäffer, A.A.: Inferring tree models of oncogenesis from comparative genomic hybridization data. J. Comput. Biol. **6**(1), 37–51 (1999)
20. Desper, R., Jiang, F., Kallioniemi, O.P., Moch, H., Papadimitriou, C.H., Schäffer, A.A.: Distance-based reconstruction of tree models for oncogenesis. J. Comput. Biol. **7**(6), 789–803 (2000)
21. Desper, R., Khan, J., Schäffer, A.A.: Tumor classification using phylogenetic methods on expression data. J. Theor. Biol. **228**(4), 477–496 (2004)
22. Eaton, J., Wang, J., Schwartz, R.: Deconvolution and phylogeny inference of structural variations in tumor genomic samples. Bioinformatics (2018). In press
23. El-Kebir, M., Oesper, L., Acheson-Field, H., Raphael, B.J.: Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. Bioinformatics **31**(12), i62–i70 (2015)
24. El-Kebir, M., Raphael, B.J., Shamir, R., Sharan, R., Zaccaria, S., Zehavi, M., Zeira, R.: Complexity and algorithms for copy-number evolution problems. Algorithms Mol. Biol. **12**(1), 13 (2017)
25. El-Kebir, M., Satas, G., Oesper, L., Raphael, B.J.: Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. Cell Syst. **3**(1), 43–53 (2016)
26. Fearon, E., Vogelstein, B.: A genetic model for colorectal tumorigenesis. Cell **61**(5), 759–767 (1990)
27. Felsenstein, J.: PHYLIP-phylogeny inference package (version 3.2). Cladistics **5**(163), 6 (1989)
28. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates Inc, Sunderland, MA (2004)
29. Frumkin, D., Wasserstrom, A., Itzkovitz, S., Stern, T., Harmelin, A., Eilam, R., Rechavi, G., Shapiro, E.: Cell lineage analysis of a mouse tumor. Cancer Res. **68**(14), 5924–5931 (2008)
30. Garey, M.R., Johnson, D.S.: The rectilinear Steiner tree problem is NP-complete. SIAM J. Appl. Math. **32**(4), 826–834 (1977)
31. Gerlinger, M., Rowan, A.J., Horswell, S., Larkin, J., Endesfelder, D., Gronroos, E., Martinez, P., Matthews, N., Stewart, A., Tarpey, P., Varela, I., Phillimore, B., Begum, S., McDonald, N.Q., Butler, A., Jones, D., Raine, K., Latimer, C., Santos, C.R., Nohadani, M., Eklund, A.C., Spencer-Dene, B., Clark, G., Pickering, L., Stamp, G., Gore, M., Szallasi, Z., Downward, J., Futreal, P.A., Swanton, C.: Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. N. Engl. J. Med. **366**(10), 883–892 (2012)
32. Gertz, E.M., Chowdhury, S.A., Lee, W., Wangsa, D., Heselmeyer-Haddad, K., Ried, T., Schwartz, R., Schäffer, A.A.: FISHtrees 3.0: Tumor phylogenetics using a ploidy probe. PLoS ONE **11**(6), e0158569 (2016)
33. Greaves, M., Maley, C.C.: Clonal evolution in cancer. Nature **481**(7381), 306–313 (2012)
34. Gusfield, D.: Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In: Proceedings of the Sixth Annual International Conference on Computational Biology, pp. 166–175. ACM (2002)
35. Hajirasouliha, I., Mahmoody, A., Raphael, B.J.: A combinatorial approach for analyzing intratumor heterogeneity from high-throughput sequencing data. Bioinformatics **30**(12), i78–i86 (2014)
36. Halperin, E., Eskin, E.: Haplotype reconstruction from genotype data using imperfect phylogeny. Bioinformatics **20**(12), 1842–1849 (2004)

37. Hastings, P., Lupski, J.R., Rosenberg, S.M., Ira, G.: Mechanisms of change in gene copy number. Nat. Rev. Genet. **10**(8), 551–564 (2009)
38. Jiang, Y., Qiu, Y., Minn, A.J., Zhang, N.R.: Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing. Proc. Natl. Acad. Sci. USA **113**(37), E5528–E5537 (2016). https://doi.org/10.1073/pnas.1522203113
39. Jiao, W., Vembu, S., Deshwar, A.G., Stein, L., Morris, Q.: Inferring clonal evolution of tumors from single nucleotide somatic mutations. BMC Bioinform. **15**, 35 (2014)
40. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of Computer Computations, pp. 85–103. Springer (1972)
41. Korbel, J.O., Kim, P.M., Chen, X., Urban, A.E., Weissman, S., Snyder, M., Gerstein, M.B.: The current excitement about copy-number variation: how it relates to gene duplications and protein families. Curr. Opin. Struct. Biol. **18**(3), 366–374 (2008)
42. Malikic, S., Jahn, K., Kuipers, J., Sahinalp, C., Beerenwinkel, N.: Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. bioRxiv (2017). https://doi.org/10.1101/234914
43. Malikic, S., McPherson, A.A., Donmez, N., Sahinalp, C.S.: Clonality inference in multiple tumor samples using phylogeny. Bioinformatics **31**(9), 1349–1356 (2015)
44. Mardis, E.R., Wilson, R.K.: Cancer genome sequencing: a review. Hum. Mol. Gen. **18**(R2), R163–R168 (2009)
45. Merlo, L.M.F., Pepper, J.W., Ried, B.J., Maley, C.C.: Cancer as an evolutionary and ecological process. Nat. Rev. Cancer **6**(12), 924–935 (2006)
46. Miller, C.A., White, B.S., Dees, N.D., Griffith, M., Welch, J.S., Griffith, O.L., R., V., Tomasson, M.H., Graubert, T.A., Walter, M.J., Ellis, M.J., Schierding, W., DiPersio, J.F., Ley, T.J., Mardis, E.R., Wilson, R.K., Ding, L.: SciClone: inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution. PLoS Comput. Biol. **10**(8), e1003665 (2014)
47. Misra, N., Szczurek, E., Vingron, M.: Inferring the paths of somatic evolution in cancer. Bioinformatics **30**(17), 2456–2463 (2014)
48. Moret, B.M.: Towards a discipline of experimental algorithmics. In: Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges, Vol. 59, pp. 197–213 (2002)
49. Moret, B.M., Shapiro, H.D.: An empirical analysis of algorithms for constructing a minimum spanning tree. In: Workshop on Algorithms and Data Structures, pp. 400–411. Springer (1991)
50. Moret, B.M., Shapiro, H.D.: Algorithms and experiments: the new (and old) methodology. J. Univ. Comput. Sci. **7**(5), 434–446 (2001)
51. Nair, N.U., Lin, Y., Manasovska, A., Antic, J., Grnarova, P., Sahu, A.D., Bucher, P., Moret, B.M.: Study of cell differentiation by phylogenetic analysis using histone modification data. BMC Bioinform. **15**(1), 269 (2014)
52. Navin, N., Kendall, J., Troge, J., Andrews, P., Rodgers, L., McIndoo, J., Cook, K., Stepansky, A., Levy, D., Esposito, D., et al.: Tumour evolution inferred by single-cell sequencing. Nature **472**(7341), 90–94 (2011)
53. Navin, N., Krasnitz, A., Rodgers, L., Cook, K., Meth, J., Kendall, J., Riggs, M., Eberling, Y., Troge, J., Grubor, V., Levy, D., Lundin, P., Må nér, S., Zetterberg, A., Hicks, J., Wigler, M.: Inferring tumor progression from genomic heterogeneity. Genome Res. **20**(1), 68–80 (2010)
54. Nowell, P.C.: The clonal evolution of tumor cell populations. Science **194**(4260), 23–28 (1976)
55. Park, Y., Shackney, S., Schwartz, R.: Network-based inference of cancer progression from microarray data. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(2), 200–212 (2009)
56. Pattengale, N.D., Alipour, M., Bininda-Emonds, O.R., Moret, B.M., Stamatakis, A.: How many bootstrap replicates are necessary? J. Comput. Biol. **17**(3), 337–354 (2010)
57. Pennington, G., Smith, C.A., Shackney, S., Schwartz, R.: Expectation-maximization method for reconstructing tumor phylogenies from single-cell data. In: Computational Systems Bioinformatics Conference, pp. 371–380 (2006)
58. Pennington, G., Smith, C.A., Shackney, S., Schwartz, R.: Reconstructing tumor phylogenies from heterogeneous single-cell data. J. Bioinform. Comput. Biol. **5**(2a), 407–427 (2007)

59. Popic, V., Salari, R., Hajirasouliha, I., Kashef-Haghighi, D., West, R.B., Batzoglou, S.: Fast and scalable inference of multi-sample cancer lineages. Genome Biol. **16**, 91 (2015)
60. Potter, N.E., Ermini, L., Papaemmanuil, E., Cazzaniga, G., Vijayaraghavan, G., Titley, I., Ford, A., Campbell, P., L., K., Greaves, M.: Single cell mutational profiling and clonal phylogeny in cancer. Genome Res. **23**(12), 2115–2125 (2013)
61. Riester, M., Attolini, C., Downey, R.J., Singer, S., Michor, F.: A differentiation-based phylogeny of cancer subtypes. PLoS Comp. Biol. **6**(5), e1000,777 (2010)
62. Roman, T., Nayyeri, A., Fasy, B.T., Schwartz, R.: A simplicial complex-based approach to unmixing tumor progression data. BMC Bioinform. **16**(1), 254 (2015)
63. Roman, T., Xie, L., Schwartz, R.: Automated deconvolution of structured mixtures from heterogeneous tumor genomic data. PLoS Comput. Biol. **13**(10), e1005815 (2017)
64. Schwartz, R., Schäffer, A.A.: The evolution of tumour phylogenetics: principles and practice. Nat. Rev. Genet. **18**(4), 213 (2017)
65. Schwartz, R., Shackney, S.E.: Applying unmixing to gene expression data for tumor phylogeny inference. BMC Bioinform. **11**, 42 (2010)
66. Schwarz, R.F., Trinh, A., Sipos, B., Brenton, J.D., Goldman, N., Markowetz, F.: Phylogenetic quantification of intra-tumour heterogeneity. PLoS Comput. Biol. **10**(4), e1003535 (2014)
67. Shlush, L.I., Chapal-Ilani, N., Adar, R., Pery, N., Maruvka, Y., Shouval, Spiro A., R., Rowe, J., Tzukerman, M., Bercovich, D., Izraeli, S., Marcucci, G., Bloomfield, C., Zuckerman T. Skorecki, K., Shapiro, E.: Cell lineage analysis of acute leukemia relapse uncovers the role of replication-rate heterogeneity and microsatellite instability. Blood **120**(3), 603–612 (2012)
68. Sottoriva, A., Spiteri, I., Shibata, D., Curtis, C., Tavaré, S.: Single-molecule genomic data delineate patient-specific tumor profiles and cancer stem cell organization. Cancer Res. **73**(1), 41–49 (2013)
69. Sridhar, S., Blelloch, G.E., Ravi, R., Schwartz, R.: Optimal imperfect phylogeny reconstruction and haplotyping (IPPH). In: Computational Systems Bioinformatics, pp. 199–210. World Scientific (2006)
70. Sridhar, S., Dhamdhere, K., Blelloch, G., Halperin, E., Ravi, R., Schwartz, R.: Algorithms for efficient near-perfect phylogenetic tree reconstruction in theory and practice. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**(4), 561–571 (2007)
71. Sridhar, S., Lam, F., Blelloch, G.E., Ravi, R., Schwartz, R.: Mixed integer linear programming for maximum-parsimony phylogeny inference. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**(3), 323–331 (2008)
72. Storchova, Z., Pellman, D.: From polyploidy to aneuploidy, genome instability and cancer. Nat. Rev. Mol. Cell Biol. **5**(1), 45 (2004)
73. Swenson, K.M., Rajan, V., Lin, Y., Moret, B.M.: Sorting signed permutations by inversions in $o(n \log n)$ time. In: Annual International Conference on Research in Computational Molecular Biology (RECOMB), pp. 386–399. Springer (2009)
74. Tolliver, D., Tsourakakis, C., Subramanian, A., Shackney, S., Schwartz, R.: Robust unmixing of tumor states in array comparative genomic hybridization data. Bioinformatics **26**(12), i106–i114 (2010)
75. Tsao, J., Zhang, J., Salovaara, R., Li, Z.H., Järvinen, H.J., Mecklin, J., Aaltonen, L., Shibata, D.: Tracing cell fates in human colorectal tumors from somatic microsatellite mutations: evidence of adenomas with stem cell architecture. Am. J. Pathol. **153**(4), 1189–1200 (1998)
76. Weinberg, R.: The Biology of Cancer. Garland science (2013)
77. Wilgenbusch, J.C., Swofford, D.: Inferring evolutionary trees with paup. Curr. Protoc. Bioinform. 6–4 (2003)
78. Williams, T.L., Moret, B.M.: An investigation of phylogenetic likelihood methods. In: Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering, pp. 79–86. IEEE (2003)
79. Xu, X., Hou, Y., Yin, X., Bao, L., Tang, A., Song, L., Li, F., Tsang, S., Wu, K., Wu, H., He, W., Zeng, L., Xing, M., Wu, R., Jiang, H., Liu, X., Cao, D., Guo, G., Hu, X., Gui, Y., Li, Z., Xie, W., Sun, X., Shi, M., Cai, Z., Wang, B., Zhong, M., Li, J., Lu, Z., Gu, N., Zhang, X., Goodman, L., Bolund, L., Wang, J., Yang, H., Kristiansen, K., Dean, M., Li, Y., Wang, J.: Single-cell exome

sequencing reveals single-nucleotide mutation characteristics of a kidney tumor. Cell **148**(5), 886–895 (2012)
80. Ye, M., Racz, G.C., Jiang, Q., Zhang, X., Moret, B.M.: NEMo: an evolutionary model with modularity for PPI networks. In: International Symposium on Bioinformatics Research and Applications, pp. 224–236. Springer (2016)
81. Yuan, K., Sakoparnig, T., Markowetz, F., Beerenwinkel, N.: BitPhylogeny: a probabilistic framework for reconstructing intra-tumor phylogenies. Genome Biol. **16**, 36 (2015)
82. Zaccaria, S., El-Kebir, M., Klau, G.W., Raphael, B.J.: The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In: International Conference on Research in Computational Molecular Biology (RECOMB), pp. 318–335. Springer (2017)
83. Zaccaria, S., El-Kebir, M., Klau, G.W., Raphael, B.J.: Phylogenetic copy-number factorization of multiple tumor samples. J. Comput. Biol. (2018). https://doi.org/10.1089/cmb.2017.0253
84. Zack, T.I., Schumacher, S.E., Carter, S.L., Cherniack, A.D., Saksena, G., Tabak, B., Lawrence, M.S., Zhang, C.Z., Wala, J., Mermel, C.H., Sougnez, C., Gabriel, S.B., Hernandez, B., Shen, H., Laird, P.W., Getz, G., Meyerson, M., Beroukhim, R.: Pan-cancer patterns of somatic copy number alteration. Nat. Genet. **45**(10), 1134 (2013)
85. Zafar, H., Navin, N., Nakhleh, L., Chen, K.: Computational approaches for inferring tumor evolution from single-cell genomic data. Curr. Opin. Syst. Biol. (2017)
86. Zakov, S., Kinsella, M., Bafna, V.: An algorithmic approach for breakage-fusion-bridge detection in tumor genomes. Proc. Natl. Acad. Sci. USA **110**(14), 5546–5551 (2013)
87. Zeira, R., Zehavi, M., Shamir, R.: A linear-time algorithm for the copy number transformation problem. J. Comput. Biol. **24**(12), 1179–1194 (2017)

Chapter 12
Clusters, Trees, and Phylogenetic Network Classes

Louxin Zhang

Abstract Rooted phylogenetic networks are rooted acyclic digraphs that are used to represent complex evolution, where reticulation events (such as horizontal gene transfer, recombination, etc.) play a role. The combinatorial study of rooted phylogenetic networks has been an active research field of phylogenetics in the past decade. It serves as the foundation for the development of fast algorithms to reconstruct recombination networks in population genetics and hybridization networks in plant science. In this expository chapter, we introduce recent developments in characterizing the classes of rooted phylogenetic networks (including tree-based, reticulation-visible, galled networks, etc.) and designing fast algorithms for the cluster and tree containment problems for rooted phylogenetic networks.

Keywords Clusters · Rooted phylogenetic trees · Galled networks · Tree-based networks · Reticulation-visibility · Near stability · Cluster containment · Tree containment

12.1 Mathematical Models of Evolution

12.1.1 Phylogenetic Trees

Let X be a set of taxa. A (rooted) *phylogenetic tree* on X is a rooted directed tree in which:

- X corresponds one-to-one to the leaves (i.e., nodes of degree 1);
- the root node is of indegree 0 and outdegree at least 2. All the edges are directed away from the root; as such, the orientation of an edge is clear once the root is given and hence usually not indicated in the tree, as shown in Fig. 12.1.
- every non-leaf and non-root node is of indegree 1 and outdegree at least 2.

A phylogenetic tree is *binary* if every non-leaf node is of outdegree exactly 2.

L. Zhang (✉)
Department of Mathematics, National University of Singapore, Singapore, Singapore
e-mail: matzlx@nus.edu.sg

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_12

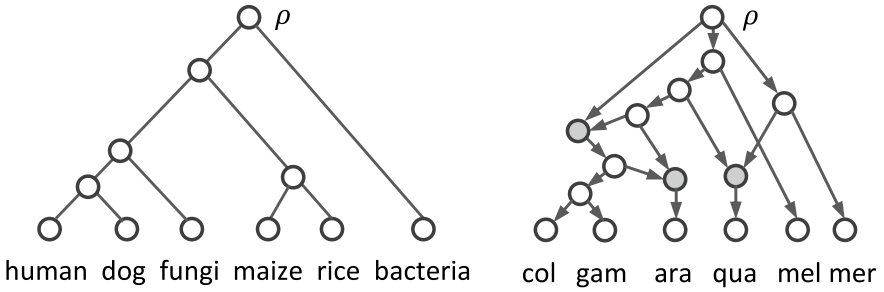


Fig. 12.1 *Left*: A binary rooted phylogenetic tree on human, dog, mushroom, maize, rice, and bacteria. *Right*: A binary rooted phylogenetic network on six members of the *An. gambiae* complex, in which there are three reticulation nodes (filled) that represent introgression events. This network is drawn on the basis of the phylogenetic relationship of the six members given in [8]. Col, *An. gambiae* M form; gam, *An. gambiae* S form; ara, *An. arabiensis*; qua, *An. quadriannulatus*, mel, *An. melas*; mer, *An. merus*

Non-leaf nodes are called *internal* nodes. The internal nodes of a phylogenetic tree represent hypothetical ancestral taxa. In particular, the root represents the "most recent common ancestor" of the taxa represented by the leaves, i.e., all the taxa in X. In molecular evolution, X may represent a set of species, a set of genes, a set of proteins, or a set of genomic sequences.

There are unweighted and weighted phylogenetic trees. In a weighted phylogenetic tree, each edge is assigned a nonnegative weight, which usually corresponds to the evolutionary duration of the lineage represented by the edge. In this chapter, phylogenetic trees are assumed to be unweighted.

12.1.2 Rooted Phylogenetic Networks

A rooted phylogenetic network is a generalization of a phylogenetic tree. These networks are often used to model both tree-like evolution and reticulation events in evolutionary genomics.

Let X be a set of taxa. Formally, a rooted phylogenetic network on X is a rooted, connected but acyclic digraph in which:

- X corresponds one-to-one to the leaves (i.e., nodes of degree 1);
- all the edges are directed away from a unique root node ρ (which is of indegree 0 and outdegree at least 2), as shown in Fig. 12.1;
- every non-leaf and non-root node is either of indegree 1 and outdegree at least 2 or is of outdegree 1 and indegree at least 2;
- there are no parallel edges between any two nodes.

In a rooted phylogenetic network, a node of indegree at most 1 is called a *tree node*. In particular, the root ρ and leaves are tree nodes. A node of indegree greater than 1 is called a *reticulate node*. Non-leaf nodes are said to be *internal*.

For a rooted phylogenetic network N, we use $\mathscr{V}(N)$, $\mathscr{T}(N)$, $\mathscr{R}(N)$ and $\rho(N)$ to denote the set of nodes, the set of internal tree nodes, the set of reticulate nodes, and the root of N, respectively. Similarly, we use $\mathscr{E}(N)$ to denote the set of edges.

For two nodes u, v of N, we say that v is the *child* of u and that u is the *parent* of v if $(u, v) \in \mathscr{E}(N)$. For a reticulate node r, we use $P(r)$ to denote the set of the parents of r and $c(r)$ to denote the unique child of r. For a tree node t, we use $p(t)$ to denote the unique parent of t and $C(t)$ to denote the set of the children of t.

We say that v is *below* u if there is a directed path from u to v. For convenience, we say a node is also below itself. We say that u is an *ancestor* of v if v is below u. Node u is said to be an ancestor of a subset of nodes if it is an ancestor of every node in the subset.

Additionally, for an edge $(u, v) \in \mathscr{E}(N)$, we call this a *tree edge* if v is a tree node and a *reticulate edge* if v is a reticulate node. For a reticulate edge e of N, $N - e$ denotes the network obtained from N by removing e, which is clearly connected. Similarly, we use $N - E$ to denote the resulting network after all the edges of E are removed for a set E of reticulate edges. Analogously $N + e$ and $N + E$ are defined for any $e \notin \mathscr{E}(N)$ and $E \cap \mathscr{E}(N) = \emptyset$.

A rooted phylogenetic network is said to be *binary* if every tree node is of outdegree 2 and every reticulate node is of indegree 2. Binary rooted phylogenetic networks have the following basic property:

Theorem 1 *Let N be a binary rooted phylogenetic network on X and $|X| = n$. Then*

$$\begin{aligned} |\mathscr{T}(N)| &= |\mathscr{R}(N)| + (n-1). \\ |\mathscr{E}(N)| &= 3|\mathscr{R}(N)| + 2(n-1). \end{aligned} \tag{12.1}$$

Proof In N, the root is of indegree 0 and outdegree 2; each internal tree node is of indegree 1 and outdegree 2, each reticulate node is of indegree 2 and outdegree 1, and each leaf is of indegree 1 and outdegree 0. Therefore, by the digraph version of the Handshaking lemma, we have

$$2 \times |\mathscr{T}(N)| + 0 \times n + 1 \times |\mathscr{R}(N)| = 1 \times (|\mathscr{T}(N)| - 1) + 1 \times n + 2 \times |\mathscr{R}(N)|,$$

which is equivalent to Eq. (12.1). Since each edge contributes 1 to the outdegree of its tail,

$$|\mathscr{E}(N)| = 2 \times |\mathscr{T}(N)| + 1 \times |\mathscr{R}(N)| = 3|\mathscr{R}(N)| + 2(n-1).$$

Since there is no reticulate node in a binary tree, the above proposition implies the following basic fact about binary trees.

Corollary 1 *Let T be a phylogenetic tree with $n \geq 2$ leaves. If T is binary, it has $n - 1$ internal nodes and $2(n - 1)$ directed edges.*

Tree-child phylogenetic networks. A tree-child network is a binary rooted phylogenetic network in which each internal node has a child that is a tree node (Fig. 12.2a).

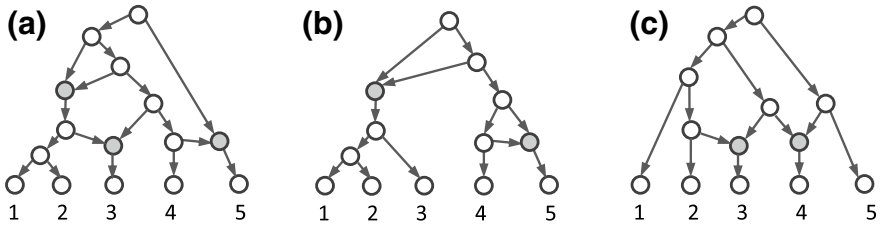


Fig. 12.2 **a** A tree-child network that is not a galled network. **b** A galled tree. **c** A galled network. It is neither a tree-child network nor a galled tree

Equivalently, a rooted phylogenetic network is tree-child if and only if for each node v, there is a path from v to a leaf such that all but v are tree nodes.

Galled trees and galled phylogenetic networks. Let N be a binary rooted phylogenetic network on X. A biconnected component of N is a maximal subgraph in which, for every pair of nodes u and v, an undirected cycle exists that contains u and v. A rooted phylogenetic network is a *galled tree* if every biconnected component contains at most one reticulate node (Fig. 12.2b). Equivalently, in a galled tree, every undirected cycle contains exactly one reticulate node. Notice that any two undirected cycles are node-disjoint in a galled tree. It is less clear that each galled tree is a tree-child phylogenetic network.

A binary rooted phylogenetic network N is a *galled network* if for each reticulate node v of N, there is a tree node u such that there are two edge-disjoint directed paths from u to v that consist only of tree nodes (Fig. 12.2c). Clearly, every galled tree is a galled network.

Temporal property of rooted phylogenetic networks. Let N be a rooted phylogenetic network. N is said to be *temporal* if there is a node labeling $\tau : \mathcal{V}(N) \to \mathbb{R}^+$ satisfying the following two "temporal" conditions:

- $\tau(u) = \tau(v)$ for any reticulate edge (u, v), and
- $\tau(u) < \tau(v)$ for any tree edge (u, v),

where $\mathbb{R}^+$ is the set of nonnegative real numbers.

Not all rooted phylogenetic networks have a temporal node labeling. However, there is a simple polynomial-time algorithm to determine whether an arbitrary network has a temporal node labeling or not [37].

12.1.3 Applications of Rooted Phylogenetic Networks

Rooted phylogenetic networks have been used to model evolutionary relationships in population genetics, plant biology, and genome evolution.

Ancestral recombination networks. The human body is built of trillions of cells. Each cell contains the hereditary material and can make copies of itself by reproduction and division. Genetic recombination may occur when the cell is divided. For example, during meiosis, two homological copies generated by replicating each of the two chromosomes in the premeiotic cell can exchange their genes. As such, the process of meiosis results in the four haploid cells that may each carry genes on either of the two chromosomes in the premeiotic cell. Genetic recombination is a key force responsible for sequence variance at the population scale. In population genetics, a rooted phylogenetic network is used to model the derivation of DNA by both recombination and mutation events, which is called a *genealogical network* [19] and sometimes called an *ancestral recombination network* in the literature.

Hybridization networks. Hybridization is the interbreeding between individuals of different species that results in a novel offspring. Hybridization is one of the important sources for genetic variation. Famous hybrid products include mules (horse $\times$ donkeys) and wheat (*Triticum aestivum*) (three wild grasses). Introgressive hybridization in genetics is gene flow from one species into the gene pool of another by the repeated backcrossing of an interspecific hybrid with one of its parent species, which is also called *introgression*.

In plant science, a rooted phylogenetic network is often used to represent the hybridization history of plant species in which a reticulate node represents a hybridization event, which is called *hybridization network*. For instance, Fig. 12.1b is a network model of the hybridization history of bread wheat [32]. It contains three retrogression events.

Tree of life or net(work) of life? Horizontal gene transfer (HGT) is an evolutionary event in which genetic material moves between organisms other than by the transmission of DNA from parent to offspring. Recent studies of genome evolution reveal that HGT happens frequently. Even pea aphids (*Acyrthosiphon pisum*) acquired multiple genes from fungi through HGT [33]. Because of HGT, gene trees are often inconsistent with the tree of the species that contain the genes in the gene tree. Therefore, Johann Peter Gogarten suggests using "the metaphor of a mosaic to describe the different histories combined in individual genomes and use the metaphor of a net(work) to visualize the rich exchange and cooperative effects of HGT among microbes" [15].

12.2 Decomposition of Rooted Phylogenetic Networks

Let N be a rooted phylogenetic network on X. Then $N - \mathscr{R}(N)$ is a forest, in which each component is a subtree rooted at either the network root or the child of a reticulate node. These components are called the *tree node components* of N. A tree node component is *trivial* if it contains exactly one network leaf. It is *nontrivial* otherwise. Notice that a nontrivial component may contain network leaves.

A reticulate node is *inner* if all its parents belong to the same tree node component. It is *cross* otherwise. In the rooted phylogenetic network given in Fig. 12.3a, the parent

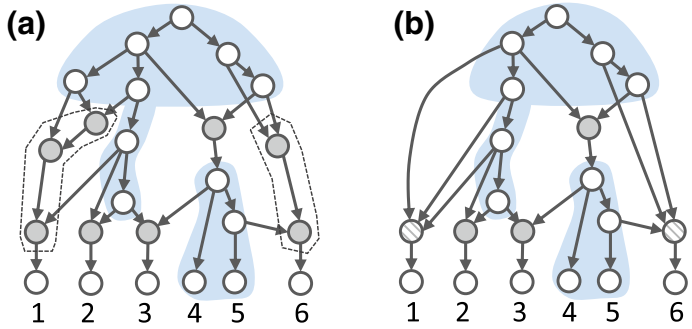


Fig. 12.3 **a** Illustration of the tree node decomposition of a rooted phylogenetic network. The network is decomposed into two nontrivial (shaded) and four trivial tree node components, together with two nontrivial reticulate components (surrounded by dotted curves) and three singleton reticulate components (middle). **b** Compression of the network in (**a**) (defined later)

of Leaf 2 is inner. By contrast, the parent r of Leaf 3 is cross, as the parents of r are in two different nontrivial tree node components.

Similarly, $N - \mathscr{T}(N) - X$ is also a forest. Each component is a rooted subtree with its root being at the *bottom*. These components are called the *reticulate components* of N. An example of such a network decomposition is illustrated in Fig. 12.3, in which the set of tree nodes are partitioned into two nontrivial and four trivial tree node components, whereas the set of reticulate nodes are decomposed into five reticulate components. In summary, we have the following facts [17]:

Theorem 2 *Let N be a rooted phylogenetic network on X. Then,*

(1) *$\mathscr{T}(N) \cup X$ contains exactly all the nodes in the union of tree node components of N.*
(2) *$\mathscr{R}(N)$ contains exactly all the nodes in the union of reticulate components of N.*
(3) *#(tree node components of N) $= 1 +$ #(reticulate components of N).*

A rooted phylogenetic network is said to be *compressed* if each reticulate component contains exactly one reticulate node.

12.3 Clusters in Rooted Phylogenetic Networks

12.3.1 Clusters in Phylogenetic Trees

Let T be a phylogenetic tree on X. For a node u of T, the *cluster* associated with u, denoted $c_T(u)$, is the subset of leaves that become separated from the root upon removal of u. Note that $c_T(u)$ is simply the set of taxa found below u. We let

$$\mathscr{C}(T) = \{c_T(u) \mid u \in \mathscr{V}(T)\}, \tag{12.2}$$

which denotes the set of clusters associated with the nodes of T. For the phylogenetic tree in Fig. 12.1, the clusters include the sets: {human, dog}, {maize, rice}, {human, dog, fungi}, and {human, dog, fungi, maize, rice}, together with X and six singleton sets each containing only one species in X. Biologists usually call clusters "clades" or "monophyletic groups".

Theorem 3 *Let T be a phylogenetic tree on X. Then $\mathscr{C}(T)$ defined in (12.2) satisfies the following two conditions:*

(**C1**) *For any two distinct clusters $A, B \in \mathscr{C}(T)$, we have $A \cap B \in \{A, B, \emptyset\}$; and*
(**C2**) *$X \in \mathscr{C}(T)$ and $\{x\} \in \mathscr{C}(T)$ for each $x \in X$.*

Conversely, given a family $\mathscr{C}$ of subsets satisfying (C1) and (C2), one can compute a unique phylogenetic tree on X such that $\mathscr{C}(T) = \mathscr{C}$.

The proof of Theorem 3 can be found in [37, p. 19]. By Theorem 3, a phylogenetic tree is uniquely determined by $\mathscr{C}(T)$. Therefore, for any two phylogenetic trees T_1 and T_2, the symmetric difference of $\mathscr{C}(T_1)$ and $\mathscr{C}(T_2)$ is used to measure the difference between T_1 and T_2. Precisely, the *Robinson–Foulds distance* between T_1 and T_2, denoted $d_{RF}(T_1, T_2)$, is equal to:

$$|\mathscr{C}(T_1) \triangle \mathscr{C}(T_2)| = |\mathscr{C}(T_1) / \mathscr{C}(T_2)| + |\mathscr{C}(T_2) / \mathscr{C}(T_1)|. \tag{12.3}$$

As an exercise, the reader is suggested to verify that the Robinson–Foulds distance satisfies the triangle inequality.

12.3.2 Cluster Networks and Regular Networks

Let N be a rooted phylogenetic network on X. For any node $u \in \mathscr{V}(N)$, the cluster associated with u, denoted $c_N(u)$, consists of leaves that are reachable by a directed path from u. It is easy to see that X is the cluster associated with the network root and $\{x\}$ is associated with the leaf for each $x \in X$.

Let $\mathscr{C}(N)$ be the collection of clusters associated with the nodes of N. For the rooted phylogenetic network in Fig. 12.2a, the clusters include {1, 2}, {4, 5}, {1, 2, 3}, and {3, 4, 5}, together with X and five singleton sets each containing only one leaf. Noticed that the condition (C1) in Theorem 3 is no longer true.

Unlike for phylogenetic trees, one may have that $\mathscr{C}(N_1) = \mathscr{C}(N_2)$ even for different N_1 and N_2. For example, the rooted phylogenetic network in Fig. 12.4 has the same collection of clusters as the network in Fig. 12.2a. Therefore, a rooted phylogenetic network is generally not determined by giving the collection of clusters in the network.

A rooted phylogenetic network N on X is a *cluster network* if it has the following three properties:

(**P1**) for any nodes u, v of N, $c_N(v) \subseteq c_N(u)$ if and only if v is below u;

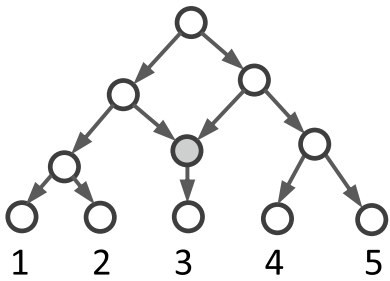


Fig. 12.4 A cluster network on {1, 2, 3, 4, 5} whose clusters consist of {1, 2}, {4, 5}, {1, 2, 3}, {3, 4, 5}, and {1, 2, 3, 4, 5}, together with all five singleton clusters

(**P2**) for different nodes u, v of N, both $c(u) = c(v)$ and v being below u imply that u is a reticulate node, v is a tree node and $(u, v) \in \mathscr{E}(N)$.

(**P3**) for any $u \in \mathscr{V}(N)$ and any child v of u, no node w exists such that $c_N(v) \subset c_N(w) \subset c_N(u)$.

Properties (P1)–(P3) are independent from each other. Property (P2) implies that cluster networks are compressed. Property (P3) implies that

(**P3′**) for any node u, v of N, we have $(u, v) \notin \mathscr{E}(N)$ if there is a directed path from u to v of length greater than 1.

Notice that the tree-child network in Fig. 12.2a does not satisfy Property (P3′).

Proposition 1 *Let N be a rooted phylogenetic network on X. If N is tree-child and satisfies Property (P3′), then N is cluster.*

Proof Let N be a tree-child network on X. We only need to prove that Properties (P1) and (P2) are both true.

Clearly, if v is below u, $c_N(v) \subseteq c_N(u)$. Conversely, assume that u and v are two nodes such that $c_N(v) \subseteq c_N(u)$. Since N is tree-child, there is a path H from v to a leaf x that passes through tree nodes only. Since $x \in c_N(v) \subseteq c_N(u)$, there is a path H' from u to x. If H' does not pass through v, then H and H' must intersect at a node w in H, implying that w is a reticulate node. This contradicts the notion that H passes through tree nodes only. Therefore, H' contains v and thus v is below u. This proves that N satisfies Property (P1).

Assume that $c_N(u) = c_N(v)$ for different u and v. By Property (P1), one is below the other. Without loss of generality, we may assume that v is below u. If u is not a reticulate node, then u is an internal tree node. Let u have the children x and y. Since N is tree-child, there is a path H' from x to a leaf ℓ' and a path H'' from y to a leaf ℓ'' that consists of only tree nodes. Since H' and H'' pass through tree nodes only, they are node-disjoint and, particularly, $\ell' \neq \ell''$. If v is not in H', then, $\ell' \in c_N(u)$ but $\ell' \notin c_N(v)$. If v is not in H'', then, $\ell'' \in c_N(u)$ but $\ell'' \notin c_N(v)$. This contradicts that $c_N(u) = c_N(v)$. If u is a reticulate node and v is not the child of u, we let w be the child of u. Then, $c_N(w) = c_N(u) = c_N(v)$ and v is also below the tree node w, which is impossible, as shown above. This shows that N satisfies Property (P2).

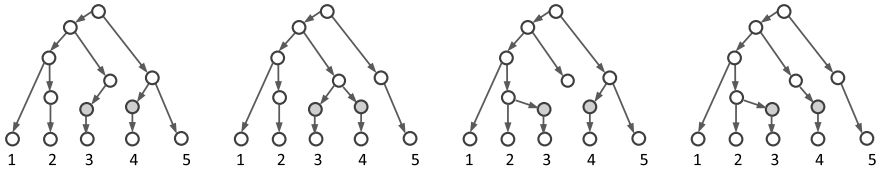


Fig. 12.5 Four possible spanning trees of the rooted phylogenetic network in Fig. 12.2c

By merging each reticulation node with its unique tree-child in a cluster network, we obtain a rooted network satisfying Property (P3′) and the following property:

(**P**1′) for any nodes u, v of N, $c_N(v) \subset c_N(u)$ if and only if v is below u;

Such networks are called *regular networks* in the literature. Note that each node represents a unique cluster in a regular network and regular networks are not rooted phylogenetic networks by definition.

Given a collection $\mathscr{C}$ of subsets of X such that $X \in \mathscr{C}$ and $\{x\} \in \mathscr{C}$ for each $x \in X$, we can construct a digraph $G(\mathscr{C})$ from $\mathscr{C}$. This digraph contains a node for each subset in $\mathscr{C}$; it has a directed edge (C', C'') if $C'' \subset C'$ and there is no set $C \in \mathscr{C}$ such that $C'' \subset C \subset C'$. It is easy to see that $G(\mathscr{C})$ is a regular network. Therefore, a set of clusters on X such that $X \in \mathscr{C}$ and $\{x\} \in \mathscr{C}$ for each $x \in X$ determines uniquely a regular network and hence a cluster network.

12.3.3 Softwired Clusters in Rooted Phylogenetic Networks

Consider a node u of a rooted phylogenetic network N. We can associate clusters other than $c_N(u)$ with u. The cluster $c_T(u)$ associated with u in a spanning tree T of N with the same leaf set X is called a *softwired cluster* of N.

Proposition 2 *Let N be a rooted phylogenetic network on X and let S be a spanning subtree of N. For any reticulate node $u \in \mathscr{R}(N)$ with a child v, $c_S(u) = c_S(v)$ if v is a tree node.*

Since there are a number of spanning trees for N, each internal node represents a set of softwired clusters. We will let $\mathscr{SC}(N)$ denote the set of softwired clusters associated with nodes of N. The rooted phylogenetic network on $X = \{1, 2, 3, 4, 5\}$ in Fig. 12.2c has four possible spanning trees (Fig. 12.5) and thus its nontrivial softwired clusters include $\{1, 2\}$, $\{2, 3\}$, $\{3, 4\}$, $\{4, 5\}$, $\{1, 2, 3\}$ and $\{1, 2, 3, 4\}$. Notice that only $\{1, 2\}$ is not a cluster for the network. In fact, it is true for any rooted phylogenetic network that every cluster is always a softwired cluster.

Proposition 3 ([24]) $\mathscr{C}(N) \subseteq \mathscr{SC}(N)$ *for any rooted phylogenetic network N.*

Proof Let C be a cluster associated with u in N. Then, for each leaf $\ell \in C$, there is a path from u to ℓ. Thus, u and all the nodes below u form a connected digraph $D(u)$

rooted at u with the leaf set C. Removal of the edges leading into $D(u)$ results in a connected spanning graph of N. Clearly, C is the cluster associated with u in any spanning tree of $D(u)$, which is also a spanning tree of N. This completes the proof.

Given a rooted phylogenetic network M, we can obtain a compressed network $C(M)$ from M by replacing each reticulate component C with a single reticulate node u_C. More specifically, let us replace C with u_C such that every direct edge that entered into C now enters into u_C and the edge that led out of C now leads out of u_C, and let us also remove all edges but one between u_C and a tree node and compress nodes of degree 2 if any. For example, the rooted phylogenetic network in Fig. 12.3b is the compression of the rooted phylogenetic network in Fig. 12.3a. $C(M)$ is called the *compression* of M.

Proposition 4 *Let N be a rooted phylogenetic network on X. Then $\mathscr{SC}(N) = \mathscr{SC}(C(N))$.*

Proof Assume that N contains k reticulate components that consist of more than one reticulate node, denoted $R_1, R_2, \ldots, R_k$. Let S be a spanning tree of N. For each $i \leq k$, let r_i be the lowest reticulate node in R_i. Then, S contains exactly one of the edges that lead into R_i and thus there is a path H_i from this incoming edge to r_i (which may consist of r_i only in some extreme cases). Additionally, the cluster associated with each reticulate node of R_i consists only of dummy leaves in S if the reticulate node is not in H_i. Compressing H_i into r_i and removing all the reticulate nodes in R_i that are not in H_i produces a spanning tree for $C(N)$. Therefore, $\mathscr{SC}(N) \subseteq \mathscr{SC}(C(N))$.

The containment $\mathscr{SC}(N) \supseteq \mathscr{SC}(C(N))$ can be proved similarly.

Open question How can we find a rooted phylogenetic network P on X such that $\mathscr{SC}(P) = C$ (if one exists) given a collection C of the subsets of X?

12.3.4 The Cluster Containment Problem

Computing $\mathscr{C}(N)$ from N is easy. However, the computation of $\mathscr{SC}(N)$ from N is intractable. Formally, we define the following problem:
The Cluster Containment Problem

Instance: A rooted phylogenetic network N on X and a subset $X' \subset X$.
Question: Is X' a softwired cluster of N?

Proposition 5 ([28]) *The cluster containment problem is NP-complete.*

To develop a fast algorithm for the cluster containment problem, we will consider the following restricted version of it.
The Small Cluster Containment Problem (SCCP)

Instance: A rooted phylogenetic network N on X, a tree node u and a subset $X' \subset X$.
Question: Is X' a softwired cluster associated with u in N?

It is easy to see that a polynomial-time algorithm for the SCCP can be extended into one for the cluster containment problem. Therefore, the SCCP is also NP-complete. In fact, there is an elegant reduction from the well-known Satisfiability (SAT) problem to this problem [28] (see also [24, p. 170]). Here, we present a simple reduction in the opposite direction, i.e., that is from the SCCP to the SAT problem.

Proposition 6 *Given a rooted phylogenetic network N on X with c reticulate components and e edges, a node $u \in \mathscr{T}(N)$ and a subset X' of X, there is a compressed network N' with at most $c+1$ reticulate nodes and at most $e+3$ edges and $u' \in \mathscr{T}(N')$ such that:*

(i) *u' is the root of a tree node component of N';*
(ii) *X' is a softwired cluster at u in N if and only if X' is a softwired cluster at u' in N'.*

In addition, N' and u' can be constructed from N and u in linear time.

Proof Let $C(N)$ be the compression of N, obtained by replacing each nontrivial reticulate component of N with a single reticulate node and removing parallel edges and degree-2 nodes if any. By Proposition 4, X' is a softwired cluster at u in N if and only if it is a softwired cluster at u in $C(N)$.

Let K_u be the tree node component containing u in $C(N)$ and ρ' is the root of K_u. If $u = \rho'$, then we simply set $N' = C(N)$. If $u \neq \rho'$, we consider two cases. When $\rho \neq \rho_{C(N)}$, we construct N' from $C(N)$ by inserting a tree node t in the edge leading into ρ, a reticulate node r in the edge leading into u and an edge (t, r) (Fig. 12.6b). When $\rho = \rho_{C(N)}$, N' is obtained in a slightly different way, as illustrated in Fig. 12.6c. Since $C(N)$ has c reticulate nodes and at most e edges, N' has $c+1$ reticulate nodes and 3 edges more than $C(N)$.

Assume that u is not the root of the tree node component K_u and that K_u does not contain the network root $\rho(C(N))$. (The argument for the case when $\rho(C(N))$ is the root of K_u is similar.) Assume that X' is a cluster associated with u in a spanning tree T of $C(N)$. Since the edges in which r and t were inserted are tree edges in $C(N)$ and thus are also contained in T, adding r and t in these two edges in T results in a spanning tree T' of N'. Clearly, X' is also the cluster of u in T', as $T'(u)$ is identical to $T(u)$, where $T'(u)$ and $T(u)$ are the subtrees rooted at u of T' and T, respectively.

Conversely, assume that X' is the cluster of u in a spanning tree T' of N'. If the added edge (t, r) is not in T', then, t and r are both of degree 2 and thus compressing t and r gives a spanning tree of $C(N)$ in which X' is the cluster of u. If (t, r) is in T', then, replacing (t, r) with the other edge leading into r in T' generates a spanning tree in which X' is still the cluster of u. This reduces to the case that (t, r) is not in the spanning tree under consideration. This completes the proof.

Let $B = \{b_1, b_2, \ldots, b_m\}$ be a set of Boolean variables. If b is a variable in B, then b and $\bar{b}$ are both said to be literals over B. A clause over B is a set of literals over U, such as $\{b_1, \overline{b_2}, b_5\}$, which is often written $b_1 \vee \overline{b_2} \vee b_5$.

A true assignment for B is a function $\mathscr{A} : B \to \{T, F\}$. If $\mathscr{A}(b) = T$, we say that b is "true" and $\bar{b}$ is "false" under $\mathscr{A}$. If $\mathscr{A}(b) = F$, we say that b is"false" and $\bar{b}$

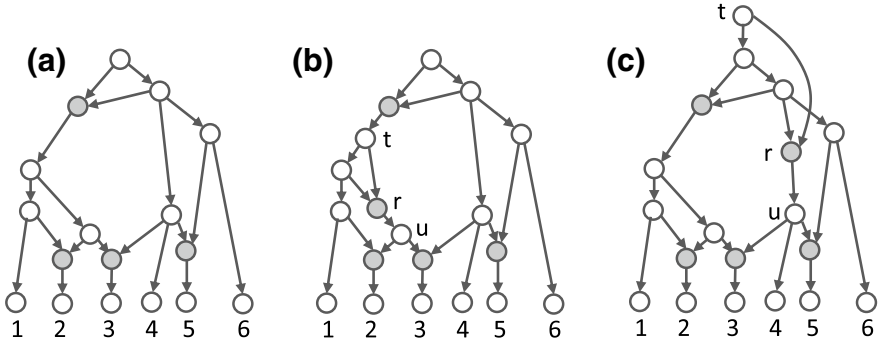


Fig. 12.6 Illustration of the transformation from $C(N)$ to N' in the proof of Proposition 6. **a** A compressed rooted phylogenetic network $C(N)$. **b** N' when the network root $\rho(C(N))$ is not the root of the tree node component C_u that contains u. **c** N' when $\rho(C(N))$ is the root of the tree node component C_u

is "true" under $\mathscr{A}$. A clause is satisfied by $\mathscr{A}$ if and only if at least one of its literals is true under $\mathscr{A}$.

An instance of the SAT problem consists of a collection of clauses over a set of Boolean variables. The problem asks whether or not there is a true assignment for the Boolean variables by which the given clauses are simultaneously satisfied.

Theorem 4 ([43]) *The SCCP can be reduced to the SAT problem in linear time.*

Proof Consider an instance Q of the small cluster containment problem consisting of a network N on X, a tree node u, and $X' \subset X$. By Proposition 6, we may assume that N is compressed and u is the root of the tree node component K_u that contains u.

Let K be a tree node component of N. K is said to be *reachable* from u if the root of K and thus all of K is below u. We denote $\Gamma(N)$ as the set of all tree node components in N and denote $\Gamma(u)$ as the set of tree node components that are reachable from u. We also denote $R(u)$ as the set of reticulate nodes that are reachable from u.

For each tree node component $K \in \Gamma(N)$, we introduce a Boolean variable $b(K)$. For each reticulate node $s \in R(u)$, let K_s be the tree node component rooted at $r(s)$ and let $K_1^{(s)}, K_2^{(s)}, \ldots, K_m^{(s)}$ be all the tree node components such that $P(s) \cap K_j^{(s)} \neq \emptyset$ for each j. We also introduce the following two clauses:

$$\begin{aligned} C_{s1} &: b(K_s) \vee \overline{b\left(K_1^{(s)}\right)} \vee \overline{b\left(K_2^{(s)}\right)} \vee \cdots \vee \overline{b\left(K_m^{(s)}\right)}; \\ C_{s2} &: \overline{b(K_s)} \vee b\left(K_1^{(s)}\right) \vee b\left(K_2^{(s)}\right) \vee \cdots \vee b\left(K_m^{(s)}\right). \end{aligned} \tag{12.4}$$

Additionally, the Boolean variables corresponding to either tree node components that contain at least a leaf in X' but do not contain any leaf not in X' or K_u will each be assigned the value "true". Hence, for each of these components K, we introduce a single-term clause:

$$A_K : b(K), \quad K \in \left\{K_u, K' : \ X' \cap \mathscr{V}(K') \neq \emptyset \ \& \ (X \backslash X') \cap \mathscr{V}(K') = \emptyset\right\} \tag{12.5}$$

Similarly, the Boolean variables corresponding to tree node components that contain a leaf that is not in X' and those not reachable from u will each take the value "false", for which we introduce the following single-term clauses:

$$B_K : \overline{b(K)}, \quad K \in \{K' : \ (X \backslash X') \cap \mathscr{V}(K') \neq \emptyset \text{ or } K \notin \Gamma(v)\}. \tag{12.6}$$

We now show that there is a true assignment under which the collection S_Q of the clauses listed in Eqs. (12.4)–(12.6) is satisfied if and only if X' is a softwired cluster of u in N.

Assume that X' is a cluster of u in a spanning tree T of N. Then the variable $b(K)$ for a tree node component K takes the value "true" if and only if the root of K is below u in T. The clauses in Eqs. (12.5)–(12.6) are clearly satisfiable.

Consider a reticulate node s in N. We let p_s and c_s be the parent and child of s in T, respectively. We also let $P_T(\rho(N), s)$ be the path from $\rho(N)$ to s in T. If $P_T(\rho(N), s)$ contains u, the tree node components containing p_s and c_s are below u and thus the corresponding variables take the value "true", which makes the two clauses defined in Eq. (12.4) for s being satisfiable. If $P_T(\rho(N), s)$ does not contain u, the tree node components containing p_s and c_s are not below u in T and thus the corresponding variables take the value "false". Again, this makes the two clauses defined in Eq. (12.4) for s satisfiable.

Conversely, assume that there is a truth assignment that makes S_Q satisfiable. Consider a tree node component K and the corresponding variable $b(K)$. If $b(K)$ takes the value "true", there must be a path from u to $\rho(K)$ that passes through the tree node components K' whose corresponding variables take the value "true". Assume that it does not hold. There must be a reticulate node s such that the variable corresponding to the tree node component rooted at the child of s is "true" but all variables corresponding the tree node components containing at least one parent of s are "false", implying that the clause C_{s2} defined in Eq. (12.4) is "false" under the assignment. This is a contradiction.

Since each variable corresponding to a tree node component that contains at least a leaf in X' but does not contain any leaf not in X' takes the value "true", the subnetwork consisting of tree nodes in all "true" components and reticulate nodes between them has a spanning subtree that as a root at u and contains all leaves in X' but not in $X \backslash X'$. This shows that X' is a softwired cluster of u in N. This completes the proof.

Because of their NP-completeness, the two cluster containment problems are unlikely to be solved in polynomial time. A practical issue is how to design algorithms for them that are fast enough for practical use. A natural way to measure the time complexity of such algorithms is to measure their run time as a function of the hybridization number of the input network N, defined as

$$H(N) = \sum_{r \in \mathscr{R}(N)} (d_{in}(r) - 1), \tag{12.7}$$

where $d_{in}(r)$ denotes the indegree of r. $H(N)$ is equal to the number of reticulate nodes if N is binary.

By Theorem 4, any algorithm for the SAT can be converted into an algorithm for the SCCP of the same time complexity. Since there is a 3-SAT algorithm that takes $O\left(p(n)2^{0.415n}\right)$ time to determine whether a SAT instance with n Boolean variables is satisfiable or not [34], we have the following result [43].

Corollary 2 *There is an algorithm for the (small) cluster containment problem that takes $O\left(p(H(N))2^{0.415H(N)}\right)$ time on binary rooted phylogenetic networks, where p is a polynomial.*

Open problem Design an algorithm for the (small) cluster containment problem that takes $O(2^{\lambda \cdot H(N)})$ basic set operations on an arbitrary network N, where λ is a constant less than 0.5.

12.3.5 Robinson–Foulds Distances

Formulating a metric on rooted phylogenetic networks that is both meaningful and efficiently computable is of great challenge. The well-known Robinson–Foulds distance is naturally generalized to phylogenetic networks in two ways:

$$\textbf{(Robinson–Foulds metric)} \qquad d_{RF}(N_1, N_2) = |\mathscr{C}(N_1) \triangle \mathscr{C}(N_2)|, \quad (12.8)$$

$$\textbf{(Soft Robinson–Foulds metric)} \quad d_{SRF}(N_1, N_2) = |\mathscr{SC}(N_1) \triangle \mathscr{SC}(N_2)|, \quad (12.9)$$

where $\mathscr{C}(N_i)$ and $\mathscr{SC}(N_i)$ are the sets of clusters and soft clusters of N_i, respectively, for $i = 1, 2$.

For regular networks, $d_{RF}(\ ,\ )$ is a distance metric. In the space of rooted phylogenetic networks, however, there are non-isomorphic-rooted phylogenetic networks N' and N'' such that $d_{RF}(N', N'') = 0$. The Robinson–Foulds metric is not a distance metric in general. The soft Robinson–Foulds metric is neither a distance metric nor computationally feasible.

12.4 Phylogenetic Trees and Rooted Phylogenetic Networks

12.4.1 Trees in Rooted Phylogenetic Networks

Let N be a rooted phylogenetic network on X. After removing all but one of the incoming edges for each reticulate node, we obtain a spanning subtree S of N, which may have new leaves that were internal nodes of N, called dummy leaves, and may have nodes of indegree and outdegree both equal to one. If we further delete from S any nodes below which there are no leaves in X and suppress any nodes of indegree

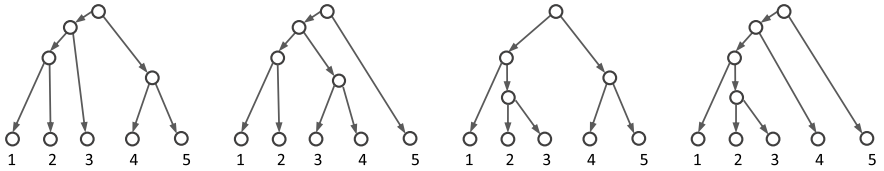


Fig. 12.7 All four phylogenetic trees displayed by the rooted phylogenetic network in Fig. 12.2c, which are derived by compressing the spanning trees of the network listed in Fig. 12.5

and outdegree both equal to one, we obtain a phylogenetic tree T on X. We say that T is displayed by N. For instance, the rooted phylogenetic network in Fig. 12.2c contains four different spanning trees that are listed in Fig. 12.5, from which we can derive four different phylogenetic trees (Fig. 12.7). As such, we consider N to be a kind of consensus of the phylogenetic trees displayed in N.

Notice that a binary rooted phylogenetic network on X with r reticulate nodes can display at most 2^r binary phylogenetic trees, as a binary phylogenetic tree can be displayed in two or more different ways. Let $\mathscr{P}(N)$ denote the set of binary phylogenetic trees on X displayed by a rooted phylogenetic network on X. We can then prove the following fact in the same way as Proposition 4.

Proposition 7 *For any rooted phylogenetic network on X, $\mathscr{P}(N) = \mathscr{P}(C(N))$, where $C(N)$ is the compression of N defined in Sect. 12.3.3.*

Open questions. (1) How do we find a rooted phylogenetic network P on X such that $\mathscr{P}(X) = C$ (if one exists), given a collection C of phylogenetic trees on X?

(2) How do we determine whether a temporal phylogenetic network P exists such that $\mathscr{P}(X) \supseteq C$, given a collection C of phylogenetic trees on X [37]?

Recently, NP-completeness is established by Döcker et al. even for two trees regarding the second open question [7].

12.4.2 Tree-Based Phylogenetic Networks

Suppose N be a binary rooted phylogenetic network on X. Any spanning subtree S of N can be obtained by removal of all but one of the incoming edges for every reticulate node in N. S may or may not contain a dummy leaf (which is not in X), as illustrated in Fig. 12.5. If S does not contain any dummy leaves, then N can be obtained from S by sequentially adding directed edges between nodes of indegree and outdegree both equal to one or, equivalently, N is obtained from T by sequentially adding directed edges between the edges of T, where T is the phylogenetic tree on X obtained from S, by suppressing nodes of indegree and outdegree both equal to one. Interestingly and a little surprisingly, it is not true that every rooted phylogenetic network has a spanning subtree without dummy leaves. For example, any spanning tree of the rooted phylogenetic network in Fig. 12.8a must contain a dummy leaf. This is because any spanning tree must not contain either (u, w) or (v, w) exclusively.

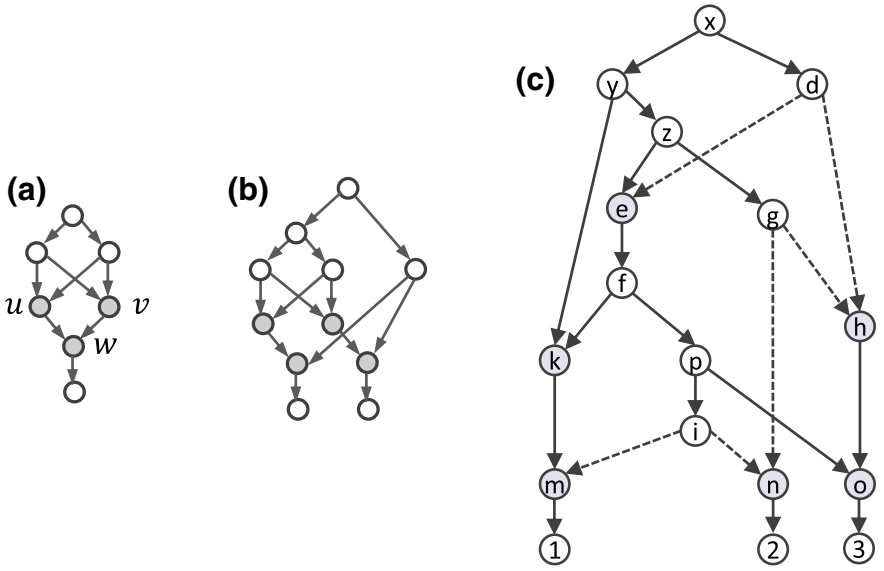


Fig. 12.8 **a** and **b** Two rooted phylogenetic networks that are not tree-based. The example in Part (**b**) is adapted from a figure in [25]. **c** A tree-based phylogenetic network, in which the dashed edges indicate a reticulate node–tree node alternating path from reticulate node m and reticulate node e

If the spanning tree does not contain the former, u is a dummy leaf. If it does not contain the latter, then v is a dummy leaf. The same fact is also true for the network in Fig. 12.8b.

A rooted phylogenetic network is *tree-based* if it has a spanning subtree without any leaves other than the leaves in X. It is less clear that the network in Fig. 12.8c is tree-based. It turns out that tree-based phylogenetic networks comprise a rather large network class that contains properly several well-studied classes, including tree-child networks, as shown below.

We first present a simple characterization of tree-based phylogenetic networks. In a rooted phylogenetic network N on X, a *reticulate node–tree node alternating chain* is a sequence of nodes $\langle v_0, v_1, v_2, \ldots, v_{2k}\rangle$ for some integer $k \geq 1$ satisfying the following two properties:

(1) Node v_i is a reticulate node when i is even and a tree node when i is odd.
(2) For each even $i \leq 2k-2$, $(v_{i+1}, v_i) \in \mathscr{E}(N)$ and $(v_{i+1}, v_{i+2}) \in \mathscr{E}(N)$.

In the rooted phylogenetic network in Fig. 12.8c, the node sequence $\langle m, i, n, g, h, d, e\rangle$ is a reticulate node–tree node alternating chain. Notice that e is an ancestor of m in this example.

A reticulate node u of N is of *type* i if it has exactly i parents that are reticulate nodes, where $i = 0, 1, 2$. The reticulate nodes e and m are of Type 0 and Type 1, respectively and there are no reticulate nodes of Type 2 in the rooted phylogenetic network in Fig. 12.8c. The node w of the network shown in Fig. 12.8a is of Type 2.

Theorem 5 ([44]) *Let N be a binary rooted phylogenetic network. The following are then equivalent.*

(i) *N is tree-based;*
(ii) *N contains neither Type 2 reticulate nodes nor reticulate node–tree node alternating chains between two reticulate nodes of Type 1.*

Proof First, since N is binary, we make the following observations.

(a) Let $e = (x, y) \in \mathscr{E}(N)$. If x is a reticulate node, then x is of outdegree 0 if e is deleted from N and hence becomes a dummy leaf in $N - \{e\}$.
(b) Let $e_1 = (x_1, y_1) \in \mathscr{E}(N)$ and $e_2 = (x_2, y_2) \in \mathscr{E}(N)$ such that $x_i \in \mathscr{T}(N)$ and $y_i \in \mathscr{R}(N)$ for $i = 1, 2$. If $x_1 = x_2$, then x_1 has outdegree 0 if e_1 and e_2 are both deleted from N and hence becomes a dummy leaf in $N - \{e_1, e_2\}$. If $y_1 = y_2$, y_1 is of indegree 0 in $N - \{e_1, e_2\}$.

We let:

$$\mathscr{R}(N) = \{r_1, r_2, \ldots, r_s\};$$
$$\mathscr{P} = \{p \in \mathscr{T}(N) \mid (p, r) \in \mathscr{E}(N), r \in \mathscr{R}(N)\} = \{p_1, p_2, \ldots, p_t\},$$

which consists of the parents of reticulate nodes that are tree nodes in N. Consider the undirected version $B(N)$ of the subgraph induced by $\mathscr{R}(N) \cup \mathscr{P}$ in N. For example, a bipartite graph obtained from the network in Fig. 12.8c is found in Fig. 12.9. Clearly, $B(N)$ is a bipartite graph with edges between $\mathscr{R}(N)$ and $\mathscr{P}$.

For a subset $E \subseteq \mathscr{E}(N) \cap (\mathscr{T}(N) \times \mathscr{R}(N))$, the observations (a) and (b) stated above imply that $N - E$ is a spanning tree with the same leaves as N (i.e., without dummy leaves) if and only if E is a matching covering (every node in) $\mathscr{R}(N)$ in $B(N)$ if the orientation of each edge of E is ignored.

Since $B(N)$ is bipartite, by Hall's theorem ([3], p. 425), there is a matching covering $\mathscr{R}(N)$ in $B(N)$ if and only if $|X'| \leq |b(X')|$ for any $X' \subseteq X$, where $b(X')$ is the set of nodes that are adjacent to at least a node of X', and hence if and only if there is such a matching covering the reticulate nodes in C for every connected component C of $B(N)$.

A reticulate node is of degree i in $B(N)$ if and only if it is a Type $(2 - i)$ node for $i = 0, 1, 2$. Each tree node p of $\mathscr{P}$ is of degree 1 or degree 2, as, by definition, p has at least one child that is reticulate. Therefore, every connected component of $B(N)$ is either a cycle or a path. Let C be a connected component in $B(N)$.

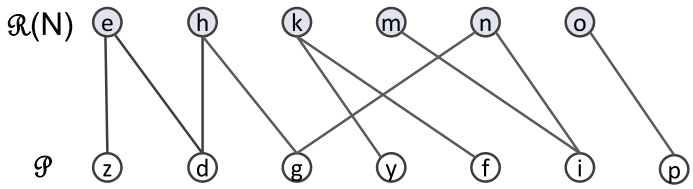


Fig. 12.9 Illustration of the bipartite graph defined in the proof of Theorem 5 for the network in Fig. 12.8c

If C is a cycle, clearly, C has a perfect matching that covers every node of C.

If C is a single node, it must be a node in $\mathscr{R}(N)$ of Type 2 and there is no matching covering the node.

If C is a path containing at least two nodes, then only the two ends of C are of degree 1. There is a matching covering every node in $C \cap \mathscr{R}(N)$ if and only if the ends cannot be both in $\mathscr{R}(N)$.

Since the node of degree 0 and 1 in $\mathscr{R}(N)$ in $B(N)$ correspond to the Type 2 and Type 1 reticulate nodes in N, respectively, we conclude that N is tree-based if and only if no Type 2 node exists in N and no Type 1 reticulate nodes are connected in $B(N)$. Noting that a path between two Type 1 nodes corresponds to a reticulate node–tree node alternating chain, we complete the proof.

The above theorem was also given independently in [27], where Jetten and van Iersel studied nonbinary tree-based phylogenetic networks. Whether or not a reticulate node is of Type 2 can be checked in constant time. Notice that two reticulate node–tree node alternating chains are either identical or node-disjoint. It takes linear time to check whether there are two reticulate nodes of Type 1 that are connected by a reticulate node–tree node alternating chain. Therefore, there is a linear-time algorithm for determining whether a binary rooted phylogenetic network is tree-based or not [11].

Proposition 8 *The following binary rooted phylogenetic networks are tree-based.*

(i) *compressed networks, and*
(ii) *tree-child networks.*

Proof (i) There are no Type 1 and Type 2 nodes in any compressed network.
(ii) Tree-child networks are compressed.

A binary phylogenetic tree is said to be a *base tree* of a rooted phylogenetic network if a subdivision of the former is a spanning subtree of the latter. A base tree of a rooted phylogenetic network is displayed by the network. However, a tree displayed by a rooted phylogenetic network may not be a base of it. In addition, a base tree for a rooted phylogenetic network may be obtained from different spanning subtrees, some of which may contain dummy leaves. This leads to another characterization of tree-child networks.

Proposition 9 ([36]) *A rooted phylogenetic network is tree-child if and only if every spanning tree of it contains no dummy leaves.*

Proof Let N be a binary rooted phylogenetic network. Assume that N is tree-child. For each reticulate node r of N, its two parents and two siblings are all tree nodes. Therefore, in the bipartite graph $B(N)$ defined in the proof of Theorem 5, each connected component C consists exactly of a reticulate node r and its two parents that are connected by two edges entering r. Clearly, each reticulate edge entering r is simply a matching covering r in C. Therefore, for any set E of reticulate edges that enter distinct reticulate nodes, $N - E$ does not contain any dummy leaves. This proves that every spanning tree of N does not contain any dummy leaves.

Conversely, if N is not tree-child, there are two reticulate nodes u and v such that u is the parent of v, or there is a tree node x whose two children y and z are both reticulate. If the former is true, u is a dummy leaf in $N-(u,v)$. Therefore, any spanning tree of $N-(u,v)$ that is also a spanning tree of N contains at least a dummy leaf u. If the latter is true, let $N'=N-\{(x,y),(x,z)\}$. Then, x is a dummy leaf in N'. This implies that any spanning tree of N' that is also a spanning tree of N contains at least a dummy leaf x. This completes the proof.

Recently, several other issues regarding tree-based phylogenetic networks have been studied. It is NP-complete to determine whether a binary phylogenetic tree is a base tree of a binary rooted phylogenetic network [1]. There are tree-based phylogenetic networks on X that have every binary phylogenetic tree on X as their base [11, 21, 44]. Finally, unrooted tree-based networks were studied in [9], in which NP-completeness is shown for the problem of determining whether a unrooted phylogenetic network is tree-based or not.

12.4.3 The Tree Containment Problem

Understanding the relationship between phylogenetic trees and rooted phylogenetic networks is one of the basic computational issues in the field of phylogenetic networks. The *tree containment problem* is formally defined as follows.

The Tree Containment Problem

Instance: A rooted phylogenetic network N on X and a phylogenetic tree T on X.
Question: Is T displayed by N?

Proposition 10 ([26, 28]) *The tree containment problem is NP-complete even for regular networks and for time consistent networks.*

Little is known about algorithm for the tree containment problem. Recently, Gunawan et al. developed an algorithm that takes $O(2^{0.669H(N)})$ basic set operations for a binary network N in [18], the idea of which will be discussed in Sect. 12.5.7. Here, $H(N)$ is the hybridization number of N defined in Eq. (12.7).

Open problem Design an algorithm for the tree containment problem that takes $O(2^{0.5H(N)})$ basic set operations for a rooted phylogenetic network N.

12.5 Reticulation-Visible Phylogenetic Networks

12.5.1 The Node Visibility Property

Let N be a rooted phylogenetic network. For a leaf ℓ of N, more than one path may exist from $\rho(N)$ to ℓ whenever N contains reticulate nodes, where $\rho(N)$ denotes the

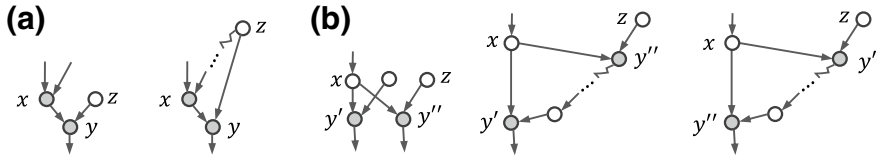


Fig. 12.10 Local structures around an internal node x with at least a child y that is reticulate when x is a reticulate (**a**) and when x is a tree node (**b**) in a binary rooted phylogenetic network. Here, the zigzag denotes a path

root of N. An internal node x is said to be *stable* on ℓ if ℓ is below x and every path from $\rho(N)$ to ℓ contains x. An internal node is *visible* if it is stable on a leaf.

Proposition 11 *Let N be a rooted phylogenetic network and let $x \in \mathcal{V}(N)$, $y \in \mathcal{V}(N)$ such that y is a child of x.*

(i) *The root $\rho(N)$ is visible. Additionally, it is stable on every leaf.*
(ii) *If y is a tree node and visible, then x is also visible.*
(iii) *If x is a reticulate node but y is a tree node, then x is visible if and only if y is.*
(iv) *If x and y are both reticulate, then x is invisible.*

Proof (i) Every path in the definition of the node stability property contains $\rho(N)$. Hence, $\rho(N)$ is stable on every leaf.

(ii) Since y is a tree node, x is the unique parent of y in N. Assume that y is stable on a leaf ℓ. For any path $P : \rho(N) = x_0, x_1, \ldots, x_i, \ldots, x_k = \ell$ from $\rho(N)$ to ℓ, $y = x_j$ for some $j \geq 1$ and hence $x = x_{j-1}$ and P also contains x. Thus, x is also stable on the same leaf ℓ.
(iii) The sufficiency follows from (ii). Similarly, the necessity can be derived from the fact that y is the unique child of x when x is reticulate.
(iv) Assume that x and y are both reticulate. Then, y is the only child of x and y has another parent z such that $z \neq x$, where z may or may not be a tree node. Since x is a reticulate node, there are only two possibilities, as illustrated in Fig. 12.10a.

Let ℓ be an arbitrary leaf below x. Then ℓ is also below y. If x is not below z, the set of paths from $\rho(N)$ to ℓ is a disjoint union of two nonempty subsets of paths. One of these two subsets consists of all paths that contain x but not z, whereas the other consists of paths that contain z but not x. Therefore, neither x nor z is stable on ℓ. If x is below z (Fig. 12.10a), then there is a path from $\rho(N)$ to ℓ that contains z but not x, implying that, again, x is not stable on ℓ. This proves that x is invisible and completes the proof.

Corollary 3 *Let N be a rooted phylogenetic network, let C be a tree node component of N and let $x \in C$. If x is stable on a leaf, then the root of C is also stable on the leaf.*

Proof Let C be a tree node component of N and let $\rho(C)$ be its root. If $x \in C$, there is a unique path from $\rho(C)$ to x that consists only of a finite number of tree nodes. Hence, by applying Part (ii) of Proposition 11 repeatedly, we conclude that if x is stable on a leaf, so is $\rho(C)$.

Proposition 12 *Let N be a rooted phylogenetic network and let $x \in \mathcal{V}(N)$, $\ell \in \mathcal{L}(N)$ and let $e = (u, v) \in \mathcal{E}(N)$ such that v is a reticulation node. If x is stable on ℓ in N, then, x is also stable on ℓ in $N - e$. Therefore, if x is visible in N, it is also visible in $N - e$.*

Proof Notice that $\rho(N - e) = \rho(N)$. Assume that x is an internal node and ℓ is a leaf such that x is stable on ℓ. Then ℓ is below x in N. Since every path from $\rho(N)$ to ℓ of N contains x and because e is a reticulate edge, there is at least one path from $\rho(N)$ to ℓ that contains x but not e, implying that ℓ is also below x in $N - e$. Any path from $\rho(N - e)$ to ℓ of $N - e$ is also a path of N, which implies that x is also stable on ℓ in $N - e$.

We conclude this section by giving another characterization of tree-child phylogenetic networks using the node visibility property, which first appeared without proof in [12].

Theorem 6 *Let N be a binary rooted phylogenetic network. The following are equivalent.*

(i) *N is tree-child.*
(ii) *Every node of N is visible.*

Proof (i) $\Rightarrow$ (ii). Let x be an internal node of N. The fact that any internal node has a child that is a tree node in N implies that there is a path from x to a leaf ℓ_x that consists only of tree nodes. Therefore, any path from $\rho(N)$ to ℓ_x must pass through x first. This implies that x is stable on ℓ_x.

(ii) $\Rightarrow$ (i). Assume that every node is visible in N and, on the contrary, N is not tree-child. There is then an internal node x neither of whose children is a tree node.

If x is reticulate, it has a unique child y and thus y must be reticulate. By Part (iv) of Proposition 11, x is invisible, a contradiction to the hypothesis that every node is visible.

If x is a tree node, x has two children y' and y'' that are both reticulate (Fig. 12.10b). Then either y' or y'' has a parent z that is neither x nor below x. Similar to the case that x is reticulate, we can prove that x is invisible. This contradicts the hypothesis that every node is visible.

12.5.2 Reticulation-Visible Networks

A rooted phylogenetic network N is said to be *reticulation-visible* if every reticulate node is visible. The network in Fig. 12.11 is reticulation-visible. This network has

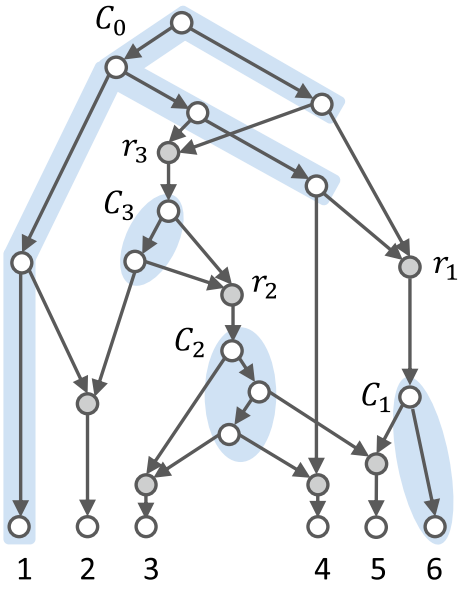


Fig. 12.11 A reticulation-visible phylogenetic network that has four tree node components C_0–C_3

seven reticulate nodes. Both r_2 and r_3 are stable on Leaf 3. Since C_1 contains Leaf 6, r_1 is stable on Leaf 6; the other four reticulate nodes from left to right are stable on Leaves 2, 3, 4, and 5, respectively.

Notice that r_2 is an inner reticulate node whose parents are both in C_3, C_2 also contains all the parents of an inner reticulate node, and C_1 contains a leaf. In fact, every tree node component contains either a leaf or all the parents of an inner reticulation node in any arbitrary reticulation-visible network. To prove this general statement, we first establish the following fact, two special cases of which have appeared in the proof of Part (ii) of Theorem 6.

Lemma 1 *Let N be a rooted phylogenetic network and C be a subtree of N consisting only of internal tree nodes such that the children of each leaf of C are both reticulate nodes of N. If there is no reticulate node r whose parents are all in C, every node of C is invisible.*

Proof Let R denote the set of reticulate nodes that have at least one parent in C. Assume that each node in R has at least one parent that is not in C. Consider any leaf ℓ below C. It must be below a reticulate node $r_1 \in R$. Since R is finite and N is acyclic, there is a sequence of reticulate nodes, $r_1, r_2, \ldots, r_k$, such that:

- each r_j has a parent p_j below r_{j+1} for $1 \leq j < k$, and
- r_k has a parent p_k that is neither in C nor below C.

Since ℓ is below r_1, there is a path $P(r_1, \ell)$ from r_1 to ℓ. Since p_j is below r_{j+1}, there is a path $P(r_{j+1}, p_j)$ from r_{j+1} to p_j for each $j < k$. Since p_k is neither in C nor below C, there is a path $P(\rho(N), p_k)$ from $\rho(N)$ to p_k that does not contain any node in C. Concatenating these paths, we obtain the path:

$$\langle P(\rho(N), p_k), P(r_k, p_{k-1}), \ldots, P(r_1, \ell)\rangle$$

from $\rho(N)$ to ℓ that does not contain any node in C. Therefore, the root C is not stable on ℓ. This has proved that any node of C is invisible.

Theorem 7 ([17]) *Let N be a rooted phylogenetic network. The following are equivalent.*

(i) *N is reticulation-visible;*
(ii) *N is compressed and every tree node component C of N contains either a leaf or all the parents of an inner reticulate node.*

Proof (i) $\Rightarrow$ (ii). Let N be a reticulation-visible network. If the child of a reticulate node is also reticulate, by Part (iv) of Proposition 11, this reticulate node is invisible and thus N is compressed. The second fact follows from Lemma 1.

(ii) $\Rightarrow$ (i). Assume that N satisfies the condition in (ii). Since N is compressed, each reticulate node has a unique child that is the root of a tree node component. By Part (iii) of Proposition 11, we only need to show that the root of every tree node component is visible.

Let C be a tree node component of N and $\rho(C)$ be its root. Since every tree node component contains either a leaf or all the parents of an inner reticulate node, there is a finite sequence of nontrivial components $C_0 = C, C_1, C_2, \ldots, C_k$ ($k \geq 0$) such that:

- C_k contains a leaf;
- The parents of $p(\rho(C_j))$ are each in C_{j-1} for $j = k, k-1, \ldots, 1$,

where $\rho(C_j)$ is the root of C_j and $p(\rho(C_j))$ is the parent of $\rho(C_j)$.

Since C_k contains a leaf ℓ, $\rho(C_k)$ is stable on ℓ. Since the parents of $p(\rho(C_k)$ are each in C_{k-1}, clearly, $r(C_{k-1})$ is also stable on ℓ. Repeating this process, we can prove that $r(C_j)$ is stable on ℓ for each $j = k-2, \ldots, 1, 0$. Hence, C is visible. This completes the proof.

12.5.3 Nearly Stable Networks

Every tree-child phylogenetic network does not contain any invisible internal node. Recently, Gambette et al. introduced the so-called nearly stable networks in the algorithmic study of the tree containment problem [12]. A rooted phylogenetic network N is *nearly stable* if either u or v is visible for any edge $(u, v) \in \mathscr{E}(N)$.

Figure 12.12 presents a nearly stable network, in which two reticulate nodes w and h comprise a reticulate component. Hence, nearly stable binary networks are not necessarily compressed and hence are not necessarily reticulation-visible. However, this network class is closely related to binary reticulation-visible networks as described below.

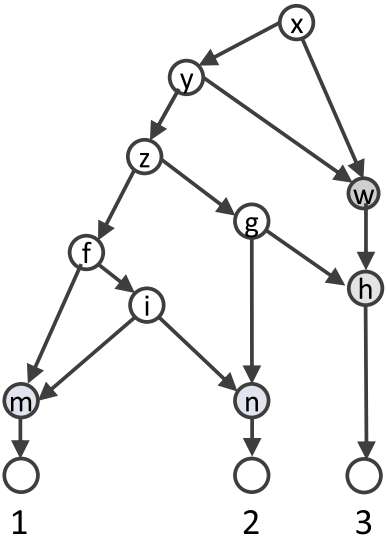


Fig. 12.12 A nearly stable phylogenetic network, in which visible internal nodes include x, y, z, f, m, n, and h, whereas invisible internal nodes include i, g and w. Notice that this network is not reticulation-visible

Let N be a nearly stable network. Assume that u is a reticulate node and v is a tree node such that $(u, v) \in \mathscr{E}(N)$. By Proposition 11, the fact that either u or v is visible implies that both u and v are visible. In other words, the lowest reticulate node of each reticulate component is always visible. Therefore, we have the following fact.

Proposition 13 *Let N be a rooted phylogenetic network. If N is nearly stable, the compression of N is reticulation-visible.*

We finish this section by presenting two topological properties of nearly stable networks, which will be used later. A cherry of a tree consists of an internal node t and all its children if every child of t is a leaf.

Proposition 14 *Let N be a nearly stable network and let K be a tree node component not containing any network leaf. Then the following hold:*

(i) *K, as a tree, contains at least two nodes one of which is a leaf.*
(ii) *Let $u \in \mathscr{V}(K)$ such that u and its children comprise a cherry. Then, there is an inner reticulate node x whose parents are below u in K (Fig. 12.13).*

Proof (i) Let K be a tree node component that does not contain any leaf of N. It is also a tree node component $C(N)$ of the compression of N. By Proposition 13, $C(N)$ is reticulation-visible. Thus by Theorem 7, K contains the two parents of an inner reticulate node.

(ii) Notice that each child of u is not a network leaf. For any child v of u, the two children of v are both reticulate nodes, as v is a leaf of K. By Lemma 1, v is invisible. Therefore u must be visible. If we apply the lemma again, there is an inner reticulate node all of whose parents are below u in K, as shown in Fig. 12.13. Since v is invisible, the inner reticulate node is visible, as it is connected to v by an edge.

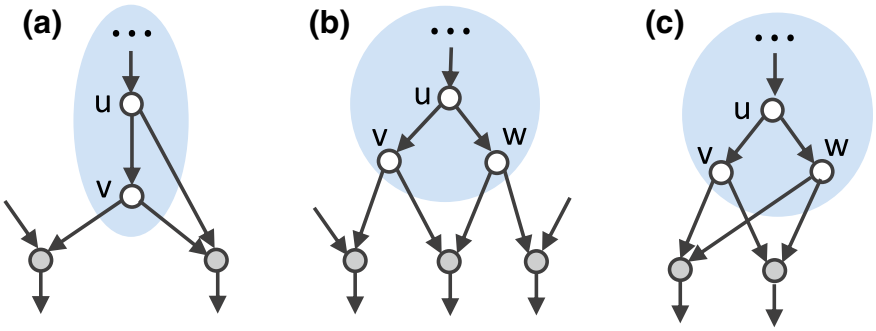


Fig. 12.13 The three possible structures around a cherry consisting of non-network leaves in a tree node component in a nearly stable network. **a** The configuration when the cherry contains only a leaf. **b** The configuration when the cherry contains two leaves; **c** is a special case of (**b**)

12.5.4 A Characterization of Galled Networks

Recall that a binary rooted phylogenetic network is galled if for every reticulate node r, there is a tree node u such that two edge-disjoint paths from u to r exist that both consist only of tree nodes. Galled networks comprise a subclass of reticulation-visible networks. Here, we present a characterization of galled networks using the concept of inner reticulate nodes.

Theorem 8 *Let N be a binary rooted phylogenetic network. The following are equivalent.*

(i) *N is galled.*
(ii) *Every reticulate node of N is inner.*

Proof (i) $\Rightarrow$ (ii). Let r be a reticulate node of a galled network N. By definition, there is a tree node u and two node-disjoint paths P' and P'' from u to r that both consist only of tree nodes. Since r has only two parents, these two parents are contained in P' and P'' and hence are in the same tree node component as u. Therefore, r is an inner reticulate node.

(ii) $\Rightarrow$ (i). Let r be a reticulate node of N, and let p_1 and p_2 be the parents of r. Since r is inner, p_1 ane p_2 are both in the same tree node component C. Hence, p_1 and p_2 are two leaves of C. Let $a(p_1, p_2)$ denote the lowest common ancestor of p_1 and p_2 in C. Clearly, there are two node-disjoint paths from $a(p_1, p_2)$ to r that both consist only of tree nodes.

12.5.5 Sizes of Reticulation-Visible Networks

A rooted phylogenetic network can have as many reticulate nodes as one wants. By definition, we can easily see that a tree-child phylogenetic network on n leaves has

at most $(n-1)$ reticulate nodes. The tight upper bound on the number of reticulate nodes had been established for reticulation-visible networks and nearly stable networks since linear upper bounds were first established for the networks within these two classes in [12].

Theorem 9 *Let N be a binary rooted phylogenetic network that has r reticulate nodes, n leaves and v nodes.*

(i) *If N is galled, $r \leq 2(n-1)$ and $v \leq 6n-5$ ([13]).*
(ii) *If N is reticulation-visible, $r \leq 3(n-1)$ and $v \leq 8n-7$ ([4]).*

Proof Let N have c tree node components and i internal tree nodes. By Theorems 1 and 2,

$$i = r + (n-1); \tag{12.10}$$
$$c = r + 1. \tag{12.11}$$

(i) Assume that N is galled. Since a galled network is reticulation-visible, by Theorem 7, each tree node component of N contains a leaf or all the parents of an inner reticulate node. For a tree node component C that does not contain a leaf, it must contain all the parents of two inner reticulate nodes and hence at least three internal tree nodes (Fig. 12.14a). Let N have c tree nodes components and i internal tree nodes. Then, we have

$$c \leq n + i/3. \tag{12.12}$$

Plugging Eqs. (12.10) and (12.11) into Eq. (12.12), we obtain

$$r + 1 \leq n + [r + (n-1)]/3.$$

Equivalently, $r \leq 2(n-1)$ and thus

$$v = n + r + i = n + r + [r + (n-1)] = 2n - 1 + 2r \leq 6n - 5.$$

(ii) Assume that N is reticulation-visible. By Theorem 7, each tree node component contains a leaf or all the parents of an inner reticulate node. For a tree node component C that does not contain a leaf, it must contain two distinct parents of an inner reticulate node and hence contain at two tree nodes (Fig. 12.14b). Then, we have

$$c \leq n + i/2. \tag{12.13}$$

Similar to the proof of Part (i), this implies that

$$r \leq 3(n-1), \quad v \leq 8n - 7.$$

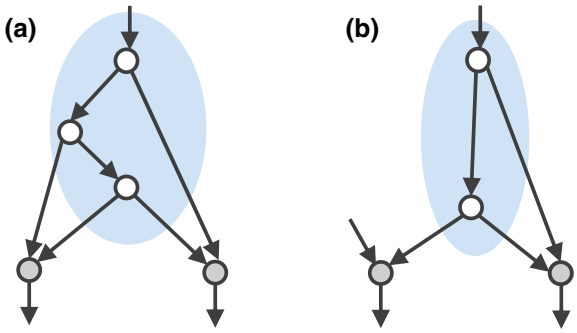


Fig. 12.14 A tree node component that does not contain a network leaf in a rooted phylogenetic network. **a** It contains at least three tree nodes if the network is galled. **b** It contains at least two tree nodes if the network is reticulation-visible

This completes the proof.

Somewhat surprisingly, nearly stable networks have the same tight size bounds as reticulation-visible networks.

Theorem 10 ([13]) *Let N be a binary rooted phylogenetic network that has r reticulate nodes, n leaves, and v nodes. If N is nearly stable, then $r \leq 3(n-1)$ and $v \leq 8n-7$.*

Proof Let c be the number of tree node components of N and let t be the number of internal tree nodes. In N, each node may or may not be visible. Thus, we consider invisible and visible nodes separately. Let r_s (resp. t_s) and r_i (resp. t_i) denote the number of visible and invisible reticulate (resp. internal tree) nodes of N, respectively. Note that $r = r_i + r_s$ and $t = t_i + t_s$.

First, for a tree node component K that contains neither of the network leaves, by Proposition 14, K contains at least two tree nodes: its root, which is visible, and another tree node that is invisible. This implies that

$$c - n \leq t_i, \tag{12.14}$$

as there are at most n components that contain at least one network leaf.

Second, consider an invisible reticulate node x. Since N is nearly stable, the parents of x are both visible tree nodes. By Lemma 1, each visible tree node has at most one child that is reticulate. Therefore, all the r_i invisible reticulate nodes have $2r_i$ distinct visible tree node parents.

Additionally, by Proposition 14, each tree node component K contains a tree node u, each child of which is either a tree node or a visible reticulate (Fig. 12.13) if K does not contain a network leaf. Thus, there are at least $c - n$ visible tree nodes such that their children are all either a tree node or a visible reticulate node.

In summary, we have shown that (1) N contains at least $2r_i$ visible tree nodes that each have an invisible reticulate child and (2) there are at least $c - n$ visible tree nodes that does not have an invisible reticulate child. Hence,

$$(c - n) + 2r_i \leq t_s. \tag{12.15}$$

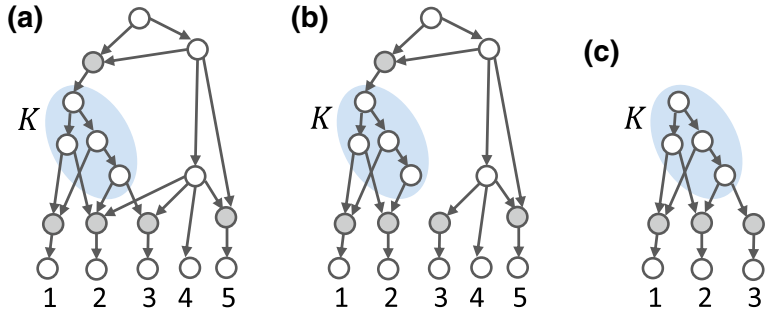


Fig. 12.15 Illustration of the construction of $N(K, +X')$. **a** N and K. **b** $N(K, +X')$ for N, K in (**a**) and $X' = \{1, 2\}$, which is derived by keeping Leaf 2 below K while sending Leaf 3 away from K. In $N(K, +X')$, all leaves below K are in X'. **c** $\overline{K}$, which is the subnetwork induced by all the nodes below the root of K

If we combine Eqs. (12.14) and (12.15),

$$2(c - n) + 2r_i \leq t_i + t_s = t. \tag{12.16}$$

Third, since the parent of the root of a tree node component is either the network root or a visible reticulate node,

$$c = 1 + r_s. \tag{12.17}$$

If we replace c with $r_s + 1$ and then $r_s + r_i$ with r in Eq. (12.16), we have

$$2r + 2 - 2n \leq t.$$

Since N is binary, by Eq. (12.1) in Theorem 1, $2r + 2 - 2n \leq r + (n - 1)$. Equivalently, $r \leq 3(n - 1)$. This also implies that $v = t + r + n = (r + n - 1) + r + n \leq 8n - 7$. This completes the proof.

12.5.6 Linear-Time Algorithms for the Cluster Containment Problem

Given an instance consisting of a rooted phylogenetic network N on X and a subset X' of X for the cluster containment problem, our task is to determine whether X' is a softwired cluster in N or not. By Proposition 4, we may assume that N is compressed.

Consider a nontrivial tree node component K of N. K is said to be *minimal* if only reticulate nodes and network leaves are found below K. For the reticulation-visible network in Fig. 12.11, C_1 and C_2 are minimal, but C_0 and C_3 are not.

For a minimal tree node component K, we define

$$\overline{K} = K + \{(p(r), r), (r, c(r)) \mid r \in \mathscr{R}(N) \text{ such that } p(r) \in P(r) \cap K\} \quad (12.18)$$

It is obtained by disconnecting r from all the tree components but K for every reticulate node r below K, as shown in Fig. 12.15c.

Additionally, let r be a reticulate node of N. It is said to be a *child reticulate node* of K if r has at least one parent contained in K. Recall that $P(r)$ denotes the set of parents of r and $c(r)$ denotes the child of r. For a minimal tree node component K and $X' \subset X$, we define

$$\begin{aligned} N(K, +X') = N &- \{(u, r) \in \mathscr{E}(N) \mid u \in P(r) \cap K \ \&\ c(r) \in X \backslash X'\} \\ &- \{(u, r) \in \mathscr{E}(N) \mid u \in P(r) \backslash K \ \&\ c(r) \in X'\}. \end{aligned} \quad (12.19)$$

It is easy to see that $N(K, +X')$ is obtained from N by sending leaves that are not in X' away from K and disconnecting leaves in X' from all the tree node components except K, as shown in Fig. 12.15b.

Proposition 15 *Let N be a rooted phylogenetic network on X, let $X' \subseteq X$ and let K be a minimal tree node component that is stable on a leaf ℓ. Then, the following hold for a tree node u that is an ancestor of K:*

(i) *If $\ell \in X'$, X' is a softwired cluster at u in N if and only if it is in $N(K, +X')$;*
(ii) *If $\ell \in X \backslash X'$, X' is a softwired cluster at u in N if and only if it is in $N(K, +(X \backslash X'))$;*
(iii) *If K is stable on both a leaf in X' and another in $X \backslash X'$, X' is not a softwired cluster of u in N.*

Proof (i) Since $N(K, +X')$ is a subnetwork of N, the sufficiency of the condition is clear. Its necessity is proved as follows.

Assume that there is a spanning tree T of N in which X' is the cluster of u. Since K is below u and stable on ℓ, the unique path P from $\rho(N)$ to ℓ in T must contain u, $\rho(K)$ (i.e., the root of K) and ℓ in sequence. Let

$$R = \{r \in \mathscr{R}(N) \mid P(r) \cap K \neq \emptyset \ \&\ c(r) \in X'\},$$

which consists of all the child reticulate nodes of K whose child is in X', and let $R' \subseteq R$ be the subset of reticulate nodes that are in R but not below $\rho(K)$ in T.

For each $r \in R'$, we let $p_T(r)$ be the parent of r in T and we also let $p_K(r)$ be a parent of r in K (i.e., $p_K(r) \in P(r) \cap K$). Since T is a spanning subtree of N, T contains all the nodes in the path from $\rho(K)$ to $p_K(r)$. This implies that $T - (p_T(r), r) + (p_K(r), r)$ is also a spanning subtree of N in which X' is still the cluster associated with u, as r is below u through the subpath of P from u to $\rho(K)$ and the path from $\rho(K)$ to $p_K(r)$ in K. Repeating this process for all reticulate nodes in R', we obtain the spanning subtree:

$$T - \{(p_T(r), r) \mid r \in R'\} + \{(p_K(r), r) \mid r \in R'\}$$

of $N(K, +X')$, in which X' is a cluster at u. This completes the proof of Part (i).

(ii) This part is symmetric to Part (i) and can be proved similarly.

(iii) Let $\ell' \in X$ and $\ell'' \in X \backslash X'$ be two leaves on which K is stable. Then the parents of $p(\ell')$ are each in K if ℓ' is not in K, as K is minimal. This fact is also true for ℓ''. Since all of K is below u, there is a path from the network root to ℓ' that passes u and then $\rho(K)$, which implies the existence of a path from u to $\rho(k)$ and then to ℓ''. Therefore, in any spanning tree of N, ℓ'' is also below u if ℓ' is below u. This suggests that X' cannot be a cluster at u in any spanning tree of N.

Proposition 16 *Let N be a rooted phylogenetic network on X, let $X' \subseteq X$ and let K be a minimal tree node component. Whether or not X' is a softwired cluster at a tree node u of K can then be determined in linear time.*

Proof A necessary condition for X' being a softwired cluster in K is that all the leaves in X' must be below K. Thus, one just needs to consider $\overline{K}$ to determine whether X' is a softwired cluster in a node of K. One can develop a dynamic programming algorithm for this purpose, a full description of which can be found in [17].

Based on Propositions 15 and 16, we can solve the small cluster containment problem by doing the following:

- Simplify the input network by eliminating the minimal tree node components one by one until K_u, the tree node component containing the given u, becomes minimal. Whenever the minimal component under examination is stable on a leaf in X' and another in $X \backslash X'$, the algorithm returns "No" and terminates.
- Execute the dynamic programming algorithm mentioned in Proposition 16 on $\overline{K_u}$, which is defined in Eq. (12.18), to determine if X' is a softwired cluster of u.

To demonstrate this algorithm, we consider whether or not $X' = \{1, 4, 6\}$ is a softwired cluster at the left child of the root in the network in Fig. 12.11, in which there are four nontrivial tree node components C_0–C_3. The first three steps of the algorithm are illustrated in Fig. 12.16.

First, we start with C_1. It is minimal and stable on 6. Since $6 \in X'$ and $5 \notin X'$, we send 5 away from C_1 and compress C_1 into a leaf labeled with 6, obtaining the second network in the first row, called N'. N' is a compression of $N(C_1, +X')$.

Second, we consider C_2 that is stable on both 3 and 5 in N'. Note that C_2 is not stable on 5 in N. Since $3 \notin X'$, $5 \notin X'$ but $4 \in X'$, we send 4 away from C_2 and compress C_2 into a leaf with the label $\{3, 5\}$, obtaining the second network in Row 2, called N''. N'' is a compression of $N'(C_2, +(X \backslash X'))$.

Third, C_3 becomes a minimal component and stable on the added leaf $\{3, 5\}$ in N''. Since $\{3, 5\} \cap X' = \emptyset$ and $2 \notin X'$, we disconnect 2 from the rest of network and compress C_3 into a new leaf with the label $\{2, 3, 5\}$, obtained the first network in the second row, called N'''.

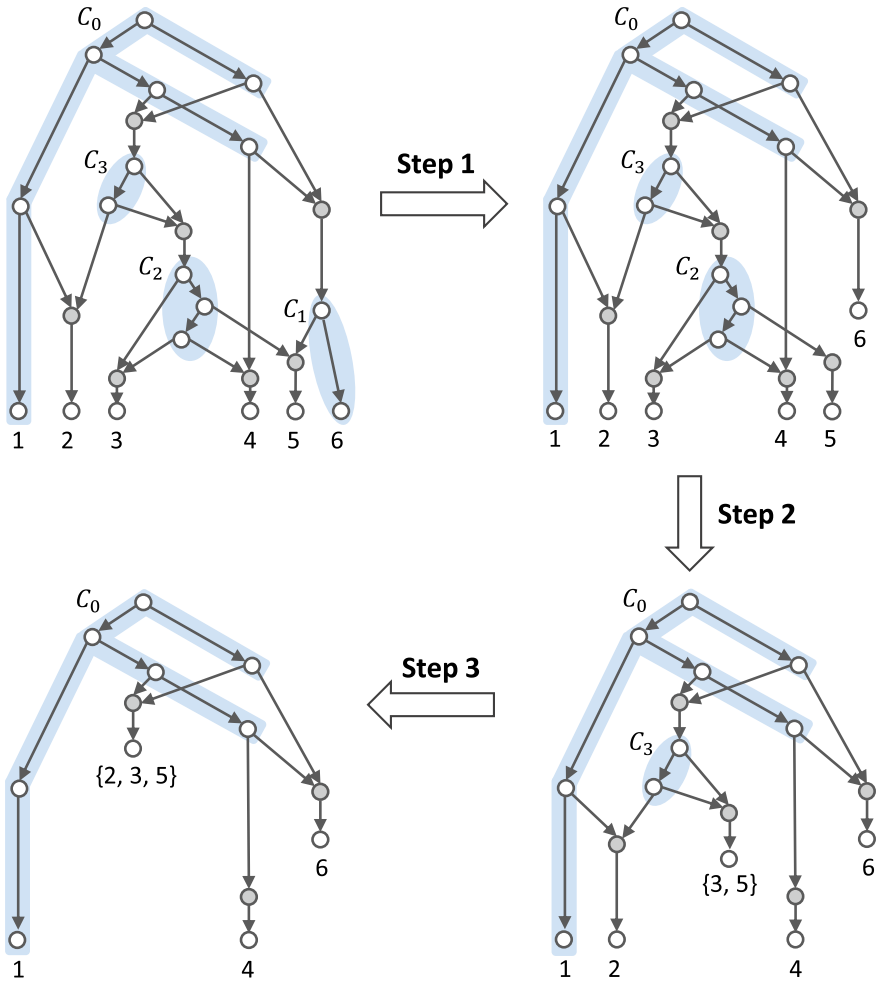


Fig. 12.16 Illustration of the linear-time algorithm for determining whether {1, 4, 6} is a softwired cluster at u in the network in Fig. 12.11

After Step 3, C_0 is the only tree node component in N''', which contains u. C_0 is clearly minimal. Hence, it is time to run the dynamic programming algorithm described in the above proposition to conclude that X' is indeed a softwired cluster at u in C_0 in N'''.

We can obtain a linear-time algorithm for the cluster containment problem by modifying the above algorithm as follows. (1) When a minimal tree node component K is eliminated, we need to check whether X' is a cluster at the root of K if K is stable only on the leaves in X'. (2) When the minimal tree node component is under consideration is stable on both a leaf in X' and another that is not in X', we need to call the dynamic programming algorithm to check if X' is a cluster at a node in

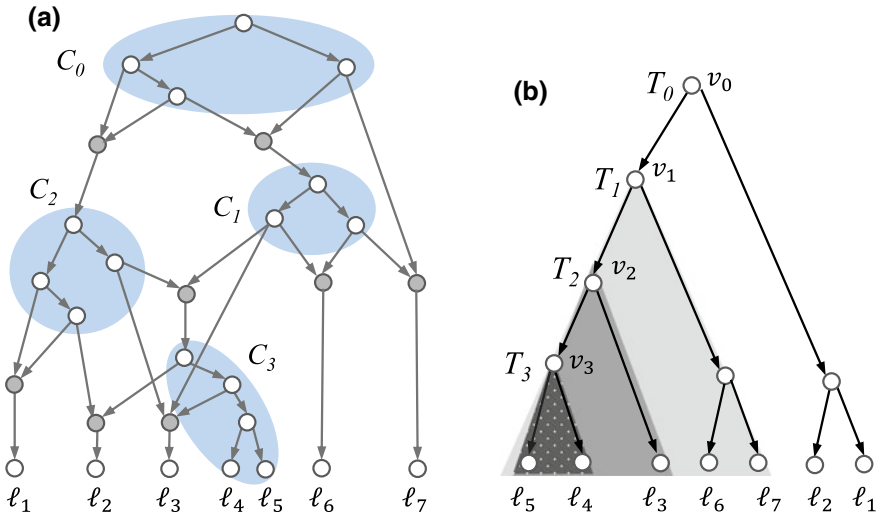


Fig. 12.17 An instance of the tree containment problem. **a** A rooted phylogenetic network N. **b** A phylogenetic tree T with same labeled leaves

this component. The complete description of this algorithm can be found in [17]. Therefore, we obtain the following theorem.

Theorem 11 ([17, 24]) *For the classes of reticulation-visible networks and nearly stable networks there is a linear-time algorithm for (i) the small cluster containment problem and (ii) the cluster containment problem.*

In Propositions 15 and 16, we only assume that K is minimal and visible. Hence, the propositions can also be used to design fast algorithms to solve the (small) cluster containment problem for arbitrary rooted phylogenetic networks [31, 43].

12.5.7 Fast Algorithms for the Tree Containment Problem

Given an instance consisting of a rooted phylogenetic network N and a binary phylogenetic tree T, both of which are on X for the tree containment problem, the task is to determine whether or not T is displayed by N. This is equivalent to determine whether there is a spanning subtree T' of N such that T' becomes a subdivision of T after the removal of all nodes below which there are no leaves in X. For convenience, we call T' an *expansion* of T in N. By Proposition 7, we assume that N is compressed.

Let $\ell \in X$. There is a unique path $P_\ell^{(T)}$ from $\rho(T)$ to ℓ in T:

$$P_\ell^{(T)}: \quad v_0 = \rho(T), v_1, \ldots, v_k = \ell,$$

where $\rho(T)$ is the root of T. We let T_i denote the subtree consisting of all the nodes below v_i for each $0 \leq i \leq k$. Then,

$$\mathscr{V}(T_k) = \{\ell\} \subset \mathscr{V}(T_{k-1}) \subset \cdots \subset \mathscr{V}(T_1) \subset \mathscr{V}(T_0) = \mathscr{V}(T).$$

For the leaf ℓ_5 of the tree in Fig. 12.17b, $k = 3$, T_1, T_2 and T_3 are highlighted in nested shaded triangles and $T_0 = T$.

For a leaf subset S such that $\ell \in S \subseteq X$, define

$$i(S, \ell) = \max\{i \mid S \subseteq \mathscr{V}(T_i)\}. \tag{12.20}$$

Continuing the example, we have $i(\{\ell_4, \ell_5\}, \ell_5) = 3$, as T_3 is the smallest subtree T_i that contains $\{\ell_4, \ell_5\}$ among the five subtrees T_i $(0 \leq i \leq 4)$.

Let K be a minimal tree node component of N that is stable on ℓ and define the following:

$$d(K, T, \ell) = \min\{i \mid T_i \text{ is displayed by } \overline{K}\}. \tag{12.21}$$

Continuing the example, $d(K, T, \ell_5) = 2$, as the largest subtree that can be displayed by $\overline{C_3}$ is T_2 for $K = C_3$ in the network in Fig. 12.17a.

It is not hard to see that the parameter $i(S, \ell)$ can be computed in a time that is linear to the number of nodes in T, given T and ℓ. However, a straightforward implementation of the dynamic programming approach takes quadratic time to compute the parameter $d(K, T, \ell)$. Recently, two better algorithms for computing this parameter have been proposed independently. Gunawan proposed a linear-time algorithm for binary K by using a topological property of the subtree consisting of the visible nodes of K [16]. Weller developed a subquadratic time algorithm for arbitrary tree node components. Formally, he proved the following interesting result [41].

Proposition 17 *For any T, K and $\ell \in X$, the parameter $d(K, T, \ell)$ can be computed in $O(|\mathscr{V}(K)| \cdot m)$, where $m = \max\{d_{in}(r) | r \in \mathscr{R}(\overline{K})\}$ and $d_{in}(r)$ is the indegree of r.*

In [17], Gunawan et al. established the following relationship between the two parameters.

Proposition 18 *Let N be a rooted phylogenetic network on X and let K be a minimal tree component that is stable on a leaf ℓ. Let S_K denote the subset of leaves on which K is stable. If T is displayed by N, then $d(K, T, \ell) \leq i(S_K, \ell)$. In addition, there is a spanning tree T' of N such that $T'(\rho(K))$ is an expansion of $T_{d(K,T,\ell)}$, where $T'(\rho(K))$ is the subtree consisting of nodes below $\rho(K)$ in T'.*

Proof Assume that T is displayed by N. Then there is a spanning tree G of N such that G is an expansion of T. For each $\ell' \in S_K$, K is stable on ℓ' and thus, in G, the unique path from $\rho(N)$ to ℓ' contains $\rho(K)$. Thus, $v_{i(S_k,\ell)}$ corresponds to a node below $\rho(K)$ and hence $T_{i(S_k,\ell)}$ is displayed by $\overline{K}$, implying that $d(K, T, \ell) \leq i(S_K, \ell)$. For

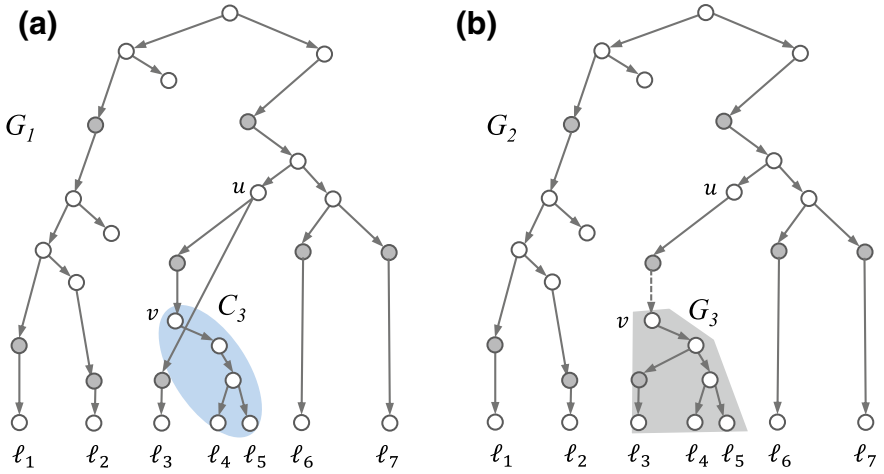


Fig. 12.18 Illustration of the proof of Proposition 18. **a** A spanning tree G_1 of the network N (Fig. 12.17a). This is a subdivision of T (Fig. 12.17b) in which the root T_2 (Fig. 12.17b) is mapped to an ancestor of v, the root of C_3. Here, $S_{C_3} = \{\ell_4, \ell_5\}$ and $d(C_3, T, \ell_5) = 2$. **b** $(G_2 \cup G_3)$(shaded) + $(c(v), v)$(dashed), where $(c(v), v)$ is the incoming edge of the root v of C_3 in N. G_2 is the subtree obtained from G_1 by removing the nodes in $\mathscr{V}(K) \cup \{\ell, p(\ell) \mid \ell \in S_{C_3}\}$

instance, for ℓ_5 of T and C_3 of N in Fig. 12.17, we know that $d(K, T, \ell_5) = 2$, while $i(S_{C_3}, \ell_5) = 3$ from the earlier discussion of the two parameters.

Let $d = d(K, T, \ell)$. Assume that G_1 is a spanning tree of N that is an expansion of T and $u \in \mathscr{V}(G_1)$ corresponds to the root of T_d. In G_1, there is a unique path P from $\rho(N)$ to ℓ that contains u and there are not any leaves other than those of T_d below u. Assume that u is not in K. Since K is stable on ℓ, $\rho(K)$ is below u in P. In addition, since G_1 is obtained from N by removing only reticulate edges, the subtree X rooted at $\rho(K)$ of G_1 consists of the nodes of K and a subset S' of reticulate nodes below K and their children, where $S_K \subseteq S'$. Let G_2 be the subtree obtained from G_1 by removing all the nodes of $\overline{K}$, together with the incident edges. On the other hand, since T_d is displayed by $\overline{K}$, there is a spanning subtree G_3 of $\overline{K}$ that is an expansion of T_d. Then, $(G_2 \cup G_3) + (p(\rho(K), \rho(K))$ is also a spanning tree of N that has the desired property (i.e., T_d is displayed below $\rho(K)$), as illustrated by Fig. 12.18.

On the basis of Proposition 18, we obtain the following algorithm for the tree containment problem on arbitrary reticulation-visible networks.

> Repeatedly simplify the input network N by replacing a minimal tree node component that is stable on ℓ of N and the subtree $T_{d(K,T,\ell)}$ of T with the same leaf ℓ in N and T, respectively, until T is empty. Whenever $i(S_K, \ell) < d(K, T, \ell)$, the algorithm returns "No" and terminates.

The full description of this algorithm can be found in [17]. By Proposition 17, this algorithm takes linear time for reticulation-visible networks with bounded indegree.

Theorem 12 ([41]) *There is a linear-time algorithm for the tree containment problem on (i) binary reticulation-visible networks and (ii) compressed nearly stable networks in which the indegree of each reticulation node is less that a constant c.*

In Proposition 18, we only assume that the minimal tree node component under consideration is visible. Hence, it can be used to design a fast tree containment algorithm for rooted phylogenetic networks [18]. It is interesting to investigate whether this approach can be strengthened to solve the open problem posed in Sect. 12.4.3.

12.6 Relationships Among Network Classes

We have introduced several classes of binary rooted phylogenetic networks: galled trees, galled networks, reticulation-visible networks, etc. and studied their combinatorial properties. By Proposition 8 and Theorems 6 and 8, we summarize their inclusion relationships in Fig. 12.19.

Noticing that both the cluster and tree containment problems remain NP-complete for binary-compressed phylogenetic networks, we list two more research problems.

Problem 1 Define a superclass of networks that contains strictly reticulation-visible networks or/and nearly stable networks for which the cluster (resp. tree) containment problem remains solvable in polynomial time.

Problem 2 Which NP-complete problems are solvable in polynomial time for tree-based networks?

Lastly, galled trees have been extensively studied as a recombination model (see [19], for example). Galled trees have nice combinatorial properties and can be reconstructed efficiently. Although reticulation-visible networks are biologically meaningful, how to reconstruct these networks from trees and sequences has yet to be investigated. This is definitely an interesting topic for future research.

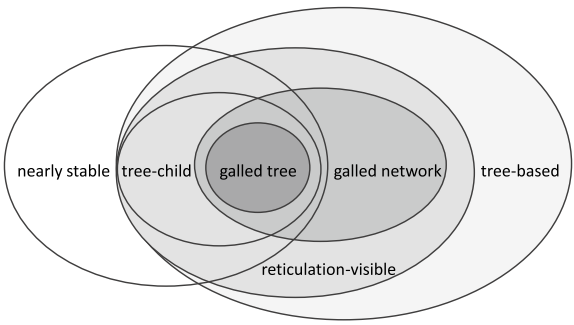


Fig. 12.19 Inclusion relationships among network classes within binary rooted phylogenetic networks

12.7 Bibliographic Notes

1. The tree-child property was introduced by Cardona et al. [6]. Galled trees were first appeared in the papers of Wang et al. [39] and Gusfield et al. [20]. A detailed coverage of galled trees as a recombination network model can be found in the book by Gusfield [19]. The galled network model was first introduced by Huson and Klöpper [22].
2. Theorem 2 was first formulated by Gunawan et al. in the study of the tree containment problem [17]. Later, it was applied to establish the tight bounds on size for galled networks and reticulation-visible networks in [13].
3. Theorem 3 is a classic result in the study of phylogenetic trees. It is the basis for hierarchical classification algorithms such as the UPGMA method (which is covered in [24, 40]). Cluster networks were introduced by Huson and Rupp (see [24]). Regular phylogenetic networks were introduced by Baroni, Semple, and Steel [2]. In [42], Willson showed how to uniquely reconstruct a rooted phylogenetic network given its clusters.
   The relationships among clusters, trees, and rooted phylogenetic networks were first investigated by Nakhleh and Wang [35], in which the cluster and tree containment problems were formally defined. The concept of softwired clusters was coined by Huson et al. [23]. The NP-completeness of the cluster containment problem (Proposition 5) was proved by Kanji et al. [28], together with the tree containment problem. An elegant proof of the NP-completeness of the SCCP appears in [24, p. 170]. Theorem 4 was proved by Gunawan et al. [43]. The Robinson–Foulds distance was first generalized from phylogenetic tree space to the space of rooted phylogenetic network spaces in [5, 24, 30].
4. It is Van Iersel who drew researchers' attention to the tree-based property [25]. The recent study of tree-based networks started with a seminal paper of Francis and Steel [11]. Several research problems posted in this paper were independently answered in the papers of Anaya et al. [1], Hayamizu [21], Jetten and van Iersel [27] and Zhang [44]. Theorem 5 is from Zhang [44], which was independently proved by Jetten and van Iersel [27]. Other characterizations of tree-based networks can be found in [10]. Proposition 9 is from Semple [36].
5. The reticulation-visibility property was defined by Huson et al. [24]. Interestingly, the stability relationships among the nodes in acyclic networks were investigated by Tarjan and Lengauer as early as the 1970s [29, 38] (see also a recent survey [14], where the relationship is called a "dominator"). Theorems 7 and 8 are from Gunawan et al. [17]. The near stability property was defined by Gambette et al. [12].
   Gambette et al. first established linear bounds on the number of reticulate nodes for reticulation-visible networks and nearly stable networks [12, 13]. Part (i) of Theorems 9, 10 and 11 appeared in [13, 17]. Part (ii) of Theorem 9 is from Bordewich and Semple [4], for which we present a different proof here.
   The first linear-time algorithm for the small cluster containment problem appears in [24, p. 171]. The linear-time algorithms for the cluster containment problems

in Sect. 12.5.6 were developed by Gambette et al. [13, 17]. Two different cubic-time algorithms for the tree containment problem were independently developed by Bordewich and Semple [4] and Gunawan et al. [17] for reticulation-visible networks. The algorithm of the latter was improved to linear-time algorithms for reticulation-visible networks, as well as for nearly stable networks [16, 41].

Acknowledgements The author thanks his collaborators Philippe Gambette, Anthony Labarre and Stéphane Vialette (Université Paris-Est) for initiating a 2-year project on rooted phylogenetic networks, which was financially supported by the Institute of France at Singapore and the National University of Singapore. This chapter is mainly based on the findings from the project. The author thanks Andreas D. M. Gunawan, who injected many ideas for studying reticulation-visible networks. The author also thanks Daniel Huson and an anonymous reviewer for useful comments on the first draft of this work.

References

1. Anaya, M., Anipchenko-Ulaj, O., Ashfaq, A., Chiu, J., Kaiser, M., Ohsawa, M.S., Owen, M., Pavlechko, E., John, K.S., Suleria, S., Thompson, K., Yap, C.: On determining if tree-based networks contain fixed trees. Bull. Math. Biol. **78**, 961–969 (2016)
2. Baroni, M., Semple, C., Steel, M.: A framework for representing reticulate evolution. Ann. Comb. **8**(4), 391–408 (2005)
3. Bondy, J.A., Murty, U.S.R.: Graph Theory. Springer, New York, USA (2008)
4. Bordewich, M., Semple, C.: Reticulation-visible networks. Adv. Appl. Math. **78**, 114–141 (2016)
5. Cardona, G., Llabrés, M., Rosselló, F., Valiente, G.: Metrics for phylogenetic networks I: Generalizations of the Robinson-Foulds metric. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(1), 46–61 (2009)
6. Cardona, G., Rossello, F., Valiente, G.: Comparison of tree-child phylogenetic networks. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(4), 552–569 (2009)
7. Döcker, J., van Iersel, L., Kelk, S., Linz, S.: Deciding the existence of a cherry-picking sequence is hard on two trees (2017). arXiv:1712.02965
8. Fontaine, M.C., Pease, J.B., Steele, A., Waterhouse, R.M., Neafsey, D.E., Sharakhov, I.V., Jiang, X., Hall, A.B., Catteruccia, F., Kakani, E., Mitchell, S.N., Wu, Y.C., Smith, H.A., Love, R.R., Lawniczak, M.K., Slotman, M.A., Emrich, S.J., Hahn, M.W., Besansky, N.J.: Extensive introgression in a malaria vector species complex revealed by phylogenomics. Science **347**(6217), 524–1258 (2015). https://doi.org/10.1126/science.1258524
9. Francis, A., Huber, K.T., Moulton, V.: Tree-based unrooted phylogenetic networks. Bull. Math. Biol. **80**(2), 404–416 (2018)
10. Francis, A., Semple, C., Steel, M.: New characterisations of tree-based networks and proximity measures. Adv. Appl. Math. **93**, 93–107 (2018)
11. Francis, A.R., Steel, M.: Which phylogenetic networks are merely trees with additional arcs? Syst. Biol. **64**(5), 768–777 (2015)
12. Gambette, P., Gunawan, A.D., Labarre, A., Vialette, S., Zhang, L.: Locating a tree in a phylogenetic network in quadratic time. In: Proceedings of the International Conference on Research in Computational Molecular Biology (RECOMB), pp. 96–107. Springer (2015)
13. Gambette, P., Gunawan, A.D., Labarre, A., Vialette, S., Zhang, L.: Solving the tree containment problem in linear time for nearly stable phylogenetic networks. Discrete Appl. Math. **246**, 62–79 (2018)
14. Georgiadis, L., Parotsidis, N.: Dominators in directed graphs: a survey of recent results, applications, and open problems. In: Proceedings of the International Symposium on Computing in Informatics and Mathematics, pp. 15–20. Epoka University, Albania (2013)

15. Gogarten, P.: Horizontal gene transfer: a new paradigm for biology. In: Proceedings of the Esalen Center for Theory and Research Conference (2000). http://archive.is/ZQVe
16. Gunawan, A.: Solving tree containment problem for reticulation-visible networks with optimal running time (2017). arXiv:1702.04088
17. Gunawan, A.D., DasGupta, B., Zhang, L.: A decomposition theorem and two algorithms for reticulation-visible networks. Inform. Comput. **252**, 161–175 (2017)
18. Gunawan, A.D., Lu, B., Zhang, L.: A program for verification of phylogenetic network models. Bioinformatics **32**(17), i503–i510 (2016)
19. Gusfield, D.: ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks. MIT Press (2014)
20. Gusfield, D., Eddhu, S., Langley, C.: The fine structure of galls in phylogenetic networks. INFORMS J. Comput. **16**(4), 459–469 (2004)
21. Hayamizu, M.: On the existence of infinitely many universal tree-based networks. J. Theoret. Biol. **396**, 204–206 (2016)
22. Huson, D.H., Klöpper, T.H.: Beyond galled trees-decomposition and computation of galled networks. In: Proceedings of the International Conference on Research in Computational Molecular Biology (RECOMB), pp. 211–225. Springer (2007)
23. Huson, D.H., Rupp, R., Berry, V., Gambette, P., Paul, C.: Computing galled networks from real data. Bioinformatics **25**(12), i85–i93 (2009)
24. Huson, D.H., Rupp, R., Scornavacca, C.: Phylogenetic Networks: Concepts, Algorithms and Applications. Cambridge University Press (2010)
25. van Iersel, L.: Different topological restrictions of rooted phylogenetic networks. Which make biological sense? (2013). http://phylonetworks.blogspot.sg/2013/03/different-topological-restrictions-of.html
26. van Iersel, L., Semple, C., Steel, M.: Locating a tree in a phylogenetic network. Inf. Process. Lett. **110**(23), 1037–1043 (2010)
27. Jetten, L., van Iersel, L.: Nonbinary tree-based phylogenetic networks. IEEE/ACM Trans. Comput. Biol. Bioinform. (2018)
28. Kanj, I.A., Nakhleh, L., Than, C., Xia, G.: Seeing the trees and their branches in the network is hard. Theoret. Comput. Sci. **401**(1–3), 153–164 (2008)
29. Lengauer, T., Tarjan, R.E.: A fast algorithm for finding dominators in a flowgraph. ACM Trans. Program. Lang. Syst. **1**(1), 121–141 (1979)
30. Linder, C.R., Moret, B.M., Nakhleh, L., Warnow, T.: Network (reticulate) evolution: biology, models, and algorithms. In: Proceedings of the Ninth Pacific Symposium on Biocomputing (PSB) (2004)
31. Lu, B., Zhang, L., Leong, H.W.: A program to compute the soft Robinson-Foulds distance between phylogenetic networks. BMC Genomics **18**(2), 111 (2017)
32. Marcussen, T., Sandve, S.R., Heier, L., Spannagl, M., Pfeifer, M., The International Wheat Genome Sequencing Consortium, Jakobsen, K.S., Wulff, B.B., Steuernagel, B., Mayer, K.F., Olsen, O.A.: Ancient hybridizations among the ancestral genomes of bread wheat. Science **345**(6194), 092–1250 (2014). https://doi.org/10.1126/science.1250092
33. Moran, N.A., Jarvik, T.: Lateral transfer of genes from fungi underlies carotenoid production in aphids. Science **328**(5978), 624–627 (2010)
34. Moser, R.A., Scheder, D.: A full derandomization of Schöning's k-SAT algorithm. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 245–252. ACM (2011)
35. Nakhleh, L., Wang, L.S.: Phylogenetic networks: properties and relationship to trees and clusters. In: Transactions on Computational Systems Biology II, pp. 82–99. Springer (2005)
36. Semple, C.: Phylogenetic networks with every embedded phylogenetic tree a base tree. Bull. Math. Biol. **78**(1), 132–137 (2016)
37. Steel, M.: Phylogeny: Discrete and Random Processes in Evolution. SIAM, Philadelphia, USA (2016)
38. Tarjan, R.: Finding dominators in directed graphs. SIAM J. Comput. **3**(1), 62–89 (1974)

39. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. J. Comput. Biol. **8**(1), 69–78 (2001)
40. Warnow, T.: Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation. Cambridge University Press, Cambridge, UK (2017)
41. Weller, M.: Linear-time tree containment in phylogenetic networks (2017). arXiv:1702.06364
42. Willson, S.: Regular networks can be uniquely constructed from their trees. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(3), 785–796 (2011)
43. Yan, H., Gunawan, A.D., Zhang, L.: S-Cluster++: a fast program for solving the cluster containment problem for phylogenetic networks. Bioinformatics **34**(17), i680–i686 (2018)
44. Zhang, L.: On tree-based phylogenetic networks. J. Comput. Biol. **23**(7), 553–565 (2016)

Chapter 13
Advances in Computational Methods for Phylogenetic Networks in the Presence of Hybridization

R. A. Leo Elworth, Huw A. Ogilvie, Jiafan Zhu and Luay Nakhleh

Abstract Phylogenetic networks extend phylogenetic trees to allow for modeling reticulate evolutionary processes such as hybridization. They take the shape of a rooted, directed, acyclic graph, and when parameterized with evolutionary parameters, such as divergence times and population sizes, they form a generative process of molecular sequence evolution. Early work on computational methods for phylogenetic network inference focused exclusively on reticulations and sought networks with the fewest number of reticulations to fit the data. As processes such as incomplete lineage sorting (ILS) could be at play concurrently with hybridization, work in the last decade has shifted to computational approaches for phylogenetic network inference in the presence of ILS. In such a short period, significant advances have been made on developing and implementing such computational approaches. In particular, parsimony, likelihood, and Bayesian methods have been devised for estimating phylogenetic networks and associated parameters using estimated gene trees as data. Use of those inference methods has been augmented with statistical tests for specific hypotheses of hybridization, like the *D*-statistic. Most recently, Bayesian approaches for inferring phylogenetic networks directly from sequence data were developed and implemented. In this chapter, we survey such advances and discuss model assumptions as well as methods' strengths and limitations. We also discuss parallel efforts in the population genetics community aimed at inferring similar structures. Finally, we highlight major directions for future research in this area.

Keywords Phylogenetic networks · Hybridization · Incomplete lineage sorting Multispecies coalescent · Maximum parsimony · Maximum likelihood · Bayesian inference

R. A. L. Elworth · H. A. Ogilvie · J. Zhu · L. Nakhleh (✉)
Rice University, Houston, USA
e-mail: nakhleh@rice.edu

R. A. L. Elworth
e-mail: r.a.leo.elworth@rice.edu

H. A. Ogilvie
e-mail: huw.a.ogilvie@rice.edu

J. Zhu
e-mail: jiafan.zhu@rice.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_13

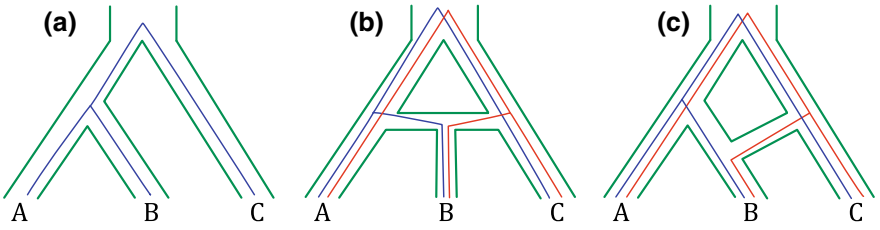


Fig. 13.1 Hybrid speciation and introgression. **a** A phylogenetic tree on three taxa, A, B, and C, and a gene tree within its branches. Genetic material is inherited from ancestors to descendants and it is expected that loci across the genome would have the shown gene tree. **b** A hybrid speciation scenario depicted by a phylogenetic network, where B is a hybrid population that is distinct from its parental species. Shown within the branches of the network are two gene trees, both of which are assumed to be very common across the genome. **c** An introgression scenario. Through hybridization and backcrossing, genetic material from (an ancestor of) C is incorporated into the genomes of individuals in (an ancestor of) B. The introgressed genetic material would have the gene tree shown in red, and the majority of loci in B's genomes would have the gene tree shown in blue. Incomplete lineage sorting would complicate all three scenarios by giving rise to loci with other possible gene trees and by changing the distribution of the various gene trees

13.1 Introduction

Hybridization is often defined as reproduction between members of genetically distinct populations [4]. This process could occur in various spatial contexts, and could have impacts on speciation and differentiation [1–3, 71–73, 96]. Furthermore, increasing evidence as to the adaptive role of hybridization has been documented, for example, in humans [91], macaques [6, 85, 108], mice [64, 104], butterflies [111, 133], and mosquitoes [30, 117].

Hybridization is "generically" used to contain two different processes: hybrid speciation and introgression [29]. In the case of hybrid speciation, a new population made of the hybrid individuals forms as a separate and distinct lineage from either of its two parental populations.[1] Introgression, or introgressive hybridization, on the other hand, describes the incorporation of genetic material into the genome of a population via interbreeding and backcrossing, yet without creating a new population [43]. As Harrison and Larson noted [43], introgression is a relative term: alleles at some loci introgress with respect to alleles at other loci within the same genomes. From a genomic perspective, and as the basis for detection of hybridization, the general view is that in the case of hybrid speciation, regions derived from either of the parental ancestries of a hybrid species would be common across the genomes, whereas in the case of introgression, regions derived from introgression would be rare across the genomes [29]. Figure 13.1 illustrates both hybridization scenarios.

[1] In this chapter, we do not make a distinction between *species*, *population*, or *subpopulation*. The modeling assumptions and algorithmic techniques underlying all the methods we describe here neither require nor make use of such a distinction.

A major caveat to the aforementioned general view is that, along with hybridization, other evolutionary processes could also be at play, which significantly complicates the identification of hybrid species and their parental ancestries. Chief among those processes are incomplete lineage sorting (ILS) and gene duplication and loss. Indeed, various studies have highlighted the importance of accounting for ILS when attempting to detect hybridization based on patterns of gene tree incongruence [16, 30, 74, 87, 91, 111, 117, 124, 133]. Furthermore, gene duplication and loss are very common across all branches of the Tree of Life. While the main focus of this chapter is on modeling and inferring hybridization, a discussion of how ILS is accounted for is also provided since recent developments have made great strides in modeling hybridization and ILS simultaneously. While signatures of gene duplication and loss are ubiquitous in genomic data sets, we do not include a discussion of these two processes in this chapter since methods that account for them in the context of phylogenetic networks are currently lacking. The chapter by El-Mabrouk and Noutahi in this volume discusses in extensive detail the evolution of gene families under gene duplication and loss, and problems that arise with respect to reconciling the evolutionary history of a gene family with that of a set of species.

When hybridization occurs, the evolutionary history of the set of species is best modeled by a *phylogenetic network*, which extends the phylogenetic tree model by allowing for "horizontal" edges to denote hybridization and to facilitate modeling bi-parental inheritance of genetic material. Figure 13.1 shows two phylogenetic networks that model hybrid speciation and introgression. It is very important to note, though, that from the perspective of existing models, both phylogenetic networks are topologically identical. This issue highlights two important issues that must be thought about carefully when interpreting a phylogenetic network. First, neither the phylogenetic network nor the method underlying its inference distinguish between hybrid speciation and introgression. This distinction is a matter of interpretation by the user. For example, the phylogenetic network in Fig. 13.1c could be redrawn, without changing the model or any of its properties, so that the introgression is from (an ancestor of) A to (an ancestor of) B, in which case the "red" gene tree would be expected to appear with much higher frequency than the "blue" gene tree. In other words, the way a phylogenetic network is drawn could convey different messages about the evolutionary history that is not inherent in the model or the inference methods. This issue was importantly highlighted with respect to data analysis in [117] (Fig. 13.7 therein). Second, the phylogenetic network does not by itself encode any specific backbone species tree that introgressions could be interpreted with respect to. This, too, is a matter of interpretation by the user. This is why, for example, Clark and Messer [16] recently argued that "perhaps we should dispense with the tree and acknowledge that these genomes are best described by a network." Furthermore, recent studies demonstrated the limitations of inferring a species tree "despite hybridization" [103, 136].

With the availability of data from multiple genomic regions, and increasingly often from whole genomes, a wide array of methods for inferring species trees, mainly based on the multispecies coalescent (MSC) model [19], have been developed [65, 67, 77]. Building on these methods, and often extending them in novel ways,

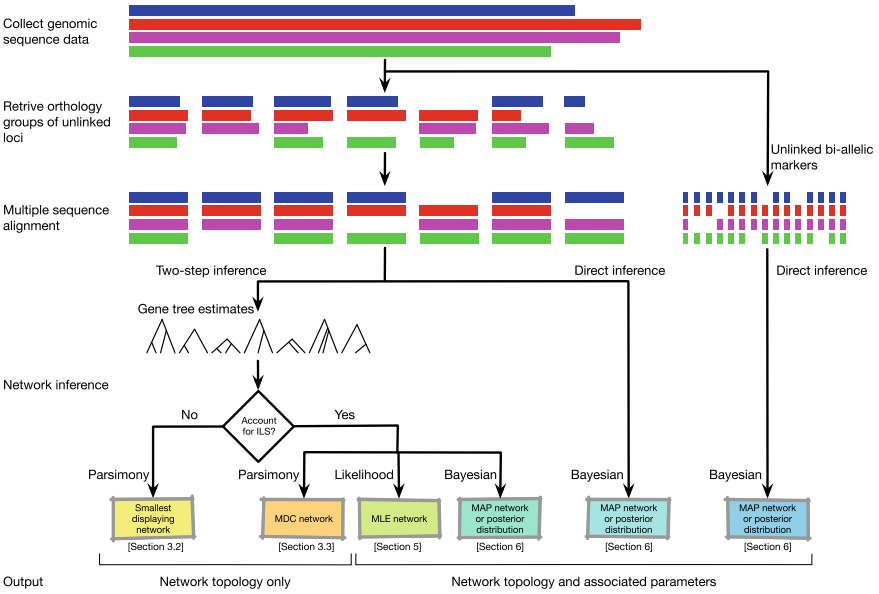


Fig. 13.2 Phylogenetic network inference process and approaches. The process of phylogenetic network inference starts with collecting the genomic data and identifying the orthology groups of unlinked loci. Multiple sequence alignments or single bi-allelic markers corresponding to the unlinked loci are then obtained; phylogenetic network inference methods use one of these two types of data. In two-step inference methods, gene trees are first estimated for the individual loci from the sequence alignment data, and these gene tree estimates are used as the input data for network inference. If incomplete lineage sorting (ILS) is not accounted for, a smallest displaying network of the gene tree estimates is sought. If ILS is accounted for, parsimony inference based on the minimizing deep coalescences (MDS) criterion, a maximum likelihood estimate (MLE), a maximum a posteriori (MAP) estimate, or samples of the posterior distribution can be obtained. In the direct inference approach, whether based on sequence alignment or bi-allelic marker data, a MAP estimate or samples of the posterior distribution can be obtained directly from the data. The two parsimony methods consider only the topologies of the gene tree estimates as input (i.e., they ignore gene tree branch lengths) and return as output phylogenetic network topologies. The likelihood and Bayesian methods that take gene tree estimates as input can operate on gene tree topologies alone or gene trees with branch lengths as well. Both methods estimate phylogenetic network topologies along with branch lengths (in coalescent units) and inheritance probabilities. The direct inference methods estimate the phylogenetic network along with its associated parameters

the development of computational methods for inferring phylogenetic networks from genome-wide data has made great strides in recent years. Figure 13.2 summarizes the general approaches that most phylogenetic network inference methods have followed in terms of the data they utilize, the model they employ, and the inferences they make.

The overarching goal of this chapter is to review the existing methods for inferring phylogenetic networks in the presence of hybridization,[2] describe their strengths and limitations, and highlight major directions for future research in this area.All the methods discussed hereafter make use of multi-locus data, where a locus in this context refers to a segment of genome present across the individuals and species sampled for a given study and related through common descent. A locus can be of varying length, coding or noncoding, and can be either functional or nonfunctional. Therefore, the use of the term "gene trees" is only historical; we use it to mean the evolutionary history of an individual locus, regardless of whether the locus overlaps with a coding region or not. Care must be taken with increasingly long loci spanning hundreds or thousands of contiguous basepairs, however, as many methods assume a locus has not been affected by recombination.

Multi-locus methods are fairly popular because the model fits several types of reduced representation genomic data sets commonly generated to study biological systems. Reduced representation refers to capturing many segments scattered throughout a genome, but only covering a fraction of the total genome sequence [35]. Reduced representation data sets which have been used with multi-locus methods include RAD-seq and genotyping by sequencing (GBS), which capture loci of roughly 100 bp associated with palindromic restriction enzyme recognition sites [28]. Another family of techniques often applied to studies of deeper time scales, sequence capture, extracts conserved sequences using probes complementary to targeted exons or ultraconserved elements [39]. Sequence capture can also be performed *in silico* when whole genomes are available [50].

The rest of the chapter is organized as follows. We begin in Sect. 13.2 by defining terminology for the nonbiologist, and give a very brief review of phylogenetic trees and their likelihood. In Sect. 13.3, we describe the earliest, and simplest from a modeling perspective, approaches to inferring parsimonious phylogenetic networks from gene tree topologies by utilizing their incongruence as the signal for hybridization. To account for ILS, we describe in Sect. 13.4 the multispecies network coalescent, or MSNC, which is the core model for developing statistical approaches to phylogenetic network inference while accounting for ILS simultaneously with hybridization. In Sects. 13.5 and 13.6 we describe the maximum likelihood and Bayesian methods for inferring phylogenetic networks from multi-locus data. In Sect. 13.7, we briefly discuss an approach aimed at detecting hybridization by using phylogenetic invariants. This approach does not explicitly build a phylogenetic network. In Sect. 13.8 we briefly discuss the efforts for developing methods for phylogenetic network inference that took place in parallel in the population genetics community (they are often referred to as "admixture graphs" in the population genetics literature). In Sect. 13.9, we summarize the available software for phylogenetic network inference, discuss the data that these methods use, and then list some of the limitations of these methods in practice. We conclude with final remarks and directions for future research in Sect. 13.10.

[2]We emphasize hybridization (in eukaryotic species) here since processes such as horizontal gene transfer in microbial organisms result in reticulate evolutionary histories, but the applicability of methods we describe in this chapter has not been investigated or explored in such a domain.

13.2 Background for Nonbiologists

In this section, we define the biological terminology used throughout the chapter so that it is accessible for nonbiologists. We also provide a brief review of phylogenetic trees and their likelihood, which is the basis for maximum likelihood and Bayesian inference of phylogenetic trees from molecular sequence data. Excellent books that cover mathematical and computational aspects of phylogenetic inference include [27, 32, 99, 107, 114].

13.2.1 Terminology

As we mentioned above, hybridization is reproduction between two members of genetically distinct populations, or species (Fig. 13.3). Diploid species (e.g., humans) have two copies of each genome. Aside from a few unusual organisms such as parthenogenic species, one copy will be maternal in origin and the other paternal. When the hybrid individual (or F_1) is also diploid, this process is called homoploid hybridization.

While each of the two copies of the genome in the hybrid individual traces its evolution back to precisely one of the two parents, this picture becomes much more complex after several rounds of recombination. Recombination is the swapping of a stretch of DNA between the two copies of the genome. Mathematically, if the two copies of the genome are given by strings u and v (for DNA, the alphabet for the strings is $\{A, C, T, G\}$), then recombination results in two strings $u' = u_1u_2u_3$ and $v' = v_1v_2v_3$, where u_1, u_2, u_3, v_1, v_2, and v_3 are all strings over the same alphabet, and $u_1v_2u_3 = u$ and $v_1u_2v_3 = v$; that is, substrings u_2 and v_2 were swapped. Observe that when this happens, u_1 and u_3 in the copy u' are inherited from one parent, and v_2, also in the copy u', is inherited from a different parent. A similar scenario happens in copy v' of the diploid genome.

This picture gets further complicated due to backcrossing, which is the mating between the hybrid individual, or one of its descendants, with an individual in one of the parental species. For example, consider a scenario in which descendants of the hybrid individual in Fig. 13.3 repeatedly mate with individuals from species A. After several generations, it is expected that the genomes of the hybrid individuals become more similar to the genomes of individuals in species A, and less similar to the genomes of individuals in species B (using the illustration of Fig. 13.3, the two copies of the genome would have much more red in them than blue).

Most models and methods for phylogenetic inference assume the two copies of a diploid genome are known separately and often only one of them is used to represent the corresponding individual. However, it is important to note that knowing the two copies separately is not a trivial task. Sequencing technologies produce data on both copies simultaneously, and separating them into their constituent copies is a well-studied computational problem known as genome phasing.

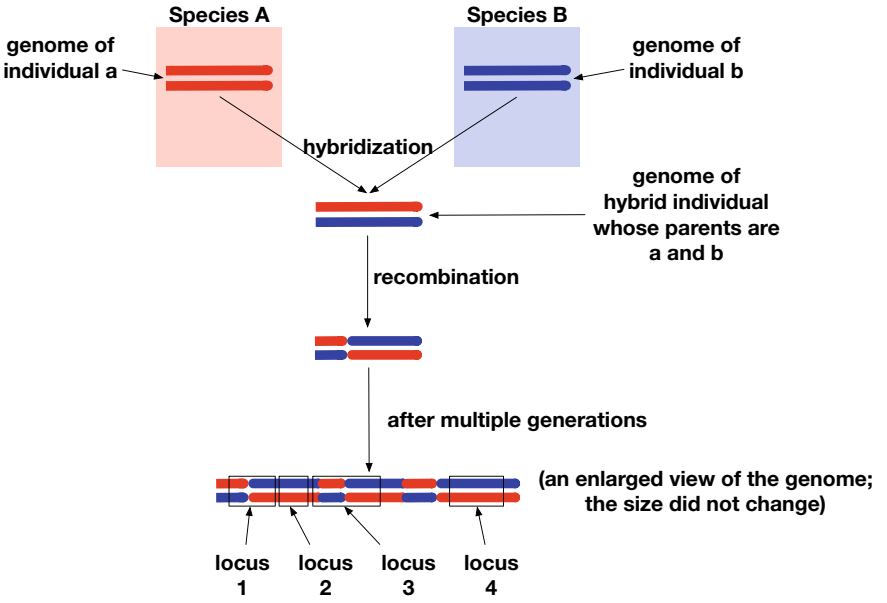


Fig. 13.3 Hybridization, recombination, and the generation of a mosaic genome. Diploid individual a from species A and individual b from species B mate, resulting in a diploid hybrid individual with one copy of its genome inherited from parent a and the other copy inherited from parent b. A recombination event results in the "swapping" of entire regions between two copies of the genome. After multiple generations in which more recombination happens, the genome becomes a mosaic. Walking across the genome from left to right, the color switches back and forth between red and blue, where switches happen at recombination breakpoints. Shown are four different loci. Loci 1 and 3 are not appropriate for tree inference since they span recombination breakpoints and, thus, include segments that have different evolutionary histories. Loci 2 and 4 are the "ideal" loci for analyses by methods described in this chapter

Biologists often focus on certain regions within the genomes for phylogenetic inference. If we consider the genome to be represented as a string w over the alphabet $\{A, C, T, G\}$, then a locus is simply a substring of w given by the start and end positions of the substring in w. The size of a locus can range anywhere from a single position in the genome to a (contiguous) stretch of 1 million or more positions in the genome. As we discussed above, when recombination happens, an individual copy of the genome would have segments with different ancestries (the blue and red regions in Fig. 13.3). A major assumption underlying phylogenetic tree inference is that the sequence data of a locus used for inference has evolved down a single tree. Therefore, the more recombination which has occurred within a locus over its evolutionary history (limited to the history connecting the species being studied), the less suitable it will be for phylogenetic inference. Conversely, loci with low recombination rates may be more suitable in terms of avoiding intra-locus recombination, although such loci are more susceptible to linked selection [42].

13.2.2 Phylogenetic Trees and Their Likelihood

An unrooted binary phylogenetic tree T on set $\mathscr{X}$ of taxa (e.g., $\mathscr{X} = \{humans, chimp, gorilla\}$) is a binary tree whose leaves are bijectively labeled by the elements of $\mathscr{X}$. That is, if $|\mathscr{X}| = n$, then T has n leaf nodes and $n - 2$ non-leaf (internal) nodes (each leaf node has degree 1 and each internal node has degree 3). A rooted binary phylogenetic tree is a directed binary tree with a single node designated as the root and all edges are directed away from the root. For n taxa, a rooted binary tree has n leaves and $n - 1$ internal nodes (each leaf node has in-degree 1 and out-degree 0; each internal node except for the root has in-degree 1 and out-degree 2; the root has in-degree 0 and out-degree 2).

Modern methods for phylogenetic tree inference make use of molecular sequence data, such as DNA sequences, obtained from individuals within the species of interest. The sequences are assumed to have evolved from a common ancestral sequence (we say the sequences are homologous) according to a model of evolution that specifies the rates at which the various mutational events could occur (Fig. 13.4).

For example, to infer a phylogenetic tree T on set $\mathscr{X} = \{X_1, \ldots, X_n\}$ of taxa, the sequence S_1 of a certain locus is obtained from the genome of an individual in species X_1, the sequence S_2 of a certain locus is obtained from the genome of an

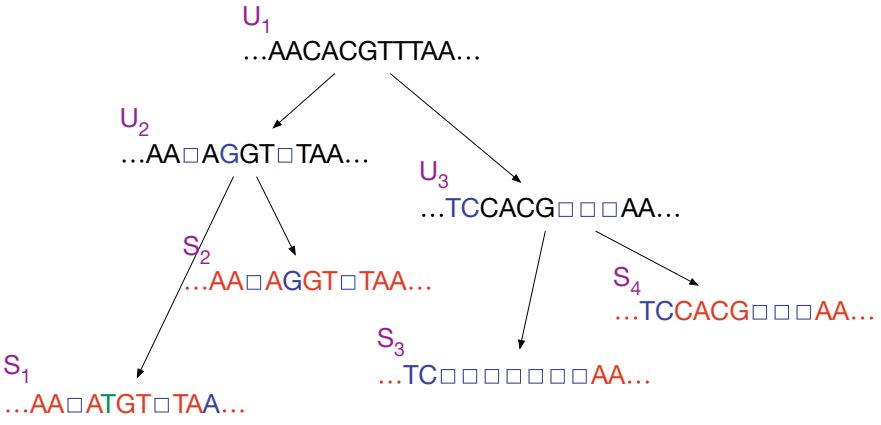


Fig. 13.4 Sequence evolution on a tree. At the top is the ancestral sequence for a certain locus in the genome of an individual. Through cell division and DNA replication, this sequence is inherited from parent to children. However, mutations could alter the inherited sequences. Boxes indicate letters that were deleted due to mutation. Letters in blue indicate substitutions (a mutation that alters the state of the nucleotide). The letter in green has mutated more than once during its evolutionary history. With respect to sequence U_1, sequence U_2 has two deletions at the third and eighth positions, and a substitution (C to G) at the fifth position. With respect to sequence U_1, sequence S_3 has 7 deletions at positions 3–9, and substitutions at the first two positions (A to T, and A to C). Sequences S_1, S_2, S_3, and S_4, with the boxes and colors unknown, are often the data for phylogenetic inference. That is, the four sequences used as data here are: AAATGTTAA, AAAGGTTAA, TCAA, and TCCACGAA

individual in species X_2, and so on until n sequences $S_1, \ldots, S_n$ are obtained. To perform phylogenetic tree inference, the n sequences must satisfy two important conditions (see Fig. 13.4):

- *The sequences are homologous*: The obtained sequences must have evolved down a single tree from a single sequence in an individual in an ancestral species. Two sequences are homologous if they evolved from a common ancestor, including in the presence of events such as duplication. Two homologous sequences are orthologs if they evolved from a common ancestor solely by means of DNA replication and speciation events. Two homologous sequences are paralogs if their common ancestor had duplicated to give rise to the two sequences.
- *The sequences are aligned*: While the obtained homologous "raw" sequences might be of different lengths due to events such as insertions and deletions, the sequences must be made to be the same length before phylogenetic inference is conducted so that positional homology is established. Intuitively, positional homology is the (evolutionary) correspondence among sites across the n sequences. That is, the sequences must be made of the same length so that the ith site in all of them had evolved from a single site in the sequence that is ancestral to all of them (Fig. 13.5).

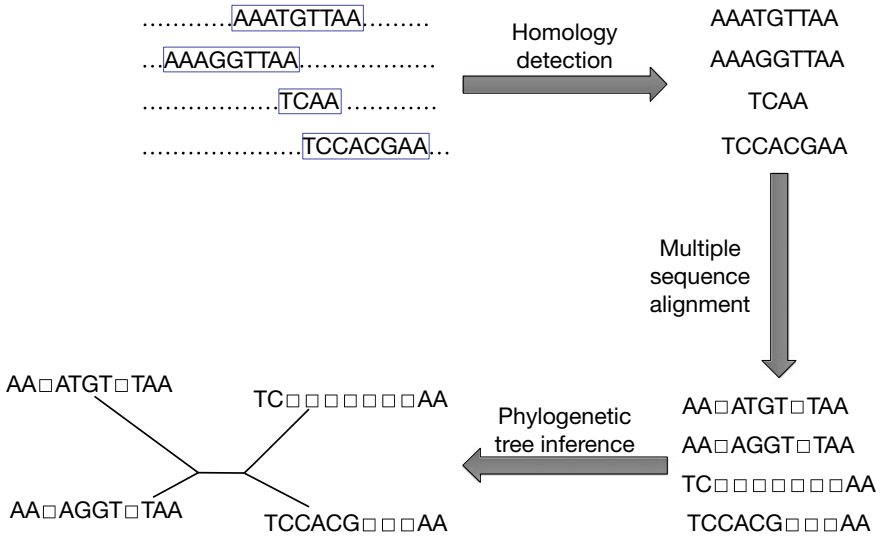


Fig. 13.5 From homologous sequences to a phylogenetic tree. Identifying homologous sequences across genomes is the first step toward a phylogenetic analysis. The homologous sequences, once identified, are not necessarily of the same length, due to insertions and deletions. Multiple sequence alignment is performed on the homologous sequences and the result is sequences of the same length where boxes indicate deleted nucleotides. Finally, a phylogenetic tree is constructed on the aligned sequences

Identifying homologous sequences across genomes is not an easy task; see, for example, [81] for a recent review of methods for homology detection. Multiple sequence alignment is also a hard computational problem, with a wide array of heuristics and computer programs currently available for it; see, for example, [13] for a recent review.

We are now in position to define a basic version of the Phylogeny Inference Problem:

Input: Set $S = \{S_1, \ldots, S_n\}$ of homologous sequences, where sequence S_i is obtained from taxon X_i, and the n sequences are aligned.
Output: A phylogenetic tree T on set $\mathscr{X}$ of taxa such that T is optimal, given the sequences, with respect to some criterion Φ.

The books we cited above give a great survey of the various criterion that Φ could take, as well as algorithms and heuristics for inferring optimal trees under the different criteria. Here, we focus on the main criterion in statistical phylogenetic inference, namely likelihood. We will make two assumptions when defining the likelihood that are (1) sites are identically and independently distributed and (2) following a DNA replication event, the two resulting sequences continue to evolve independently of each other.

To define the likelihood of a tree T, we first assign lengths $\lambda : E(T) \to \mathbb{R}^+$ to its branches, so that $\lambda(b)$ is the length of branch b in units of expected number of mutations per site per generation. Furthermore, we need a model of sequence evolution $\mathscr{M}$. Most models of sequence evolution are Markov processes where the probability of observing a sequence S at node u depends only on the sequence at u's parent, the length of the branch that links u to its parent, and the parameters of the model of sequence evolution. If we denote by $p_{uv}^{(i)}(t)$ the probability that the ith nucleotide in the sequence at node u evolves into the ith nucleotide in the sequence at node v over time t (measured in units of expected number of mutations as well), then the likelihood of a tree T and its branch lengths λ is

$$L(T, \lambda | S) = P(S|T, \lambda) = \prod_i \left(\sum_R \left(p(root^{(i)}) \cdot \prod_{b=(u,v)\in E(T)} p_{uv}^{(i)}(\lambda_b) \right) \right). \quad (13.1)$$

Here, the outer product is taken over all sites i in the sequences; i.e., if each of the n sequences is of length m, then $1 \leq i \leq m$. The summation is taken over R, which is the set of all possible labelings of the internal nodes of T with sequences of length m. Inside the summation, $p(root^{(i)})$ gives the stationary distribution of the nucleotides at position i. The likelihood as given by Eq. (13.1) is computed in polynomial time in m and n using Felsenstein's "pruning" algorithm [26].

Finally, the maximum likelihood estimate for solving the Phylogeny Inference Problem is given by

$$(T^*, \lambda^*) \leftarrow \text{argmax}_{(T,\lambda)} L(T, \lambda | S).$$

Computing the maximum likelihood estimate from a set S of sequences is NP-hard [15, 97]. However, much progress has been made in terms of developing heuristics that scale up to thousands of taxa while achieving high accuracy, e.g., [106].

13.3 From Humble Beginnings: Smallest Displaying Networks

Early work and, still, much effort in the community has focused on inferring the topology of a phylogenetic network from a set of gene tree topologies estimated for the individual loci in a data set. In this section, we discuss parsimony approaches to inferring phylogenetic network topologies from sets of gene trees.

13.3.1 The Topology of a Phylogenetic Network

As discussed above, a reticulate, i.e., non-treelike, evolutionary history that arises in the presence of processes such as hybridization and horizontal gene transfer is best represented by a phylogenetic network.

Definition 1 A *phylogenetic $\mathscr{X}$-network* (Fig. 13.6), or $\mathscr{X}$-network for short, Ψ is a rooted, directed, acyclic graph (rDAG) with set of nodes $V(\Psi) = \{r\} \cup V_L \cup V_T \cup V_N$, where

- $indeg(r) = 0$ (r is the *root* of Ψ);
- $\forall v \in V_L$, $indeg(v) = 1$ and $outdeg(v) = 0$ (V_L are the *external tree nodes*, or *leaves*, of Ψ);
- $\forall v \in V_T$, $indeg(v) = 1$ and $outdeg(v) \geq 2$ (V_T are the *internal tree nodes* of Ψ); and,
- $\forall v \in V_N$, $indeg(v) = 2$ and $outdeg(v) = 1$ (V_N are the *reticulation nodes* of Ψ).

For *binary* phylogenetic networks, the out-degree of the root and every internal tree node is 2. The network's set of edges, denoted by $E(\Psi) \subseteq V \times V$ is bipartitioned into *reticulation edges*, whose heads are reticulation nodes, and *tree edges*, whose heads are tree nodes (internal or external). Finally, the leaves of Ψ are bijectively labeled by the *leaf-labeling* function $\ell : V_L \to \mathscr{X}$.

13.3.2 Inferring Smallest Displaying Networks

Early work on phylogenetic networks focused on the problem of identifying a network with the fewest number of reticulation nodes that summarizes all gene trees in the input. More formally, let Ψ be a phylogenetic network. We say that Ψ *displays*

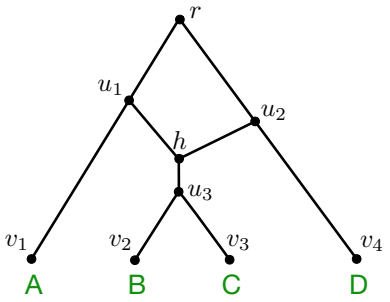


Fig. 13.6 An example of a phylogenetic network Ψ with a single reticulation event. This network is made up of leaf nodes $V_L = \{v_1, v_2, v_3, v_4\}$, internal tree nodes $V_T = \{u_1, u_2, u_3\}$, reticulation nodes $V_N = \{h\}$, and the root r. The nodes are connected by branches belonging to the set of phylogenetic network edges $E(\Psi)$. The branches are: (r, u_1), (r, u_2), (u_1, h), (u_2, h), (h, u_3), (u_1, v_1), (u_2, v_4), (u_3, v_2), and (u_3, v_3). The leaves are labeled by set $\mathscr{X} = \{A, B, C, D\}$ of taxa: $\ell(v_1) = A$, $\ell(v_2) = B$, $\ell(v_3) = C$, and $\ell(v_4) = D$

phylogenetic tree t if t can be obtained from Ψ by repeatedly applying the following operations until they are not applicable:

1. For a reticulation node h with two incoming edges $e_1 = (u_1, h)$ and $e_2 = (u_2, h)$, remove one of the two edges.
2. For a node u with a single parent v and a single child w, remove the two edges (v, u) and (u, w), and add edge (v, w).

The set of all trees displayed by the phylogenetic network is

$$\mathscr{T}(\Psi) = \{t : \ \Psi \text{ displays } t\}.$$

For example, for the phylogenetic network Ψ of Fig. 13.6, we have $\mathscr{T}(\Psi) = \{T_1, T_2\}$, where $T_1 = ((A, (B, C)), D)$ and $T_2 = (A, ((B, C), D))$.

Using this definition, the earliest phylogenetic network inference problem was defined as follows:

- **Input**: A set $\mathscr{G} = \{g_1, g_2, \ldots, g_m\}$ of gene trees, where g_i is a gene tree for locus i.
- **Output**: A phylogenetic network Ψ with the smallest number of reticulation nodes such that $\mathscr{G} \subseteq \mathscr{T}(\Psi)$.

This problem is NP-hard [113] and methods were developed for solving it and variations thereof, some of which are heuristics [86, 112, 120, 121]. Furthermore, the view of a phylogenetic network in terms of the set of trees it displays was used for pursuing other questions in this domain. For example, the topological difference between two networks could be quantified in terms of the topological differences among their displayed trees [80]. The parsimony and likelihood criteria were extended to the case of phylogenetic networks based on the assumption that each site (or, locus) has evolved down one of the trees displayed by the network [51–55, 78]. The concepts

of character compatibility and perfect phylogeny were also extended to phylogenetic networks based on the notion of displayed trees [58, 79, 113]. Furthermore, questions related to distinguishability of phylogenetic networks based on their displayed trees have been pursued [57] and relationships between networks and trees have been established in terms of this definition [31, 132]. The chapter by L. Zhang in this volume discusses in detail problems relating to phylogenetic networks and their sets of displayed trees and clusters.

However, the computational complexity of this problem notwithstanding, the problem formulation could be deficient with respect to practical applications. For one thing, solving the aforementioned problem only yields the topology of a phylogenetic network, but no other parameters. In practice, biologists would be interested in divergence times, population parameters, and some quantification of the amount of introgression in the genomes. These quantities are not recoverable under the given formulation. Moreover, for the biologist seeking to analyze her data with respect to hypotheses of reticulate evolutionary events, solving the aforementioned problem could result in misleading evolutionary scenarios for at least three reasons. First, the smallest number of reticulations required in a phylogenetic network to display all trees in the input could be arbitrarily far from the true (unknown) number of reticulations. One reason for this phenomenon is the occurrence of reticulations between sister taxa, which would not be detectable from gene tree topologies alone. Second, a smallest set of reticulations could be very different from the actual reticulation events that took place. Third, and probably most importantly, some or even all of gene tree incongruence in an empirical data set could have nothing to do with reticulation. For example, hidden paralogy and/or incomplete lineage sorting could also give rise to incongruence in gene trees. When such phenomena are at play, seeking a smallest phylogenetic network that displays all the trees in the input is the wrong approach and might result in an overly complex network that is very far from the true evolutionary history. To address all these issues, the community has shifted its attention in the last decade toward statistical approaches that view phylogenetic networks in terms of a probability distribution on gene trees that could encompass a variety of evolutionary processes, including incomplete lineage sorting.

13.3.3 Phylogenetic Networks as Summaries of Trees

Before we turn our attention to these statistical approaches, it is worth contrasting smallest phylogenetic networks that display all trees in the input to the concept of consensus trees. In the domain of phylogenetic trees, consensus trees have played an important role in compactly summarizing sets of trees. For example, the strict consensus tree contains only the clusters that are present in the input set of trees, and nothing else. The majority-rule consensus tree contains only the clusters that appear in at least 50% of the input trees. When there is incongruence in the set of trees, these consensus trees are most often nonbinary trees (contain "soft polytomies") such that each of the input trees can be obtained as a binary resolution of the consensus tree.

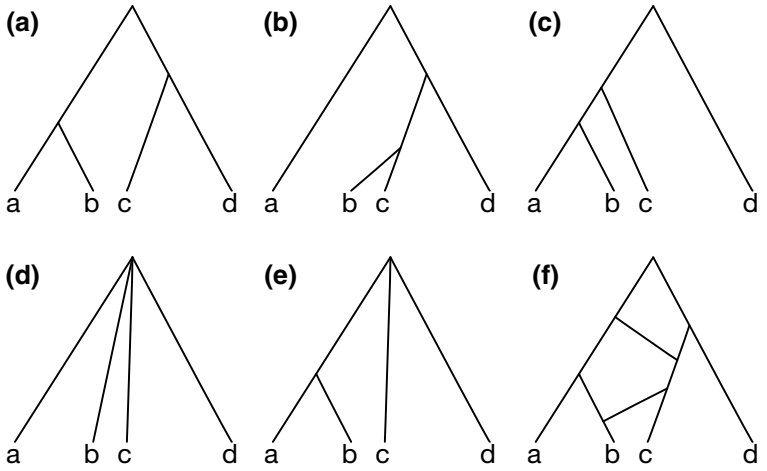


Fig. 13.7 Consensus trees and phylogenetic networks as two contrasting summary methods. **a**–**c** Three (input) gene trees whose summary is sought. **d** The strict consensus of the input trees. **e** The 70% majority-rule consensus of the input trees. **f** A smallest phylogenetic network that displays all three trees in the input. The strict consensus could be resolved to yield 15 different binary trees, only three of which are in the input. The majority-rule consensus tree could be resolved to yield three possible trees, two of which are the trees in (**a**) and (**d**), but the third, which is (((**a**, **b**), **d**), **c**), is not in the input. Furthermore, the tree in panel (**b**) is not included in the summary provided by the majority-rule consensus. The phylogenetic network displays four trees, three of which are the input trees, and the fourth is ((**a**, (**b**, **c**)), **d**), which is not in the input

Notice that while the consensus tree could be resolved to yield each tree in the input, there is no guarantee in most cases that it cannot also be resolved to generate trees that are not in the input. Smallest phylogenetic networks that display all trees in the input could also be viewed as summaries of the trees, but instead of removing clusters that are not present in some trees in the input, they display all clusters that are present in all trees in the input. Similarly to consensus trees, a smallest phylogenetic network could also display trees not in the input (which is the reason why we use $\subseteq$, rather than $=$, in the problem formulation above). These issues are illustrated in Fig. 13.7.

As discussed above, ILS is another process that could cause gene trees to be incongruent with each other and complicates the inference of phylogenetic networks since incongruence due to ILS should not induce additional reticulation nodes. Before we move on to discuss statistical approaches that account for ILS in a principled probabilistic manner under the coalescent, we describe an extension of the *minimizing deep coalescences*, or, MDC, criterion [69, 70, 109, 128], to phylogenetic networks, which was devised in [122].

13.3.4 A Step Toward More Complexity: Minimizing Deep Coalescences

Let Ψ be a phylogenetic network and consider node $u \in V(\Psi)$. We denote by $B_u \subseteq V(\Psi)$ the set of nodes in Ψ that are below node u (that is, the set of nodes that are reachable from the root of Ψ via at least one path that goes through node u).

Definition 2 A coalescent history of a gene tree g and a species (phylogenetic) network Ψ as a function $h : V(g) \rightarrow V(\Psi)$ such that the following conditions hold:

- if w is a leaf in g, then $h(w)$ is the leaf in Ψ with the same label (in the case of multiple alleles, $h(w)$ is the leaf in Ψ with the label of the species from which the allele labeling leaf w in g is sampled); and
- if w is a node in g_v, then $h(w)$ is a node in $B_{h(v)}$.

Given a phylogenetic network Ψ and a gene tree g, we denote by $H_\Psi(g)$ the set of all coalescent histories of gene tree g within the branches of phylogenetic network Ψ.

Given a coalescent history h, the *number of extra lineages* arising from h on a branch $b = (u, v)$ in phylogenetic network Ψ is the number of gene tree lineages exiting branch b from below node u toward the root, minus one. Finally, $XL(\Psi, h)$ is defined as the sum of the numbers of extra lineages arising from h on all branches $b \in E(\Psi)$.

Using coalescent histories, the minimum number of extra lineages required to reconcile gene tree g within the branches of Ψ, denoted by $XL(\Psi, g)$ is given by

$$XL(\Psi, g) = \min_{h \in H_\Psi(g)} XL(\Psi, h). \tag{13.2}$$

Under the MDC (minimizing deep coalescence) criterion, the optimal coalescent history refers to the one that results in the fewest number of extra lineages [69, 109], and thus,

$$XL(\Psi, g) = \sum_{e \in E(\Psi)} [k_e(g) - 1] \tag{13.3}$$

where $k_e(g)$ is the number of extra lineages on edge e of Ψ in the optimal coalescent history of gene tree g (Fig. 13.8).

A connection between extra lineages and the displayed trees of a phylogenetic network is given by the following observation.

Observation 1 *If gene tree g is displayed by phylogenetic network Ψ, then $XL(\Psi, g) = 0$.*

The implication of this observation is that if one seeks the phylogenetic network that minimizes the number of extra lineages, the problem can be trivially solvable by finding an overly complex network that displays every tree in the input. Therefore,

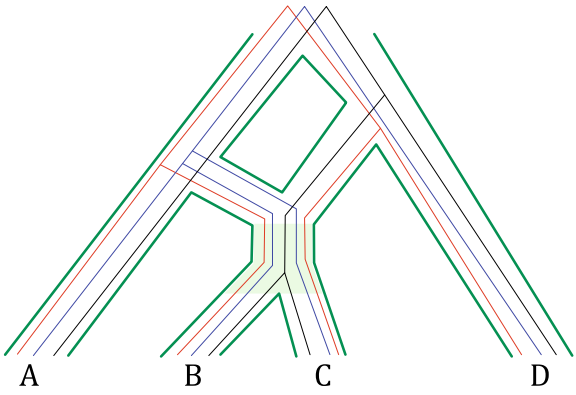


Fig. 13.8 The MDC criterion on phylogenetic networks. A phylogenetic network and coalescent histories within its branches of the three gene trees in Fig. 13.7a–c. The highlighted branch that separates the hybridization event from the MRCA of B and C has two extra lineages arising from the three shown coalescent histories. All other branches have 0 extra lineages. Therefore, the total number of extra lineages in this case is 2

inferring a phylogenetic network Ψ from a collection of gene tree topologies $\mathscr{G}$ based on the MDC criterion is more appropriately defined by

$$\hat{\Psi}(m) = \text{argmin}_{\Psi(m)} \left(\sum_{g \in \mathscr{G}} XL(\Psi(m), g) \right),$$

where we write $\Psi(m)$ to denote a phylogenetic network with m reticulation nodes. While the number of reticulations m is unknown and is often a quantity of interest, there is a trade-off between the number of reticulation nodes and number of extra lineages in a network: Reticulation edges can be added to reduce the number of extra lineages. Observing this reduction in the number of extra lineages could provide a mechanism to determine when to stop adding reticulations to the network [122].

13.4 Phylogenetic Networks: A Generative Model of Molecular Sequence Data

In the previous section, we focused on two parsimony formulations for inferring a phylogenetic network from a collection of input gene tree topologies: The first seeks a network with the fewest number of reticulations that displays each of the input gene trees, and the second seeks a network that does not have to display every gene tree in the input, but must minimize the number of "extra lineages" that could arise within a given number of reticulations. Both formulations result in phylogenetic network

topologies alone and make use of only the gene tree topologies. In this section, we introduce the multispecies network coalescent, or MSNC [118], as a generative process that extends the popular multispecies coalescent (MSC) model [19] that is the basis for most multi-locus species tree inference methods. The MSNC allows for the coalescent to operate within the branches of a phylogenetic network by viewing a set of populations—extant and ancestral—glued together by a rooted, directed, acyclic graph structure.

13.4.1 Parameterizing the Network's Topology

In addition to the topology of a phylogenetic network Ψ, as given by Definition 1 above, the nodes and edges are parameterized as follows.

Associated with the nodes are divergence/reticulation times, $\tau : V(\Psi) \rightarrow \mathbb{R}^+$, where $\tau(u)$ is the divergence time associated with tree node u and $\tau(v)$ is the reticulation time associated with reticulation node v. All leaf nodes u in the network have $\tau(u) = 0$. Furthermore, if u is on a path from the root of the network to a node v, then $\tau(u) \geq \tau(v)$.

Associated with the edges are population mutation rate parameters, $\theta : E(\Psi) \rightarrow \mathbb{R}^+$, where $\theta_b = 4N_b\mu$ is the population mutation rate associated with edge b, N_b is the effective population size associated with edge b, and μ is the mutation rate per site per generation.

Divergence times associated with nodes in the phylogenetic network could be measured in units of years, generations, or coalescent units. Branch lengths in gene trees are often given in units of expected number of mutations per site. The following rules are used to convert back and forth between these units:

- Given divergence time τ in units of expected number of mutations per site, mutation rate per site per generation μ and the number of generations per year g, $\tau/(\mu g)$ represents divergence times in units of years.
- Given population size parameter θ in units of the population mutation rate per site, $2\tau/\theta$ represents divergence times in coalescent units.

In addition to the divergence times and population size parameters, the reticulation edges of the network are associated with *inheritance probabilities*. For every reticulation node $u \in V_N$, let $left(u)$ and $right(u)$ be the "left" and "right" edges incoming into node u, respectively (which of the two edges is labeled left and which is labeled right is arbitrary). Let $E_R \subseteq E(\Psi)$ be the set of reticulation edges in the network. The inheritance probabilities are a function $\gamma : E_R \rightarrow [0, 1]$ such that for every reticulation node $u \in V_N$, $\gamma(left(u)) + \gamma(right(u)) = 1$. In the literature, γ is sometimes described as a vector Γ.

13.4.2 The Multispecies Network Coalescent and Gene Tree Distributions

As an orthologous, non-recombining genomic region from a set $\mathscr{X}$ of species evolves within the branches of the species phylogeny of $\mathscr{X}$, the genealogy of this region, also called the *gene tree*, can be viewed as a discrete random variable whose values are all possible gene tree topologies on the set of genomic regions. When the gene tree branch lengths are also taken into account, the random variable becomes continuous. Yu et al. [123] gave the probability mass function (pmf) for this discrete random variable given the phylogenetic network Ψ and an additional parameter Γ that contains the inheritance probabilities associated with reticulation nodes, which we now describe briefly.

The parameters Ψ and Γ specify the *multispecies network coalescent*, or MSNC (Fig. 13.9), and allow for a full derivation of the mass and density functions of gene trees when the evolutionary history of species involves both ILS and reticulation [123, 124]. This is a generalization of the *multispecies coalescent*, which describes the embedding and distribution of gene trees within a species tree without any reticulate nodes [19].

It is important to note that a single reticulation edge between two nodes does not mean a single hybridization event. Rather, a reticulation edge abstracts a continuous

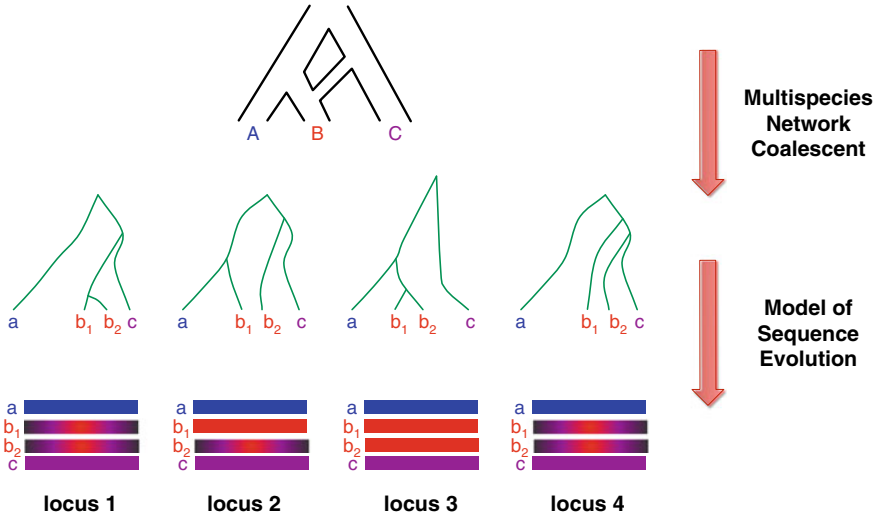


Fig. 13.9 Layers of the multispecies network coalescent model. A phylogenetic network describes the relationship between species (top). The MSNC describes the distribution of gene trees within the network, in which alleles from the same species can have different topologies and inheritance histories due to reticulation and/or ILS (middle). Some kind of mutation process occurs along the gene trees, resulting in observed differences between alleles in the present, which vary between genes based on their individual trees

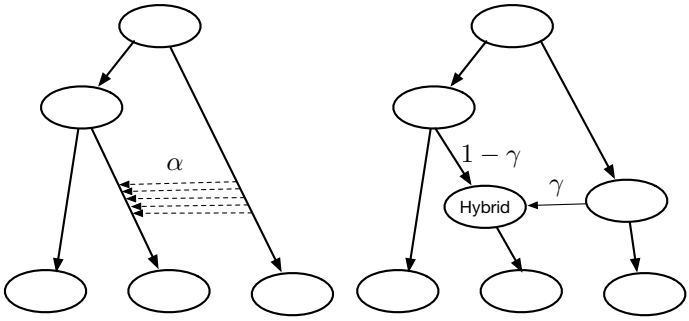


Fig. 13.10 Reticulation edges as abstractions of gene flow epochs. (Left) An epoch of gene flow from one population to another with migration rate α per generation. (Right) A phylogenetic network with a single reticulation edge that abstracts the gene flow epoch, with inheritance probability γ

epoch of repeated gene flow between the two species, as illustrated in Fig. 13.10. The two models shown in Fig. 13.10 were referred to as the "gene flow" model (left) and "intermixture" model (right) of hybridization in [68]. While the "gene flow" model is used by the IM family of methods [45] to incorporate admixture, the MSNC adopts the "intermixture" model. In this model, the γ inheritance probabilities indicate the ratio of genetic materials of a hybrid coming from its two parents. This means that unlinked loci from a hybrid species will have independent evolutionary histories, and will have evolved through the "left" or "right" parent with some probability γ and $1 - \gamma$, respectively. The performance of phylogenetic network inference on data simulated under the gene flow model was demonstrated in [116].

Wen and Nakhleh [116] derived the density function of the probability of gene trees given a phylogenetic network, with its topology, divergence/migration times, population mutation rates and inheritance probabilities. The divergence/migration times are in units of expected number of mutations per site, and population mutation rates are in units of population mutation rate per site. Based on the MSNC, and by integrating out all possible gene trees, Zhu et al. [135] developed an algorithm to compute the probability of a bi-allelic genetic marker given a phylogenetic network.

13.5 Maximum Likelihood Inference of Phylogenetic Networks

Phylogenetic networks are more complicated than a tree with some reticulation edges. The gene tree topology with the highest mass probability may not be one of the backbone trees of the network with four or more taxa [136]. Also, not all networks can be obtained by simply adding edges between the original edges of a tree [31].

Therefore inferring phylogenetic networks is not a trivial extension of methods to infer species trees. Most phylogenetic network methods [116, 118, 122, 124, 125,

131, 134, 135] sample from whole-network space rather than simply adding reticulations to a backbone tree. As such methods walk the space from one phylogenetic network to another, the point estimate or posterior distribution of networks is not tree-based, does not return or imply a backbone tree, and can include networks which cannot be described by merely adding reticulate branches between tree branches.

13.5.1 Inference

Sequential inference was initially developed to estimate species trees under the multispecies coalescent [66, 75], and in recent years has been extended to species networks [101, 118, 124–126]. These methods follow a two-step approach, where the first step is to estimate gene trees from multiple sequence alignments. The second step is to estimate a species tree or network from the distribution of estimated gene tree topologies.

As described above and illustrated in Fig. 13.9, two key requirements have facilitated the development of several methods for phylogenetic inference from multilocus data, including those that follow the two-step approach. One key requirement for current methods is that the segments are the result of speciation and not gene duplication, that is, sequences from different individuals and species are orthologs and not paralogs. Meeting this requirement ensures that the nodes in each gene tree represent coalescent events and can be fit to a coalescent model within each species network branch. A model which accounts for gene duplication and loss in addition to coalescent events has been developed to reconcile gene family trees with a fixed species tree [95, 130]. The most recent implementation of this model can also use it to estimate the species tree [21], but this model has yet to be extended to work with species networks.

A second key requirement is that the evolutionary history of the locus can be accurately modeled using a single tree. Recombination or gene fusion should not have occurred within a locus, otherwise, a gene network would be required to model that locus, breaking the MSNC model of gene trees within a species network [127]. Because of this requirement, multi-locus methods should be used with short contiguous sequences. The results of a previous study on mammal phylogenetics suggest that individual exons are an appropriate target sequence [98].

Under these two key requirements, each gene tree is considered to be a valid and independent sample from an underlying distribution of gene trees conditional on some unobserved species network. Of course, this assumption in sequential inference is violated as the gene trees are only estimates. Particular methods may be more or less sensitive to gene tree estimation errors. For species trees, methods which infer unrooted species trees (e.g., ASTRAL [75]) appear to be more robust relative to methods which infer rooted species trees (e.g., MP-EST [66]). This is because unrooted methods take unrooted gene trees as input and do not rely on correct rooting of the gene trees [100].

An estimate $\hat{g}$ of the true gene tree is typically made using phylogenetic likelihood (see Sect. 13.2). The phylogenetic likelihood of the sequence alignments can be combined sequentially or simultaneously with the MSNC probability densities of the gene trees to estimate a species network from sequence data.

Given gene trees where each node represents a coalescent event, the probability densities, and masses of those gene trees given a species network can be calculated [124]. This can be based on the topologies and node heights of the gene trees, or based on the topologies alone (see Sect. 13.4.2).

Maximum likelihood (ML) methods seek a phylogenetic network (along with its parameters) that maximizes some likelihood function. In a coalescent context, these methods search for the species network which maximizes the likelihood of observing a sample of gene trees given the proposed species network. The sample of gene trees can include branch lengths, in which case the likelihood is derived from the time intervals between successive coalescent events [124]. In the absence of branch lengths, the likelihood is derived from the probability mass of each gene tree topology [123, 124]. This probability is marginalized over every coalescent history h, which is all the ways for a gene tree to follow the reticulate branching of the network:

$$P(g|\Psi, \Gamma, \theta) = \sum_{h \in H_\Psi(g)} P(h|\Psi, \Gamma, \theta) \tag{13.4}$$

and the ML species network is therefore:

$$\hat{\Psi} = \text{argmax}_\Psi \prod_{g \in \mathcal{G}} P(g|\Psi, \Gamma, \theta). \tag{13.5}$$

ML inference of species networks has been implemented as the `InferNetwork_ML` command in PhyloNet [124], which identifies the ML species network up to a maximum number of reticulations.

Similar to our discussion of the MDC criterion above, absent any explicit stopping criterion or a penalty term in the likelihood function, obtaining an ML estimate according to Eq. (13.5) can result in overly complex phylogenetic networks since adding more reticulations often improve the likelihood of the resulting network. Therefore, it is important to parameterize the search by the number of reticulations sought, m, and solve

$$\hat{\Psi}(m) = \text{argmax}_{\Psi(m)} \prod_{g \in \mathcal{G}} P(g|\Psi(m), \Gamma, \theta), \tag{13.6}$$

where the value m is experimented with by observing the improvement in the likelihood for varying values of m (for example, maximum likelihood inference of phylogenetic networks in PhyloNet implements information criteria, such as AIC and BIC, for this purpose [124]).

The computational cost of the likelihood calculation increases with larger species networks and gene trees. Not only does this increase the number of branches and coalescent times, but as more reticulations are added many more possible coalescent histories exist to be summed over. Even with one reticulation edge attached onto a tree, the difficulty of the problem is exponential to tree cases. The computational complexity of the likelihood calculation is highly related to the size of the set of all coalescent histories of a gene tree conciliated in a network. Zhu et al. [134] proposed an algorithm to compute the number of coalescent histories of a gene tree for a network, and demonstrated that the number can grow exponentially after adding merely one reticulation edge to a species tree.

To show how running time of likelihood computation varies in a network with a single reticulation, we generated 150 random 1-reticulation networks with 5 taxa, then simulated 10,000 bi-allelic markers with 4 individuals per species. When a reticulation node exists in a phylogenetic network, this will induce a cycle in the unrooted equivalent of the acyclic rooted network. The "diameter" is the length of that cycle, and we ran the likelihood computation in [135] and summarized the maximum running time according to values of diameter and number of taxa under reticulation. Figure 13.11 shows that the complexity of the likelihood computation is highly related to the structure of the network. The running time in the worst case is hundreds of times slower than that of the best case.

A faster way to estimate a species network is to calculate a pseudo-likelihood instead of a full likelihood. The `InferNetwork_MPL` command in PhyloNet implements a maximum pseudo-likelihood (MPL) method for species networks.

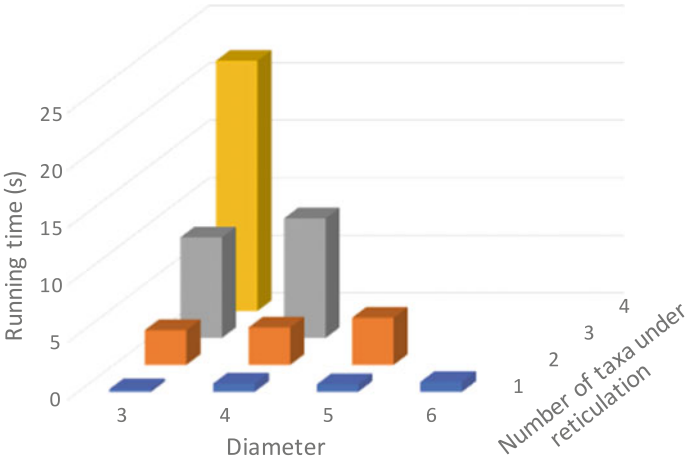


Fig. 13.11 Running time of computing the likelihood of a phylogenetic network given gene tree topologies. 150 1-reticulation phylogenetic networks with 5 species and 4 individuals per species were used, and the data consisted of 10,000 bi-allelic markers. The networks varied in terms of the diameter of a reticulation node (the number of edges on the cycle in the underlying undirected graph) and the number of taxa (leaves) under the reticulation nodes

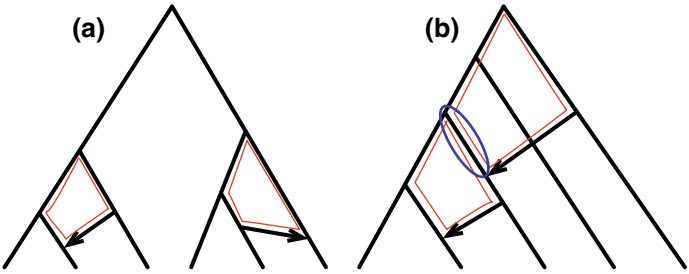


Fig. 13.12 Level-1 network definition. Reticulation nodes induce cycles in the (undirected graphs underlying the) phylogenetic networks. The edges of the cycles are highlighted with red lines. **a** A level-1 network is one where no edge of the network is shared by two or more cycles. **b** A non-level-1 network is one where at least one edge is shared by at least two cycles (the shared edge in this case is the one inside the blue circle)

This method is based on rooted triples, which is akin to the MP-EST method for species tree inference [66, 125].

Unlike phylogenetic trees, a given phylogenetic network is not necessarily uniquely distinguished by its induced set of rooted triples. Therefore this method cannot distinguish the correct network when other networks induce the same sets of rooted triples [125]. However, it is much more scalable than ML methods in terms of the number of taxa [44].

Another MPL method, SNaQ, is available as part of the PhyloNetworks software [101, 102]. SNaQ is based on unrooted quartets, akin to the ASTRAL method for species tree inference [75]. It is even more scalable than `InferNetwork_MPL` [44], but can only infer level-1 networks (Fig. 13.12).

13.6 Bayesian Inference of Phylogenetic Networks

Maximum likelihood estimation of phylogenetic networks, as described in the previous section, has three main limitations:

- As discussed, without penalizing the complexity of the phylogenetic network, the ML estimate could be an overly complex network with many false reticulations.
- The inference results in a single point estimate that does not allow for assessing confidence in the inferred network.
- The formulation does not allow for making use of the sequence data directly, but is based on gene tree estimates.

One way of addressing these limitations is to adopt Bayesian inference where an estimate of the posterior distribution on networks is sought directly from the sequence data of the individual loci, and where the prior distribution on phylogenetic networks accounts for model complexity in a principled manner.

Before we describe the work on Bayesian inference, it is important to note that while maximum likelihood estimation is not satisfactory, we cannot say that Bayesian estimation is without challenges. Such methods like [116, 131, 135] are based on reversible-jump MCMC [36] with varying numbers of parameters. Mixing problems arise when they involve dimension changing moves: adding a reticulation and removing a reticulation. This is because while walking over the space of phylogenetic networks, these methods jump between probability spaces of different models. Therefore moves should be carefully designed to account for mixing issues.

13.6.1 Probability Distributions Over Species Networks

It is useful to define probability distributions over species trees or networks without reference to sequence data or gene trees. Among other uses, these probability distributions can be applied as prior distributions in Bayesian inference. The two most common types of prior distributions used for species trees are birth–death tree priors and compound priors. Both types have been extended to create probability distributions over species networks.

As their name implies, birth–death tree priors combine a rate of *birth* with a rate of *death*. These are the rates at which one lineage splits into two, and one lineage ceases to exist respectively [34]. In the context of species trees these rates are more informatively called *speciation* and *extinction*. When the extinction rate is set to zero, this is known as a Yule prior [129]. Birth–death tree priors have been extended to support incomplete sampling in the present, and sampling-through-time [105]. A birth–death prior for species networks has been developed, called the birth-hybridization prior. This prior combines a rate of birth (or speciation) with a rate of hybridization, which is the rate at which two lineages merge into one. This model does not include a rate of extinction [131].

All birth–death tree priors induce a uniform probability distribution over ranked tree topologies, regardless of the rates of speciation and extinction. This means that birth–death priors favor symmetric over asymmetric trees, as symmetric trees have more possible ranked histories. Empirical trees generally have more asymmetric shapes than predicted by birth–death models [41]. For the birth-hybridization prior the probability distribution over network topologies is not invariant to the hybridization rate, which when set to zero reduces to the Yule model, and any topology containing a reticulation will have zero probability.

All birth–death priors are generative, as is the birth-hybridization prior. This means that not only can these distributions be used as priors for Bayesian inference, but they can be used to simulate trees and networks. These simulated distributions can then be used for ABC inference, which is used for models which are difficult to implement using MCMC. They can also be used for posterior predictive checks, which is an absolute measure of model goodness of fit [33].

While birth–death priors induce a probability distribution over topologies and branch lengths, compound tree and network priors are constructed from separate

distributions on both. Typically compound tree priors combine a uniform distribution over unranked tree topologies, favoring more asymmetric trees. Empirical trees generally have more symmetric shapes than predicted by this distribution [41]. Then a continuous distribution such as gamma can be applied to branch lengths or node heights. Compound priors are used for network inference by adding a third distribution describing the number of reticulations [116]. A Poisson distribution is a natural fit for this parameter as it describes a probability on nonnegative integers. The probability distribution for each network topology can still be uniform for all networks given k reticulations.

Unlike birth–death priors, compound priors are not generative, so it is not straightforward to simulate trees or networks from those distributions. The most obvious way to simulate such trees and networks would be running an existing Markov Chain Monte Carlo sampler without any data, and subsampling states from the chain at a low enough frequency to ensure independence between samples.

13.6.2 Sampling the Posterior Distribution

The ML species network with $k + 1$ reticulations will always have a higher likelihood than the ML network with k reticulations. For this reason, some threshold of significance must be applied to estimate the number of reticulations. This threshold may be arbitrary or it may be theoretically based, for example, the Akaike information criterion (AIC) and Bayesian information criterion (BIC) measures of relative fit [10].

In contrast, Bayesian methods of species network inference are able to naturally model the probability distribution over species networks including the number of reticulations by using a prior (see Sect. 13.6.1). In a Bayesian model, the posterior probability of a species network $P(\Psi)$ is proportional to the likelihood of the gene trees $P(G|\Psi, \Gamma, \theta)$, multiplied by the prior on the network and other parameters of the model $P(\Psi, \Gamma, \theta)$, and marginalized over all possible values of Γ and θ:

$$P(\Psi) \propto \iint P(G|\Psi, \theta) \cdot P(\Psi, \Gamma, \theta)\, \mathrm{d}\Gamma\, \mathrm{d}\theta. \tag{13.7}$$

When a decaying prior is used on the number of reticulations or on the rate of hybridization, the prior probability of species networks with large numbers of reticulations will be very low, and so will the posterior probability (Fig. 13.13).

Bayesian methods for phylogenetic inference typically use the Metropolis–Hastings Markov Chain Monte Carlo (MCMC) algorithm to estimate the posterior distribution of trees or networks. MCMC is a random walk where each step depends on the previous state, and it is flexible enough to be used for implementing extremely complex models such as species network inference with *relative* ease. Bayesian estimation of species networks from gene trees is implemented in the PhyloNet command `MCMC_GT`.

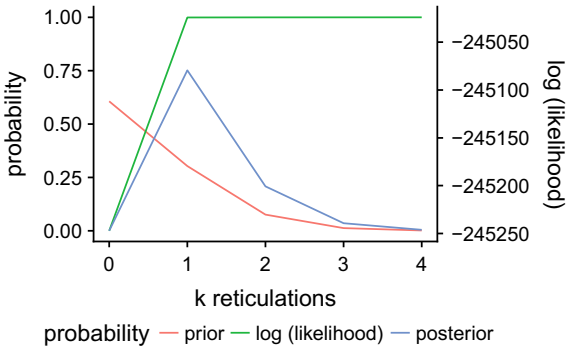


Fig. 13.13 Bayesian inference of the number of reticulations. In this example the topology of the network Ψ is fixed, the true number of reticulations is 1, and the likelihood is calculated for a topology with no reticulations, the true reticulation, and additional reticulations, with maximum likelihood branch lengths. The posterior probability was normalized to sum to 1, although as this is not integrated over branch lengths, the typical Bayesian posterior probability might be a bit different

The posterior probability of a species network is equal to the integral in Eq. 13.7 multiplied by a normalizing constant Z known as the marginal likelihood. In the case of sequential multi-locus inference, this constant is equal to $Z = P(G)^{-1}$ The marginal likelihood is usually intractable to calculate, but MCMC sidesteps this calculation by sampling topologies and other parameters with frequencies proportional to their probability mass or density. The posterior probability of a species network Ψ can, therefore, be approximated as the proportion of steps in the MCMC chain where the network topology Ψ_i at the end of the step i is equal to Ψ.

The value of any particular parameter, for example an inheritance probability γ for a given reticulation node v, can be estimated by averaging its value over the set of X steps where the state includes that parameter. In this case it is averaged over the states where the species network includes that node, i.e., the set $X = \{i : v \in \Psi_i\}$:

$$E(\gamma) = \frac{1}{|X|} \sum_{X}^{i} \gamma_i \tag{13.8}$$

Bayesian inference has also enabled the inference of species trees and networks directly from multi-locus sequence data. Instead of first estimating individual gene trees from multiple sequence alignments, these methods jointly infer the gene trees and species network using an application of Bayes' rule:

$$P(G, \Psi) \propto \iint P(D|G) \cdot P(G|\Psi, \Gamma, \theta) \cdot P(\Psi, \Gamma, \theta)\, \mathrm{d}\Gamma\, \mathrm{d}\theta. \tag{13.9}$$

Here, $P(D|G)$ is the likelihood of the data overall gene trees. In practice, this is the sum of phylogenetic likelihoods $\sum_i P(d_i|g_i)$ for every sequence alignment d

and associated gene tree g. As with sequential Bayesian inference of species trees and networks, the use of MCMC avoids the calculation of the marginal likelihood, which for joint inference can be expressed as $Z = P(D)^{-1}$. Joint Bayesian inference was first developed for species trees, and now has several popular implementations including StarBEAST2 [83] and BPP [93].

Joint Bayesian inference of species networks has been implemented independently as the PhyloNet command `MCMC_SEQ`, and as the BEAST2 package "SpeciesNetwork" [116, 131]. These two methods are broadly similar in their model and implementation, with a few notable differences. `MCMC_SEQ` uses a compound prior to the species network, whereas SpeciesNetwork has a birth-hybridization prior (see Sect. 13.6.1). SpeciesNetwork is able to use any of the protein and nucleotide substitution models available in BEAST2. `MCMC_SEQ` can be used with any nested GTR model but with fixed rates and base frequencies. So the rates (e.g., the transition/transversion ratio for HKY) must be estimated before running the analysis, or Jukes-Cantor is used where all rates and base frequencies are equal.

13.6.3 *Inference Under MSC Versus MSNC When Hybridization Is Present*

We simulated 128 loci on the phylogenetic network of Fig. 13.14a. The program `ms` [47] was used to simulate 128 gene trees on the network, and each gene tree was used to simulate a sequence alignment of 500 sites using the program Seq-Gen [92] under the GTR model and $\theta = 0.036$ for the population mutation rate. The exact command used was:

```
seq-gen -mgtr -s0.018 -f0.2112, 0.2888, 0.2896, 0.2104 -r0.2173,
0.9798, 0.2575, 0.1038, 1, 0.2070 -l500
```

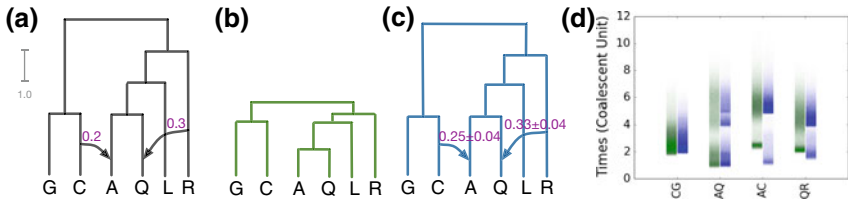


Fig. 13.14 Inference under the MSC and MSNC when the evolutionary history involves hybridization. **a** The true phylogenetic network with the shown inheritance probabilities and branch lengths (in coalescent units). **b** The MPP (maximum a posteriori probability) species tree estimated under the MSC by StarBEAST (frequency of 94% in the 95% credible set) with the average divergence times. **c** The MPP phylogenetic network along with the inheritance probabilities estimated under the MSNC by `MCMC_SEQ` [116] (the only network topology in the 95% credible set). The scale bar of divergence times represents 1 coalescent unit for (**a**–**c**). **d** The coalescent times of the MRCAs of (C, G), (A, Q), (A, C), (Q, R) from co-estimated gene trees inferred by StarBEAST (green) and `MCMC_SEQ` (blue)

We then ran both StarBEAST and `MCMC_SEQ`, as inference methods under the MSC and MSNC models, respectively, for 6×10^7 iterations each. The results are shown in Fig. 13.14.

A few observations are in order. First, while StarBEAST is not designed to deal with hybridization, it inferred the tree topology (Fig. 13.14b) that is obtainable by removing the two hybridization events (the two arrows) from the true phylogenetic network (the backbone tree). Second, `MCMC_SEQ` identified the true phylogenetic network as the one with the highest posterior (Fig. 13.14c). Furthermore, the estimated inheritance probabilities are very close to the true ones. Third, and most interestingly, since StarBEAST does not account for hybridization, it accounts for all heterogeneity across loci as being caused by incomplete lineage sorting (ILS) by underestimating all branch lengths (that is, "squashing" the divergence times so as to explain the heterogeneity by ILS). Indeed, Fig. 13.14d shows that the minimum coalescent times of the co-estimated gene trees by StarBEAST force the divergence times in the inferred species tree to be very low. `MCMC_SEQ`, on the other hand, accurately estimates the branch lengths of the inferred phylogenetic network since networks differentiate between divergence and hybridization times. For example, Fig. 13.14d shows that the coalescent times of clade (C, G) across all co-estimated gene trees is a continuum with a minimum value around 2, which defines the divergence time of these two taxa in the phylogenetic network. `MCMC_SEQ` clearly identifies two groups of coalescent times for each of the two clades (A, C) and (Q, R): The lower group of coalescent times corresponds to hybridization, while the upper group of coalescent times correspond to the coalescences above the respective MRCAs of the clades. We also note that the minimum value of coalescent times corresponding to (Q, R) is larger than that corresponding to (A, Q), which correctly reflects the fact that hybridization from R to Q happened before hybridization from C to A, as indicated in the true phylogenetic network. Finally, for clade (A, Q), three groups of coalescence times are identified by `MCMC_SEQ`, which makes sense since there are three common ancestors of A and Q in the network: at the MRCA of (A, Q) in the case of no hybridization involving either of the two taxa, at the MRCA of (A, Q, L, R) in the case of the hybridization involving Q, and at the root of the network in the case of the hybridization involving A. More thorough analysis and comparison of inferences under the MSC and MSNC can be found in [116].

These results illustrate the power of using a phylogenetic network inference method when hybridization is involved. In particular, if hybridization had occurred, and the practitioner did not suspect it and ran StarBEAST instead, they would get wrong inferences. In this case, the errors all have to do with the divergence time estimates. However, the topology of the inferred tree could be wrong as well, depending on the hybridization scenarios.

13.7 Phylogenetic Invariants Methods

The focus of this chapter up to this point has largely been on the MSC and MSNC models. A parallel effort has been led to detect reticulate evolution by using the notion of *phylogenetic invariants* [12, 62]. Phylogenetic invariants are polynomial relationships satisfied by frequencies of site patterns at the taxa labeling the leaves of a phylogenetic tree (and given a model of sequence evolution). Invariants that are predictive of particular tree topologies could then be used for inferring the tree topology by focusing on the space of site patterns rather than the space of tree topologies [27]. As Felsenstein wrote in his book, "invariants are worth attention, not for what they do for us now, but what they might lead to in the future." With the availability of whole-genome data and, consequently, the ability to obtain better estimates of site frequencies, the future is here. Indeed, methods like SVDQuartets [14] use phylogenetic invariants to estimate species trees under the MSC model.

A detailed discussion of phylogenetic invariants, in general, is beyond the scope of this manuscript. Interested readers should consult the excellent exposition on the subject in Felsenstein's seminal book (Chapter 22 in [27]). In this section, we briefly review phylogenetic invariants-based methods for detecting reticulation, starting with the most commonly used one, known as the D-statistic or the "ABBA-BABA" test.

The D-Statistic [22] is a widely known and frequently applied statistical test for inferring reticulate evolution events. The power of the test to infer reticulate evolution derives from the likelihood calculations of the MSNC. Despite this, the test itself is simple to calculate and formalize. The D-Statistic is given by

$$\frac{N_{ABBA} - N_{BABA}}{N_{ABBA} + N_{BABA}} \tag{13.10}$$

To calculate these quantities, we are given as input the four-taxon tree including outgroup of Fig. 13.15 and a sequence alignment of the genomes of P1, P2, P3, and O. Given this alignment, N_{ABBA} is calculated as the number of occurrences of single sites in the alignment where P1 and O have the same letter and P2 and P3 have the same letter, but these two letters are not the same, i.e., CTTC or GCCG. Similarly, N_{BABA} can be calculated as the number of occurrences in the alignment where the letters of $P1 = P3$ and $P2 = O$ with no other equalities between letters.

Upon calculating the D-Statistic, a significant deviation away from a value of 0 gives evidence for reticulate evolution. As shown in Fig. 13.15, a strong positive value implies introgression between P2 and P3 while a strong negative value implies introgression between P1 and P3. No such conclusions can be made from a D value very close to 0.

The crux of the theory behind the D-Statistic lies in the expectation of the probabilities of discordant gene trees given the overall phylogeny of Fig. 13.15. If we remove the two reticulation events in Fig. 13.15, we end up with a species tree Ψ. Given Ψ, the two gene trees whose topologies disagree with that of the species tree are equally probable under the MSC. On the other hand, when a reticulation between

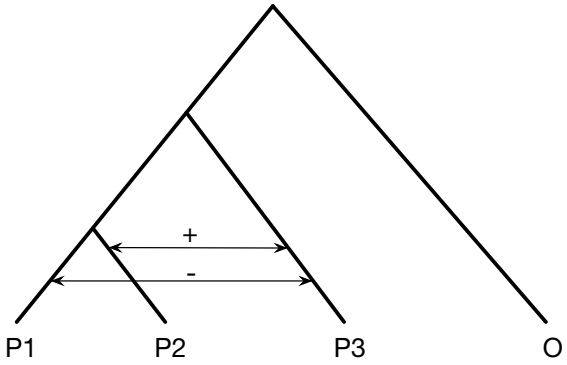


Fig. 13.15 The four-taxon tree topology used for the *D*-statistic. Significant deviations away from a value of 0 of the *D*-statistic (Eq. (13.10)) support introgression between P3 and either P1 or P2. As shown, a significant positive value supports introgression between P2 and P3. A significant negative value supports introgression between P1 and P3

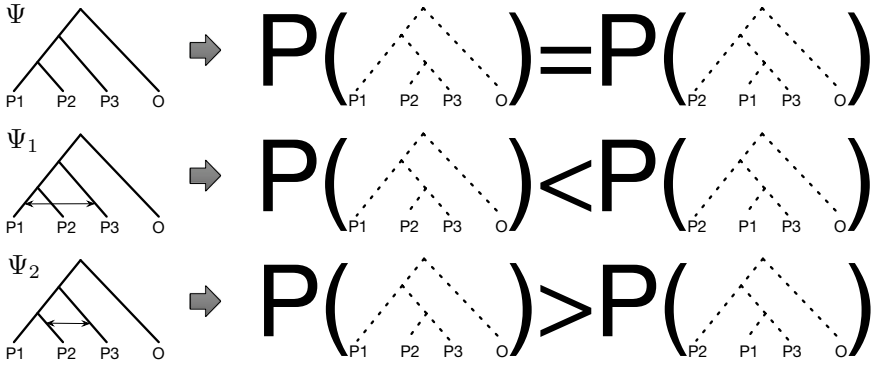


Fig. 13.16 The three scenarios of probabilities of the two gene trees that are discordant with the species tree in the case of a single reticulation event between P3 and one of the other two (in-group) species. If the evolutionary history of the species is a tree (Ψ), the two discordant gene trees are equally probable. However, if the evolutionary history of the species is non-treelike, as given by phylogenetic networks Ψ_1 and Ψ_2, then the probabilities of the two discordant gene trees are unequal in different ways

P1 and P3 occurs, this results in an increase in the probability of the discordant gene tree that groups P1 and P3 as sister taxa, as compared to the other discordant gene tree. Similarly, when a reticulation between P2 and P3 occurs, this results in an increase in the probability of the discordant gene tree that groups P2 and P3 as sister taxa. These three scenarios are illustrated in Fig. 13.16.

Assuming an infinite sites assumption of sequence evolution, the frequencies of gene trees ((P1, (P2, P3)), O) and (((P1, P3), P2), O) directly correlate with the values N_{ABBA} and N_{BABA}, respectively, explaining the rationale behind Eq. (13.10). To apply the *D*-statistic, frequencies of the *ABBA* and *BABA* site patterns are counted across an

alignment of four genomes, the value of Eq. (13.10) is calculated, and deviation from 0 is assessed for statistical significance. A significant deviation is taken as evidence of introgression.

Since the introduction of the *D*-Statistic, work has been done to extend this framework. Recently, the software package HyDe [5] was introduced with several extensions including handling multiple individuals from four populations as well as identifying individual hybrids in a population based on the method of [59]. In HyDe, higher numbers of individuals are handled through calculating statistics on all permutations of quartets of the individuals. Another recent extension to move the *D*-Statistic beyond four taxa is the D_{FOIL} framework introduced by [88]. In it, we see the same derivation used in the *D*-Statistic on a particular five-taxon tree. This derivation includes isolating gene trees whose probabilities go from equal to unequal when going from the tree case to the network case as well as converting these gene trees to corresponding site patterns to count in an alignment. Finally, Elworth et al. recently devised a heuristic, D_{GEN}, for automatically deriving phylogenetic invariants for detecting hybridization in more general cases than the *D*-Statistic and D_{FOIL} can handle [25]. The rationale behind the approach of Elworth et al. is that invariants could be derived by computing the probabilities of gene trees under a given species tree (e.g., using the method of [20]) and then computing the probabilities of the same trees under the same species tree with any reticulation scenarios added to it (using the method of [123]), and contrasting the two to identify sets of gene trees whose equal probabilities under the tree model get violated under the network model.

The *D*-Statistic is very simple to implement and understand, and it can be calculated on 4-genome alignments very efficiently, making it an appealing choice of a test for detecting introgression. Indeed, applications of the *D*-Statistic are widespread in the literature, reporting on introgression in ancient hominids [22, 37], butterflies [111], bears [61], and sparrows [23], just to name a few. However, it is important to note here that the derivation of the *D*-Statistic (and its extensions) relies on many assumptions that can easily be violated in practice. One major cause of such a violation is that the mathematics behind the *D*-Statistic relies on the coalescent which makes many simplifying assumptions about the evolutionary model and processes taking place. Of course, this shortcoming applies as well to all phylogenetic inference methods that employ the MSC or MSNC models. A second cause of such a violation is that in practice, more than a single reticulation could have taken place and ignoring those could result in erroneous inferences [25]. A third violation stems from the way the *D*-Statistic is applied.[3] In propositional logic, the statement "If p, then q" and its converse "If q, then p" are logically not equivalent. That is, if one is true, it is not necessarily the case the other is. Looking back at Fig. 13.16, the statement illustrated by the figure is: If there is no reticulation (i.e., the species phylogeny is a tree), then the probabilities of the two discordant trees are equal. The converse (if the probabilities of the two discordant trees are equal, then the species phylogeny is a tree) does not follow logically. However, this is how the test is used in practice. In all fairness, though, this logical fallacy is commonplace in inferences in biology,

[3]We especially thank David Morrison for requesting that we highlight this issue.

including when inferring species trees and networks under the MSC and MSNC, respectively. The fallacy is always dealt with by resorting to the "simplest possible explanation" argument. For example, why various scenarios could have given rise to equal frequencies of the frequencies of the *ABBA* and *BABA* site patterns, the species tree scenario is considered the simplest such possible explanation and is invoked as such.

Last but not least, Peter [89] recently provided a review and elegant connections between the *D*-Statistic and a family of statistics known as the *F*-statistics.

13.8 Phylogenetic Networks in the Population Genetics Community

The population genetics community has long adopted rooted, directed acyclic graphs as a model of evolutionary histories, typically of individuals within a single species. Ancestral recombination graphs, or ARGs, were introduced [38, 46] to model the evolutionary history of a set of genomic sequences in terms of the coalescence and recombination events that occurred since their most recent common ancestor. Statistical methods for inference of ARGs from genome-wide data have also been developed [94]. Gusfield's recent book [40] discusses algorithmic and combinatorial aspects of ARGs. However, while ARGs take the shape of a phylogenetic network as defined above, they are aimed at modeling recombination and methods for their inference are generally not applicable to hybridization detection.

Efforts in the population genetics community that are aimed at modeling admixture and gene flow are more relevant to hybridization detection. Here we discuss one of the most popular methods in this domain, namely TreeMix [90]. In population genetics, the counterparts to species trees and phylogenetic networks are population trees and admixture graphs, respectively. The difference between these models boils down to what labels their leaves: If the leaves are labeled by different species, then the models are called species trees/networks, and if the leaves are labeled by different subpopulations of the same species, then the models are called population trees and admixture graphs. Of course, it is not always easy to identify whether a species or subpopulations have been delimited, and hence what particular tree/network should be called in that case, as species and populations may exist on a continuum [18].

13.8.1 TreeMix

TreeMix [90] models the evolution of a set of SNPs, where the input data consists of allele frequencies of these SNPs in a set of populations whose evolutionary history is given by a population tree (in the case of no migration) or an admixture graph (in the case where migration is included).

The basis of the model used in TreeMix is in the notion of modeling drift over time as a diffusion process, where an original allele frequency of x_1 of a given SNP undergoes drift by an amount c to give rise to a new allele frequency x_2 [11, 17, 82], as given by

$$x_2 = x_1 + N(0, c \cdot x_1[1 - x_1]). \tag{13.11}$$

It is worth noting here that, as pointed out in [90], $c = t/2N_e$ for drift over small time scales where the time scale is on the same order of the effective population size [82].

When there are multiple populations under the effects of drift that evolved down a tree, the drift processes become linked and can no longer be described with independent Gaussian additions. This process is modeled with a covariance matrix derived from the amounts of drift occurring along the branches of the evolutionary tree. Finally, to incorporate reticulate evolution into the model one only needs to alter this covariance matrix based on the rate of gene flow along reticulate edges in the admixture graph.

In its current implementation, the authors of TreeMix assume the evolutionary history of the sampled, extant populations is very close to a tree-like structure. Based on this assumption, the search for a maximum likelihood admixture graph proceeds by first estimating a rooted tree, and then adding migration events one at a time until they are no longer statistically significant (however, as the authors point out, they "prefer to stop adding migration events well before this point so that the result graph remains interpretable."). Clearly, adding additional edges connecting the edges of a tree in this way will infer a tree-based network, which is a more limited class of networks compared with phylogenetic networks [31].

13.9 Data, Methods, and Software

Given the interest in reticulate evolution from both theoreticians and empirical researchers, it is perhaps unsurprising that software to infer hybridization has proliferated in recent years. Such methods have been developed for a variety of data types, including multi-locus data, SNP matrices, and whole genomes (Table 13.1). Some of these methods are able to infer a phylogenetic network, whereas others infer introgression between species tree lineages. With the exception of TreeMix and MixMapper, methods that infer phylogenetic networks are not constructed around a backbone tree, and so do not assume that tree-like evolution is the dominant process.

Regardless of the input and output, most of these methods allow for ILS in addition to hybridization (Table 13.1), which is necessary to infer phylogenetic networks representing reticulate evolution in biological systems where ILS is a possibility. Likelihood (including Bayesian) methods incorporate the possibility or effect of ILS into the likelihood function. Maximum parsimony methods that minimize deep coalescences, for example `InferNetwork_MP`, essentially attempt to infer the

Table 13.1 Common methods to infer hybridization

Method	Platform	Inference	ILS	Input	Output
5-taxon ABBA-BABA [88]	D_{FOIL}	Phylogenetic invariants	Yes	Genomic data	Presence/absence of introgression[a]
ABBA-BABA [22]	D_{FOIL}, HyDe, etc.	Phylogenetic invariants	Yes	Genomic data	Presence/absence of introgression[a]
D_{GEN} [25]	ALPHA [24]	Phylogenetic invariants	Yes	Genomic data	Presence/absence of introgression[a]
Blischak et al. [5]	HyDe	Phylogenetic invariants	Yes	Genomic data	Hybrid species
AIM [76]	BEAST2	Bayesian	Yes	Multi-locus sequences	Rooted tree w/gene flow[b]
IMa2 [45]	Standalone	Bayesian	Yes	Multi-locus sequences and backbone tree	Evolutionary parameters[c]
PIRN [121]	Standalone	Maximum parsimony	No	Rooted gene trees	Rooted network
CASS [112]	Dendroscope [49]	Maximum parsimony	No	Rooted gene trees	Rooted network
InferNetwork_ML [124]	PhyloNet [110, 119]	Maximum likelihood	Yes	Rooted gene trees	Rooted network
InferNetwork_MP [122]	PhyloNet [110, 119]	Maximum parsimony	Minimized	Rooted gene trees	Rooted network
InferNetwork_MPL [125]	PhyloNet [110, 119]	Pseudo-likelihood	Yes	Rooted gene trees	Rooted network
MCMC_BiMarkers [135]	PhyloNet [110, 119]	Bayesian	Yes	Bi-allelic sites	Rooted network
MCMC_GT [118]	PhyloNet [110, 119]	Bayesian	Yes	Rooted gene trees	Rooted network
MCMC_SEQ [116]	PhyloNet [110, 119]	Bayesian	Yes	Multi-locus sequences	Rooted network
MLE_BiMarkers [134]	PhyloNet [110, 119]	Maximum (pseudo)likelihood	Yes	Bi-allelic sites	Rooted network
MixMapper [63]	Standalone	Pseudo-likelihood	Yes	Allele frequencies	Rooted network
Neighbor-net [9]	Splitstree [48]	Agglomeration	No	Genomic data	Splits graph (unrooted network)
SNAQ [101]	PhyloNetworks [102]	Pseudo-likelihood	Yes	Unrooted gene trees	Unrooted, level-1 network
SpeciesNetwork [131]	BEAST2 [8]	Bayesian	Yes	Multi-locus sequences	Rooted network
STEM-hy [60]	Standalone	Maximum likelihood	Yes	Molecular clock gene trees	Rooted tree w/hybridizations[d]
TreeMix [90]	Standalone	Pseudo-likelihood	Yes	Allele frequencies	User-rooted, tree-based network

[a] Along with statistical significance
[b] Between contemporaneous lineages
[c] Effective population sizes, migration rates, divergence times
[d] Hybridizations limited to sister lineages

tree or network that minimizes the quantity of ILS, but do not necessarily eliminate all genetic discordance.

Methods which do not allow for ILS will instead infer phylogenetic networks representing conflicting signals [9]. Reticulation is one such conflicting signal, but so is ILS, so reticulate branches in these networks should not be blindly interpreted as necessarily representing introgression or hybridization.

13.9.1 Limitations

The biggest limitation of methods to infer introgression and hybridization, including species network methods, is scalability.

Methods which infer a species network directly from multi-locus sequences have only been used with a handful of taxa, and less than 200 loci. A systematic study of the species tree method StarBEAST found that the number of loci used has a power law relationship with a large exponent with the required computational time, making inference using thousands of loci intractable [84]. Although no systematic study of computational performance has been conducted for equivalent species network methods such as `MCMC_SEQ`, anecdotally they suffer from similar scaling issues.

Methods which scale better than direct multi-locus inference have been developed, but they are no silver bullet. Species networks can be estimated directly from unlinked bi-allelic markers by integrating over all possible gene trees for each marker, which avoids having to sequentially or jointly estimate gene trees. Bi-allelic methods make the inference of species trees and networks from thousands of markers possible, at the cost of using less informative markers.

Pseudo-likelihood inference has been developed for both bi-allelic and multi-locus methods [125, 134]. This reduces the computational cost of computing the likelihood of a species network as the number of taxa increases, and enabled the reanalysis of an empirical data set with 1070 genes from 23 species [125].

The ABBA-BABA test and similar phylogenetic invariant methods are capable of analyzing an enormous depth of data (whole genomes), but can be limited in taxonomic breadth based on hard limits of four or five taxa for the D-Statistic and D_{FOIL}, respectively, or by computational requirements for the case of D_{GEN} (Table 13.1). In addition, the D-Statistic and D_{FOIL} are limited to testing a specific hypothesis for introgression given a fixed species tree topology of a specific shape. This can be understood as a trade-off, where the flexibility of species network methods is sacrificed for the ability to use more data.

Beyond scalability, another present limitation is visualizing or summarizing posterior or bootstrap distributions of networks. Methods have been developed to visualize whole distributions of trees, or summarize a distribution as a single tree. Equivalent tools for networks are underdeveloped, leaving researchers to report the topology or set of topologies with the highest posterior or bootstrap support.

13.10 Conclusions and Future Directions

Great strides have been made over the past decade in the inference of evolutionary histories in the presence of hybridization and other processes, most notably incomplete lineage sorting. Species networks can now be inferred directly from species-level data which do not assume any kind of backbone tree, and instead, put reticulate evolution on an equal basis with speciation.

To some extent, the development of species network methods has recapitulated the development of species tree methods, starting with maximum parsimony and transitioning to likelihood methods, both maximum likelihood and Bayesian. To improve computational performance and enable the analysis of large data sets, pseudo-likelihood species network methods have been developed, inspired by similar species tree methods.

Phylogenetic invariant methods such as the ABBA-BABA test are able to test for reticulate evolution across whole genomes, uncovering chromosomal inversions and other features associated with hybridization and introgression. Last but not least, the population genetics community has long been interested in and developing methods for phylogenetic networks mainly to model the evolution of subpopulations in the presence of admixture and gene flow. In this chapter, we surveyed the recent computational developments in the field and listed computer software programs that enable reticulate evolutionary analyses for the study of hybridization and introgression, and generally to infer more accurate evolutionary histories of genes and species.

Empirical biologists feel constrained by the computational performance of existing species network methods. For species trees, phylogenetic invariant methods can be combined with quartet reconciliation to infer large species trees from genomic data, as in SVDquartets [14]. For networks, phylogenetic invariant methods to identify the true network with a limited number of edges needs to be developed, as do methods to reconcile the resulting subnets.

Even for species trees, Bayesian methods have practical limitations in terms of the amount of data they can be used with. Bayesian methods for trees and networks, with few exceptions, have been built on Markov chain Monte Carlo (MCMC). This technique is inherently serial and hence unsuited to modern workstations, which contain many CPU and GPU cores working in parallel. It is important to continue to explore other Bayesian algorithms which work in parallel such as sequential Monte Carlo [7], or algorithms which are orders of magnitude faster than MCMC such as variational Bayes [115].

Phylogenetic methods for species tree inference have a huge head start on methods for species network inference. Not only is the problem of species network inference much more complicated, but species tree methods have been in development for much longer. For example, MDC for species trees was first described in 1997, and extended to phylogenetic networks 14 years later [69, 127]. In this light, the progress made is remarkable. However, as evolutionary biology is moving toward data sets containing whole genomes for hundreds or even thousands of taxa, methods developers must

focus on improving the scalability of their methods without sacrificing accuracy so that the full potential of this data may be realized.

While preliminary studies exist of the performance of the different methods for phylogenetic network inference [56], more thorough studies are needed to assess the accuracy as well as computational requirements of the different methods.

Last but not least, it is important to highlight that all the development described above excludes processes such as gene duplication and loss, and so may be susceptible to errors and artifacts which can be present in data such as hidden paralogy. Furthermore, the multispecies network coalescent already has its own population-genetic assumptions, almost all of which are not necessarily realistic for analyses in practice. Accounting for these is a major next step (though it is important to point out that these have not been fully explored in the context of species tree inference either), but the mathematical complexity will most likely add, extensively, to the computational complexity of the inference step.

Acknowledgements Luay Nakhleh started working on phylogenetic networks in 2002 in a close collaboration with Tandy Warnow, Bernard M. E. Moret, and C. Randal Linder. At the time, their focus was on phylogenetic networks in terms of displaying trees. This focus led to work on the inference of smallest phylogenetic networks that display a given set of trees, as well as on comparing networks in terms of their displayed trees. While the approaches pursued at the time were basic, they were foundational in terms of pursuing more sophisticated models and approaches by Nakhleh and his group. Therefore, we would like to acknowledge the role that Bernard played in the early days of (explicit) phylogenetic networks.

The authors would also like to acknowledge James Mallet, Craig Moritz, David Morrison, and Mike Steel for extensive discussions and detailed comments that helped us significantly improve this chapter. The authors thank Matthew Hahn and Kelley Harris for their discussion of the definition of phylogenetic invariant methods, and Dingqiao Wen for the results in Sect. 13.6.3.

This work was partially supported by NSF grants DBI-1355998, CCF-1302179, CCF-1514177, CCF-1800723, and DMS-1547433.

References

1. Abbott, R., Albach, D., Ansell, S., Arntzen, J., Baird, S., Bierne, N., Boughman, J., Brelsford, A., Buerkle, C., Buggs, R., Butlin, R.K., Dieckmann, U., Eroukhmanoff, F., Grill, A., Cahan, S.H., Hermansen, J.S., Hewitt, G., Hudson, A.G., Jiggins, C., Jones, J., Keller, B., Marczewski, T., Mallet, J., Martinez-Rodriguez, P., Möst, M., Mullen, S., Nichols, R., Nolte, A.W., Parisod, C., Pfennig, K., Rice, A.M., Ritchie, M.G., Seifert, B., Smadja, C.M., Stelkens, R., Szymura, J.M., Väinölä, R., Wolf, J.B.W., Zinner, D.: Hybridization and speciation. J. Evolut. Biol. **26**(2), 229–246 (2013)
2. Arnold, M.: Natural Hybridization and Evolution. Oxford University Press (1997)
3. Barton, N.: The role of hybridization in evolution. Mol. Ecol. **10**(3), 551–568 (2001)
4. Barton, N.H., Hewitt, G.M.: Analysis of hybrid zones. Ann. Rev. Ecol. Syst. **16**(1), 113–148 (1985)
5. Blischak, P.D., Chifman, J., Wolfe, A.D., Kubatko, L.S.: HyDe: a Python package for genome-scale hybridization detection. Syst. Biol. p. syy023 (2018)
6. Bonhomme, M., Cuartero, S., Blancher, A., Crouau-Roy, B.: Assessing natural introgression in 2 biomedical model species, the rhesus macaque (Macaca mulatta) and the long-tailed macaque (Macaca fascicularis). J. Heredity **100**(2), 158–169 (2009)

7. Bouchard-Côté, A., Sankararaman, S., Jordan, M.I.: Phylogenetic inference via sequential Monte Carlo. Syst. Biol. **61**(4), 579–593 (2012). http://dx.doi.org/10.1093/sysbio/syr131
8. Bouckaert, R., Heled, J., Khnert, D., Vaughan, T., Wu, C.H., Xie, D., Suchard, M.A., Rambaut, A., Drummond, A.J.: BEAST 2: a software platform for Bayesian evolutionary analysis. PLOS Comput. Biol. **10**(4), 1–6 (2014). https://doi.org/10.1371/journal.pcbi.1003537
9. Bryant, D., Moulton, V.: Neighbor-Net: An agglomerative method for the construction of phylogenetic networks. Mol. Biol. Evolut. **21**(2), 255–265 (2004). http://dx.doi.org/10.1093/molbev/msh018
10. Buckland, S.T., Burnham, K.P., Augustin, N.H.: Model selection: an integral part of inference. Biometrics **53**(2), 603–618 (1997). http://www.jstor.org/stable/2533961
11. Cavalli-Sforza, L.L., Edwards, A.W.: Phylogenetic analysis: models and estimation procedures. Evolution **21**(3), 550–570 (1967)
12. Cavender, J.A., Felsenstein, J.: Invariants of phylogenies in a simple case with discrete states. J. Classification **4**(1), 57–71 (1987)
13. Chatzou, M., Magis, C., Chang, J.M., Kemena, C., Bussotti, G., Erb, I., Notredame, C.: Multiple sequence alignment modeling: methods and applications. Brief. Bioinform. **17**(6), 1009–1023 (2015)
14. Chifman, J., Kubatko, L.: Quartet inference from SNP data under the coalescent model. Bioinformatics **30**(23), 3317–3324 (2014). http://dx.doi.org/10.1093/bioinformatics/btu530
15. Chor, B., Tuller, T.: Maximum likelihood of evolutionary trees is hard. In: Miyano, S., Mesirov, J., Kasif, S., Istrail, S., Pevzner, P., Waterman, M. (eds.), Research in Computational Molecular Biology. Lecture Notes in Computer Science, vol. 3500, pp. 995–995. Springer, Berlin/Heidelberg (2005)
16. Clark, A.G., Messer, P.W.: Conundrum of jumbled mosquito genomes. Science **347**(6217), 27–28 (2015)
17. Coop, G., Witonsky, D., Di Rienzo, A., Pritchard, J.K.: Using environmental correlations to identify loci underlying local adaptation. Genetics **185**(4), 1411–1423 (2010)
18. De Queiroz, K.: Species concepts and species delimitation. Syst. Biol. **56**(6), 879–886 (2007). https://doi.org/10.1080/10635150701701083
19. Degnan, J.H., Rosenberg, N.A.: Gene tree discordance, phylogenetic inference and the multispecies coalescent. Trends Ecol. Evolut. **24**(6), 332–340 (2009)
20. Degnan, J.H., Salter, L.A.: Gene tree distributions under the coalescent process. Evolution **59**, 24–37 (2005)
21. Du, P., Nakhleh, L.: Species tree and reconciliation estimation under a duplication-loss-coalescence model. In: The 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB), pp. 376–385. Association for Computing Machinery (ACM) (2018)
22. Durand, E.Y., Patterson, N., Reich, D., Slatkin, M.: Testing for ancient admixture between closely related populations. Mol. Biol. Evolut. **28**(8), 2239–2252 (2011). http://mbe.oxfordjournals.org/content/28/8/2239.abstract
23. Elgvin, T.O., Trier, C.N., Tørresen, O.K., Hagen, I.J., Lien, S., Nederbragt, A.J., Ravinet, M., Jensen, H., Sætre, G.P.: The genomic mosaicism of hybrid speciation. Sci. Adv. **3**(6), e1602,996 (2017)
24. Elworth, R.L., Allen, C., Benedict, T., Dulworth, P., Nakhleh, L.: ALPHA: a toolkit for automated local phylogenomic analyses. Bioinformatics **1**, 3 (2018)
25. Elworth, R.L., Allen, C., Benedict, T., Dulworth, P., Nakhleh, L.: D_{GEN}: a test statistic for detection of general introgression scenarios. In: Proceedings of the 18th Workshop on Algorithms in Bioinformatics (WABI) (2018)
26. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evolut. **17**(6), 368–376 (1981)
27. Felsenstein, J.: Inferring Phylogenies. Sinauer, Sunderland, MA (2004)
28. Fernández-Mazuecos, M., Mellers, G., Vigalondo, B., Sáez, L., Vargas, P., Glover, B.J.: Resolving recent plant radiations: power and robustness of genotyping-by-sequencing. Syst. Biol. **67**(2), 250–268 (2018). http://dx.doi.org/10.1093/sysbio/syx062

29. Folk, R.A., Soltis, P.S., Soltis, D.E., Guralnick, R.: New prospects in the detection and comparative analysis of hybridization in the tree of life. Am. J. Botany **105**(3), 364–375 (2018)
30. Fontaine, M.C., Pease, J.B., Steele, A., Waterhouse, R.M., Neafsey, D.E., Sharakhov, I.V., Jiang, X., Hall, A.B., Catteruccia, F., Kakani, E., Mitchell, S.N., Wu, Y.C., Smith, H.A., Love, R.R., Lawniczak, M.K., Slotman, M.A., Emrich, S.J., Hahn, M.W., Besansky, N.J.: Extensive introgression in a malaria vector species complex revealed by phylogenomics. Science **347**(6217), 1258,524 (2015)
31. Francis, A.R., Steel, M.: Which phylogenetic networks are merely trees with additional arcs? Syst. Biol. **64**(5), 768–777 (2015)
32. Gascuel, O.: Mathematics of Evolution and Phylogeny. Oxford University Press, Oxford (2005)
33. Gelman, A., Meng, X.L., Stern, H.: Posterior predictive assessment of model fitness via realized discrepancies. Statistica Sinica **6**(4), 733–760 (1996). http://www.jstor.org/stable/24306036
34. Gernhard, T.: The conditioned reconstructed process. J. Theoret. Biol. **253**(4), 769–778 (2008). https://doi.org/10.1016/j.jtbi.2008.04.005
35. Good, J.M.: Reduced representation methods for subgenomic enrichment and next-generation sequencing. In: Orgogozo, V., Rockman, M.V. (eds.) Molecular Methods for Evolutionary Genetics, pp. 85–103. Humana Press, Totowa, NJ (2011)
36. Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**(4), 711–732 (1995)
37. Green, R.E., Krause, J., Briggs, A.W., Maricic, T., Stenzel, U., Kircher, M., Patterson, N., Li, H., Zhai, W., Fritz, M.H.Y., Hansen, N.F., Durand, E.Y., Malaspinas, A.S., Jensen, J.D., Marques-Bonet, T., Alkan, C., Prafer, K., Meyer, M., Burbano, H.A., Good, J.M., Schultz, R., Aximu-Petri, A., Butthof, A., Hober, B., Hoffner, B., Siegemund, M., Weihmann, A., Nusbaum, C., Lander, E.S., Russ, C., Novod, N., Affourtit, J., Egholm, M., Verna, C., Rudan, P., Brajkovic, D., Kucan, O., Guic, I., Doronichev, V.B., Golovanova, L.V., Lalueza-Fox, C., de la Rasilla, M., Fortea, J., Rosas, A., Schmitz, R.W., Johnson, P.L.F., Eichler, E.E., Falush, D., Birney, E., Mullikin, J.C., Slatkin, M., Nielsen, R., Kelso, J., Lachmann, M., Reich, D., Paabo, S.: A draft sequence of the Neandertal genome. Science **328**(5979), 710–722 (2010). http://www.sciencemag.org/content/328/5979/710.abstract
38. Griffiths, R., Marjoram, P.: Ancestral inference from samples of DNA sequences with recombination. J. Comput. Biol. **3**, 479–502 (1996)
39. Grummer, J.A., Morando, M.M., Avila, L.J., Sites, J.W., Leaché, A.D.: Phylogenomic evidence for a recent and rapid radiation of lizards in the Patagonian Liolaemus fitzingerii species group. Mol. Phylogenet. Evolut. (2018). https://doi.org/10.1016/j.ympev.2018.03.023, http://www.sciencedirect.com/science/article/pii/S1055790317307303
40. Gusfield, D.: ReCombinatorics: the algorithmics of ancestral recombination graphs and explicit phylogenetic networks. MIT Press (2014)
41. Hagen, O., Hartmann, K., Steel, M., Stadler, T.: Age-dependent speciation can explain the shape of empirical phylogenies. Syst. Biol. **64**(3), 432–440 (2015). http://dx.doi.org/10.1093/sysbio/syv001
42. Hahn, M.W.: Toward a selection theory of molecular evolution. Evolution **62**(2), 255–265 (2008). https://doi.org/10.1111/j.1558-5646.2007.00308.x
43. Harrison, R.G., Larson, E.L.: Hybridization, introgression, and the nature of species boundaries. J. Heredity **105**(S1), 795–809 (2014)
44. Hejase, H.A., Liu, K.J.: A scalability study of phylogenetic network inference methods using empirical datasets and simulations involving a single reticulation. BMC Bioinform. **17**(1), 422 (2016). https://doi.org/10.1186/s12859-016-1277-1
45. Hey, J.: Isolation with migration models for more than two populations. Mol. Biol. Evolut. **27**(4), 905–920 (2010). http://dx.doi.org/10.1093/molbev/msp296
46. Hudson, R.R.: Gene genealogies and the coalescent process. In: Futuyma, D, Antonovics, J. (eds.) Oxford Surveys in Evolutionary Biology, vol. 7, pp. 1–44. Oxford University Press (1991)
47. Hudson, R.R.: Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics **18**, 337–338 (2002)

48. Huson, D.: SplitsTree: a program for analyzing and visualizing evolutionary data. Bioinformatics **14**(1), 68–73 (1998)
49. Huson, D., Richter, D., Rausch, C., Dezulian, T., Franz, M., Rupp, R.: Dendroscope: an interactive viewer for large phylogenetic trees. BMC Bioinform. **8**(1), 460 (2007)
50. Jarvis, E.D., Mirarab, S., Aberer, A.J., Li, B., Houde, P., Li, C., Ho, S.Y.W., Faircloth, B.C., Nabholz, B., Howard, J.T., Suh, A., Weber, C.C., da Fonseca, R.R., Li, J., Zhang, F., Li, H., Zhou, L., Narula, N., Liu, L., Ganapathy, G., Boussau, B., Bayzid, M.S., Zavidovych, V., Subramanian, S., Gabaldón, T., Capella-Gutiérrez, S., Huerta-Cepas, J., Rekepalli, B., Munch, K., Schierup, M., Lindow, B., Warren, W.C., Ray, D., Green, R.E., Bruford, M.W., Zhan, X., Dixon, A., Li, S., Li, N., Huang, Y., Derryberry, E.P., Bertelsen, M.F., Sheldon, F.H., Brumfield, R.T., Mello, C.V., Lovell, P.V., Wirthlin, M., Schneider, M.P.C., Prosdocimi, F., Samaniego, J.A., Velazquez, A.M.V., Alfaro-Núñez, A., Campos, P.F., Petersen, B., Sicheritz-Ponten, T., Pas, A., Bailey, T., Scofield, P., Bunce, M., Lambert, D.M., Zhou, Q., Perelman, P., Driskell, A.C., Shapiro, B., Xiong, Z., Zeng, Y., Liu, S., Li, Z., Liu, B., Wu, K., Xiao, J., Yinqi, X., Zheng, Q., Zhang, Y., Yang, H., Wang, J., Smeds, L., Rheindt, F.E., Braun, M., Fjeldsa, J., Orlando, L., Barker, F.K., Jønsson, K.A., Johnson, W., Koepfli, K.P., O'Brien, S., Haussler, D., Ryder, O.A., Rahbek, C., Willerslev, E., Graves, G.R., Glenn, T.C., McCormack, J., Burt, D., Ellegren, H., Alström, P., Edwards, S.V., Stamatakis, A., Mindell, D.P., Cracraft, J., Braun, E.L., Warnow, T., Jun, W., Gilbert, M.T.P., Zhang, G.: Whole-genome analyses resolve early branches in the tree of life of modern birds. Science **346**(6215), 1320–1331 (2014)
51. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Efficient parsimony-based methods for phylogenetic network reconstruction. Bioinformatics **23**, e123–e128 (2006). Proceedings of the European Conference on Computational Biology (ECCB 06)
52. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Maximum likelihood of phylogenetic networks. Bioinformatics **22**(21), 2604–2611 (2006)
53. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Inferring phylogenetic networks by the maximum parsimony criterion: a case study. Mol. Biol. Evolut. **24**(1), 324–337 (2007)
54. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: A new linear-time heuristic algorithm for computing the parsimony score of phylogenetic networks: theoretical bounds and empirical performance. In: Mandoiu, I., Zelikovsky, A. (eds.) Proceedings of the International Symposium on Bioinformatics Research and Applications. Lecture Notes in Bioinformatics, vol. 4463, pp. 61–72 (2007)
55. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Parsimony score of phylogenetic networks: hardness results and a linear-time heuristic. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(3), 495–505 (2009)
56. Kamneva, O.K., Rosenberg, N.A.: Simulation-based evaluation of hybridization network reconstruction methods in the presence of incomplete lineage sorting. Evolut. Bioinform. **13**, 1176934317691,935 (2017)
57. Kanj, I., Nakhleh, L., Than, C., Xia, G.: Seeing the trees and their branches in the network is hard. Theoret. Comput. Sci. **401**, 153–164 (2008)
58. Kanj, I., Nakhleh, L., Xia, G.: The compatibility of binary characters on phylogenetic networks: complexity and parameterized algorithms. Algorithmica **51**, 99–128 (2008)
59. Kubatko, L., Chifman, J.: An invariants-based method for efficient identification of hybrid species from large-scale genomic data. bioRxiv, p. 034348 (2015)
60. Kubatko, L.S.: Identifying hybridization events in the presence of coalescence via model selection. Syst. Biol. **58**(5), 478–488 (2009)
61. Kumar, V., Lammers, F., Bidon, T., Pfenninger, M., Kolter, L., Nilsson, M.A., Janke, A.: The evolutionary history of bears is characterized by gene flow across species. Sci. Rep. **7**, 46,487 (2017)
62. Lake, J.A.: A rate-independent technique for analysis of nucleic acid sequences: evolutionary parsimony. Mol. Biol. Evolution **4**(2), 167–191 (1987)
63. Lipson, M., Loh, P.R., Levin, A., Reich, D., Patterson, N., Berger, B.: Efficient moment-based inference of admixture parameters and sources of gene flow. Mol. Biol. Evolut. **30**(8), 1788–1802 (2013). http://dx.doi.org/10.1093/molbev/mst099

64. Liu, K., Steinberg, E., Yozzo, A., Song, Y., Kohn, M., Nakhleh, L.: Interspecific introgressive origin of genomic diversity in the house mouse. Proc. Nat. Acad. Sci. **112**(1), 196–201 (2015)
65. Liu, L., Xi, Z., Wu, S., Davis, C.C., Edwards, S.V.: Estimating phylogenetic trees from genome-scale data. Ann. New York Acad. Sci. **1360**(1), 36–53 (2015)
66. Liu, L., Yu, L., Edwards, S.V.: A maximum pseudo-likelihood approach for estimating species trees under the coalescent model. BMC Evolut. Biol. **10**(1), 302 (2010). https://doi.org/10.1186/1471-2148-10-302
67. Liu, L., Yu, L.L., Kubatko, L., Pearl, D.K., Edwards, S.V.: Coalescent methods for estimating phylogenetic trees. Mol. Phylogenet. Evol. **53**, 320–328 (2009)
68. Long, J.C.: The genetic structure of admixed populations. Genetics **127**, 417–428 (1991)
69. Maddison, W.: Gene trees in species trees. Syst. Biol. **46**(3), 523–536 (1997)
70. Maddison, W.P., Knowles, L.L.: Inferring phylogeny despite incomplete lineage sorting. Syst. Biol. **55**, 21–30 (2006)
71. Mallet, J.: Hybridization as an invasion of the genome. TREE **20**(5), 229–237 (2005)
72. Mallet, J.: Hybrid speciation. Nature **446**, 279–283 (2007)
73. Mallet, J., Besansky, N., Hahn, M.W.: How reticulated are species? BioEssays **38**(2), 140–149 (2016)
74. Marcussen, T., Sandve, S.R., Heier, L., Spannagl, M., Pfeifer, M.: The international wheat genome sequencing consortium, Jakobsen, K.S., Wulff, B.B.H., Steuernagel, B., Mayer, K.F.X., Olsen, O.A., Ancient hybridizations among the ancestral genomes of bread wheat. Science **345**(6194), 1250,092 (2014)
75. Mirarab, S., Reaz, R., Bayzid, M.S., Zimmermann, T., Swenson, M.S., Warnow, T.: ASTRAL: genome-scale coalescent-based species tree estimation. Bioinformatics **30**(17), i541–i548 (2014). http://dx.doi.org/10.1093/bioinformatics/btu462
76. Mueller, N.F., Ogilvie, H.A., Zhang, C., Drummond, A.J., Stadler, T.: Inference of species histories in the presence of gene flow. bioRxiv (2018). https://doi.org/10.1101/348391, https://www.biorxiv.org/content/early/2018/06/17/348391
77. Nakhleh, L.: Computational approaches to species phylogeny inference and gene tree reconciliation. Trends Ecol. Evolut. **28**(12), 719–728 (2013)
78. Nakhleh, L., Jin, G., Zhao, F., Mellor-Crummey, J.: Reconstructing phylogenetic networks using maximum parsimony. In: Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference (CSB2005), pp. 93–102 (2005)
79. Nakhleh, L., Ringe, D.A., Warnow, T.: Perfect phylogenetic networks: a new methodology for reconstructing the evolutionary history of natural languages. Language **81**(2), 382–420 (2005)
80. Nakhleh, L., Sun, J., Warnow, T., Linder, C.R., Moret, B.M., Tholse, A.: Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In: Biocomputing 2003, pp. 315–326. World Scientific (2002)
81. Nichio, B.T., Marchaukoski, J.N., Raittz, R.T.: New tools in orthology analysis: a brief review of promising perspectives. Front. Genet. **8**, 165 (2017)
82. Nicholson, G., Smith, A.V., Jónsson, F., Gústafsson, Ó., Stefánsson, K., Donnelly, P.: Assessing population differentiation and isolation from single-nucleotide polymorphism data. J. R. Stat. Soc. Series B (Stat. Method.) **64**(4), 695–715 (2002)
83. Ogilvie, H.A., Bouckaert, R.R., Drummond, A.J.: StarBEAST2 brings faster species tree inference and accurate estimates of substitution rates. Mol. Biol. Evolut. **34**(8), 2101–2114 (2017). http://dx.doi.org/10.1093/molbev/msx126
84. Ogilvie, H.A., Heled, J., Xie, D., Drummond, A.J.: Computationalperformanceand statistical accuracy of *BEAST and comparisons with other methods. Syst. Biol. **65**(3), 381–396 (2016). https://doi.org/10.1093/sysbio/syv118, http://sysbio.oxfordjournals.org/content/65/3/381.abstract
85. Osada, N., Uno, Y., Mineta, K., Kameoka, Y., Takahashi, I., Terao, K.: Ancient genome-wide admixture extends beyond the current hybrid zone between Macaca fascicularis and M. mulatta. Mol. Ecol. **19**(14), 2884–2895 (2010)

86. Park, H., Nakhleh, L.: MURPAR: A fast heuristic for inferring parsimonious phylogenetic networks from multiple gene trees. In: Proceedings of the International Symposium on Bioinformatics Research and Applications (ISBRA 12). Lecture Notes in Bioinformatics, vol. 7292, pp. 213–224 (2012)
87. Pease, J.B., Haak, D.C., Hahn, M.W., Moyle, L.C.: Phylogenomics reveals three sources of adaptive variation during a rapid radiation. PLoS Biol. **14**(2), e1002,379 (2016)
88. Pease, J.B., Hahn, M.W.: Detection and polarization of introgression in a five-taxon phylogeny. Syst. Biol. **64**(4), 651–662 (2015)
89. Peter, B.M.: Admixture, population structure, and F-statistics. Genetics **202**(4), 1485–1501 (2016)
90. Pickrell, J.K., Pritchard, J.K.: Inference of population splits and mixtures from genome-wide allele frequency data. PLoS Genet. **8**(11), e1002,967 (2012)
91. Racimo, F., Sankararaman, S., Nielsen, R., Huerta-Sánchez, E.: Evidence for archaic adaptive introgression in humans. Nat. Rev. Genet. **16**(6), 359–371 (2015)
92. Rambaut, A., Grassly, N.C.: Seq-gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Comp. Appl. Biosci. **13**, 235–238 (1997)
93. Rannala, B., Yang, Z.: Efficient Bayesian species tree inference under the multispecies coalescent. Syst. Biol. **66**(5), 823–842 (2017). http://dx.doi.org/10.1093/sysbio/syw119
94. Rasmussen, M.D., Hubisz, M.J., Gronau, I., Siepel, A.: Genome-wide inference of ancestral recombination graphs. PLoS Genet. **10**(5), e1004,342 (2014)
95. Rasmussen, M.D., Kellis, M.: Unified modeling of gene duplication, loss, and coalescence using a locus tree. Gen. Res. **22**(4), 755–765 (2012). https://doi.org/10.1101/gr.123901.111, http://genome.cshlp.org/content/22/4/755.abstract
96. Rieseberg, L.H.: Hybrid origins of plant species. Ann. Rev. Ecol. Syst. **28**, 359–389 (1997)
97. Roch, S.: A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. IEEE Trans. Comput. Biol. Bioinf. **3**(1), 92–94 (2006)
98. Scornavacca, C., Galtier, N.: Incomplete lineage sorting in mammalian phylogenomics. Syst. Biol. **66**(1), 112–120 (2017). http://dx.doi.org/10.1093/sysbio/syw082
99. Semple, C., Steel, M.: Phylogenetics. Oxford Series in Mathematics and its Applications (2004)
100. Simmons, M.P., Gatesy, J.: Coalescence versus concatenation: sophisticated analyses versus first principles applied to rooting the angiosperms. Mol. Phylogenet. Evolut. **91**, 98–122 (2015). https://doi.org/10.1016/j.ympev.2015.05.011, http://www.sciencedirect.com/science/article/pii/S1055790315001487
101. Solís-Lemus, C., Ané, C.: Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting. PLoS Genet. **12**(3), e1005,896 (2016)
102. Solís-Lemus, C., Bastide, P., Ané, C.: PhyloNetworks: a package for phylogenetic networks. Mol. Biol. Evolut. **34**(12), 3292–3298 (2017). http://dx.doi.org/10.1093/molbev/msx235
103. Solís-Lemus, C., Yang, M., Ané, C.: Inconsistency of species-tree methods under gene flow. Syst. Biol. (2016). https://doi.org/10.1093/sysbio/syw030
104. Song, Y., Endepols, S., Klemann, N., Richter, D., Matuschka, F.R., Shih, C.H., Nachman, M.W., Kohn, M.H.: Adaptive introgression of anticoagulant rodent poison resistance by hybridization between old world mice. Curr. Biol. **21**(15), 1296–1301 (2011). https://doi.org/10.1016/j.cub.2011.06.043, http://www.sciencedirect.com/science/article/pii/S0960982211007160
105. Stadler, T.: Sampling-through-time in birth-death trees. J. Theoret. Biol. **267**(3), 396–404 (2010). https://doi.org/10.1016/j.jtbi.2010.09.010, http://www.sciencedirect.com/science/article/pii/S0022519310004765
106. Stamatakis, A.: RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21), 2688–2690 (2006)
107. Steel, M.: Phylogeny: discrete and random processes in evolution. SIAM (2016)
108. Stevison, L., Kohn, M.: Divergence population genetic analysis of hybridization between rhesus and cynomolgus macaques. Mol. Ecol. **18**(11), 2457–2475 (2009)
109. Than, C., Nakhleh, L.: Species tree inference by minimizing deep coalescences. PLoS Comput. Biol. **5**(9), e1000,501 (2009)

110. Than, C., Ruths, D., Nakhleh, L.: PhyloNet: a software package for analyzing and reconstructing reticulate evolutionary relationships. BMC Bioinform. **9**(1), 322 (2008)
111. The Heliconious Genome Consortium: Butterfly genome reveals promiscuous exchange of mimicry adaptations among species. Nature **487**(7405), 94–98 (2012). http://dx.doi.org/10.1038/nature11041
112. Van Iersel, L., Kelk, S., Rupp, R., Huson, D.: Phylogenetic networks do not need to be complex: using fewer reticulations to represent conflicting clusters. Bioinformatics **26**(12), i124–i131 (2010)
113. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. In: Proceedings of the 2001 ACM Symposium on Applied Computing, SAC '01, pp. 46–50. ACM, New York, NY, USA (2001). http://doi.acm.org/10.1145/372202.372271
114. Warnow, T.: Computational phylogenetics: an introduction to designing methods for phylogeny estimation. Cambridge University Press (2017)
115. Waterhouse, S.R., MacKay, D., Robinson, A.J.: Bayesian methods for mixtures of experts. In: Advances in Neural Information Processing Systems, pp. 351–357 (1996)
116. Wen, D., Nakhleh, L.: Co-estimating reticulate phylogenies and gene trees from multi-locus sequence data. Syst. Biol. **67**(3), 439–457 (2018)
117. Wen, D., Yu, Y., Hahn, M., Nakhleh, L.: Reticulate evolutionary history and extensive introgression in mosquito species revealed by phylogenetic network analysis. Mol.Ecol. **25**, 2361–2372 (2016)
118. Wen, D., Yu, Y., Nakhleh, L.: Bayesian inference of reticulate phylogenies under the multispecies network coalescent. PLoS Genet. **12**(5), e1006,006 (2016)
119. Wen, D., Yu, Y., Zhu, J., Nakhleh, L.: Inferring phylogenetic networks using PhyloNet. Syst. Biol. **67**(4), 735–740 (2018)
120. Wu, Y.: Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees. Bioinformatics **26**(12), i140–i148 (2010)
121. Wu, Y.: An algorithm for constructing parsimonious hybridization networks with multiple phylogenetic trees. J. Comput. Biol. **20**(10), 792–804 (2013)
122. Yu, Y., Barnett, R., Nakhleh, L.: Parsimonious inference of hybridization in the presence of incomplete lineage sorting. Syst. Biol. **62**(5), 738–751 (2013)
123. Yu, Y., Degnan, J., Nakhleh, L.: The probability of a gene tree topology within a phylogenetic network with applications to hybridization detection. PLoS Genet. **8**, e1002,660 (2012)
124. Yu, Y., Dong, J., Liu, K., Nakhleh, L.: Maximum likelihood inference of reticulate evolutionary histories. Proc. Nat. Acad. Sci. **111**(46), 16,448–6453 (2014)
125. Yu, Y., Nakhleh, L.: A maximum pseudo-likelihood approach for phylogenetic networks. BMC Genom. **16**, S10 (2015)
126. Yu, Y., Ristic, N., Nakhleh, L.: Fast algorithms and heuristics for phylogenomics under ILS and hybridization. BMC Bioinform. **14**(Suppl 15), S6 (2013)
127. Yu, Y., Than, C., Degnan, J., Nakhleh, L.: Coalescent histories on phylogenetic networks and detection of hybridization despite incomplete lineage sorting. Syst. Biol. **60**(2), 138–149 (2011)
128. Yu, Y., Warnow, T., Nakhleh, L.: Algorithms for mdc-based multi-locus phylogeny inference: beyond rooted binary gene trees on single alleles. J. Comput. Biol. **18**(11), 1543–1559 (2011)
129. Yule, G.U.: A mathematical theory of evolution based on the conclusions of Dr. J.C. Willis, F.R.S. Phil. Trans. R. Soc. Lond. B **213**, 21–87 (1924)
130. Zhang, B., Wu, Y.C.: Coestimation of gene trees and reconciliations under a duplication-loss-coalescence model. In: Cai, Z., Daescu, O., Li, M. (eds.) Bioinformatics Research and Applications, pp. 196–210. Springer International Publishing, Cham (2017)
131. Zhang, C., Ogilvie, H.A., Drummond, A.J., Stadler, T.: Bayesian inference of species networks from multilocus sequence data. Mol. Biol. Evolut. **35**(2), 504–517 (2018). http://dx.doi.org/10.1093/molbev/msx307
132. Zhang, L.: On tree-based phylogenetic networks. J. Comput. Biol. **23**(7), 553–565 (2016)
133. Zhang, W., Dasmahapatra, K.K., Mallet, J., Moreira, G.R., Kronforst, M.R.: Genome-wide introgression among distantly related Heliconius butterfly species. Genome Biol. **17**, 25 (2016)

134. Zhu, J., Nakhleh, L.: Inference of species phylogenies from bi-allelic markers using pseudo-likelihood. Bioinformatics **34**, i376–i385 (2018). https://doi.org/10.1093/bioinformatics/bty295
135. Zhu, J., Wen, D., Yu, Y., Meudt, H.M., Nakhleh, L.: Bayesian inference of phylogenetic networks from bi-allelic genetic markers. PLOS Comput. Biol. **14**(1), 1–32 (2018). https://doi.org/10.1371/journal.pcbi.1005932
136. Zhu, J., Yu, Y., Nakhleh, L.: In the light of deep coalescence: revisiting trees within networks. BMC Bioinform. **17**(14), 415 (2016)

Chapter 14
A Perspective on Comparative and Functional Genomics

Daniel Doerr and Jens Stoye

Abstract Comparing genomes based on the order of genes provides insights into their evolutionary history and further allows to identify sets of genes with associated function. In the past two decades, many methods have been developed for identifying genomic regions that share homologous genes, which can be subsequently tested for functional associativity. As these methods are flexible by tolerating duplicate, missing, and intruding genes, we now study a case in which relationships between genes are established through a hierarchical relationship and thereby turn the problem of identifying regions with common functional associations inside out: We use a measure of dissimilarity between genes defined on a gene ontology hierarchy to identify collections of genomic regions with low functional dissimilarity.

Keywords Gene order comparison · Functional genomics · Functional dissimilarity

14.1 Introduction

Gene order comparison is an established tool of comparative genomics to gain insights into the functional organization of the genome. Evidence has been gathered in all kingdoms of life that genes with associated functionality are more densely co-localized on their chromosomal sequences than genes that are not functionally associated [1, 13, 17].

The study of gene order conservation has given rise to a number of computational models that characterize whether or how much blocks of two or more genome sequences are *conserved*. The most simple models require genes to occur colinearly in all blocks. Others allow some degree of variability between the blocks'

D. Doerr · J. Stoye (✉)
Genome Informatics, Faculty of Technology and Center for Biotechnology, Bielefeld University, Bielefeld, Germany
e-mail: jens.stoye@uni-bielefeld.de

D. Doerr
e-mail: daniel.doerr@uni-bielefeld.de

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_14

gene orders and gene content. Most prominent are qualitative models such as *common intervals* [18], *approximate common intervals* [10], and *gene teams* [12]. Other quantitative models include *maximum adjacency disruption*, *summed adjacency disruption*, and *breakpoint metric* [16].

A fundamental prerequisite of gene order comparison is the knowledge how genes relate to each other across all genomes of the studied dataset. Most commonly, *homology* is used as underlying relationship, although in practice, sequence similarity is used in turn as proxy for homology. In doing so, inference of homology is framed as a classification problem and thus can be tackled computationally. Initial methods treated genomes as permutations, or collections of sequences where each gene occurs exactly once. Subsequently, these models have been extended to general sequences. Today's models allow to tolerate the effects that gene duplication, loss, and gain have on the genomes' gene orders. Some models, such as proposed in [7, 9] or tools for synteny detection, e.g., i-AdHoRe [14], Cyntenator [15], MCScanX [19] do not impose the requirement of a global alphabet of gene families by allowing non-transitive relationships between genes. In the most general model, gene sequences are embedded as multi-partite graph and are connected with each other by weighted edges indicating their degree of (dis)similarity [8].

The research described in the following is motivated by our aim to develop an integrated method that may provide new biological insights. More specifically, the integrated discovery of common functional regions over multiple related genomes can help to study functional constraints of genome architecture and strengthen the predictive power in assigning specific functions to genes within these genomic regions. There can be multiple biological explanations for two or more regions that are found to be functionally similar by our algorithm: (i) All regions are descendants of a common ancestral region, i.e., are homologous. Such regions would also be found by current homology-based gene cluster methods, given that the provided homology information is adequate and correct. (ii) Multiple regions from different locations of the genome share a functional association and are jointly homologous to sets of regions from other genomes. A well-known example is the HOX gene cluster that is split up into multiple subclusters often found on different chromosomes, e.g., in human, four HOX gene clusters are found that are dispersed on four different chromosomes. While traditional gene cluster methods would be able to identify all four subclusters separately, the lack of functional information would not allow to discover their overall functional association. (iii) Regions could be functionally similar as a result of convergent evolution. Such events occur rarely in evolution, but, if found, provide deep insights into the function of their corresponding organisms. (iv) Regions may be found functionally similar, yet their genes occupy distinct subfunctions that share no commonality with others. Such regions identified by our method may be discarded by further, subsequent analysis that exploits further information in order to distinguish between too general, too specific, or wrong functional annotation.

We propose a method that minimizes both, intra- and inter-functional dissimilarity of regions from distinct genomes. More specifically, we investigate the scenario where relationships between genes are established through a structured dissimilarity measure. In doing so, we use functional classification rather than homology to

establish relationships between genes of different species. Regions with common functional association are then identified by means of *Gene Ontology* (GO) data.

The *Gene Ontology Consortium* [2] is an initiative dedicated to collect and organize the ever-increasing knowledge of gene functions into a standardized data format. To this end, the consortium maintains three continuously expanding hierarchical ontologies (*GO hierarchies*) that describe gene functions by means of their *molecular function*, their association to *cellular components*, and involvement in *biological processes*. Each of the GO hierarchies constitutes a set of *GO terms* that are used in genome annotation for classifying the function of genes, thereby allowing for varying degrees of specificity. In this work, we quantify the functional dissimilarity of sets of genes annotated with GO terms relating to biological processes. In doing so, we make use of a gene ontology based pairwise distance measure described by Diaz-Diaz et al. [6]. Regions of conserved gene order are discovered based on the concept of *gene teams* [3].

14.2 Background

We subsequently make use of the following notation: For string S of length $|S| = n$ over some alphabet Σ, $S[i]$ refers to the character at position i in S, $1 \leq i \leq n$. A *substring* starting at position i and ending at position j of S, $1 \leq i \leq j \leq n$, is denoted by $S[i, j] := S[i] \cdot S[i+1] \cdots S[j]$. Further, a *subsequence* of string S is a sequence of characters $S[i_1] \cdot S[i_2] \cdots S[i_m]$ such that $1 \leq i_1 < i_2 < \cdots < i_m \leq n$. An *indeterminate string* S is a string drawn from $\mathscr{P}(\Sigma)$, the power set of Σ, such that for each position i in S holds $\emptyset \neq S[i] \subseteq \Sigma$. The *character set* of an indeterminate string S is defined as $\mathscr{C}(S) := \bigcup_{i=1}^{|S|} S[i]$.

Let $\mathscr{G}$ denote the universe of genes. A *genome* is a collection of linear chromosomes. A *chromosome* is a string of unique genes drawn from $\mathscr{G}$, i.e., each gene is unambiguously associated with a single position in exactly one chromosome. For two genes s and t, the *genomic distance* is defined as

$$\Delta(s,t) = \begin{cases} |j-i| & \text{if } \exists \text{ chromosome } G \text{ s.t. } s,t \in G \text{ and } G[i]=s,\, G[j]=t \\ \infty & \text{otherwise.} \end{cases}$$

A GO hierarchy $\tau = (V, E)$ is a *directed acyclic graph* (DAG) with a single root node r, i.e., node r is the common ancestor of all other nodes. We denote by $P_\tau(u, v)$ the (possibly empty) set of all directed paths from u to v in τ. Given a GO hierarchy $\tau = (V, E)$, an *annotation* is a multivalued function $\alpha : \mathscr{G} \to \mathscr{P}(V)$ mapping genes to sets of vertices in τ. Moreover, for a collection of genes $(g_1, \ldots, g_m)$, a *configuration* $(a_1, \ldots, a_m)$ denotes an element of V^m such that $a_i \in \alpha(g_i)$ for all i, $1 \leq i \leq m$. Let $A = (a_1, \ldots, a_m)$ be a configuration for a set of m genes, *functional dissimilarity* [6] is defined as

$$f_\tau(A) = \frac{1}{\binom{m}{2}} \cdot \sum_{\{a,b\} \subseteq A} \frac{\min_{c \in V}(\{|X| + |Y| \mid (X, Y) \in P_\tau(c, a) \times P_\tau(c, b)\}) + 2}{l_\tau(a) + l_\tau(b)},$$

where $l_\tau(v) := \max_{U \in P_\tau(r,v)} |U| + 1$ with r being the root node of the GO hierarchy.

We are interested in finding sets of genes and their configurations that have significantly low functional dissimilarity. In doing so, we may need to test many possible gene sets and their configurations, which is computationally demanding. Therefore, we use the following upper bound of functional dissimilarity that can be computed more efficiently:

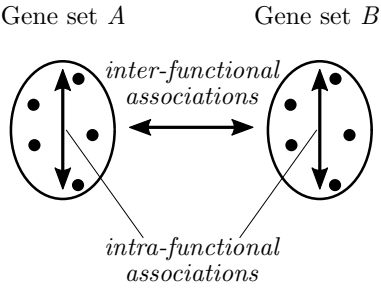
$$\bar{f}_\tau(A) = \frac{1}{m} \cdot \min_{b \in LCA_\tau(A)} \left\{ \sum_{a \in A} \frac{\min_{X \in P_\tau(b,a)} |X| + 1}{l_\tau(a) + \min_{c \in A} l_\tau(c)} \right\},$$

where $LCA_\tau(U)$ is the set of least common ancestors (LCAs) of vertex set $U \subseteq V$ in τ. Note that, unlike in general DAGs, because of the unique root node every subset of vertices is guaranteed to have at least one common ancestral node in a GO hierarchy.

Because GO hierarchies are sparse, i.e., the average out-degree of vertices is much smaller than their number, the LCAs can be computed in constant time using a preprocessed data structure [5]. As a result, the upper bound can be computed in linear time, whereas the functional dissimilarity requires quadratic time. Because we are interested in finding those sets of vertices that have a very low functional dissimilarity, in our experiments $\bar{f}_\tau$ has proven close enough to the exact measure. We will thus use the upper bound for all our calculations.

14.3 Comparative Detection of Functional Regions

We differentiate between intra- and inter-functional associations of two gene sets:



In the following, we assume that we are given a GO hierarchy $\tau = (V, E)$, a collection of genomes $S_1, \ldots, S_k$, an annotation α, a quorum threshold q, and a minimum size threshold s.

Identifying Intra-functional Associations.

A particular challenge in identifying a region of dedicated functionality is posed by the facts that it can be disrupted by unassociated genes, and that each gene can be annotated with no, or any number of GO terms of varying specificity. Number and specificity of GO terms depend much on the gene's versatility and the gathered knowledge about its function(s). As a result, identifying such a region is difficult whenever genes annotated with GO terms of higher specificity are interspersed with genes annotated with lower specificity or no annotation at all. Therefore, a discovery method must permit *gaps*, allowing to discard genes that would otherwise inflate the regions' functional dissimilarity and also permit that regions overlap with others of different associated function.

One notion of local sequence similarity that has been proposed in the comparative genomics literature and that satisfies the conditions above is that of δ-*chains* [3]: A subsequence of genes $(g_1, \ldots, g_m)$ in a chromosome is a δ-*chain* if and only if $\Delta(g_i, g_{i+1}) - 1 \leq \delta$ for all $1 \leq i < m$.

Accordingly, the problem of finding a set of genes with intra-functional association in chromosome S can be phrased as that of finding a δ-chain $(g_1, \ldots, g_m)$ in S with $m \geq s$ that has a configuration $(a_1, \ldots, a_m)$ with significantly low functional dissimilarity $\bar{f}_\tau(a_1, \ldots, a_m)$, where *significance* will be further discussed in Sect. 14.4.

Identifying Inter-functional Associations.

The varying degree of specificity in the genes' annotation with GO terms poses also a major challenge in discovering regions with common functional associations. Further, some genomes of the dataset may contain multiple copies of a particular functional region, whereas others may contain this region not at all. Thus, we formulate the problem of finding such regions as identifying a collection of gene sets $C_1, \ldots, C_l$ with configurations $A_1, \ldots, A_l$ from at least q out of k given genome sequences such that (i) each set C_i, $1 \leq i \leq l$, contains at least s genes and (ii) the sum of functional dissimilarities $\sum_{a_1 \in A_1} \cdots \sum_{a_l \in A_l} \bar{f}_\tau(a_1, \ldots, a_l)$ is significantly low. Again, our particular choice of "significantly low" will be given in Sect. 14.4.

More precisely, we aim to find those collections of δ-chains $C_1, \ldots, C_l$ with configurations $A_1, \ldots, A_l$ that have a significantly low joint functional dissimilarity $\bar{f}_\tau(A_1 \cup \cdots \cup A_l)$. We follow a heuristic strategy based on identifying LCAs of the configuration $A_1 \cup \cdots \cup A_l$. Recall that the smaller the sum of distances between GO terms of a configuration and their LCA, the lower is the functional dissimilarity of the configuration. Thus, processing the GO hierarchy in bottom-up order, i.e., from more specific to more general functions, the best candidates are identified first.

To ensure that a configuration corresponds to a collection of δ-chains, we use indeterminate strings constructed from the GO annotation of chromosomes. For a given GO term $v \in V$, we denote the set of all vertices on all paths from the root node r of the GO hierarchy to vertex v by $\phi_\tau(v) := \bigcup_{U \in P_\tau(r,v)} U$. Let S be a chromosome, then indeterminate string S^ϕ is defined position-wise as follows:

Algorithm 1 $0-$ RUNS

Input: chromosome S and associated indeterminate string S^ϕ ending with sentinel $\emptyset$
Output: all pairs $(start, end)$ of genes of 0-runs in S

```
1: initialize table RUNS: for each character in 𝒞(S^φ), create entry in RUNS corresponding to an
   empty list of gene pairs (start, end)
2: for i = 1 ... |S| do
3:   for each character c at position i in S^φ do
4:     if i = 1 or c ∉ S^φ[i − 1] then
5:       add new entry e to end of list corresponding to character c in RUNS
6:       e.start ← S[i]
7:     end if
8:   end for
9:   if i > 1 then
10:    for each character c in S^φ[i − 1] \ S^φ[i] do
11:      e ← last entry of list corresponding to character c in RUNS
12:      e.end ← S^φ[i − 1]
13:    end for
14:  end if
15: end for
```

$$S^\phi[i] := \begin{cases} \bigcup_{v \in \alpha(S[i])} \phi_\tau(v) & \text{if } \alpha(S[i]) \neq \emptyset \\ \{-\} & \text{otherwise} \end{cases}$$

Candidate regions with common functional associations are identified by discovering consecutive occurrences of GO terms in these indeterminate strings: Given a GO term $v \in V$, a δ-run of v in S^ϕ is a subsequence $S^\phi[i_1] \dots S^\phi[i_m]$ such that $v \in S^\phi[i_1] \cap \dots \cap S^\phi[i_m]$. A δ-run of GO term v is *maximal* if it cannot be extended within $S^\phi[i_1, i_m]$, nor in either direction.

The complete strategy for finding regions with common functional associations based on GO hierarchy $\tau = (V, E)$ is composed of five steps:

1. Construct indeterminate strings $S_1^\phi, \dots, S_l^\phi$ from all chromosomes of the dataset.
2. Discover all intervals in genomes $G_1, \dots, G_k$ associated with maximal δ-runs of size at least s in indeterminate strings $S_1^\phi, \dots, S_l^\phi$.
3. Discard δ-runs whose associated GO terms are ancestors of GO terms of identical other δ-runs.
4. Label vertices of the GO hierarchy with associated intervals of Step 3.
5. Process the GO hierarchy in bottom-up fashion:
   - If a vertex has intervals from at least q genomes, report this interval set.
   - If an interval set does not cover all genomes, push it to all ancestral vertices.

The discovery of maximal δ-runs (step 2) is achieved in two stages, corresponding to Algorithm 1 and Algorithm 2: First, all 0-runs are discovered, i.e., those runs that do not have a gap. Then, for each GO term, the list of 0-runs is iterated to identify

Algorithm 2 EnumerateDeltaRuns

Input: Table Runs containing GO terms and their corresponding sorted lists of 0-runs; universe of GO terms C; minimum size s; gap threshold δ
Output: List of δ-runs with minimum size s

```
1: for each GO term c in C do
2:    initialize empty list L
3:    i ← 1
4:    r ← Δ(Runs[c][i].start, Runs[c][i].end) + 1
5:    for j = 2 ... |Runs[c]| do
6:       if Δ(Runs[c][j − 1].end, Runs[c][j].start) − 1 > δ then
7:          if r ≥ s then
8:             append (∪_{i'=i...j−1} Runs[c][i']) to L
9:          end if
10:         i ← j
11:         r ← 0
12:      end if
13:      r ← r + Δ(Runs[c][j].start, Runs[c][j].end) + 1
14:   end for
15:   report L
16: end for
```

maximal δ-runs. Algorithm 1 describes the procedure to discover 0-runs in indeterminate strings in further detail. The algorithm computes table Runs, in which each character is contrasted with a list of gene pairs of chromosome S corresponding to the start and end positions of 0-runs in indeterminate string S^ϕ. The list is constructed and maintained in sorted order. After initialization of this table (line 1), all positions of indeterminate string S^ϕ are iterated. First, for each character c that has not been observed in the previous position, a new entry is appended to list Runs$[c]$ (lines 4–7), marking the position currently processed as the starting position of the interval. Then, the ending position of each interval whose character occurs only in the previous position (if such exists) is set at its corresponding entry in Table Runs (lines 10–13). If the character sets at each position of the indeterminate string are maintained in sorted order, both iterations can be performed simultaneously and the algorithm enumerates all 0-runs in $\Theta(\|S^\phi\|)$ time, where $\|S^\phi\| := \sum_{i=1}^{|S^\phi|} |S[i]|$.

Algorithm 2 describes the procedure for enumerating intervals corresponding to δ-runs of minimal size. The algorithm iterates over each GO term and processes its corresponding 0-runs independently (lines 1–16). To this end, the intervals are processed in sorted order and agglomerated as long as they are not more than δ positions apart (lines 5–14). The algorithm outputs only collections of δ-runs of minimal size m (lines 7–9). Because the algorithm processes each previously identified 0-run exactly once, the algorithm requires only $\Theta(|\text{Runs}|)$ time.

14.4 Statistical Analysis

A functional dissimilarity value is considered *significantly low* if the probability of obtaining such a value under the *null* distribution falls below a given threshold α. In doing so, the members of a set of genes of interest $G = \{g_1, \ldots, g_m\} \subseteq \mathscr{G}$ are assumed to be as functionally dissimilar as the members of any other equally sized set of genes drawn from the universal gene pool $\mathscr{G}$. Because genes can be associated with multiple GO terms, their functional dissimilarity is defined as the lowest value over all possible configurations [6]:

$$f_\tau^{\downarrow}(G) = \min_{A \in \bigotimes_{g \in G} \alpha(g)} f_\tau(A)\,.$$

In practice, we use upper bound $\bar{f}_\tau^{\downarrow}(G)$, defined analogously. Using a bottom-up processing of τ, the computation of functional dissimilarity can be tremendously sped up, which was already reported by Diaz-Diaz [6]. Still, in assessing empirical p-values, the functional dissimilarity of several millions of samples must be calculated, for which Diaz-Diaz' approach remains computationally intractable.

Observe that summands of $\bar{f}_\tau$ can be treated as random variables that are interdependent through their lowest common ancestral vertex. Trading accuracy for speed, the true distribution can be approached by assuming their independence. To this end, random samples are drawn from the sample pool that constitutes the *multiset* of pairs $(d, l_\tau(v))$ over all possible lengths d of directed paths ending at any annotation v of GO hierarchy τ that is associated to any gene of the universal gene pool:

$$\Omega = \{(d, l_\tau(v)) \mid d = 1 \ldots |X|, X \in P_\tau(r, v), v \in \alpha(g), g \in \mathscr{G}\}\,.$$

The p-value of a set of genes of size m is then estimated by drawing random samples from $\omega \in \Omega^n$ and computing their functional dissimilarity:

$$\tilde{f}_\tau(\omega) = \frac{1}{n} \cdot \sum_{(d,l) \in \omega} \frac{d}{l + \min_{(d',l') \in \omega} l'}\,.$$

In our experiments, the distribution of $\tilde{f}_\tau$ covers that of $\bar{f}_\tau^{\downarrow}$ for most gene sets except for those that are very small, as indicated in Fig. 14.1. Moreover, the figure also shows that the distribution of functional dissimilarity values of regions inferred by our proposed method (shown as distribution with circle hatch pattern) overlaps only marginally with both, $\tilde{f}_\tau$ and $\bar{f}_\tau^{\downarrow}$.

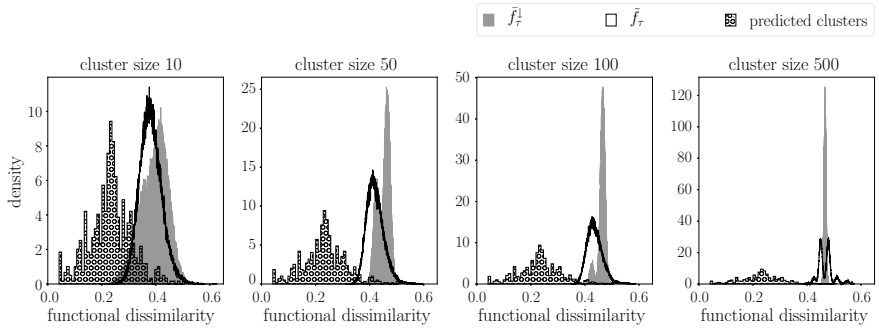


Fig. 14.1 Distributions of $\bar{f}_\tau^{\downarrow}$ (gray, filled) and $\tilde{f}_\tau$ (black, no hatch pattern) of sets of genes of size 10, 50, 100, and 500 drawn from gene ontology corresponding to domain "biological process". For comparison, the plots include the distribution of functional dissimilarity values (black, circle hatch pattern) for all 1,065 collections of regions predicted by our method with $q = 7$ in seven amniote genomes. Further details on the dataset can be found in Sect. 14.5

14.5 Analysis of Seven Amniote Genomes

Despite ongoing efforts, the number of fully (GO-)annotated genomes is limited. The Gene Ontology Consortium[1] offers annotations for only 27 genomes. Most commonly, GO annotations are transferred from one genome to another on the basis of homology (e.g. using tools such as Blast2GO [4]). Such approaches defy the efforts of this work, which seeks to compare genes based on their functional associations rather than homology. Therefore, the subsequent analysis is limited to annotations provided by the Gene Ontology Consortium and more specifically to those annotations corresponding to the domain "Biological Process".

From the fully annotated genomes provided by the Gene Ontology Consortium, we chose the largest subset that falls within a somewhat closer phylogenetic range, consisting of seven amniote genomes. The gene order information was obtained from the UCSC Genome Browser [11]. Table 14.1 provides some key information on the dataset. The number of genes per genome ranges from roughly 1,600 (dog) to over 22,000 (human). The annotations of each gene were pruned to exclude those referring to the root node of the hierarchy (GO:0008150) and those that are ancestors of other associated GO terms. After pruning, more than 80% of all genes remained annotated, and the average associated number of GO terms per position was still above 5.

We implemented our method in the programming language Python. All computations were performed on a Dell RX815 machine with 64 2.3 GHz AMD Opteron processors and 512 GB of shared memory. We ran our method with fixed minimum size threshold $s = 3$, and varying quorum $q = 3, 4, 5, 6, 7$ and gap threshold $\delta = 2, 3, 4, 5$. On all twenty runs, the method took between 10 min ($q = 7, \delta = 2$) and 21 min ($q = 3, \delta = 5$) to infer all collections of regions with common functional associations. For subsequent analysis, we grouped all collections with the same quo-

[1] http://www.geneontology.org, as of 14 Feb 2018.

Table 14.1 Genomic dataset of the seven amniotes. Columns from left to right: species name and parenthesized colloquial name; assembly version in the UCSC genome browser; number of protein coding genes; percentage of genes with one or more GO annotations; average number of GO terms associated with GO-annotated genes

Species	Version	# Genes	% Annotated	∅ Annot./gene
Gallus gallus (chicken)	`5.0`	6031	78.9	5.76
Bos taurus (cow)	`UMD_3.1.1`	13,227	80.8	5.57
Canis familiaris (dog)	`3.1`	1651	84.2	8.67
Homo sapiens (human)	`GRCh38`	22,753	85.4	7.06
Mus musculus (mouse)	`GRCm38`	21,511	80.4	6.42
Sus scrofa (pig)	`11.1`	4025	77.0	5.61
Rattus norvegicus (rat)	`6.0`	17,220	86.8	7.18

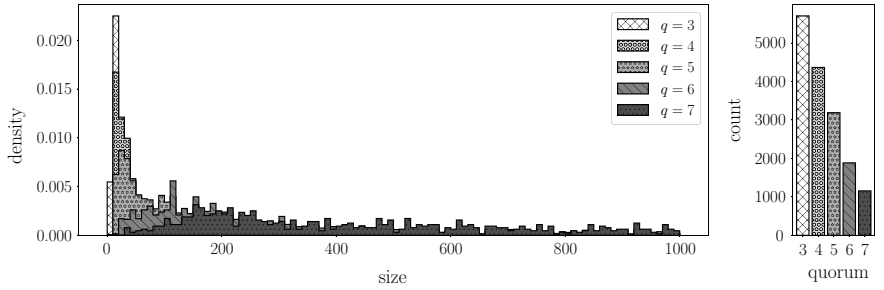


Fig. 14.2 Size density distribution (left) and total count (right) of predicted collections of regions with common functional associations under quorum settings $q = 3, 4, 5, 6, 7$

rum into a single bin. The distributions of collection sizes smaller than 1000 genes are shown in the left plot of Fig. 14.2. The right side of the figure illustrates the total number of collections per bin, ranging from 1,156 ($q = 7$) to 5,706 ($q = 3$). For all collections that contain less than 1,000 genes (5,115 for $q = 3$; 3,787 for $q = 4$; 2,614 for $q = 5$; 1,342 for $q = 6$; and 637 for $q = 7$) empirical p-values were computed. The sampling process was limited to 100 set sizes. To this end, sizes were clustered into bins using k-means. For each of the 100 sizes, 100,000 samples were drawn from distribution $\tilde{f}_\tau$, requiring in total 1 h 50 min of computation time. Out of all ($\sim$15,000) calculated empirical p-values, only 175 are larger than 1×10^{-5}.

14.6 Conclusion and Outlook

In broader perspective, this work continues the path of research devoted to *phylogenetic transfer of knowledge* (PTK), a task to which Bernard Moret has heavily contributed [20]. More specifically, the presented work studies the interaction between gene function and gene order within the realm of genome evolution. Whereas current

gene order methods make no use of functional information, this study is a first attempt at integrating such data in comparative gene order analysis. To this end, we measure dissimilarities between genes based on their associated gene ontology terms within the gene ontology hierarchy of the domain "Biological Process". By combining two strategies, one that clusters genes within the hierarchy, the other that identifies close neighborhood of genes in their respective gene orders, we developed a fast heuristic for an otherwise computationally intractable problem. We evaluated the approach on a dataset consisting of seven GO-annotated amniote genomes that have been provided by the Gene Ontology Consortium. We could show that our approach can identify many collections of regions with significantly low functional dissimilarity. These collections could be the basis of further investigation of the interplay between gene function and gene order conservation.

The presented method can be improved by ranking candidate regions by the joint probability over functional associativity and gene order conservation. Future studies towards this general goal could make use of additional functional information from databases such as UniProt, Interpro, or KEGG. This work could serve as starting point for investigations of rare evolutionary events such as convergent evolution. More specifically, one could investigate those collections of regions with common functional associations that contain non-orthologous genes.

References

1. Arst, H.N., MacDonald, D.W.: A gene cluster in Aspergillus nidulans with an internally located cis-acting regulatory region. Nature **254**(5495), 26–31 (1975)
2. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene ontology: tool for the unification of biology. Nat. Genet. **25**(1), 25–29 (2000)
3. Beal, M., Bergeron, A., Corteel, S., Raffinot, M.: An algorithmic view of gene teams. Theor. Comput. Sci. **320**(2–3), 395–418 (2004)
4. Conesa, A., Götz, S., García-Gómez, J.M., Terol, J., Talón, M., Robles, M.: Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. Bioinformatics **21**(18), 3674–3676 (2005)
5. Dash, S.K., Scholz, S.B., Herhut, S., Christianson, B.: A scalable approach to computing representative lowest common ancestor in directed acyclic graphs. Theor. Comput. Sci. **513**(C), 25–37 (2013)
6. Díaz-Díaz, N., Aguilar-Ruiz, J.S.: GO-based functional dissimilarity of gene sets. BMC Bioinform. **12**(1), 360 (2011)
7. Doerr, D., Stoye, J., Böcker, S., Jahn, K.: Identifying gene clusters by discovering common intervals in indeterminate strings. BMC Gen. **15**(Suppl 6), S2 (2014)
8. Doerr, D., Thévenin, A., Stoye, J.: Gene family assignment-free comparative genomics. BMC Bioinform. **13**(Suppl 19), S3 (2012)
9. Ghiurcuta, C.G., Moret, B.M.E.: Evaluating synteny for improved comparative studies. Bioinformatics **30**(12), i9–18 (2014)
10. Jahn, K.: Efficient computation of approximate gene clusters based on reference occurrences. J. Comput. Biol. **18**(9), 1255–1274 (2011)
11. Kent, W.J., Sugnet, C.W., Furey, T.S., Roskin, K.M., Pringle, T.H., Zahler, A.M., Haussler, D.: The human genome browser at UCSC. Gen. Res. **12**(6), 996–1006 (2002)

12. Luc, N., Risler, J.L., Bergeron, A., Raffinot, M.: Gene teams: a new formalization of gene clusters for comparative genomics. Comput. Bio. Chem. **27**(1), 59–67 (2003)
13. Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G.D., Maltsev, N.: The use of gene clusters to infer functional coupling. P. Natl. Acad. Sci. USA **96**(6), 2896–2901 (1999)
14. Proost, S., Fostier, J., De Witte, D., Dhoedt, B., Demeester, P., Van de Peer, Y., Vandepoele, K.: i-ADHoRe 3.0–fast and sensitive detection of genomic homology in extremely large data sets. Nucleic Acids Res. **40**(2), e11–e11 (2012)
15. Rödelsperger, C., Dieterich, C.: CYNTENATOR: progressive gene order alignment of 17 vertebrate genomes. PLoS ONE **5**(1), e8861–e8861 (2010)
16. Sankoff, D., Haque, L.: Power boosts for cluster tests. In: McLysaght, A., Huson, D.H. (eds.) Comparative Genomics, LNCS, vol. 3678, pp. 121–130. Springer, Berlin, Heidelberg (2005)
17. Thevenin, A., Ein-Dor, L., Ozery-Flato, M., Shamir, R.: Functional gene groups are concentrated within chromosomes, among chromosomes and in the nuclear space of the human genome. Nucleic Acids Res. **42**(15), 9854–9861 (2014)
18. Uno, T., Yagiura, M.: Fast algorithms to enumerate all common intervals of two permutations. Algorithmica **26**(2), 290–309 (2000)
19. Wang, Y., Tang, H., Debarry, J.D., Tan, X., Li, J., Wang, X., Lee, T.h., Jin, H., Marler, B., Guo, H., Kissinger, J.C., Paterson, A.H.: MCScanX: a toolkit for detection and evolutionary analysis of gene synteny and collinearity. Nucleic Acids Res. **40**(7), e49–e49 (2012)
20. Zhang, X., Ye, M., Moret, B.: Phylogenetic transfer of knowledge for biological networks. PeerJ PrePrints **2**, e401v1 (2014)

Chapter 15
Integer Linear Programming in Computational Biology: Overview of ILP, and New Results for Traveling Salesman Problems in Biology

Dan Gusfield

Abstract Integer linear programming (ILP) is a powerful and versatile technique for framing and solving hard optimization problems of many types. In the last several years, ILP has become widely used in computational biology, although predominantly by computationally and mathematically trained researchers, such as Bernard Moret. In an effort to reach a broader set of researchers, this chapter begins with an introduction to ILP, illustrated by the phenomena of cliques and independent sets in biological graphs. Then, the focus shifts to new research results on the use of ILP to solve traveling salesman problems, using *compact* ILP formulations. Such formulations have been largely declared useless in the optimization literature. However, in this chapter, I argue that the correct compact formulation can be very effective for problems of the size and structure that arise in computational biology. These empirical results, and some additional arguments, then bring into question the relevance of the concept of strength of an ILP formulation as a predictor of the speed that it will be solved.

Keywords Integer programming · Biological networks · Clique finding · Independent set · Traveling salesman problem · Strength · Beauty · Efficiency

15.1 Introduction

Bernard Moret has been, and still is, a leader in many areas in computational biology, both in biological focus, and in computational technique. Among the techniques where Bernard has been an early-adopter and early-innovator is *integer linear programming (ILP)*. He recognized that ILP can often solve *realistic-size* instances of hard computational problems in biology, even when there is no general, efficient (in a provable, worst-case sense) solution to the problem. And, he focused his understanding of the potential of ILP on very hard computational problems in biology, such as sorting genes using various rearrangement operations [45–48]. As a tribute

D. Gusfield (✉)
University of California, Davis, CA, USA
e-mail: gusfield@cs.ucdavis.edu

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3_15

to Bernard,[1] and to reflect his contributions and interest in integer linear programming in computational biology, I present here results of a study of *compact* integer programming formulations for traveling salesman problems in computational biology.

Integer linear programming (ILP) is a powerful modeling and solution method for complex problems that have long been used in industry. More recently, ILP has been used in nontraditional and inventive ways in computational and systems biology. Some examples of the uses of ILP in computational biology appear in a review, by Lancia, [29], which covers much of the literature up to 2008. A deeper treatment of a narrower range of selected applications appears in [28]. Another overview of several problems and uses in computational biology, appears in [4], written by E. Althaus, G. W. Klau, O. Kohlbacher, H. P. Lenhof, and K. Reinert. The paper [14] also surveys several computational problems in computational biology, but the problems are first cast as *quadratic* integer programming problems, and later converted to *linear* integer programming problems. A textbook level, broad introduction to ILP in computational and systems biology, will appear in [20].

This chapter begins with a brief introduction to integer linear programming through the lens of computational and systems biology questions relating to biological networks. We specifically discuss applications of the *maximum clique*, and equivalently, *independent set*, and how the later problem is solved using ILP.[2] Then, I turn to the research focus of the paper: the use of the traveling salesman problem (TSP, or TS problem) in Computational Biology, and on the effectiveness of solving TS problems[3] by *compact* integer linear programming formulations.

The results reported on the TS problem show the effectiveness of (the right) compact formulation; show that one particular compact formulation is far superior to the other well-known compact formulations; show that the common assertion that compact TSP formulations cannot solve large TS problems is not correct for all ILP formulations; show that there are compelling counter-arguments to the current belief that the *strength* of an ILP formulation is a good predictor of how efficiently the formulation will solve in practice; and show that some (mathematically correct) theoretical results on ILP strength are misleading. Thus, this paper is intended both as a contribution to computational biology and to mathematical programming.

To allow the readers to verify (or maybe object to) the results in this paper, computer code to produce the ILP formulations for the methods discussed in this paper, along with some of the test data, can be downloaded[4] from the author's website at http://csiflabs.cs.ucdavis.edu/~gusfield.

[1] And the other Bernard (the bookish nerd) in *Death of a Salesman*.

[2] The introduction is related and partly derived from several sections in [20].

[3] Terminology note: We use "TS" as an abbreviation for "traveling salesman", which is sometimes followed by "tour" or "path", as appropriate.

[4] This is a practice I recommend for all empirical, computation-based papers.

15.2 Brief Overview of ILP

Integer linear programming formulations use integer-valued variables, and linear functions defined on those variables. A linear function of a set of variables is the sum of terms, where each term is the product of one variable times one constant. For example, if the variables are $\{X, Y, Z\}$, then $7X + 5.3Y - 22.3Z$ is a linear function of those variables. An integer-valued variable is one whose value must be an integer. ILP formulations have three components:

First, an objective function, which either *maximizes* or *minimizes* a *linear* function of a (sub)set of the ILP variables; second, a set of *linear* (in)equalities (constraints), each defined on a (sub)set of the ILP variables; and third, a set of *bounds*, each defined on a single ILP variable. Each bound is actually a constraint, and so could be considered as part of the constraints, but are historically distinguished from the other constraints. When all of the variables are allowed to take on fractional values, the formulation is called a *linear program*.

15.2.1 LP- and ILP-Solvers

There are several *algorithms* that can take any concrete LP formulation and find an optimal solution; or determine that the formulation is infeasible; or that the solution value is unbounded. The first and most famous LP *algorithm* is called the *simplex algorithm*, developed by George Dantzig shortly after World War II. It is still the basis for many practical LP-solvers, although additional refinements have been made to the original method. Further, other algorithms were later developed that are based on very different ideas than the Simplex Algorithm. Some of these later algorithms have theoretical properties that the Simplex Algorithm lacks. For example, some LP-algorithms are *provably efficient* in worst-case theoretical sense (i.e., they solve the problem in worst-case time bounded by a polynomial function of the size of the LP formulation), which is a property that does not hold for the simplex algorithm, despite its efficiency in practice.

LP-Solvers

When the details of an LP-algorithm are written into an executable computer program, the program is called an *LP-Solver*. An LP-Solver takes in a concrete LP formulation (in some, usually rigid, format), and returns the value of the optimal solution, together with values assigned to the LP variables in the solution. The major LP-solvers are Gurobi Optimizer, and Cplex, which are commercial solvers but free now to researchers and students; and GLPK, which is an open source free solver.

ILP-Solvers

The LP-solvers above are also ILP-solvers,[5] returning an integer optimal solution, or determining that no integer feasible solution exists, or that the solution value is unbounded. To specify an ILP formulation, the user must specify which variables are restricted to having only integer values, or only binary values, in addition to specifying the LP formulation.

At the high level, the algorithms that ILP-solvers use to find an integer optimal solution are quite different from the algorithms used to solve LP formulations. But ILP methods usually require creating and solving many concrete LP formulations. Thus, the time to solve an LP formulation is generally much less than the time needed for the same ILP formulation where all the variables are required to have integer values. Further, unlike the case of linear programming, where theoretically efficient (in the worst case) LP-solvers have been created, no such ILP-solver exists. In fact, the problem of solving ILP formulations is *NP-hard*, but despite that, highly tuned ILP-solvers such as Gurobi Optimizer and Cplex are surprisingly effective on a wide range of specific ILP formulations, including the ones discussed in this chapter.

15.3 Biological Graphs and Networks

We now begin to discuss Biological Graphs and Networks, as a segue to discussing the maximum clique and (equivalent) independent set problems on biological networks and their solution using ILP.

Graphs and networks are used extensively in biology to represent *relationships* between biological elements, or to represent *processes* that the elements participate in. There are hundreds of *types* of graphs and networks used in biology, and *thousands* of published networks and graphs that display specific biological information. A number of such networks are discussed in [20], but here we will only mention one type, the *protein–protein intersection* (PPI) network, which is currently one of the most widely studied biological networks.

> Protein interactions form a network whose structure drives cellular function and whose organization informs biological inquiry. [22]

Protein–protein interaction (PPI) networks (which are often undirected graphs, but can have directed edges) are extremely important networks in computational biology. They depict a wide range of information. For example, they can depict *physical* contacts between *pairs* of proteins; or depict pairs of proteins that are part of the same protein *pathway* or the same *complex*; or depict pairs of proteins that are present at the *same time in the same region* of a cell; or depict pairs of proteins that are regulated together; or depict pairs of proteins that are both involved in a

[5] Another noncommercial ILP-solver (which is not an LP-solver) that has a good reputation is called SCIP, but I have not had much experience with it.

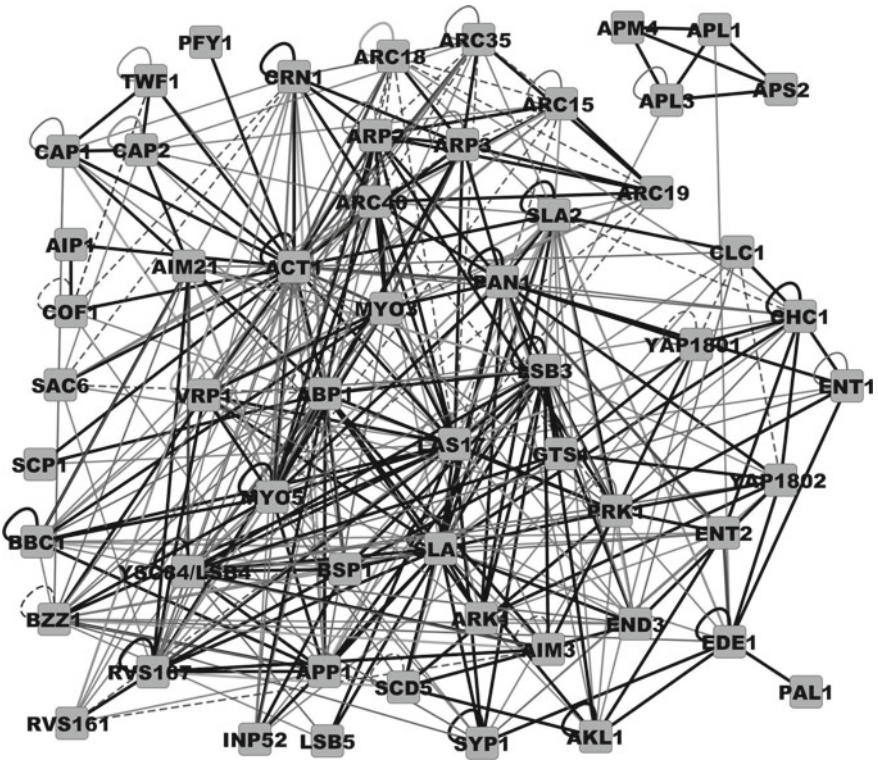


Fig. 15.1 The protein–protein interaction graph of 56 clathrin associated proteins and 386 interactions. This is a small-size biological network but already much too big for processing by hand, or by brute-force computation. Figures like this are called "hairballs", and they are not very informative. This is Fig. 15.2 from [23], used by creative commons CC0 public domain dedication

specific biological *function*; or depict the relationship of proteins that are expressed in a specific *organ* or in the *brain*. See Fig. 15.1 for an example. When edges in a PPI network are directed, the direction usually represents the *action* of one protein on another, but other kinds of information, such as chronological ordering, can be represented by edge direction.

15.3.1 High-Density Subgraphs: A Nontrivial Biological Feature

A feature of biological graphs that is widely thought to have significant biological importance is the existence of *high-density* subgraphs, which in the extreme case are *cliques*.

Cliques in Graphs

Suppose G is an undirected graph with n nodes. If there is an undirected edge between every pair of nodes in G, there will be exactly $n(n-1)/2$ edges, so every graph has that number of edges or fewer.

A *clique*, K, in an undirected graph $G = (V, E)$ is a *subset* of V (possibly all of V) with the property that for *every* two nodes u, v in K, there is an edge between u and v in G. Note that if the clique has k nodes, it will have $k(k-1)/2$ edges.

A similar concept is that of an *independent set* of nodes. This is a subset I of nodes in V such that there is no edge between any two nodes in I. Although some applications are discussed in terms of cliques, and others in terms of independent sets of nodes, the two concepts are equivalent in terms of graph theory. A clique in a graph $G = (V, E)$ is an independent set in the *complement graph* of G. The complement graph, G^c, of G is formed by adding an edge between every pair of nodes in V, (forming a clique), and then removing all of the edges in the original set E. So, the problem of finding a clique in G is equivalent to the problem of finding an independent set of nodes in G^c.

A *maximum* (or maximum-size) clique in G is a clique that has a number of nodes that are at least as large as the number of nodes in any other clique in G.

15.3.1.1 Cliques and High-Density Subgraphs in Biological Graphs

High-density *subgraphs* (which in the extreme are cliques) are probably the most studied nontrivial feature of biological graphs. There are hundreds (perhaps thousands) of publications in biology (and even more outside of biology) that use high-density subgraphs (often cliques) to define features of importance in many different kinds of networks. Many of these are discussed in [20]. Explicit applications of independent sets in biological graphs also appear in the computational biology literature, although less often than for cliques. One example is found in [42].

Functional modules and protein complexes can often be identified in graphs and networks (such as PPI networks) as highly connected subgraphs. These are discussed in detail in [20]. Additional examples of cliques in computational and systems biology that are discussed in [20] include applications to reducing inconsistencies in phylogenetics [13, 21]; to finding biological motifs in graphs; to the study of metastasis in pancreatic, ovarian, and prostate cancers [42]; to protein structure [25], and RNA folding; to the study of genes involved in epilepsy [32]; and to molecular sequence analysis [8]. Examples of high-density subgraphs that are not cliques are also discussed in [20].

15.4 The Maximum Clique and Maximum Independent Set Problems and Their Solutions Using ILP

Having motivated biological networks and graphs, and cliques and independent sets in graphs, we now turn to the question of how integer linear programming can be used to *find* them. The purpose of this discussion is to introduce and illustrate the use of integer programming on a simple, but biologically relevant, problem (before we discuss integer programming on the traveling salesman problem, where the ILP formulations are more complex). For such purposes, we could either discuss clique finding or independent set finding, since they are equivalent in the sense of graph theory. Here we take the latter approach. For the former, see [20].

The Maximum Independent Set Problem:

> Given an undirected graph G, find a *maximum-size* independent set I in G. That is, find an independent set of nodes, I, in G, that is as large, or larger, than any other independent set in G.

15.4.1 An Abstract ILP Formulation for the Maximum Independent Set Problem

The Logic

Expanding on what it means for I to be an independent set, it is required that *if* a node i is chosen to be in I, *and* a node j is chosen to be in I, *then* (i, j) *must not* be an *edge* in G. That is equivalent to saying that *if* (i, j) is an edge in G, then we *cannot* choose *both* i and j to be in I. That is the logic that we will implement with integer linear inequalities.

The ILP Variables

In the abstract ILP formulation for the independent set problem, we create one *binary* variable,[6] $I(i)$, for each node i of G.

We use variable $I(i)$ to indicate whether or not node i will be included in a selected set of nodes, called I^*. It will be clear that I^* *must* be a maximum-sized independent set in G, so the $I(i)$ variables in the optimal ILP solution will specify a maximum independent set in G.

[6] A binary variable can only be set to value 0 or 1.

The Inequalities

For each pair of nodes i, j in G, we create the following inequality if (and only if) there *is* an edge in G between nodes i and j:

$$I(i) + I(j) \leq 1. \tag{15.1}$$

To see the correctness of this formulation, note that when there is an edge between nodes i and j, inequality (15.1) would be violated if both variables $C(i)$ and $C(j)$ were set to 1. Therefore, two nodes i, j can *both* be in I^* *only if* there *isn't* an edge between nodes i and j. And since this applies to *any* two nodes included in I^*, it follows that I^* must form an independent set of nodes in G.

The Objective Function

Because want to find a *maximum* independent set, we use the following objective function:

$$\texttt{Maximize} \sum_{i=1}^{n} I(i)$$

The objective function, along with the inequalities specified in (15.1), ensure that an optimal ILP solution will specify a *maximum-size* independent set in G.

15.5 New Results on the Traveling Salesman Problem (TSP) in Biology

15.5.1 Introduction to TSP

We have discussed the components of an integer linear program, and discussed the simple, but biologically important, problem of finding a maximum independent set, to demonstrate how ILP formulations are created. Now we begin to discuss new results on the *Traveling Salesman Problem (TSP)*. We will mention several uses of TSP in computational biology and then discuss ILP formulations and methods for solving concrete TSP formulations.[7]

The classical traveling salesman problem (TSP) requires finding a *minimum-cost* route for a salesman to *visit* each city in a given set of cities (nodes in a graph) *exactly* once. Input to a problem instance is a map (or graph) showing which pairs of cities (nodes) are directly connected by a road (edge), and the cost of traversing each road.

[7]The introductory material on TSP, and the descriptions of TSP formulations, are extracted from [20]. The research results and conclusions are new.

For now, we assume that the cost of traversing a road is the same in either direction. The cost of a route that visits each city once is the sum of the costs of the roads used on the route. Clearly, for any graph with more than two nodes, no edge can be traversed in both directions.

The TSP comes in two high-level variants. In the TS *Tour* (cycle) version, the salesman is required to start and end at the same city, hence traveling around a tour, visiting each city exactly once. In the TS *Path* version, the salesman is required to start and end at *different* cities, hence traveling a path, visiting each city exactly once. Further, there are two variants of the TS path problem, one where the start and end cities are specified as part of the problem input, and one where they are not.

15.6 The Traveling Salesman Problem in Genomics

With current genomic technology, many tasks involving DNA maps, sequences, and sequencing lead to computational problems that concern the *ordering* or *permutation* of DNA sequence *fragments*. Most problems come from experimental techniques in genomics that give information (often partial, or containing errors) about (possibly overlapping) fragments of a linear molecule (e.g., a chromosome). However, the experiments don't explicitly determine how those fragments are ordered on the chromosome. Other experimental techniques give information (often partial) about the *relative* positions of molecular *markers* (sites) in a linear molecule but again do not fully specify the complete order of those markers. The computational problems must use such information to try to *deduce* the correct, complete order of the fragments or markers in the chromosome.

The specific details of the experimental techniques generating fragment data determine specific details of the computational problems. However, there are many diverse experimental techniques and they change rapidly. But regardless of the experimental basis for the data, most of the ordering problems come down to trying to determine the best *permutation* of some set of elements, *subject to* various *constraints* on what permutations are permitted. Often, these permutation problems in genomics are naturally modeled as *traveling salesman problems* (TSP), or variants of the TSP. There are also TS-like problems that arise in computational biology.

15.6.1 Examples of the TS Problems in Computational Biology

There are two well-exposed applications of the TSP in computational biology. The first is a very important problem in DNA sequencing, the *DNA assembly problem*. The second is the *marker-ordering* problem, studied in [3]. Both of these problems, and their relationship to TS problems, are discussed in detail in [19] and [20], and

won't be repeated here. However, we report empirical results for the *marker-ordering problem*, solving it as a TSP using several different ILP formulations.

Additional genomic applications of TS problems (that we will not discuss) appear in [7] and [43], where methods for gene rearrangements (reversals, transpositions, translocations, etc.), and for building phylogenies based on rearrangements, successively formulate and solve TS problems; in [31], where TSP is used to deduce a plausible *signaling pathway* in cervical cancer; in [1], where TSP is used to model *radiation-hybrid mapping* (which is a technology that is no longer current, but the structure of the problem is similar to the marker-ordering problem, and will likely arise in future applications); in [16], where an optimal TS tour reconstructs a putative ordering of genes expressed during the cyclic *cell cycle*; in [24], where an optimal TS path is used to order *protein interaction data* in order to reveal common function; in [26], where a near-optimal *multiple sequence alignment* (under the sum-of-pairs objective) is built from a TS tour through the sequences; and in a related paper, [27], where an optimal TS tour is computed as a first step in a method to build phylogenetic trees. Those trees solve the *weighted maximum parsimony problem* in phylogenetics, when the edge scores satisfy certain technical conditions that the authors believe are biologically meaningful.

Finally, being a chapter in a festschrift for Bernard Moret, it would almost be criminal not to note here the paper [34] by Bernard, together with David Bader and Tandy Warnow, that describes algorithm engineering to dramatically reduce the time needed for the computations in the abovementioned paper [7] for finding the "breakpoint phylogeny", a problem that arises in the context of constructing phylogenies for a collection of genomes that have evolved under rearrangement events. The algorithms described in [34] for the breakpoint phylogeny involve formulating and solving many instances of the TS tour problem. See the chapters in this book by Jijun Tang and by Zeira and Shamir for more about the breakpoint phylogeny and other genome rearrangement phylogeny problems.

15.7 Solving TS Problems with Integer Linear Programming

15.7.1 DFJ: The Classical ILP Formulation

The first and most highly studied ILP formulation for the TSP is due to Dantzig, Fulkerson, and Johnson (DFJ) [12], in 1954. It is shown in Fig. 15.2. This formulation works for both directed (asymmetric) and undirected (symmetric) graphs.

The first two sets of constraints are called "assignment constraints", which say that each node must be entered exactly once, and exited exactly once. The third set of constraints are the "subtour elimination" constraints which ensure that no assignment of values to the F variables defines a cycle of length less than n. Hence, any feasible solution to the constraints defines a *single* cycle containing all n nodes.

$$\texttt{Minimize} \quad \sum_{i \neq j} D(i,j) \times F(i,j)$$

Subject to
For each node i in V,

$$\sum_{j \neq i} F(i,j) = 1$$

and

$$\sum_{j \neq i} F(j,i) = 1$$

For each non-trivial subset of nodes, $S \subset V$,

$$\sum_{i,j \in S} F(i,j) + F(j,i) \leq |S| - 1$$

All variables $F(i,j)$ are binary.

Fig. 15.2 The *Full* DFJ formulation for the TS tour problem. $D(i,j)$ is the cost of of traversing an edge from node i to node j; and $F(i,j)$ set to value 1 has the meaning that node i is followed by node j in the tour

Note that the number of inequalities in this formulation grows *exponentially* as a function of n. Therefore, this *full* variant of the DFJ formulation can only be used on very small problem instances. We tested it for $n = 20$. In practice, the DFJ formulation is used with a *separation method*, which starts by solving an ILP with only the assignment constraints. If the assignment solution contains exactly one cycle, then an optimal TS tour has been found. Otherwise, find a cycle (subtour) in the ILP solution, and introduce the subtour elimination constraint that prohibits that cycle. Then resolve the larger ILP. Iterate this, (successively solving larger ILP formulations, checking for subtours, and adding violated subtour elimination constraints) until obtaining an optimal solution to the expanded ILP, consisting of only a single cycle. At that point, the cycle must be a optimal TS Tour.

The separation strategy based on the DFJ formulation works spectacularly for TS problems, dominating, in terms of speed, every other known ILP formulation for TS problems. Still, there are reasons to be interested in those other formulations.

15.7.1.1 DFJ Contrasts with Compact ILP Formulations

Since the size of the full DFJ formulation grows exponentially with n, and the separation strategy is somewhat complex, many other formulations for the TS problem have been devised, where the number of variables and inequalities grows *polynomially* with n. These are called "compact formulations". About 30 compact formulations for TSP are known, and interest in compact TSP formulations continues. See [35] for a description of 24 compact TSP formulations.

Why are compact formulations still of interest given the dominance of the DFJ formulation solved by the separation technique?[8] We quote from [40]:

> An efficient implementation of a separation routine for subtour inequalities is beyond the ability of most practitioners and undergraduate/MS students.[9] ... most problems encountered in practice are not pure TS problems, but only TS-like. ... For TS-like problems the TSP formulations are usually not hard to generalize, but the TSP separation routines cannot be used.

Further, the program Concorde cannot be modified by users for other TS-like problems. More generally, understanding other TSP formulations and their behaviors expands our repertoire of tools that can be used for mathematical optimization. Finally, for many problems in Computational Biology, the best compact formulations suffice and are easier to use and modify, even if they are slower than DFJ with separation.

15.7.1.2 The New Results in This Paper

The main results are the following:

1. One of the compact formulations, called *GG*, consistently and often dramatically, solves faster than the others, under *all* the conditions tested. This empirical observation is *inconsistent* with the accepted belief that the *strength* of an ILP formulation (discussed in Sect. 15.11) is a good predictor of its efficiency in practice. The empirical observation is also inconsistent with statements in the literature which say that *compact* (weak) TSP formulations are "nearly useless". In particular, *GG* is practical for TSP instances much larger than have been previously tried using compact TSP formulations, and is quite practical for instances of TSP that currently arise in computational biology.
2. Comparisons of other compact TSP formulations also show the weakness of strength as a predictor of efficiency. Moreover, the spectacular effectiveness of the DFJ TSP formulation, when combined with a separation strategy, is not explained by appealing to the strength of the *full* DFJ formulation.
3. Some of the theoretical results on strength that have been established in the literature are less informative than they may at first appear.

15.8 A Compact ILP Solution to the TS Tour Problem on G'

Graph G' is the input graph G augmented with a new node 0, and an edge from node 0 to each of the nodes in G. The weight of any of those edges is zero. A solution of a TS

[8] Moreover, a freely available, highly engineered program called *Concorde* mixes many techniques and tricks to solve very large TS problems in practice.

[9] This remains true in 2018, according to people who teach courses on ILP.

tour problem on G' is a solution to a TS path problem on G. Since in computational biology problems we are more interested in paths than tours, we give special attention to TS tour problems on G'.

15.8.1 The GG TSP Formulation

We start by displaying a *complete ILP formulation* for the TS Tour Problem, shown in Fig. 15.3. This formulation is called the *GG* formulation, and is considered *compact*, because the number of variables is bounded by $O(n^2)$, as is the number of inequalities. This formulation was developed in [17], a technical report from MIT that was never published.

The constant $D(i, j)$ is the cost of traversing edge (i, j) from node i to node j. The binary variable $F(i, j)$ is set to 1 to indicate that node i is followed by node j in the tour. Variable $b(i, j)$ can be interpreted as the number of cities left to visit when entering node j from node i.

Inequalities (i) and (ii) are the inequalities of the assignment ILP; inequality (iii) sets variable $b(0, j)$ to n, for the unique node j where $F(0, j)$ has value 1; the inequalities in (iv) force the value of $b(i, j)$ to be 0, *unless* the value of $F(i, j)$ is 1; and the inequalities in (v) say that for each node $j > 0$ in G, exactly one unit of whatever it is (snake oil perhaps) that the salesman is carrying must be delivered to each city. Note that in (v), index i can have value 0, although j must be in the range 1 to n. Inequality (o) says that an edge can be traversed in only one direction. Although it is natural to include it, it is not part of the assignment formulation. This will be discussed next.

It is possible to interpret the inequalities in (v) in a way that is more consistent with several other ILP formulations, both for TSP and other problems. In that interpretation, the inequalities in (v) describe a single-commodity flow originating at the root node 0, where the other nodes are sinks, each consuming one unit of flow. For a more general discussion of these kinds of models, see [5].

Well, Not Quite

The version of the GG formulation shown in Fig. 15.3 is not quite the same as the version considered in some of the ILP literature. In particular, inequality (o) is redundant and can be omitted. That natural inequality simply states that an edge can be traversed in *only* one direction, which is almost a fundamental part of the *definition* of a TS problem.[10]

The difference between the variants of GG that include or don't include the inequalities in (o) turns out to *be important*, as we will discuss in Sect. 15.11. Since

[10]Certainly, if one states the TS problem to students without including that constraint, an alert student will ask about it.

$$\texttt{Minimize} \sum_{edge\ (i,j)\ in\ G'} D(i,j) \times F(i,j) + D(i,j) \times F(j,i)$$

such that:

(o) For each edge (i, j):

$$F(i,j) + F(j,i) \leq 1$$

(i) For each node i, from 0 to n:

$$\sum_{j \neq i} F(i,j) = 1$$

(ii)

$$\sum_{j \neq i} F(j,i) = 1$$

(iii) For each node j from 1 to n:

$$b(0,j) = n \times F(0,j)$$

(iv) For each edge (i, j):

$$b(i,j) \leq n \times F(i,j)$$
$$b(j,i) \leq n \times F(j,i)$$

(v) For each node j from 1 to n:

$$\sum_{\texttt{edge}\ (i,j)} b(i,j) - \sum_{\texttt{edge}\ (i,j)} b(j,i) = 1$$

All F variables are binary, and all b variables are integral with values between 1 and n.

Fig. 15.3 A complete abstract ILP formulation for the traveling salesman problem on graph G'. This formulation is called the GG formulation after the authors who developed it in an unpublished MIT working paper [17]. The method is also detailed in [2]. This formulation works for both directed and undirected graphs

inequality (o) is natural, and helpful (as we will see), and only adds $\Theta(n^2)$ inequalities to a formulation that is already of that size, I have included it in all of the simulations I did of GG, except as noted. I call the version of GG without (o) the *pure GG*.

15.8.2 The MTZ Formulation

The first compact ILP formulation for the TS problem was published in [33], and is shown in Fig. 15.4. It is the most widely known and most studied compact ILP formulation for the TSP. The *Wikipedia* discussion of the TSP presents this formulation as the exemplar of the compact formulations for TSP. The textbook [11] discusses MTZ in depth but does not even mention the GG formulation or cite [17]. But, as we will see, the MTZ formulation is inferior in practice to the GG formulation. One hoped-for contribution of this chapter is to replace MTZ with GG as the accepted exemplar of compact TSP formulations.

$$\texttt{Minimize} \quad \sum_{edge\ (i,j)\ in\ G'} D(i,j) \times F(i,j) + D(i,j) \times F(j,i)$$

such that:

$$(o) \quad U(0) = n+1$$

(i) For each node i, from 0 to n:

$$\sum_{j \neq i} F(i,j) = 1$$

(ii) For each node i, from 0 to n:

$$\sum_{j \neq i} F(j,i) = 1$$

(iii) For each node $i \neq 0$:

$$1 \leq U(i) \leq n$$

For each ordered pair of nodes (i, j) where i and j each range from 1 to n:

$$(iv) \quad F(i,j) + F(j,i) \leq 1$$

$$(v) \quad U(i) - U(j) + n \times F(i,j) \leq n-1$$

All F variables are binary, and all U variables are integral.

Fig. 15.4 The *MTZ* ILP formulation [33] for the TS Tour Problem on G'. This is the compact formulation discussed in Wikipedia

As in the case of inequality (o) in GG, the inequalities in (iv) say that an edge can be used in only one direction. They are redundant, but natural, and I include them in the simulations I do for the MTZ formulation, unless otherwise noted. The version of MTZ without (iv) is called the *pure MTZ*.

At first exposure, the MTZ formulation seems very similar to the GG formulation. The major difference is the use of $U()$ variables in MTZ, and the use of $b()$ variables in GG. But each $U()$ variable is associated with a *node*, while each $b()$ variable is associated with an *edge*. The meaning of inequality (v) in the GG formulation is that *if* the tour traverses edge (i, j) from i to j, and then traverses edge (j, k), for some k, then the value of $b(i, j)$ must be *exactly* one larger than the value of $b(j, k)$. So, those values specify (in reverse) the order that the edges (and hence the nodes) are traversed in a solution to the GG formulation.

In contrast, the meaning of inequality (v) in the MTZ formulation is that *if* the tour goes from node i to node j along the edge (i, j), then $U(j)$ must be larger than $U(i)$. In the end, because of inequality (iii), which says that all the $U()$ values must be between 1 and n, the values of the $U()$ variables give the order of the nodes in the tour that is found. But, that consequence is less direct and explicit than in the GG formulation. That is, to me, the most significant difference between the GG and MTZ formulations. Why that leads to faster solutions for GG formulations, compared to MTZ formulations, is unclear (to me).

15.9 Empirical Results

We examined the empirical behavior of *five* different compact ILP formulations for the TSP problem, in comparison to both the *full* DFJ formulation, and its solution with the separation approach. The compact TSP formulations are GG and MTZ, already described, and ILP formulations called FGG3 and FGG4 [15], and CLAUS [10]. The five compact formulations were selected because of their prominent roles in the literature; for their range of sizes (number of variables and inequalities); and for the different theoretical strengths (to be discussed later) established in the literature. Several other compact formulations[11] were also implemented and initially examined, but their empirical performance was so bad that they were quickly discarded.

Extensive empirical tests were conducted using three types of data: (1) Randomly generated instances of the *marker-ordering problem*, and a specialization of it called the *Consecutive Ones Problem*; (2) Instances of TS problems on *randomly* generated graphs with varying numbers of nodes and edge densities; (3) Instances of TS tour and path problems from established *benchmark* test sets of varying sizes, in both undirected and directed graphs.

15.9.1 Results for the GG Formulation

The empirical results for the GG formulation, both by themselves and in comparison with the other four compact formulations and the full DFJ formulation, establish it as the best compact formulation, by far. I first discuss results for GG alone, and then in comparison to other formulations.

Random Graphs

First, I tested the ILP formulation on *random* graphs of different sizes, where each potential edge was selected to be in the graph with varying probabilities. Details of these tests are in Table 15.1. As a sample of the results, consider the case where the probability of each edge is 0.25, and edge costs assigned randomly (uniformly) in a range from 1 to 100. For $n = 20$, the GG ILP solves in under a *one-tenth* of a second, averaged over 10 trials; for $n = 50$, it solves in under *half* of 1 s, on average; for $n = 100$, the ILP solves in 2.27 s, on average; for $n = 200$, it solves in 64 s, on average; and for $n = 500$, it solves in about 42 min, on average. These numbers illustrate the tremendous efficiency of the GG TSP formulation, compared to *brute-force enumeration* and examination of all $n!$ (n-factorial) potential TS tours in a graph with n nodes. For example, according to Wikipedia $n!$ is 2,432,902,008,176,640,000 (a huge number) for n equal to 20 (a small number). For $n = 100$, $n!$ is larger than 10^{157}, a truly gargantuan number.

[11] Including one that I came up with, which had nearly the worst performance of all.

Benchmark Tests

I also tested the GG TSP formulation on some of the classic benchmark (nonbiological) problem instances, from city data in TSPLIB [41], and from the National TSP Collection at U. Waterloo. The results were more varied but still demonstrate the practicality of the GG formulation for instances of meaningful size. These results contradict expectations stated in the literature. Details of these tests are shown in Table 15.2.

Surprising Results

The first dramatic result for the GG formulation is for benchmark problem called *br17* with 17 cities. The expectation in the literature is expressed by the quote:

> ... a small instance, (such) as br17, can be surprisingly hard using a weak formulation ... (If one wants to assign only one problem in a student project, this probably should be the one!) [40]

But, using the GG formulation, the optimal TS *tour*, with cost of 39, was found using Gurobi 7.5 on my MacBook pro, in 0.21 s, and the optimal TS *path* problem was solved in 0.11 s.[12]

The second dramatic result for the GG formulation is for the benchmark problem called *p43*. The author of [39] points to *p43* as a "*particularly hard*" TS problem instance to solve, and states that it was unsolvable using the MTZ formulation. A more recent paper [44] (with empirical tests) in 2016, calls it the "the most challenging problem", and also states that it was unsolvable with the MTZ formulation.

However, the concrete ILP formulation of "the most challenging problem", p43, using the GG formulation (which was not examined in either [39] or [44]) was solved in 20 s on my laptop using Gurobi 7.5, and in 10 s using Gurobi 8! The TS path version of the p43 problem was solved in 3.14 s with Gurobi 7.5.

Other Results

Highlighting other results for the GG formulation on benchmark problems: instance *wi29* with 29 cities in the western Sahara solved in under *one* second; *berlin52* with 52 locations in Berlin, solved in 3.2 s; problem instance ch130, with 130 cities in China solved in under *seven minutes* (with Gurobi 7.5), but took about 12 min with Gurobi 8 (why? You got me!). The path version of ch130 solved in under *six* minutes with Gurobi 7.5, and under 4 min with Gurobi 8.

---

[12] Certainly, even my laptop is a faster machine than the one used in 2003 by the author of [40]. But, the increased machine speed does not account for the difference between the observation today, and the understanding in 2003. It should be noted however, that my experience with MTZ on br17 also contradicts the statement in [40], since the MTZ formulation for *br17* solved in 0.54 s.

These benchmark results also establish the practicality of the GG formulation. However, the GG formulation was considerably slower for benchmark problem *a280*, with 280 circuit-board drilling points. Gurobi 6.5 solved it in 44 h. This is long, but still practical for many applications; and the execution took under 2 h to reduce the gap between the best solution value (*ub*) and the best lower bound (*lb*) to 5.98%. Further, the feasible solution that was found after *three* hours was in fact the *optimal* solution, although it took another 41 h to get a matching lower bound, *lb*. After Gurobi 7.5 was released, I reran the ILP for *a280*, and it solved in only 28 h, although this time it did not attain the optimal solution until the 27 h mark. Then it took another hour or so to attain a matching lower bound.

Marker-Ordering

I also tested simulated data that roughly mimic variants of realistic *marker-ordering* problems, which translate into instances of TS problems with 500 nodes. Using the GG TSP formulation, Gurobi 6.5 solved each of the generated problem instances in at most *seven* minutes. The number of nodes in these tests is much larger than the number of nodes in nonbiological TS problems which are previously reported to be solved by compact TSP formulations. This suggests that TS problem instances that arise in biology are *easier* to solve than the classic *benchmark* instances. This is because many problems in biology have more *structure*, and the set of solutions are less *symmetric*, and they have lower *density* (the ratio of the number of edges to the number of nodes) than do the benchmark problem instances. Benchmark problems are mainly Euclidean distance problems or problems with very high symmetry, such as chip design and circuit-board drilling.

15.9.2 Comparing Empirical Results for the Other Compact Formulations

Tables 15.1 and 15.2 give a detailed comparison of the efficiency of three compact TSP formulations, showing clearly the superiority of the GG TSP formulation.

Random Graphs

Comparisons of GG, MTZ, and CLAUS on *random* graphs are shown in Table 15.1. Our interest in GG and MTZ has been discussed earlier. CLAUS is of interest because it *provably* has the *same* strength (discussed later) as does the full DFJ formulation [35, 38], and the strength of DFJ is commonly cited as the reason for its success when used with branch-and-cut and branch-and-bound [38].

The data show that CLAUS is much slower, and quickly becomes impractical, compared to GG and MTZ, as the number of nodes and edge density increase. For example, for complete graphs with 50 nodes, the average solution time for GG is under 1 s, for MTZ it is under 2 s, but for CLAUS it is over ten minutes. For $n = 100$ with density of 0.25, the average solution time for GG is only 2.27 s, for MTZ, it is under 15 s, but is one hour and 25 min for CLAUS. For complete graphs with one hundred nodes, the respective average times for GG and MTZ are 11 s, and 40 s, but is 3 h and 40 min for CLAUS. Because of those results, and similar results on benchmark data, CLAUS was not tested for random graphs with more than 100 nodes.

The data also show that MTZ is considerably slower than GG under all test parameters, but the gap is reduced as the graph gets denser. For example, for $n = 400$ and density = 0.1, the average solution time for GG is about nine minutes, and about 200 min for MTZ, so MTZ takes about 22 times longer than GG. But when the density is increased to 1, the respective averages are about 47 min and 267 min, so MTZ takes 5.7 times longer than GG.

In a different trial of 500 randomly generated graphs with 60 nodes and edge density of 0.7, the GG formulations took an average of 0.18 s to solve, while the MTZ formulations took an average of 2.22 s. These results again show that MTZ is slower than GG, but not fatally slower. However, the *maximum* time for those 500 GG formulations was 6.75 s, while for the MTZ formulations it was 3606 s, with another taking 534 s.[13] So MTZ can occasionally be unreliable, while GG appears quite stable in comparison.

Benchmark Data

The results for benchmark data (excluding the data for *a280*) are shown in Table 15.2. Even *more* convincingly than the results on random graphs, these results show that GG is superior to the other compact formulations on all the benchmark data. In particular, they show the inefficiency and unreliability of the FGG4 and CLAUS formulations, and the widening gap between GG and MTZ as the problem size increases. The table also shows the time for the *pure* DFS formulation on the smallest problem, and the times for DFJ with the use of separations, on some of the larger problems.

In the test of the TS *Path* problem, using *br17* data, Gurobi solved the ILP formulation in 0.11 s using the GG; in 0.54 s using MTZ; in 1.52 s with CLAUS ; in 3.83 and 5.17 s using the FFG3 and FFG4 formulations, respectively; and in 44.87 s using the full DFJ formulation. Note that these results are inconsistent with the statement in [39] "... a small instance, (such) as br17, can be surprisingly hard using a weak formulation ... [40]"

[13] On the instance where MTZ took over 1 h, the GG formulation took 2.38 s to solve (with Gurobi 8), and the DFJ formulation with separation took 0.02 s. So, DFJ with separation is unquestionably dominant, but the speed of GG here contributes to the new understanding that the right compact TSP formulation is practical, while the instability of MTZ makes it much less reliable.

The data on random graphs showed that MTZ is consistently slower than GG, and occasionally unreliable. But the benchmark results suggest that MTZ is unreliable more frequently. Recall that the author of [39] points to the benchmark TS problem *p43* as a *particularly hard* TS problem instance to solve, and states that it was unsolvable using the MTZ formulation. Another paper [44] reports that two variants of the MTZ formulation

> ... failed to prove optimality for the most challenging problem p43.atsp.

And, in fact, when I tried to solve instance *p43* using the MTZ formulation, Gurobi 7.5 quickly stalled with a gap of about 54%—it stayed there for nine hours before I killed the execution.[14] So, that observed performance is consistent with the point made in [39], that the MTZ compact ILP formulation is "nearly useless".

But, the larger point made in [39, 40] and in other discussions of ILP, is that *weak* (to be discussed later) ILP formulations are often useless, and their use of MTZ is just an exemplar of that point. However, GG is *also a weak* ILP formulation, but using it, Gurobi 8 solved p43 in *ten* seconds on my laptop. This illustrates the difficulty of using theory to predict which ILP formulations will solve well in practice. We will discuss this more in Sect. 15.11.

Marker-Ordering

The results using marker-ordering and consecutive ones problems are consistent with those on random graphs and benchmark data. In my experiments, when concrete instances of the *MTZ* formulation can be solved, the ILP-solver takes *much* longer to finish, as compared to the GG formulation. Further, the *MTZ* formulation does not terminate on large problem instances, where the GG formulation does. As one of many examples, I created a random instance of the *consecutive ones* problem, with 300 rows, 300 columns, and 30 entries of value 1 in each row. Using the GG formulation, Gurobi 6.5 solved the problem instance in 100 s (the newer Gurobi 7.5 took 186 s—go figure!). However, using the MTZ formulation, Gurobi 7.5 took more than 6.5 h to finish. It arrived at the optimal solution in about 1.5 h, but still didn't have a matching lower bound for another 5 h—a gap of 1.16% remained for those five hours.

Similarly, the FGG4 formulation [15] was vastly slower than the GG formulation. It could not solve even modest-size problem instances and was much slower on small instances it could solve. For example, for an instance of the consecutive ones problem in a 20-by-20 matrix with five 1s per row, Gurobi 7.5 solved the GG formulation of the TSP in 0.07 s, but took 256 s to solve the FGG formulation. Cplex 12.6 solved the GG formulation in 0.08 s, but with the FGG formulation, it ran for two hours without even finding a first feasible solution and was then terminated (for cause!).

[14] A later attempt with Gurobi 8 suffered the same fate.

Good Intuition is Elusive

The compact TSP formulations GG and FGG provide another illustration of how difficult it is to have reliable *intuition* about which ILP formulations will solve well. The FGG formulations published in [15] are vastly inferior to the GG formulation on all of the moderate and large-sized tests that I performed. But, the GG formulation was only written in an *unpublished* working paper [17], while the FGG formulations were published in a prominent journal. Further, the GG formulation is related to the FGG formulations, and since the two authors who developed the GG formulation later (along with a third coauthor, K. Fox) developed the FGG formulation, the GG formulation could have naturally been discussed in the published paper. This suggests that the authors, or the reviewers, did not appreciate how superior GG was to FGG.

15.10 Take Home Lessons

The empirical results conclusively show that FGG3, FGG4, and CLAUS are "essentially useless" in comparison to GG, and even in comparison to MTZ, except on very small problem instances. Overall, the empirical results for the GG formulation are amazing and surprising. They contradict what I was taught and believed until recently, that more sophisticated solution methods (such as the use of separations) would always be *required* to solve any but the smallest TSP instances. For example, in 2003, it was generally accepted [40] that any compact ILP formulation would only be able to solve TS problems for 50 cities or less.

> Solving a reasonably large (with at least, say, 50 cities) problem to optimality is only possible using the subtour formulation; [40]

Now we know that this is no longer true, and the consequences for computational biology can be truly transformative. These empirical results show that a compact ILP approach can solve a wide range of TS problem instances that model real phenomena in biology (particularly genomics). However, this knowledge comes with two caveats.

Caveats

First, the choice of ILP-solver *really* does matter. In my experiments, Cplex 12.6 was able to solve TS tour instances on random graphs of size up to $n = 200$ as efficiently as Gurobi 6.5, and on some problem instances it was faster than Gurobi. But it was unable to solve instances of size $n = 500$ that Gurobi solved in under 25 min, even allowing Cplex to run for more than a day. And, when terminated, the gap between the best solution obtained and the lower bound was still large (in the 50% range). Also, the newer Gurobi 7.5 was sometimes slower than Gurobi 6.5 (taking about twice as much time), and when it was faster, it was only by a small amount. So, the solver matters, and the most recent release is not always the best.

Second, the specific compact, abstract ILP *formulation* matters. The GG formulation leads to concrete ILPs that can be solved *much* faster in practice than the

other similar formulations, even though all the formulation are mathematically correct. In our, genomic-oriented, experiments, the GG formulation consistently solved faster, and sometimes *vastly* faster, than the other four compact formulations we tried. So, the ILP approach can be practical on problem instances of meaningful size and structure in biology, but getting that result may require art as well as science; and sometimes you have to experiment with several different ILP formulations, and different ILP solvers.

15.11 On Strength

> When I am weak, then am I strong. (2 Corinthians)

Several papers in the mathematical programming literature concern the efficiency of different compact ILP formulations for the TSP [18, 30, 35–38, 49]. In particular, they concern the relative *strengths* and *execution times* of pure compact formulations, compared to the classic, but non-compact, DFJ formulation [12], solved using the separation method discussed in Sect. 15.7.1.

Strength is measured by the *difference* between the value of the optimal *integer* solution to an ILP, and the value of the optimal *fractional* solution to the *relaxed* LP problem (i.e., where the variables can take on nonnegative fractional values). The smaller the difference, the greater the strength of the formulation.

The current belief is that the stronger the formulation, the faster the ILP will solve, especially in the context of a branch-and-bound or branch-and-cut solution strategy [6]. Branch-and-bound is the high-level solution strategy that ILP-solvers generally employ (supplemented with other techniques such as the use of cutting planes). Those strategies repeatedly change the ILP formulation and compute optimal LP (not ILP) solutions to use as bounds to guide the branching process. It is therefore intuitively appealing to think that stronger formulations will result in faster overall solution times for the ILP, and given two ILP formulations for the same problem, the stronger formulation is generally thought to be best. This is particularly believed if the two formulations have a similar number of variables and inequalities (which impacts the speed of the LP computations). Consequently, the relative strengths of several compact formulations for TSP have been *theoretically* established, and widely taught as important for mastering effective integer programming [6, 11, 39, 40]. This belief is reflected in the following quotes:

> There are several lessons to be learned from these experiments: ... the foremost one is the connection between the strength of a formulation and the required solution time. [40]

> Although the MTZ formulation is correct, we will show ... that it produces weaker bounds for branch-and-cut algorithms than the DFJ formulation. It is for this *reason* that the latter is preferred in practice. [11]

And later, after establishing the fact that MTZ is weaker than DFJ, they state:

> It is therefore *not surprising* that the MTZ formulation is effective only for small values of n. For large instances, tighter formulations such as $P_{subtour}$ (DFJ) are needed. [11]

Elsewhere in [11], they state that "medium to large" instances are those with $n > 30$, suggesting that "small" here means $n \leq 30$.

> Solving a reasonably large (with at least, say, 50 cities) problem to optimality is only possible using the subtour formulation; at least, we are not aware of any published computational studies that use the pure MTZ formulation. There is an *analytical explanation* of this fact: the LP relaxation of the MTZ formulation is much weaker. [40]

In [36], in discussing a variant of FGG3 and FGG4, the authors state:

> ... it is also remarkably bad in terms of the strength of its Linear Programming relaxation and *therefore* the slowness of its overall running time.

The choice to italicize "reason", "not surprising", "analytical explanation", and "therefore" is mine, and is done to emphasize the general acceptance of the dogma that the relative strength of ILP formulations is what determines efficiency in practice. These specific quotes relate to the relative strength of *full* DFJ formulation to pure MTZ, rather than to GG, but pure GG is *also weaker* than the *full* DFJ [35]. So, conclusions about MTZ *based* on the relative weakness of pure MTZ compared to full DFJ should hold for GG as well. More generally, the view that the strength of an ILP formulation is critical is reflected in the following:

> The main message of this chapter is that strong formulations are central to being able to solve integer optimization problems efficiently. The quality of a formulation is judged by the closeness of its linear relaxation to the convex hull of integer feasible solutions. [6]

What is the Evidence for the Importance of Strength in TS Problems?

In the case of the traveling salesman problem, empirical validation of the strength dogma is largely missing from the literature. The reported, tested cases have been small in size and number. The most recent discussion of the strengths of ILP formulations (24 of them) [35] only reports the time for the LP *relaxations* of several test cases and not the time to solve the ILP problems. An earlier paper, [36], only tested the TS problem with ten cities, and the empirical tests reported in [40] are limited to a six benchmark cases whose size ranges from 17 to 70 cities. Moreover, as we will see, some of the theoretical results on strength use variants of ILP formulations that do not agree with the way those formulations are realistically used, and therefore some of the stated results are not as informative as they may appear.

Here, I discuss how the empirical results we obtained impact the question of the importance of strength. The relevant relations that have been mathematically proven are summarized in [35]: Full DFJ and pure CLAUS have equal strength, which is greater than the strength of pure GG, which is greater than the strength of pure MTZ. The strengths of pure FGG3 and pure FGG4 are incomparable with the full DFJ, but pure FGG4 is stronger than pure FGG3, which is stronger than pure GG. Also, since all of these formulations contain the assignment constraints, each is more constrained than the assignment ILP. Hence, the assignment formulation is as weak or weaker than all of the other formulations, and in fact is *strictly* weaker. For additional examples of the determination and comparison of strength in other ILP formulations, see [5, 9].

15.11.1 Is Strength a Good Predictor of Efficiency in Practice?

The pure FGG3 and pure FGG4 TSP formulations are stronger than the pure GG and pure MTZ formulations, but the FGG formulations are essentially useless, as established in our empirical testing. The total *matrix volume* of the matrices for these four formulations are the same, $O(n^4)$: GG and MTZ have $O(n^2)$ variables and constraints, while FGG3 and FGG4 have $O(n^3)$ variables and $O(n)$ constraints. Similarly, pure CLAUS is stronger than pure GG and pure MTZ, but, in our testing, CLAUS quickly becomes useless in comparison to GG and MTZ. The CLAUS formulation has $O(n^3)$ variables and constraints. The full DFJ formulation is stronger than all the pure formulations studied, other than pure CLAUS, but is essentially useless past $n = 20$. Of course, the full DFJ formulation has an exponential number of inequalities, perhaps making irrelevant any consideration of strength. Clearly, these empirical results show that a simple use of the established strength relations as a predictor of efficiency is not very effective.

CLAUS v. GG v. DFJ

Of the five compact formulations considered, the theoretically strongest is pure CLAUS, which has in every problem instance, the same LP-relaxation value as does the full DFJ formulation, which is stronger than the pure GG formulation. But, we have seen empirically that the CLAUS formulation is vastly inferior to GG.

The comparison of CLAUS to DFJ with separation is particularly telling. The pure CLAUS formulation has the same strength as the full DFJ but only has polynomial size (which influences the speed of the LP solutions). Further, pure CLAUS is solved using brand-and-bound, where the strength of a formulation is presumed to be important [6]. And, in practice, pure CLAUS is much stronger than the full DFJ formulation, for the simple reason that the full DFJ formulation is *not used*.

What is used initially is the *assignment formulation* (which is extremely weak),[15] and afterwords, what is used is the assignment formulation augmented with a small number of the subtour elimination constraints. But, pure CLAUS, or CLAUS, is only effective on small TSP instances, whereas DFJ with elimination works amazingly well for large instances.

What then is the relevance of the strength of the full DFJ formulation (which is not used) to the formulations, derived from the assignment formulation? The quotes above assert that it is the strength of the full DFJ formulation which explains the amazing effectiveness of DFJ solved using the separation technique. But where is the evidence, either theoretical or empirical that strength is the explanation?

We come to a similar conclusion when looking at FGG3 or FGG4 and GG and MTZ. Both pure FGG3 and pure FGG4 are stronger than pure GG and pure MTZ, and all four have the same matrix volumes ($O(n^4)$) overall. But FGG3 and FGG4 run horribly compared to GG and even MTZ.

If Not Strength, Then What?

The real key to the behavior of DFJ with separation is that the number of iterations is extremely small compared to the number of subtour elimination constraints in the full DFJ formulation; that the assignment ILP solves extremely fast, and that each of the successive subproblems solves quickly, even as we add subtour elimination constraints. Does the strength of the full DFJ formulation explain this? I am not an expert in TSP or ILP theory, but I am not aware of it, if it does.

The Theory is Misleading

Theory has established strength relations between several compact TSP formulations, and the full DFJ formulation. But, the strength relations are *only* proved for the *pure* variants of the formulations—the proofs break down if the formulations *explicitly* prohibit any edge from being used in both directions.[16] That prohibition is so natural, almost a fundamental part of the TS problem definition, that I used it in all the formulations except FGG4 (which was just by accident). And, we have seen that with these more natural formulations, the strength relations of their corresponding pure variants are not good predictors of the relative efficiencies of the formulations.

The empirical results do not contradict the mathematical results, but they call into question their *relevance*. Prohibiting any edge from using both directions is natural (almost part of the definition), and really helps (reducing the solution time of GG to about one-third of the pure GG formulation); and it never increases the asymptotic

[15] For example, in the benchmark data *ch130* of 130 cities, the ILP optimal is 6110, the LP-opt for GG is 5608 but the assignment optimal is only 4377.

[16] However, I have never seen this stated in the literature.

size of a formulation. So why not use them, and why focus on formulations that don't use them?[17]

The Theory is Brittle

Not only are the proven strength relations not good predictors of efficiency, and not only do the proofs break, but the relations themselves are easily broken in the more natural formulations. Table 15.2 shows several such occurrences. For example, the LP-Opt for GG on the br17 tour problem is 27, but for full DFJ, it is only 25. Similarly, the LP-Opt for GG on the ch130 path problem is 5608 but is only 4321 for FGG4. We have also found similar phenomena in MTZ and GG. Pure GG is strictly stronger than pure MTZ, but this is not true for GG and MTZ, where the LP-Opt of an MTZ formulation can be *larger* than the LP-Opt of the GG formulation. The following edge-weight matrix is one such example:

```
0    0    0    0    0    0
0    0   24   56    4   11
0   24    0   53   23   52
0   56   53    0   34   68
0    4   23   34    0   18
0   11   52   68   18    0
```

The LP-relaxation to the pure GG formulation has solution value 79.6, while the LP relaxation of the pure MTZ formulation has solution value 71.2, which is in agreement with the established theoretical results. However, using GG and MTZ, which explicitly prohibit edges from being used in both directions, the LP relaxation of the GG formulation has solution value 84.4, while the LP relaxation of the MTZ formulation has a larger solution value, 86. This, and many other examples, show a reversal of the established strength relations when using the more *natural* TSP formulations, rather than the *pure* formulations used for the theoretical results.

Acknowledgements This research was supported by NSF grant 1528234. The research was done partly while on sabbatical at the Simons Institute for Computational Theory, UC Berkeley. I would also like to thank Thong Le for help on understanding proofs about strength; Jim Orlin, T. L. Magnanti, and David Shmoys for helpful communications. Finally, I thank Tandy Warnow, Mohammed El-Kebir, and the anonymous reviewers who provided many helpful suggestions.

[17] In fact, I put that prohibition into my first ILP implementations without even thinking about it. Then, when I looked at the empirical results and saw cases where the LP results violated (correct) mathematical theory, I was perplexed until I realized that the theory is only established for the pure formulations.

Appendix 1: Data for Random Graphs

See Table 15.1 for results of experiments with different compact TSP formulations on a range of random graphs that differ in the number of nodes they contain, and their edge density.

Appendix 2: Data from Benchmark Tests

Experiments on several well-known TSP benchmark test sets covering a range of sizes are shown in Table 15.2. All the formulations, except FGG4, have inequalities that prohibit an edge from being traversed in both directions. The numerals in each ID give the number of cites, from 17 to 229. The letter 'A' or 'S' indicates whether the problem is for a directed (asymmetric) or an undirected (symmetric) graph. Both the optimal tour cost and the optimal path cost (with no designated start or stop nodes) were computed and written next to the problem ID. Each of the ILP formulations is for an optimal TS *path*, unless "tour" is indicated.[18]

The entry in the column for "gap" is empty if the computation ran to completion, and otherwise is the gap when the computation was terminated. An entry for "Time" is the time at completion or termination of the ILP computation; and an entry for "LP-Opt" gives the optimal cost of the LP-relaxation of the TS problem, as reported by Gurobi. The LP-Opt cost can be compared to the cost indicated next to the problem ID, as a measure of the *strength* of the ILP formulation.[19] These LP-opt values can also be compared to each other to validate known theory about the strength of ILP formulations, or to question the relevance of that theory. This is discussed in Sect. 15.11.

[18] But remember that the path computation for the input graph G is actually a tour computation on the derived graph G'. Hence, what we learn from these computations concerns TSP tours and ILP formulations for the TS tour problem.

[19] Note however, what looks like a contradiction in the case of DFJ with separation. In the case of ch130, the LP-Opt reported is 5582. However, the LP-opt for the *assignment ILP* for ch130 is only 4377, and the two values should be the same if LP-opt was computed exactly as discussed in Sect. 15.7.1. A possible explanation is the fact that the computation of DFJ with separation, implemented by a Gurobi program, added subtour-elimination constraints even before the LP-opt was reported. So, the Gurobi code implementing the separation approach seems not to exactly follow the description given in Sect. 15.7.1. However, in all experiments that did not use the DFJ formulation, the LP-opt value was identical to the value obtained by running the LP-relaxation of the ILP formulation.

Table 15.1 The edge density, d, and the number of datasets (replicates), r, generated for this case. The three columns show the results for the GG, MTZ, and CLAUS formulations respectively. All three formulations contain inequalities that explicitly prohibit an edge from being traversed in both directions. The times and standard deviation are all reported in seconds. Any number larger than ten has been rounded to the nearest one-decimal point accuracy. Any number larger than 200 has been rounded to the nearest integer. The large inefficiency of the CLAUS formulation, compared to MTZ and GG, is clearly established with graphs containing only 100 nodes, so no computations with the CLAUS formulation were done for a larger number of nodes. Similarly, due to the inefficiency of MTZ with 400 nodes and edge density of 0.25, more challenging computations were not done with the MTZ formulation. All the computations were done on a MacBook Pro laptop, except for the entry for 100, 0.5, marked with a '*'. Those ran on a somewhat faster iMac desktop

N, d, r	GG (avg, min, max, std)	MTZ (avg, min, max, std)	CLAUS (avg, min, max, std)
20, 0.25, 10	0.037, 0.01, 0.08, 0.023	0.088, 0.02, 0.18, 0.05	0.278, 0.18, 0.45, 0.08
20, 0.5, 10	0.047, 0.02, 0.09, 0.028	0.096, 0.02, 0.27, 0.087	0.58, 0.31, 1.52, 0.35
20, 1, 10	0.056, 0.03, 0.16, 0.04	0.07, 0.03, 0.14, 0.04	0.67, 0.55, 1.07, 0.16
50, 0.1, 9	0.13, 0.05, 0.30, 0.088	1.58, 0.17, 6.59, 1.94	6.9, 3.86, 20.4, 5
50, 0.25, 9	0.38, 0.13, 0.76, 0.18	1.24 0.32, 2.85, 0.9	18.2, 8, 45.75, 11.8
50, 0.5, 9	0.56, 0.18, 1.11, 0.34	0.93, 0.25, 2.59, 0.76	58.6, 18.3, 171.0, 43.9
50, 1, 9	0.89, 0.46, 1.43, 0.3	1.46, 0.41, 3.47, 1.07	380, 43.6, 1592, 454
100, 0.1, 9	1.73, 0.34, 6.09, 1.71	23.5, 2.42, 57.7, 17.6	1433, 108.6, 3.8K, 1178
100, 0.25, 9	2.27, 0.94, 4.94, 1.46	14.9, 2.26, 45.1, 14.1	5101, 198, 15.5K, 4.3K
100, 0.5, 9*	5.1, 3.32, 7.72, 1.42	39.1, 3.02, 107.0, 37.7	6062, 815, 14K, 4K
100, 1, 6	11.0, 7.3, 14.5, 2.78	39.5, 4.06, 119, 39.1	13220, 206, 24K, 10.5K
200, 0.1, 10	23.9, 6.54, 61.3, 15.7	503, 75.9, 1720, 452	
200, 0.25, 11	63.4, 14.5, 152.3, 48.0	434, 19.7, 1091, 332	
200, 0.5, 10	93.3, 58.8, 133, 24.1	490, 123.5, 1203, 332	
200, 1, 10	236, 159.1, 338, 64.3	416, 162.1, 990, 296	
300, 0.1, 5	89.2, 15.1, 209, 77.9	780, 98, 1778, 595	
300, 0.25, 5	206, 90.7, 370, 92.8	1921, 91.8 4117, 1562	
300, 0.5, 5	495, 315, 769, 169.7	2551, 1854, 3415, 510	
300, 1, 5	1861, 862, 4582, 1373	2565, 1600, 3503, 671	
400, 0.1, 5	543, 239, 1014, 299	12025, 6057, 19366, 5K	
400, 0.25, 5	644, 562, 794, 80	9848, 2986, 31945, 11K	
400, 0.5, 25	1787, 880, 3697, 738	12201, 1593, 43724, 9952	
400, 1, 11	2850, 2307, 3616, 434	16067, 877, 93115, 25051	
500, 0.1, 7	949, 306, 1716, 531		
500, 0.25, 7	2505, 1020, 5606, 14K		
500, 0.5, 7	7630, 3978, 16238, 3773		
500, 1, 7	6165, 4981, 8468, 1231		

Table 15.2 Data for benchmark TS problems. The ID gives the name of the test, and is followed by its optimal TSP cost, and whether the test problem is for a path or a tour, and whether the test problem is symmetric (S) or asymmetric (A). The next entry is the ILP formulation used to solve the test. Each of these formulations is for the TS path problem, unless "tour" is stated. The next column, labeled GAP, is empty if the computation ran to completion but shows the size of the gap if the computation was terminated before completion. The next column, time, gives the time for the computation, either to its completion or to its termination. The final column, LP-Opt, gives the optimal value for LP relaxation of the test problem

ID ILP-OPT	Formulation	Gap	Time	LP-Opt
br17 25 (path) A				
br17 39 (tour) A				
	GG (path)		0.11 s	15.6
	GG (tour)		0.21 s	27
	MTZ		0.54 s	10
	CLAUS		1.52 s	25
	FGG4		3.19 s	12.82
	full DFJ		45 s	25
p43 553 (path) A				
p43 5620 (tour) A				
	GG (path)		3.19 s	170.6
	GG (tour)		20 s	845
	MTZ	54%	10 h	160
	CLAUS		46 s	553
	FGG4		3 h 46 min	113.7
berlin52 7544.17				
	GG (tour)		2.21 s	7194
eil76 521.4 (path) S				
eil76 544.4 (tour) S				
	GG (path)		7.85 s	518
	GG (tour)		8.36 s	540.85
	MTZ		37 s	518
	CLAUS		48 min	520.59
	FGG4	18.5%	4 h 39 min	475.7
	DFJ (tour)		0.19 s	540.7
ch130 5890 (path) S				
ch130 6110 (tour) S				
	GG (path)		5 min 45 s	5397
	GG (tour)		6 min 52 s	5608
	MTZ		2 h 18 min	5372
	CLAUS	88%	6 h 37 min	5850
	FGG4	46.1%	3 h 43 min	4321
	DFJ (tour)		1.87 s	5582

(continued)

Table 15.2 (continued)

ID ILP-OPT	Formulation	Gap	Time	LP-Opt
qa194 8946 (path) S				
qa194 9352 (tour) S				
	GG (path)		7 min 49 s	8614
	GG (tour)		14 min 48 s	9028
	MTZ	1.38%	4 h	8582
	CLAUS	crashed	1 h	
	FGG4	no integer solution	10 h	
	DFJ (tour)		8 s	9006.48
gr229 952 (path) S				
gr229 1014.8 (tour) S				
	GG (path)		3 h 10 min	842
	GG (tour)	6.8%	3 h 41 min	903.5
	MTZ	24.5%	12 h	832.5
	DFJ (tour)		21 s	885.6

References

1. Agarwala, R., Applegate, D.L., Maglott, D., Schuler, G.D., Schäffer, A.A.: A fast and scalable radiation hybrid map construction and integration strategy. Genome Res. **10**(3), 350–364 (2000)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
3. Alizadeh, F., Karp, R.M., Weisser, D., Zweig, G.: Physical mapping of chromosomes using unique probes. J. Comput. Biol. **2**, 159–184 (1995)
4. Althaus, E., Klau, G.W., Kohlbacher, O., Lenhof, H.P., Reinert, K.: Integer linear programming in computational biology. In: Festschrift Mehlhorn, LNCS 5760, pp. 199 – 218. Springer (2009)
5. Álvarez-Miranda, E., Ljubić, I., Mutzel, P.: The maximum weight connected subgraph problem. In: Junger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 245–270. Springer (2013)
6. Bertsimas, D., Weismantel, R.: Optimization Over Integers, vol. 13. Dynamic Ideas, Belmont (MA) (2005)
7. Blanchette, M., Bourque, G., Sankoff, D.: Breakpoint phylogenies. In: Miyano, S., Takagi, T. (eds.) Genome Informatics, pp. 25–34. University Academy Press (1997)
8. Blum, C., Festa, P.: Metaheuristics for String Problems in Bio-informatics. Wiley (2016)
9. Chimani, M., Rahmann, S., Bocker, S.: Exact ILP solutions for phylogenetic minimum flip problems. In: Proceedings of the First ACM-BCB Conference, pp. 147–153 (2010)
10. Claus, A.: A new formulation for the travelling salesman problem. SIAM J. Algebr. Discr. Methods **5**, 21–25 (1984)
11. Conforti, M., Cornuejols, G., Zambelli, G.: Integer Programming. Springer (2014)
12. Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale travelling-salesman problem. Oper. Res. **2**, 393–410 (1954)
13. Felsenstein, J.: Inferring Phylogenies. Sinauer (2004)
14. Forrester, R., Greenberg, H.J.: Quadratic binary programming models in computational biology. Alg. Oper. Res. **3**, 110129 (2008)

15. Fox, K., Gavish, B., Graves, S.: An n-constraint formulation of the (time-dependent) traveling salesman problem. Oper. Res. **28**, 101821 (1980)
16. Frumkin, J.P., Patra, B.N., Sevold, A., Ganguly, K., Patel, C., Yoon, S., Schmid, M.B., Ray, A.: The interplay between chromosome stability and cell cycle control explored through gene-gene interaction and computational simulation. Nucleic Acids Res. **44**, 8073–8085 (2016)
17. Gavish, B., Graves, S.: The travelling salesman problem and related problems. Working Paper OR 078-78. Technical Report. MIT, Operations Research Center (1978)
18. Gouveia, L., Vos, S.: A classification of formulations for the (time-dependent) traveling salesman problem. Europ. J. Oper. Res. **83**, 69–82 (1995)
19. Gusfield, D.: Algorithms on Strings, Trees and Sequence. Computer Science and Computational Biology. Cambridge University Press (1997)
20. Gusfield, D.: Integer linear programming in computational and systems biology: an entry-level text and course. Cambridge University Press (2019)
21. Gusfield, D., Frid, Y., Brown, D.: Integer programming formulations and computations solving phylogenetic and population genetic problems with missing or genotypic data. In: Proceedings of 13th Annual International Conference on Combinatorics and Computing, pp. 51–64. LNCS 4598, Springer (2007)
22. Huttlin, E.L., Ting, L., Bruckner, R.J., Gebreab, F., Gygi, M.P., Szpyt, J., Tam, S., Zarraga, G., Colby, G., Baltier, K., Dong, R., Guarani, V., Vaites, L.P., Ordureau, A., Rad, R., Erickson, B.K., Whr, M., Chick, J., Zhai, B., Kolippakkam, D., Mintseris, J., Obar, R.A., Harris, T., Artavanis-Tsakonas, S., Sowa, M.E., Camilli, P.D., Paulo, J.A., Harper, J.W., Gygi, S.P.: The BioPlex network: a systematic exploration of the human interactome. Cell **162**, 425–440 (2015)
23. Johnson, M., Hummer, G.: Interface-resolved network of protein-protein interactions. PLoS Comput. Biol. **9**, e1003,065 (2013)
24. Johnson, O., Liu, J.: A traveling salesman approach for predicting protein functions. Source Code Biol. Med. **1**, (2006)
25. Kingsford, C.L., Chazelle, B., Singh, M.: Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics **21**, 1028–1036 (2005)
26. Korostensky, C., Gonnet, G.: Near optimal multiple sequence alignments using a traveling salesman problem approach. In: Proceedings of String Processing and Information Retrieval Symposium, p. 105. IEEE (1999)
27. Korostensky, C., Gonnet, G.: Using traveling salesman problem algorithms for evolutionary tree construction. Bioinformatics **16**, 619–627 (2000)
28. Lancia, G.: Integer programming models for computational biology problems. J. Comp. Sci. Tech. **19**, 6077 (2004)
29. Lancia, G.: Mathematical programming in computational biology: an annotated bibliography. Algorithms **1**, 100129 (2008)
30. Langevin, A., Soumis, F., Desrosiers, J.: Classification of travelling salesman problem formulations. Oper. Res. Let. **9**, 12732 (1990)
31. Lorenzo, E., Camacho-Caceres, K., Ropelewski, A.J., Rosas, J., Ortiz-Mojer, M., Perez-Marty, L., Irizarry, J., Gonzalez, V., Rodríguez, J.A., Cabrera-Rios, M., Isaza, C.: An optimization-driven analysis pipeline to uncover biomarkers and signaling paths: cervix cancer. Microarrays **4**(2), 287–310 (2015)
32. Mazza, A., Klockmeier, K., Wanker, E., Sharan, R.: An integer programming framework for inferring disease complexes from network data. Bioinformatics **32**, i271–i277 (2016)
33. Miller, C., Tucker, R., Zemlin, R.: Integer programming formulation of traveling salesman problems. J. Assoc. Comput. Mach. pp. 326–329 (1960)
34. Moret, B., Bader, D.A., Warnow, T.: High-performance algorithm engineering for computational phylogenetics. J. Supercomput. **22**, 99–111 (2002)
35. Oncan, T., Altnel, I., Laporte, G.: A comparative analysis of several asymmetric traveling salesman problem formulations. Comp. Oper. Res. **36**, 637654 (2009)
36. Orman, A., Williams, H.: A survey of different integer programming formulations of the travelling salesman problem. Technical Report, Department of Operational Research, London School of Economics and Political Science (2004)

37. Orman, A., Williams, H.P.: A survey of different integer programming formulations of the travelling salesman problem. In: Kontoghiorghes, E., Gatu, C. (eds.) Optimisation, Econometric and Financial Analysis, vol. 9, pp. 91–104. Springer, Berlin, Heidelberg (2007)
38. Padberg, M., Sung, T.Y.: An analytical comparison of different formulations of the travelling salesman problem. Math. Prog. **52**, 315–357 (1991)
39. Pataki, G.: The bad and the good-and-ugly. Technical Report, Columbia University, IEOR (2000). CORC 2000-1
40. Pataki, G.: Teaching integer programming formulations using the traveling salesman problem. SIAM Rev. **65**, 116–123 (2003)
41. Reinelt, G.: TSPLIB-A traveling salesman problem library. ORSA J. Comp. **3**, 376–384 (1991)
42. Reiter, J., Makohon-Moore, A., Gerold, J., Bozic, I., Chatterjee, K., Iacobuzio-Donahue, C., Vogelstein, B., Nowak, M.: Reconstructing metastatic seeding patterns of human cancers. Nat. Commun. **8**, (2017)
43. Sankoff, D., Blanchette, M.: Multiple genome rearrangement and breakpoint phylogeny. J. Comp. Biol. **5**, 555–570 (1998)
44. Sawik, T.: A note on the Miller-Tucker-Zemlin model for the asymmetric traveling salesman problem. Bull. Polish Acad. Sci. Tech. Sci. **64**, 517–520 (2016)
45. Shao, M., Lin, Y., Moret, B.M.: An exact algorithm to compute the DCJ distance for genomes with duplicate genes. J. Comput. Biol. **22**(5), 425–435 (2015)
46. Shao, M., Moret, B.M.E.: Comparing genomes with rearrangements and segmental duplications. Bioinformatics **31**(12), i329–i338 (2015)
47. Shao, M., Moret, B.M.E.: A fast and exact algorithm for the exemplar breakpoint distance. J. Comput. Biol. **23**(5), 337–346 (2016)
48. Shao, M., Moret, B.M.E.: On computing breakpoint distances for genomes with duplicate genes. J. Comput. Biol. **24**(6), 571–580 (2017)
49. Wong, R.: Integer programming formulations of the traveling salesman problem. In: Rabbat, G. (ed.) Proceedings of ICCC 80, IEEE Conference on Circuits and Computing, pp. 149–152 (1980)

Index

T. Warnow (ed.), *Bioinformatics and Phylogenetics*, Computational Biology 29,
https://doi.org/10.1007/978-3-030-10837-3