Computational Biology

Cedric Chauve
Nadia El-Mabrouk
Eric Tannier *Editors*

Models and Algorithms for Genome Evolution

ISCB INTERNATIONAL SOCIETY FOR COMPUTATIONAL BIOLOGY

Springer

Computational Biology

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.

Author guidelines: springer.com > Authors > Author Guidelines

For further volumes:
www.springer.com/series/5769

Cedric Chauve • Nadia El-Mabrouk • Eric Tannier
Editors

Models and Algorithms for Genome Evolution

Springer

Editors
Cedric Chauve
Department of Mathematics
Simon Fraser University
Burnaby, British Columbia, Canada

Eric Tannier
Biometry and Evolutionary Biology
INRIA Rhône-Alpes
University of Lyon
Villeurbanne, France

Nadia El-Mabrouk
Computer Science and Operations Research
University of Montreal
Montreal, Québec, Canada

ISSN 1568-2684 Computational Biology
ISBN 978-1-4471-5297-2 ISBN 978-1-4471-5298-9 (eBook)
DOI 10.1007/978-1-4471-5298-9
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013949678

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

It is hard to overestimate the impact of David Sankoff's research on today's computational biology. His fingerprints are present in every curriculum and textbook. Even more importantly, Sankoff's studies have had a tremendous effect on the way we think about computational problems in evolution and more broadly in biology. This book contains articles written by experts in computational biology, summarizing some of the past directions in the field, and paving the way into new directions for the future. Not surprisingly, many of them build on ideas pioneered by Sankoff.

The volume starts with a chapter by Gene Myers, who describes the complexity-theoretic ideas developed in the 1980s that led to the creation of the BLAST search engine, perhaps the number one "killer application" of computational biology. In Chap. 2 David Bryant and Jamie Kydd revisit a largely forgotten 1972 study by Sankoff, a paper that turns out to be one of the pioneering treatise on model-based phylogeography. They put this paper in the context of modern methods and show how its merits have withstood the test of time. Chapter 3 by Miklos Csuros reviews the problems of reconstructing evolutionary features of ancestors in a known phylogeny, taking as a springboard a 1975 paper of Sankoff and Rousseau.

Chapter 4, by Chauve, El-Mabrouk, Gueguen, Smeria and Tannier, discusses integrative models of evolution that simultaneously address the nucleotide, the gene and the genome levels. The authors put a pioneering 2000 work of Sankoff and El-Mabrouk in the context of novel developments. The following chapter, by Anne Bergeron and Jens Stoye, describes the intuitions and methods that led to the double cut and join (DCJ) distance formula, and put the formula in a broader theoretical context. Chapter 6 by Tandy Warnow discusses new algorithms aimed to estimate alignment and phylogeny on ultra-large datasets. Both theoretical and empirical aspects of the performance are portrayed.

In Chap. 7, Bernard Moret, Yu Lin and Jijun Tang discuss a way to use likelihood methods with rearrangement models. The good performance of the method is attributed to injecting a simple bias in the ground probabilities, an idea originally proposed by Sankoff in 1998. Chapter 8, by Liqing Zhang, Mingming Liu and Layne Watson, reviews the impact of insertion and deletion variants in human biology, evolution and health. In addition, it describes extant and novel tools for predicting the

functional effects of insertions and deletions. In the following chapter, Binhai Zhu surveys the research on genomic distance in genomes with gene repetitions, missing and redundant genes, pioneered by Sankoff. It also describes the consequences of this research beyond computational biology.

In Chap. 10, Joao Meidanis and Sophia Yancopoulos discuss the DCJ distance and the so-called algebraic distance, introduced recently by Feijao and Meidanis. They compare the two distances using a new graph-based method, and explore possible weighting of the operations and their effects. In Chap. 11, David Sankoff and Chunfang Zheng study the combined effect of whole-genome duplication and fractionation, the process by which exactly one of the two copies of a duplicated gene is lost. They develop a new strategy to use consolidated intervals in order to reconstruct the ancestral gene order in the genome preceding duplication, and demonstrate it on two plant species.

Chapter 12, by Maneul Lafond, Krister Swensen and Nadia El-Mabrouk, investigates several possible error sources in gene tree reconstruction, and suggests new models to detect such errors. It demonstrates these error detection methods in analyzing gene trees of several fish species. Chapter 13, by Braga, Chauve, Doerr, Jahn, Stoye, Thevenin and Wittler, studies approaches in comparative genomics that do not rely on identification of gene families. The authors investigate the effect of such approaches on key problems in rearrangement phylogeny. The final chapter, by Daniel Platt, Filippo Utro, Marc Pybus and Laxmi Parida, uses simulations of human population histories to assess what fraction of genetic events are recoverable from present data population data. The authors conclude that no more than two thirds of the events are recoverable, but that the signal is still sufficient for resolution between populations on many levels.

The broad spectrum of papers in this volume is a tribute to the richness of the huge tree of research that David has developed, which undoubtedly will continue to bear fruit, develop offshoots and shape research in biology for many years. At the same time this volume also reflects the love and admiration of numerous colleagues and generations of students that David has raised. Many happy returns, David!!

Tel Aviv University, Israel Ron Shamir

Models and Algorithms for Genome Evolution—Preface

Fifty years ago, Pauling and Zuckerkandl introduced the idea to reconstruct ancestral protein sequences from the comparison of extant protein sequences [1]. Roughly at the same time, a young David Sankoff published his first scientific article [2], a first step in an exceptional scientific career marked by fundamental contributions in computational molecular evolution, building a theoretical ground to the idea of Pauling and Zuckerkandl.

David Sankoff's scientific contribution goes beyond molecular evolution, as he is also very well known for his work on models and algorithms for comparative linguistic, social sciences and music. However, the present volume "Models and Algorithms for Genome Evolution" (MAGE) focuses on computational biology, and celebrates David Sankoff's work and impact in this field.

Indeed, as mathematicians and computer scientists interested in molecular evolution, we feel that, in our research, almost every path we follow has been opened, if not paved, by David Sankoff. This prompted us to dedicate the MAGE conference, held in August 2013 near Montréal, Canada, to the 50th anniversary of his first scientific article. MAGE is also intended to follow up on a previous conference, organized in 2000 by David Sankoff and Joseph Nadeau, near Montréal, entitled *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families* [3], that provided a retrospective in comparative genomics and opened new avenues following the first sequence of the human genome.

In the same spirit, the 2013 MAGE conference also offered a retrospective on the past 50 years of the young field of computational molecular evolution, as well as a prospective view on the challenges we expect to face in the near future.

The contributions presented here offer a wide panorama on comparative genomics today. We also learn from Gene Myers (Chap. 1) why the Blast heuristic, although universally adopted by biologists and bioinformaticians, is sometimes unfairly disregarded by theoreticians. It is then recalled that some standard results in computational biology are due to David Sankoff, such as the dynamic programming principle for ancestral character reconstruction along a phylogenetic tree (Chap. 3). Chapter 2, together with Chaps. 4, 7 and 9, illustrate that David Sankoff's contribu-

tion stands out by the problems he introduced and the research avenues he opened, that were followed by many others. David Sankoff is also an efficient publicist for new models, approaches, and techniques, that turn out to become standard, as illustrated by his role in establishing the Double Cut and Join distance (Chaps. 5 and 10). The present book also includes overviews on well established research fields (Chaps. 6 and 8) and of new ideas which are likely to develop into active research programs (Chaps. 11–14).

Although the research activity around the mathematical and computational aspects of molecular evolution has reached a certain maturity, thanks to influential researchers such as David Sankoff, Joe Felsenstein, Gene Myers, Michael Waterman and many others, yet challenges are still overwhelmingly numerous. Fifty years after Pauling and Zuckerkandl seminal work, we are probably still not aware of the diversity of genome structures nor of the diversity of means genomes use to evolve. We believe that the MAGE conference has been a good opportunity to share the latest developments and discuss about future directions. We hope the book will give the occasion to learn about and from the past of this young field, and will be a source of inspiration for many researchers.

We thank reviewers for their work, Mathieu Blanchette, Guillaume Bourque, Anthony Labarre, Anne Bergeron, Chunfang Zheng, Krister Swenson, David Bryant. Thanks to Ron Shamir for writing the foreword.

References

1. Pauling, L., Zuckerkandl, E.: Chemical paleogenetics, molecular "restoration studies" of extinct forms of life. Acta Chem. Scand. **17**, 9–16 (1963)
2. Friesen, J., Sankoff, D., Siminovitch, L.: Radiobiological studies of vaccinia virus. Virology **21**, 411–424 (1963)
3. Sankoff, D., Nadeau, J. (eds.): Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families. Kluwer Academic, Dordrecht (2000)

LaBRI, Université Bordeaux I, Talence, France
Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada — Cedric Chauve

DIRO, Université de Montréal, Montréal, QC, Canada — Nadia El-Mabrouk

LBBE, Université Lyon I Claude Bernard, Villeurbanne, France
INRIA Rhône-Alpes, Montbonnot, France — Eric Tannier

Contents

Contributors

Anne Bergeron Lacim, Université du Québec à Montréal, Montréal, Canada

Marília D.V. Braga Inmetro, Duque de Caxias, Brazil

David Bryant University of Otago, Dunedin, New Zealand

Cedric Chauve Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada; LaBRI, Université Bordeaux I, Talence, France

Miklós Csűrös Department of Computer Science and Operations Research, University of Montréal, Montreal, QC, Canada

Daniel Doerr Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany; Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

Nadia El-Mabrouk Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, Montréal, QC, Canada

Laurent Guéguen LBBE, Université Lyon I Claude Bernard, Lyon, France

Katharina Jahn Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany; Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

Jamie Kydd University of Auckland, Auckland, New Zealand

Manuel Lafond Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, Montréal, QC, Canada

Yu Lin Laboratory for Computational Biology and Bioinformatics, EPFL, Lausanne, Switzerland

Mingming Liu Department of Computer Science, Virginia Tech, Blacksburg, VA, USA

Joao Meidanis Scylla Bioinformatics, Campinas, SP, Brazil; University of Campinas, Campinas, SP, Brazil

Bernard M.E. Moret Laboratory for Computational Biology and Bioinformatics, EPFL, Lausanne, Switzerland

Gene Myers MPI for Cellular Molecular Biology and Genetics, Dresden, Germany

Laxmi Parida IBM T.J. Watson Research, New York, USA

Daniel E. Platt IBM T.J. Watson Research, New York, USA

Marc Pybus Institute of Evolutionary Biology (CSIC-UPF), Barcelona, Spain

David Sankoff University of Ottawa, Ottawa, ON, Canada

Magali Semeria LBBE, Université Lyon I Claude Bernard, Lyon, France

Jens Stoye Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany; Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

Krister M. Swenson Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, Montréal, QC, Canada; McGill University, Montreal, QC, Canada

Jijun Tang Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA

Eric Tannier LBBE, Université Lyon I Claude Bernard, Lyon, France; INRIA Rhône-Alpes, France

Annelyse Thévenin Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany; Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

Filippo Utro IBM T.J. Watson Research, New York, USA

Tandy Warnow Department of Computer Science, University of Texas at Austin, Austin, TX, USA

Layne T. Watson Dapartment of Computer Science, Virginia Tech, Blacksburg, VA, USA; Dapartment of Mathematics, Virginia Tech, Blacksburg, VA, USA

Roland Wittler Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany; Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

Sophia Yancopoulos The Feinstein Institute for Medical Research, Manhasset, NY, USA

Liqing Zhang Dapartment of Computer Science, Virginia Tech, Blacksburg, VA, USA

Chunfang Zheng University of Ottawa, Ottawa, ON, Canada

Binhai Zhu Department of Computer Science, Montana State University, Bozeman, MT, USA

Part I
Emergence of Standard Algorithms

Chapter 1
What's Behind Blast

Gene Myers

Abstract The BLAST search engine was published and released in 1990. It is a heuristic that uses the idea of a neighborhood to find seed matches that are then extended. This approach came from work that this author was doing to lever these ideas to arrive at a deterministic algorithm with a characterized and superior time complexity. The resulting $O(en^{\text{pow}(e/p)} \log n)$ expected-time algorithm for finding all e-matches to a string of length p in a text of length n was completed in 1991. The function $\text{pow}(\epsilon)$ is 0 for $\epsilon = 0$ and concave increasing, so the algorithm is truly sublinear in that its running time is $O(n^c)$ for $c < 1$ for ϵ sufficiently small. This paper reviews the history and the unfolding of the basic concepts, and it attempts to intuitively describe the deeper result whose time complexity, to this author's knowledge, has yet to be improved upon.

1.1 The Meeting

The 1980s were an active decade for basic advances in sequence comparison algorithms. Michael Waterman, Temple Smith, Esko Ukkonen, Webb Miller, Gad Landau, David Lipman, Bill Pearson, and myself, among others, were all very active in this period of time and were working out the basic algorithms for comparing sequences, approximate pattern matching, and database searching (e.g. [1–6]). During this time, the BLAST heuristic was developed and deployed at the National Library of Medicine in 1990 and the paper that described it became one of the most highly cited papers in science [7]. This paper is about how the design for the algorithm came about, from my point of view, and its relationship to the theoretical underpinnings of sequence comparison that I was exploring at the time that ultimately lead to an efficient, deterministic, expected-time algorithm for finding approximate matches using a precomputed index [8].

In 1988, Webb Miller and I organized a small bioinformatics meeting in Bethesda, Maryland that included such notable figures as David Sankoff, Michael

G. Myers (✉)
MPI for Cellular Molecular Biology and Genetics, 01307 Dresden, Germany
e-mail: myers@mpi-cbg.de

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_1,

Waterman, Temple Smith, Eric Lander, Zvi Galil, Esko Ukkonen, David Lipman and other great investigators of that time. At the meeting Zvi Galil gave a talk about suffix trees [9] and raised two questions that ultimately were answered:

1. Can suffix trees be built in a way that is independent of alphabet size s?
2. Can a precomputed index such as a suffix tree of a large text be used to speed up searches for approximate matches to a string?

To understand the first question, one must recall that one can either use an s-element array in each suffix tree node to permit a search for a string of length p in a text of length n in $O(p)$ time but requiring $O(ns)$ space for the suffix tree, or one can use only $O(n)$ space by using binary trees to decide which edge to follow out of a node, but resulting in $O(p \log s)$ time for the search. This question ultimately led Udi Manber and I to develop suffix arrays in late 1990 [10], where a suffix array occupies $O(n)$ space, independent of s, and takes $O(p + \log n)$ time to search for a string, again independent of s. This data structure in turn enables the Burroughs–Wheeler Transform or BWT [11], that is now greatly in vogue for next-gen sequencing (NGS) applications [12], to be computed in $O(n)$ time.

1.2 Filters and Neighborhoods

But it was the second question on the use of an index, such as a suffix tree, to speed searches for approximate matches that captured my attention immediately after the meeting. Most algorithmicists work on *deterministic* search algorithms meaning that the method finds exactly the set of locations in the text where a query approximately matches within some specified threshold, whereas a *heuristic* is an algorithm that finds most of the matches sought, but may miss a few, called a *false negative*, and may further report a few locations where a match doesn't actually occur, called a *false positive*. In between these two types of algorithms, a *filter* is an algorithm that has no false negatives but may produce false positives. That is, it produces a superset of the instances sought, or equivalently it filters out most of the locations where matches do not occur. A filter can be the first step of a deterministic algorithm simply by running a deterministic checking algorithm on the subset of locations reported by the filter. If the filter is much more efficient than the deterministic checker, then one ends up with a much more efficient search.

At the time, there were surprisingly no published methods using the simple idea of finding *exact* matches to k-mers (strings of length k) from the query string [13, 14] even though this was fairly obvious and had been used in the heuristic method of FASTA. Shortly after the Bethesda meeting, I had the first and most important idea of looking for exact matches to strings in the *neighborhood* of k-mers selected from the query string. Let δ be a sequence comparison measure that given two strings v and w returns a numeric measure $\delta(v, w)$ of the degree to which they differ (e.g. the generalized Levenshtein metric). Given a string w the τ-neighborhood of w with respect to δ, $\aleph_\tau^\delta(w)$ is the set of all strings v whose best alignment with w under scoring scheme δ is less than τ, i.e. $\{v : \delta(v, w) \leq \tau\}$.

For this paper, except where mentioned otherwise, we are focusing on the approximate match problem where δ is the *simple Levenshtein* metric, which is the minimum number of insertions, deletions, and substitutions possible in an alignment between the two strings in question. That is, we seek matches of a query of length p to a text of length n, where up to e differences are allowed. Another way to phrase this, which we will use interchangeably, is that we seek ϵ-matches where $\epsilon = e/p$ is the *length relative* fraction of differences allowed. To illustrate the idea of a neighborhood under this metric, the 1-neighborhood of *abba* (or 25 %-neighborhood) is $\aleph_1(abba) = \{aaba, aabba, abaa, aba, abaa, ababa, abba, abbaa, abbab, abb, abbb, abbba, babba, bba, bbba\}$.

For the example above, notice that wherever one finds *abaa* one will also finds *aba* as it is a prefix of the former. So to find all matches to neighborhood strings it suffices to look up in an index only those that are *not* an extension of a shorter string in the same neighborhood. Let the *condensed* τ-neighborhood of w be the subset of these strings, i.e. $\bar{\aleph}^{\delta}_{\tau}(w) = \{v : v \in \aleph^{\delta}_{\tau}(w)$ and $\nexists u \in \aleph^{\delta}_{\tau}(w)$ such that u is a prefix of $v\}$. For our example, the condensed 1-neighborhood of *abba* is $\bar{\aleph}_1(abba) = \{aaba, aabba, aba, abb, babba, bba, bbba\}$, a considerably smaller set of strings.

To illustrate the advantage of using (condensed) neighborhoods, consider looking for a match with nine differences to a query of length say 100. If one partitions the query into 10 strings of length 10, then by the Pigeon Hole principle, one of the 10 strings must exactly match in the database. So one can filter the text by looking for one of these 10 strings of length 10. But if one partitions the query into 5 strings of length 20, then by the Pigeon Hole principle, a string in the 1-neighborhoods of the five query parts must exactly match in the database. A rough estimate for the number of strings in the condensed e-neighborhood of a string of length k is $\bar{\aleph}_e(k) = \binom{k}{e}(2s)^e$. Thus in our example we can filter the text by looking for one of 800 strings of length 20. Which filter is better? The probability of a random false positive for the k-mer filter is $10/s^{10}$ and for the neighborhood filter it is $800/s^{20}$. Thus the later filter produces $s^{10}/80$ fewer false positives. If s is 4 (e.g. the DNA alphabet) and n is 3×10^9 (e.g. the size of the human genome) then the neighborhood filter produces 13,000 times fewer false positives, and reports in expectation 2.18 false positive, whereas the k-mer filter reports over 28,600!

1.3 Version 0.1

For every true positive location, i.e. an approximate match is present, one must spend time proportional to the best algorithm available for aligning one sequence to another. While there are some quite sophisticated algorithms, in practice, one of the best is still the $O(pe)$ algorithm discovered by Ukkonen [2] and a year later by myself [6], where I further proved that the algorithm runs in $O(p + e^2)$ expected time, and can be modified with a suffix tree and $O(1)$ lca-finding [15] to take this much time in the worst-case. If a search results in h true hits, we will assume for simplicity that $O(hep)$ time will be taken to confirm and report all the matches

and their alignments. The goal of a filter is to deliver the hits efficiently and to waste as little time as possible on false positives. That is, the goal is to optimize the time a filter would take on a random text that in expectation has no hits to the query. The initial simple idea that I was working with in early 1989 was as follows:

1. Partition the query into p/k k-mers.
2. Generate every string in the (ϵk)-neighborhood of the query k-mers and find all the exact matches to these strings using an index.
3. Check each location reported above with the $O(ep)$ algorithm.

The question is what k-mer size leads to the best expected-time performance of the filter over a random text? Roughly, the number of neighborhood strings is $(p/k)\bar{\aleph}_{\epsilon k}(k)$ and the time spent looking up each is $O(k)$ excluding the time to check hits for each. Thus the lookup phase takes $O(p\bar{\aleph}_{\epsilon k}(k))$ time. The expected number of hits is $(p/k)\bar{\aleph}_{\epsilon k}(k)(n/s^k)$ and thus the expected time for checking proposed locations is $O((\epsilon p^3/k)\bar{\aleph}_{\epsilon k}(k)(n/s^k))$. Thus the total expected time for the filter is

$$O\left(p\bar{\aleph}_{\epsilon k}(k)\left(1+\frac{\epsilon p^2 n}{ks^k}\right)\right) \tag{1.1}$$

I was unable to produce an analytic formula for the value of k that as a function of n, p, and ϵ gives the minimum time. However, using Stirling's Approximation, I was able to demonstrate (unpublished) that the best value of k is always bigger than $\log_s n$ and less than $(1+\alpha)\log_s n$ where α becomes increasingly closer to 0 as n/p goes to infinity. For typical values of the parameters, and especially when n is much larger than p, α is quite close to zero. Thus one instinctively knows that the k-mer size should be on the order of $\log_s n$. We will use this observation later on.

1.4 BLAST

In May of 1989, I spent two weeks at the National Center for Biotechnology Information (NCBI) with David Lipman. I was a smoker at the time and was having a cigarette outside when David came out and showed me an article in *Science* in which Lee Hood was extolling the virtues of a new systolic array chip built by TRW called the Fast Data Finder (FDF) [16]. I proceeded, as a firm believer that special hardware is not the way to solve problems, to explain to David my new ideas for approximate search and how I thought we could do such searches just as well in software rather than spend money on relatively expensive hardware. David had previously developed FASTA with Bill Pearson [5], which at the time was the best heuristic for searching protein databases. David listened carefully and started to think about how the ideas could be used in a heuristic and efficient way to search for significant locally aligned regions of a protein query against a protein database under a general scoring scheme such as the PAM or BLOSSUM metrics. In short order we had the following heuristic adaption of my first filter:

1. Consider the $p - k + 1$ *overlapping* k-mers of the query (to increase the chance of not missing a true hit).
2. Generate every string in the τ-neighborhood of the query k-mers under a similarity-based protein metric δ and find all the exact matches to these strings by some means (an index may not be in practice the fastest way to do this).
3. Extend each seed match into a local alignment of significance by some means, and report it if its score is sufficiently high.

Over the next several months a number of versions of codes based on the above template were developed by myself, Webb Miller, and Warren Gish.

Webb tried a simple index for the look up and reported that it was quite slow. In hindsight this was just at the time when the mismatch in speed between memory access and processor speed was becoming severe enough that being aware of cache-coherence was becoming essential for good performance. I still wonder if a better design of an index and the order of lookups within it, e.g. sorting the strings to be looked up, would not lead to a much speedier implementation. The other idea and faster implementation was to generate a finite automaton of the neighborhood strings and in an $O(n)$ scan of the text with the automaton find all potential match locations. Each state of the automaton had an s-array table of transitions. Gish realized that if a Mealy machine [17] was used instead of a Moore machine [18] (i.e. report hits on transitions rather than on states), a factor of s is saved in space. Given that s is 20 for protein alphabets this was a significant space saving.

For the extension step we tried simply extending forward and backward with no indels. I proposed the idea that an extension step stop when the score of the extension dropped too far below the best score seen (the X-factor). I also wrote a version that extended with indels, again observing the X-factor, but Lipman deemed that it was too slow and not worth the additional computer time. He later reversed his position in 1989 with a second release and publication of BLAST [19], albeit with Miller reinventing the gapped extension strategy.

Warren also wrote all the code for practical matters such as low-complexity sequence filtering and he built the initial web server [20]. Altschul, the first author, added the calculation of the significance of each match based on work he and Sam Karlin had published earlier in the year [21]. He also tested the sensitivity and performance of the method and wrote the paper. An unfortunate consequence of this was that the algorithm was inadequately described and led to much confusion about what the BLAST algorithm was over the ensuing years. However, the use of the match statistics was a great advance and enhanced the popularity of the engine, as previously there had been much optimistic reporting of statistically insignificant matches in the formal molecular biology literature.

1.5 Doubling Extension of $\log_s n$ Seeds

While BLAST was being developed, I continued to pursue the quest for a provably efficient deterministic algorithm. The simple seed and test strategy hadn't yielded

an analytic expected-time complexity, but it did suggest that $k = \log_s n$ might be a good seed size. Indeed, I liked immediately that since $s^{\log_s n} = n$, a simple $2n$ integer index of all the k-mers in the text permits $O(p + h)$ expected-time lookup of all h exact matches to a string of length p in the text. Later I was further able to give an analytic estimate for the size of neighborhoods of strings of this special size, but at the time I was focused on the extension step, as it was the aspect that was not yielding an analytic bound. Later I would prove it, but at the time I intuited that if ϵ is small enough, then the probability, $\Pr(p, \epsilon)$, of a random ϵ-match to a string of length p is less than $1/\alpha^p$ for some fixed $\alpha > 1$ that is a function of ϵ (just as an exact match has probability $1/s^p$). If this is true, then as shown below, the time for an extension strategy based on progressively doubling and checking seed hits telescopes for false hits.

The basic "double and check" idea is as follows. Suppose a k-mer of the query, s_0, ϵ-matches a substring t_0 of the database. The idea of doubling and checking, is to try a $2k$-mer s_1 of the query that spans s_0 and check with the customary zone-based dynamic programming algorithm if there is a string t_1 spanning t_0 that ϵ-matches s_1. If not, then one can, under the right doubling protocol to be given shortly, conclude that an ϵ-match to the query does not exist that spans t_0. Otherwise, a match to the query is still possible, so one proceeds to double s_1 to a substring s_2 of the query of length $4k$ and then check for an ϵ-match to it spanning t_1. And so on, until either there is a failure to match at some doubling stage, or until all of the query is found to match a string spanning the seed hit t_0.

Returning to the complexity claim, if one assumes $\Pr(p, \epsilon) < 1/\alpha^p$ for some α, then one starts with $h = (p/k)(n/\alpha^k)$ expected random k-mer seed matches. The idea is to check if these can be extended to ϵ-matches of length $2k$, and then to ϵ-matches of length $4k$, and so on. For a text that is random with respect to the query, the extensions that survive diminish hyper-geometrically. Specifically, there are $n(p/k)/\alpha^{2^{x-1}k}$ surviving hits at doubling stage x and it takes $O(\epsilon(2^x k)^2)$ time to check each implying that the total expected time to eliminate *all* of these random seeds is

$$n(p/k) \sum_{x=1}^{\log_2 p/k} \epsilon\left(2^x k\right)^2 / \alpha^{2^{x-1}k} = nek/\alpha^k \sum_{x=1}^{\log_2 p/k} 4^x / \alpha^{(2^{x-1}-1)k}$$
$$= O\left(nek/\alpha^k\right) \tag{1.2}$$

But how are the doublings of the seeds arranged? To keep it simple, suppose k divides p, and $p/k = 2^\pi$ is a power of 2. If a query w of length p has an ϵ-match to a substring v of the text, then by the Pigeon Hole principle either the first or second half of w, defined as w_0 and w_1, ϵ-matches a prefix v_0 or suffix v_1 of v, respectively, where $v_0 v_1 = v$. Inductively if w_x has an ϵ-match to a string v_x, then by the Pigeon Hole principle either the first or second half of w_x, defined as w_{x0} and w_{x1}, ϵ-matches a prefix v_{x0} or suffix v_{x1} of v, respectively, where $v_{x0} v_{x1} = v_x$. In other words, if there is an ϵ-match to the query w, then there is at least one binary string α of length π such that w_β has an ϵ-match to a string v_β for all prefixes β of α

where it is further true that $v_{\beta x}$ is a prefix or suffix of v_β according to whether x is 0 or 1, respectively. So now reverse the logic and imagine one has found a seed ϵ-match to a piece w_α of the query where $\alpha = a_1 a_2 \ldots a_\pi$. To determine if w has a match involving this seed match, one considers checking for ϵ-matches to the doubling sequence of strings $w_{(a_0 a_1 \ldots a_{\pi-1})}, w_{(a_0 a_1 \ldots a_{\pi-2})}, \ldots, w_{(a_0 a_1)}, w_{(a_0)}, w_{(\epsilon)} = w$, discovering the prefix and/or suffixes v_β at each level, until either w is confirmed or a check fails. This strategy is deterministic as it never misses a match, yet the expected time spent on a false positive seed is $O(\epsilon k^2)$.

When there is a match present, the time spent confirming the match is

$$\sum_{x=1}^{\pi} \epsilon \left(2^x k\right)^2 = \epsilon k^2 \sum_{x=1}^{\pi} 4^x = \epsilon k^2 \left(4^{\pi+1}/3 - 1\right) < 4/3 \epsilon p^2 = O(ep) \tag{1.3}$$

So in the case that there are exactly h real matches to the query, the extension and reporting phase of an algorithm using this strategy takes expected time:

$$O\left(nek/\alpha^k + hep\right) \tag{1.4}$$

When in particular k is chosen to be $\log_s n$, then $\alpha^k = n^{\log_s \alpha}$ and so the extension step takes $O(en^{1-\log_s \alpha} \log n + hep)$ time. This is exciting because as long as α is greater than 1 (how much so depends on ϵ), then the time is $O(n^c)$ for $c < 1$ and hence truly sublinear in n. In the next section, we will confirm that indeed $\Pr(k, \epsilon) < 1/\alpha^k$ for an α that is a function of ϵ, and thus that the complexity of this section holds.

1.6 Neighborhood Size

I was fairly confident I could come up with a good algorithm for generating neighborhoods that was proportional to the size of the condensed neighborhood, but I was less certain about arriving at an analytic upper bound for the size of a condensed d-neighborhood of a string of length k, $\bar{\aleph}_d^k$. In the introduction I (inaccurately) estimated it as $\binom{k}{d}(2s)^d$ and in the previous section I guessed such a bound would have the form $\alpha(\epsilon)^k$ where α depends on $\epsilon = d/k$. I embarked on developing recurrences for counting the number of sequences of d distinct edits that one could perform on a string of length k. Rather than consider induction over the sequence of edits, I thought about an induction along the characters of the string from left to right. At each character one can either leave it alone, delete it, insert some number of symbols after it, or substitute a different symbol for it and optionally insert symbols after it. Note carefully that redundant possibilities, such as deleting the symbol and then inserting a character after it, or substituting a symbol and then deleting it, need not be counted. While I took some care to produce tight recurrences at the time, I recently noted that I could improve the recurrences but interestingly I could not prove a better complexity bound than with the original recurrence. We will present the new recurrence with the idea that another investigator might prove a better bound.

Suppose one has k symbols left in the query, and needs to introduce d differences into this string of remaining characters where insertions before the first symbol are not allowed. Let $S(k,d)$ be the number of such d-edit scripts. The new lemma is

Lemma *If $k \leq d$ or $d = 0$ then $S(k,d) = \bar{\aleph}_d(k) = 1$. Otherwise,*

$$\begin{aligned} S(k,d) &= S(k-1,d) + (s-1)S(k-1,d-1) \\ &\quad + (s-1)\sum_{j=0}^{d-1} s^j S(k-2,d-1-j) \\ &\quad + (s-1)^2 \sum_{j=0}^{d-2} s^j S(k-2,d-2-j) + \sum_{j=0}^{d-1} S(k-2-j,d-1-j) \end{aligned}$$

$$\bar{\aleph}_d(k) \leq S(k,d) + \sum_{j=1}^{d} s^j S(k-1,d-j)$$

Proof The new recurrences begins with the observations that (a) a deletion followed by an insertion is the same as a substitution, (b) a deletion followed by a substitution is the same as a substitution followed by a deletion, (c) an insertion followed by a substitution is the same as a substitution followed by an insertion, and (d) an insertion followed by a deletion is the same as doing nothing. Therefore we need only consider scripts in which deletions can only be followed by deletions or an unchanged character, and in which insertions can only be followed by other insertions or an unchanged character. A substitution or unchanged character can be followed by any edit (or no edit). Furthermore, it is redundant to substitute a character for itself implying there are only $s-1$ choices for a substitution at a given position. Moreover, an insertion following an unchanged or substituted character is redundant if the inserted character is equal to the one behind it, because the net effect is the same as inserting the given character *before* the symbol it follows. So there are only $s-1$ non-redundant characters for the first insert in a sequence of inserts. Finally, we need only produce *condensed* neighborhoods, so when $t \leq d$ the number of scripts is 1 as the null string is in the neighborhood and hence the condensed neighborhood contains only this string. Thus it is clear that $S(k,d) = 1$ when either $k \leq d$ or $d = 0$. For all other values of k and d it follows from the "rules" above that

$$\begin{aligned} S(k,d) &= S(k-1,d) + (s-1)\big(S(k-1,d-1) + I(k-1,d-1)\big) \\ &\quad + (s-1)^2 I(k-1,d-2) + D(k-1,d-1) \end{aligned}$$

where $I(k,d)$ is the number of d edit scripts that immediately follow one or more inserts after the $(k+1)$st symbol in the query string, and $D(k,d)$ is the number of d edit scripts that immediately follow a deletion of the $(k+1)$st symbol in the query

string. It follows from the "rules" that

$$I(k,d) = sI(k,d-1) + S(k-1,d)$$
$$D(k,d) = D(k-1,d-1) + S(k-1,d)$$

Solving the recurrences for I and D in terms of S and substituting these back into the recurrence for S gives the final recurrence of the lemma for S, and the bound for $\bar{\aleph}_d(k)$ simply considers that one can have one or more inserts before the first character of the query. □

A simple but tedious exercise in induction reveals that for any value $c \geq 1$, $S(k,d) \leq B(k,d,c)$ and $\aleph_d(k) \leq \frac{c}{c-1}B(k,d,c)$ where $B(k,d,c) = (\frac{c+1}{c-1})^k c^d s^{\bar{d}}$. It further follows that $B(k,d,c)$ is minimized for $c = c^\star = \epsilon^{-1} + \sqrt{1+\epsilon^{-2}}$ where $\epsilon = d/k$. Given that $\epsilon \in [0,1]$, it follows that $c^\star \in [1+\sqrt{2}, \infty]$ implying $c^\star$ is always larger than 1 and that $\frac{c}{c-1}$ is always less than $1+\sqrt{0.5}$. Therefore,

$$S(k,d) \leq B\big(k,d,c^\star\big) \quad \text{and} \quad \aleph_d(k) \leq 1.708B\big(k,d,c^\star\big) \tag{1.5}$$

As in the original paper, one can similarly and easily develop recurrences for the probability $\Pr(k,\epsilon)$ of an ϵ-match to a string of length k in a uniformly random text and show that the recurrence is bounded by $\frac{c}{c-1}B(k,d,c)/s^k$ where $d = \lceil \epsilon k \rceil$. Therefore:

$$\Pr(k,\epsilon) \leq 1.708/\alpha(\epsilon)^k \quad \text{where } \alpha(\epsilon) = \left(\frac{c^\star - 1}{c^\star + 1}\right) c^{\star -\epsilon} s^{1-\epsilon} \tag{1.6}$$

proving that the bound used in Sect. 1.5, and hence also the complexity of the extension step of the algorithm derived in that section.

Now consider the function $\text{pow}(\epsilon) = \log_s \frac{c^\star+1}{c^\star-1} + \epsilon \log_s c^\star + \epsilon$. A little algebra and the bounds in Eqs. (1.5) and (1.6) allow us to conclude the following rather striking bounds:

$$\aleph_\epsilon(k) = O\Big(\big(s^{\text{pow}(\epsilon)}\big)^k\Big) \quad \text{and} \quad \alpha(\epsilon) = O\big(s^{1-\text{pow}(\epsilon)}\big) \tag{1.7}$$

The first bound effectively says that one can think of each position in the string as have a certain "flex factor" at a given rate ϵ, namely $s^{\text{pow}(\epsilon)}$, so that the neighborhood size is the kth power of the flex factor. The second bound effectively says that the "match specificity" of each position in the set of neighborhood strings is $s^{(1-\text{pow}(\epsilon))}$, so that the probability of matching any string in the ϵ-neighborhood of a string of length k is 1 over the kth power of this match specificity.

While quite complex in form, note that $\text{pow}(\epsilon)$ is monotone increasing and concave in ϵ and $\text{pow}(0) = 0$. The last fact implies $\aleph_0(k) = 1$ and $\alpha(0) = s$ as expected. It rises to the value of 1 before ϵ becomes 1, and does so at a point that depends on the size s of the alphabet. We plot it below in Fig. 1.1 for several value of s. Finally, note that if $k = \log_s n$ then $\aleph_\epsilon(k) = O(n^{\text{pow}(\epsilon)})$.

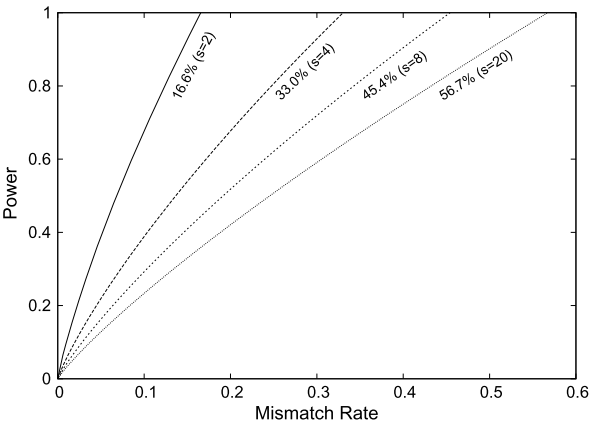


Fig. 1.1 Pow(ϵ) plotted for several values of s. For each curve the percentage mismatch at which Pow becomes 1 is given

1.7 Generating Condensed Neighborhoods

With the analysis of complexity in hand, the only remaining problem was to efficiently generate all the strings in a condensed d-neighborhood of a string w of length k. The basic idea is to explore the tree of all strings over the underlying alphabet, computing row by row the dynamic programming matrix of w versus the string on the current path in the tree. That is, given the last row, $L_{v,w}$, of the dynamic programming matrix for w versus a string v, one computes, in $O(k)$-time, the last row of the dynamic program matrix for w versus va for every letter a in the alphabet. For the simple Levenshtein measure, the smallest value in a row is monotonically increasing and therefore once a row R has a minimum value, $\min(R)$, greater than d, one can certainly eliminate the current string and all extensions of it as belong to the condensed neighborhood. Conversely, once a row is reached that has d as its last entry, then a string in the condensed neighborhood has been reached and one should report the current string and then backtrack. In pseudo code, one calls Search(ϵ, [012...k]), where *Search* is the routine:

```
Search(v, R)
  if R[k] = d then
    Report v
  else if min(R) ≤ d then
    for a ∈ Σ do
      Compute S = L_{va,w} from R
      Search(va, R)
```

The big problem above is that too much time is taken visiting words that are not in the condensed neighborhood. As soon as $\min(R)$ is d, we know that the only possible words in the condensed neighborhood are those that are extended by the suffix w^x for each x such that $R[x] = d$, where w^x is the suffix of w consisting of its last $k - x$ symbols. This gives us the algorithm:

Search(v, R)
1. if min(R) = d then
2. for each x s.t. R[x] = d do
3. Report $v \cdot w^x$
4. else # min(R) < d #
5. for $a \in \Sigma$ do
6. Compute $S = L_{va,w}$ from R
7. Search(va, S)

Now the number of terminal strings visited (i.e., those for which no further recursion is pursued) is less than the number of words in the condensed neighborhood as at least one member is reported in lines 2 and 3. Moreover, the number of interior strings is also less than the number of terminal strings as the recursion tree is an s-ary complete tree. Thus the total number of calls to Search is $O(\bar{\aleph}_d(k))$ and each call takes $O(k)$ time.

Next, immediately note that one need only compute the $2d + 1$ values of the dynamic programming matrix that are in the band between diagonals $-d$ and d as one can easily show that any value outside this band must be greater than d. Thus the relevant part of each row is computed in $O(d)$ time. But then how does one look up $v \cdot w^x$ in an index in less than $O(k)$ time given that $|w^x|$ can be on the order of k?

The answer comes from the fact that $k = \log_s N$ and thus we can build a very simple index based on directly encoding every k-mer w as an s-ary number, code(w), in the range $[0, s^k - 1] = [0, n - 1]$. It is an easy exercise (and shown in the earlier paper [8]) that with two n-vectors of integers, one can build an index that for each k-mer code delivers the positions, in the underlying text (of length n), at which that k-mer occurs. So with such a simple structure, reporting a string in the neighborhood requires only delivering its code. First note that one can incrementally compute code(va) = code(v) $\cdot\, s$ + code(a) in $O(1)$ time, and second, that one can *precompute* code(w^z) and power(z) = code(s^z) for every z in a relatively minuscule $O(k)$ time before starting the search. So during the generation of neighborhoods, one gets the code for $v \cdot w^x$ in $O(1)$ time by way of the fact that code($v \cdot x$) = code(v) $\cdot$ power($k - x$) + code(w^x).

The careful reader will note that there is one remaining problem. Namely, that in line 2 of *Search* there could be two or more entries in R, say x and $z > x$ such that $R[x] = R[z] = e$ and it could be that $v \cdot w^x$ is not in the condensed neighborhood because w^z is a prefix of w^x! To me it was fascinating that the answer lies in the failure function, ϕ, of the Knuth–Morris–Pratt (KMP) algorithm for finding matches to a string in a linear time scan of a text. Recall that for a string $v = a_1 a_2 \ldots a_k$ that $\phi(x)$ is the maximum y such that $a_1 a_2 \ldots a_y$ is a suffix of $a_1 a_2 \ldots a_x$. A table of $\phi[x]$ for all x was shown by KMP to be constructible in $O(k)$ time. Building ϕ on the reverse of w gives us exactly the relationships we want. That is, $\phi(x)$ is the maximum y such that w^y is a prefix of w^x. To test in the general case that w^z is a prefix of w^x, we test if $\phi^k(x) = z$ for some number of application k of ϕ. Since only $2e + 1$ contiguous suffixes will be considered at the most, using the failure function to test if any one is a prefix of another takes $O(e)$ time with a simple marking strategy.

Thus, we have an $O(d\aleph_d(k))$ algorithm for generating all the words in the condensed d neighborhood of a string w of length k. Given seed size k, there are p/k seeds and so generating the strings in the condensed neighborhoods of all of them takes $O(p/k \cdot \epsilon k \aleph_\epsilon(k)) = O(e\aleph_\epsilon(k))$ time.

1.8 Total Running Time

Putting the time complexities of Sect. 1.7 for the generation of neighborhoods and Sect. 1.5 for the extension of seed matches, we have as a function of seed size k:

$$O\left(e\,\aleph_\epsilon(k) + nek/\alpha(\epsilon)^k + hep\right) \tag{1.8}$$

Using Eq. (1.7) from Sect. 1.6, the complexity excluding the $O(hep)$ true positive term is

$$O\left(e\left(s^{\mathrm{pow}(\epsilon)k} + nk/s^{(1-\mathrm{pow}(\epsilon))k}\right)\right) = O\left(e\left(s^k\right)^{\mathrm{pow}(\epsilon)}\left(1 + k\frac{n}{s^k}\right)\right) \tag{1.9}$$

When one chooses $k = \log_s n$ as the seed size, then $s^k = n$ and we formally arrive at the expected-time complexity for the entire algorithm given in the following theorem.

Theorem *Using a simple $O(n)$ precomputed index, one can search for ϵ-matches to a query of length p in a text of length n, in*:

$$O\left(en^{\mathrm{pow}(\epsilon)} \log n + hep\right) \textit{ expected time} \tag{1.10}$$

where $\mathrm{pow}(\epsilon) = \log_s \frac{c^\star+1}{c^\star-1} + \epsilon \log_s c^\star + \epsilon$ *and* $c^\star = \epsilon^{-1} + \sqrt{1+\epsilon^{-2}}$.

1.9 Final Remarks and Open Problems

In a hopefully intuitive style, the reader has been introduced to a fairly involved set of theoretical ideas that underlie the BLAST heuristic and that give a deterministic, expected-time algorithm that is provably sublinear in the size of the text for suitably small ϵ. That is the algorithm's running time is $O(n^c)$ for $c < 1$ (and not $O(cn)$ for $c < 1$ as in a "sublinear" method such as the Boyer-Moore exact match algorithm). Interestingly, the author is not aware of any work that builds on this approach or another that has a superior time complexity.

There are at least two interesting questions. The first involves the analysis of neighborhood sizes. Is there a tighter bound to the recurrences formulated here and/or are there better recurrences? Such bounds would give a tighter characterization of the running time of the algorithm. The second question is a bit harder to formulate, but the essence of it is whether or not this algorithm can be shown to be a

lower bound on the time required to find all ϵ-matches. In other words, one wonders whether the idea of using a $\log_s n$ seed size and then carefully doubling such hits is essential for ruling out false positive locations. That is, must one spend this amount of time eliminating a near miss? If true, it would also explain why a better result of its kind has not been forthcoming in the last 20 years since the first publication of this result.

References

1. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
2. Ukkonen, E.: Algorithms for approximate string matching. Inf. Control **64**, 100–119 (1985)
3. Myers, E., Miller, W.: Optimal alignments in linear space. Comput. Appl. Biosci. **4**(1), 11–17 (1988)
4. Landau, G., Vishkin, U.: Efficient string matching with k mismatches. Theor. Comput. Sci. **43**, 239–249 (1986)
5. Pearson, W.R., Lipman, D.J.: Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA **85**, 2444–2448 (1988)
6. Myers, E.: An $O(ND)$ difference algorithm and its variations. Algorithmica **1**(2), 251–266 (1986)
7. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. **215**(3), 403–410 (1990)
8. Myers, E.: A sublinear algorithm for approximate keyword searching. Algorithmica **12**(4/5), 345–374 (1994)
9. Weiner, P.: Linear pattern matching algorithm. In: 14th Annual IEEE Symposium on Switching and Automata Theory, pp. 1–11 (1973)
10. Manber, U., Myers, E.: Suffix arrays: a new method for on-line searches. In: Proc. 1st ACM-SIAM Symp. on Discrete Algorithms, pp. 319–327 (1990)
11. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation (1994)
12. Li, H., Ruan, J., Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res. **18**(11), 1851–1858 (2008)
13. Jokinen, P., Ukkonen, E.: Two algorithms for approximate string matching in static texts. In: Proc. of MFCS'91. LNCS, vol. 520, pp. 240–248 (1991)
14. Ukkonen, E.: Approximate string-matching with q-grams and maximal matches. Theor. Comput. Sci. **92**(1), 191–211 (1992)
15. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. **13**(2), 338–355 (1984)
16. Roberts, L.: New chip may speed genome analysis. Science **244**, 655–656 (1989)
17. Mealy, G.: A method for synthesizing sequential circuits. Bell Syst. Tech. J. **34**, 1045–1079 (1955)
18. Moore, E.: Gedanken-experiments on sequential machines. In: Automata Studies. Annals of Mathematical Studies, vol. 34, pp. 129–153. Princeton University Press, Princeton (1956)
19. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. **25**(17), 3389–3402 (1997)
20. http://blast.ncbi.nlm.nih.gov/Blast.cgi
21. Karlin, S., Altschul, S.: Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA **87**, 2264–2268 (1990)

Chapter 2
Forty Years of Model-Based Phylogeography

David Bryant and Jamie Kydd

Abstract The roots of model-based phylogeography are usually traced back to the celebrated papers of Wright, Kimura, Cavalli-Sforza and Edwards, and Thompson. Here we discuss a 1972 paper of Sankoff which we believe also belongs among the foundational papers of the field. In it, Sankoff presents a joint model of geographic subdivision and genetics and shows how both geography and phylogeny can be estimated simultaneously from data. We review the paper, and discuss how it connects to contemporary work in the area.

2.1 Introduction

Spatial structure has played a fundamental role in the evolutionary history of most organisms and any attempt at reliable phylogenetic inference needs to take this into account. The real problem is how to do this in practice: how to incorporate sample locations, present and past geography, and the effect of spatial correlations into phylogenetic inference. Integrating geography with phylogeny leads one quickly to a quagmire of difficult modeling and methodological issues, many of which remain unresolved.

It will come of no surprise to computational biologists that one of the first papers to develop methodology for model-based phylogeography is by David Sankoff. His 1972 paper, *Reconstructing the History and Geography of an Evolutionary Tree* [28] was, in many ways, 30–40 years ahead of its time. It describes a stochastic model incorporating *both* genetics and geography and shows how both geography and phylogeny can be estimated simultaneously from data. Sankoff's model captures important features of a dynamic spatial structure without being bogged down in a mass of geographic and environmental data. Curiously, the paper has been seldom cited, and almost entirely only within the field of lexicostatistics. It appears

D. Bryant (✉)
University of Otago, Dunedin, New Zealand
e-mail: david.bryant@otago.ac.nz

J. Kydd
University of Auckland, Auckland, New Zealand
e-mail: jamie.b.kydd@gmail.com

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_2,

to have been completely missed by the phylogeographers. Our opinion is that the novelty of ideas and models places this work among the classic early papers of model-based phylogeography.

In this chapter we review Sankoff's article and argue that this paper merits revisiting. First we consider the problem of modeling geography, outlining the model in Sankoff's paper and demonstrating some links with related models. We then consider the interaction of genetics and geography. Sankoff introduces an appealing method for reconstructing phylogeographic patterns, one which has analogues in Markov random field theory. We compare this approach to some contemporary methodology in phylogenetics. We argue that Sankoff's approach is genuinely different and that it is not without its advantages.

2.2 Modeling Geography

2.2.1 Background

There are two key design decisions to be made in any model integrating geography and genetics [13]. The first is the statistical unit of analysis: does the model describe individuals, family units, villages, populations, species, or something in between. The second is the effect of geography on movement between these groups, or more correctly, on the gene flow between these groups.

The theoretical foundation for the most commonly used migration models in population genetics is provided by the island models of Wright [32], the stepping stone models [15] and their generalization to arbitrary migration matrices [3]. Under these models, the 'islands' or subpopulations are fixed a priori. The rate of migration or gene flow between different islands might be different for each pair, or equal to a constant, or capturing some aspect of the geographic structure. In many cases the choice of migration rates is governed more by mathematical convenience than biological realism.

We note that there are several models which do not break the population up into discrete chunks but instead consider a distribution of individuals in space. These have had far less impact than the island-based models. One important reason for the discrepancy is that it is straightforward to set up a working discrete population model, whereas continuous space models have hidden difficulties. The continuous space model of Malecot [23] appears, at least superficially, to be quite reasonable and conservative. It is nevertheless internally inconsistent, as demonstrated by Felsenstein [12].

2.2.2 A Joint Model for Phylogeny and Geography

In Sankoff's 1972 paper, the (sub)-populations are represented as vertices in a planar graph or, equivalently, as regions in the plane. Migrations occur at a fixed and constant rate between populations connected by an edge. There is, however, a ma-

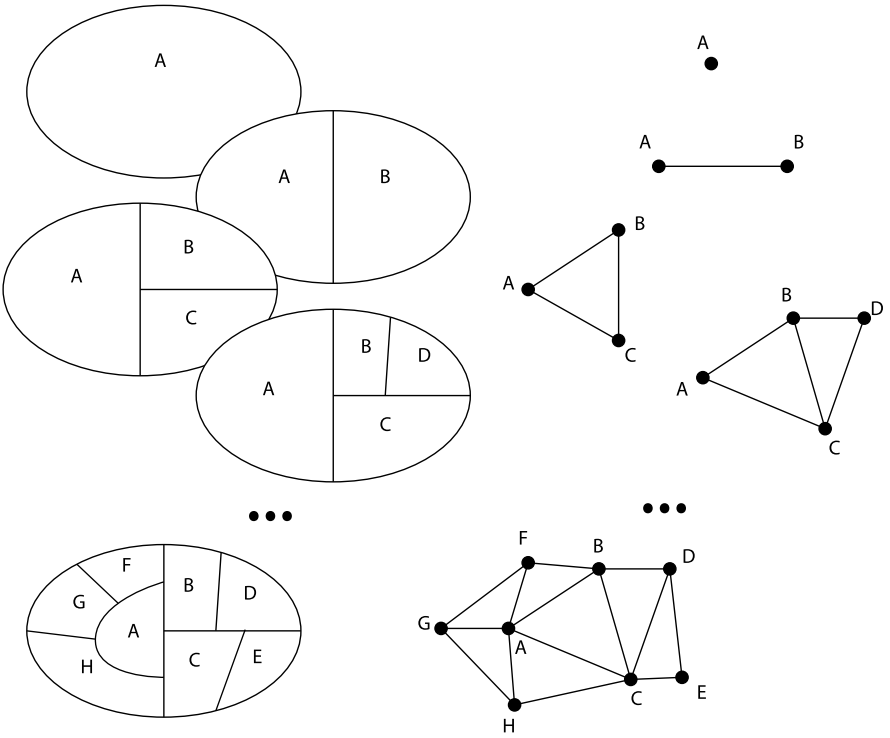


Fig. 2.1 The splitting processed studied by Sankoff. Initially there is a single region, here represented by a single region *A* (*on the left*) or a single vertex in the adjacency graph (*on the right*). At each splitting, a region is subdivided by selecting two edges and joining them with a new boundary line. The corresponding adjacency (dual) graphs appear *on the right*

jor difference between Sankoff's model and others based on networks on the plane: Sankoff's model is dynamic. The model includes not only the migration patterns between contemporary populations; it also describes how these connections change over time.

Initially, there is one region, corresponding to population at the root of the tree. After each population split (divergence/speciation) the corresponding region is subdivided, thereby adding one new vertex to the dual graph describing adjacencies (Fig. 2.1). This process continues until we obtain one region for every contemporary population. In a way, the model describes a process of successive allopatric speciations.

A splitting is carried out as follows. A region is chosen uniformly at random. Two of the edges bounding that region are picked uniformly and the region is split in a way that subdivides both edges. In the case that there is only one sequence, the boundary of that region is subdivided twice and the points of subdivision are joined by a new edge, creating two regions (Fig. 2.1). There is considerable scope for different selection schemes, and Sankoff says as much in his paper.

One splitting process is illustrated in Fig. 2.1. Initially there is a single region A. This is subdivided to give two regions A and B. Then B is subdivided, giving three regions A, B, C, and a further splitting of B gives a map with four regions A, B, C, D. The adjacencies between these regions are represented by the graphs on the right.

One aspect of the model that is not completely clear from Sankoff's paper is whether the external region could be split. The process would still be well defined, and these splittings could model expansion into new territory which lies outside the boundary of the original region A. The adjacency graph would be modified to include a specially marked vertex representing the external region, with edges to every region adjacent to the exterior boundary.

In either case, the process captures many aspects of connectivity relating to spatial structure. What is surprising is how much it leaves out. It contains no information about region size, or different environments, or even (beyond adjacency) the shape of the regions. This sparsity of information could turn out to be particularly useful. The model is clearly more believable than, say, models of populations distributed on a torus, or populations with no spatial structure at all. The model is clearly less 'realistic' than those incorporating landscape simulators and small-scale geographic niches. However, realism in any model is only relative, and a hyper-realistic model is useless if it is not tractable.

2.2.3 Properties of Splitting

Sankoff's splitting process produces random planar adjacency graphs. The first natural question is whether there is anything special about the particular graphs produced.

To answer this, we start by looking not at the graph, but at the configurations of regions (or cells) produced by subdivision. Observe that the faces of the adjacency graph correspond to places where more *at least* three regions meet at a point, and that every time this happens, *exactly* three regions meet at that point. As a consequence all of the faces of the adjacency graph, except perhaps the external face, are triangular. It is not hold to show that this will always be the case, provided the number n of regions is at least three.

In the second version of the splitting model we permit splittings of the external vertex. In this case, all faces of the adjacency graph will be triangular, so that the adjacency graph (as a drawing) is a planar triangulation. There has been a great deal of work on these triangulations due to their application in surface visualization [14], finite element methods [5] and spatial data analysis [22]. The splitting operation of Sankoff corresponds exactly to *vertex splitting*; the reverse operation is called *edge contraction*.

An edge is said to be *contractible* if contracting that edge produces a valid triangulation. It can be easily shown that an edge e is contractible if and only if

- e does not lie on any triangle of the graph which is not a face of the graph; and
- The triangulation is not K_4 embedded on a sphere.

Steinitz and Rademacher [30] proved in 1934 that every triangulation of the sphere (assuming $n \geq 4$) can be converted into K_4 by a sequence of (valid) edge contractions. As a consequence, we see that the graphs produced by this version of the splitting process are exactly the triangulations: planar graphs with triangular faces.

Now consider the version of the splitting process where splittings of the external region are *not* allowed. This change makes the analysis a little more complicated. The adjacency graph is still planar, and every face except the exterior face is triangular. The problem of finding valid edge contractions in this case is known as *polygon reduction*, a problem with applications to 3D graphics in the gaming industry [24]. Conditions for a valid edge contraction in this instance were established by [14]:

- e does not lie on any non-facial triangle; and
- the triangulation is not K_3 embedded in the plane.

An analogue of Steinitz and Rademacher's result for this version follows from Theorem 3 of [14] (which establishes that the triangle is a *subdivision* of a triangulation) and Lemma 4 of [14] (which shows that a subdivision of these triangulations can be obtained by edge contractions). Hence the adjacency graphs produced by this version of Sankoff's process are exactly the plane graphs with all triangular faces except the exterior face (also called *simplicial surfaces* [14]).

There are many more avenues for mathematical investigation here. The splitting model generates a distribution on triangulations; what is that distribution? Is it possible to compute, in polynomial time, the probability of a given planar triangulation? This question has a similar flavor to analyses of processes generating random trees [29].

2.3 Modeling Genetics

2.3.1 The Model

The genetic model described in Sankoff's paper is essentially the *infinitely many alleles* model, first introduced by [15]. It assumes a set Γ of genetic sites, and at each site the populations (regions) have a particular state (allele). Every mutation produces a new and unique type, and the only information available is whether two individuals carry the same type of allele at a site. There is a further assumption that there is no variation *within* each region (polymorphism). This is akin to assuming that the population sizes are small so that any mutant is quickly lost or fixed. A similar assumption is made by [27].

There are three distinct processes contributing to gene dynamics in the model. First, mutation: for each region, mutations occur at a fixed rate, and each mutation creates a new and distinct genetic type. Second, migration: at a fixed rate the type of a region is transferred from one of the adjacent regions. Random processes of this form are examples of *Markov random fields*, and appear in numerous guises in fields ranging from statistical physics to epidemiology. Kindermann and Snell [16]

credit the first work on Markov random fields to mathematicians in the former Soviet Union (e.g. [10]). Their appearance in the West coincided roughly with the publication of Sankoff's paper.

Of course the connection with Markov random fields only applies *between* splitting times, when the adjacency graph remains constant. This third process, splitting, adds a fairly novel twist to the analysis (compare the 'splitting operator' of [31]).

For the moment, consider the dynamics of an individual site. The relevant state information at a time t is then just the partition of the populations into types. We can analyze the model as a Markov chain with a state space equal to the set of partitions. A mutation takes a population and puts it into a class by itself. A migration (borrowing) transfers a population from one class to another. A splitting duplicates an element contained in one of the classes.

2.3.2 Dynamic Similarity

The approach taken by Sankoff is to bypass computations over the space of partitions, and instead concentrate on the dynamics for pairs of populations. The resulting calculations are still exact and take account of all populations simultaneously, they just do not capture all of the higher-order dependencies between populations.

Let $\mathbf{X}_t, \mathbf{Y}_t$ be two populations at some time t in the past, where t ranges from $t = 0$ (the present) to $t = -T$ (time of the first splitting). Let $s(\mathbf{X}_t, \mathbf{Y}_t)$ denote the proportion of sites at which $\mathbf{X}_t$ and $\mathbf{Y}_t$ share the same state. Sankoff's calculations assume that the set Γ of sites is large enough that $s(\mathbf{X}_t, \mathbf{Y}_t)$ coincides with the *probability* that the populations have the same type at a *particular* site, or equivalently that $s(\mathbf{X}_t, \mathbf{Y}_t)$ is the *expected* proportion of sites at which the two populations have the same state.

A system of differential equations can be derived for $s(\mathbf{X}_t, \mathbf{Y}_t)$ by means of a case-by-case analysis. For the moment, just consider time periods between population splits, so that the only processes to analyse are mutations and migrations. For each region X, let $N_{\mathbf{X}}$ denote the set of neighboring vertices (excluding X itself) and let $k(\mathbf{X}) = |N_{\mathbf{X}}|$. Let r denote the rate of mutation (loss) of an allele at a single locus and a the rate at which a region $\mathbf{X}$ adopts a type (borrows) from one of its $k(\mathbf{X})$ neighbors.

We then have

$$\frac{ds(\mathbf{X}_t, \mathbf{Y}_t)}{dt} = -2rs(\mathbf{X}_t, \mathbf{Y}_t) \tag{2.1}$$

$$+ \big(1 - s(\mathbf{X}_t, \mathbf{Y}_t)\big)a\big(1/k(\mathbf{X}) + 1/k(\mathbf{Y})\big) \tag{2.2}$$

$$+ \frac{a}{k(\mathbf{X}) - 1} \sum_{\mathbf{Z} \in N_{\mathbf{X}} - \{\mathbf{Y}\}} \big(1 - s(\mathbf{X}_t, \mathbf{Y}_t)\big)s(\mathbf{Y}_t, \mathbf{Z}_t)$$

$$- s(\mathbf{X}_t, \mathbf{Y}_t)\big(1 - s(\mathbf{Y}_t, \mathbf{Z}_t)\big) \tag{2.3}$$

$$+\frac{a}{k(\mathbf{Y})-1}\sum_{\mathbf{Z}\in N_{\mathbf{Y}}-\{\mathbf{X}\}}\big(1-s(\mathbf{X}_t,\mathbf{Y}_t)\big)s(\mathbf{X}_t,\mathbf{Z}_t)$$

$$-s(\mathbf{X}_t,\mathbf{Y}_t)\big(1-s(\mathbf{X}_t,\mathbf{Z}_t)\big). \tag{2.4}$$

Here (2.1) corresponds to the loss of identity following mutation in X or Y; (2.2) follows from the gain in identity when $\mathbf{X}$ or $\mathbf{Y}$ obtain a state from each other; (2.3) corresponds to the event when a state is transferred from a neighbor of $\mathbf{X}$ to $\mathbf{X}$ which either restores or removes identity;[1] while (2.4) is the symmetric case for $\mathbf{Y}$. Unaware of Sankoff's work, Bryant re-derived analogous equations in [6], a paper modeling network breaking in the Polynesian languages.

Sankoff showed that if we are provided with the adjacency graph at time 0 (the present) as well as the quantities $s(\mathbf{X}_0,\mathbf{Y}_0)$ for all $\mathbf{X},\mathbf{Y}$, the entire history of splittings can be reconstructed. The proof works by induction on the number of populations. Let $\mathbf{U},\mathbf{V}$ be the regions created in the most recent split, and suppose that this occurred at time τ. We then have

- $s(\mathbf{U}_\tau,\mathbf{V}_\tau)=1$;
- $s(\mathbf{X}_\tau,\mathbf{Y}_\tau)<1$ for all X,Y not resulting from a split at time τ;
- $s(\mathbf{X}_t,\mathbf{Y}_t)<1$ for all $t>\tau$ and all X,Y including $\mathbf{U},\mathbf{V}$.

The case of multiple splittings at exactly the same time can be dealt with by picking one pair arbitrarily.

The values $s(\mathbf{X}_t,\mathbf{Y}_t)$ can be computed for $t\geq\tau$ by solving the initial value problem (2.1)–(2.4) with initial values $s(\mathbf{X}_0,\mathbf{Y}_0)$. In this way, $\mathbf{U},\mathbf{V}$ and τ can be identified as well as all of the values $s(\mathbf{X}_\tau,\mathbf{Y}_\tau)$. Replacing $\mathbf{U}$ and $\mathbf{V}$ by a single population we continue to obtain the second most recent splitting, and so on.

The analysis in Sankoff's paper assumes that the rate of mutation r and the rate of adoptions a are known. It is clear that we cannot identify both parameters given just the similarity values $s(\mathbf{X}_0,\mathbf{Y}_0)$: if we scale the rates and times simultaneously we can obtain identical similarity values. In fact, the situation is even more difficult. It was shown in [19] that in some cases one of these rates cannot be identified even though the other is known.

Sankoff makes the convenient (and acknowledged) assumption that the probabilities of identity $s(\mathbf{X}_t,\mathbf{Y}_t)$ are known without error. Any serious application of the approach to real data will require some degree of uncertainty quantification. In a paper on Polynesian languages [6], Bryant used parametric bootstrapping to estimate variance in parameter estimates. Unfortunately, parametric bootstrapping is computationally inefficient, and it can be problematic when faced with substantial model error on top of sampling error.

2.3.3 Multiway Similarities

One way to potentially address the problem of parameter estimation and uncertainty quantification is to follow the lead of Markov random field theory (e.g. [16, p. 76])

[1] A typo in Sankoff's original version of (2.3) was pointed out, rather excitedly, by [11].

and compute probabilities of identity for not just pairs, but triples and larger sets of regions.

Let $\mathcal{R}_t$ denote the set of regions present at time t. For $\mathcal{X}_t \subseteq \mathcal{R}_t$ we let $s(\mathcal{X}_t)$ denote the probability that all of the regions $\mathbf{X}_t \in \mathcal{X}_t$ have the same state at a site. Hence $s(\{\mathbf{X}_t, \mathbf{Y}_t\}) = s(\mathbf{X}_t, \mathbf{Y}_t)$ for all pairs $\mathbf{X}_t, \mathbf{Y}_t$. We note that if we define

$$\delta(\mathcal{X}) = 1 - s(\mathcal{X})$$

for all subsets $\mathcal{X}$ then (for generic t) the function δ satisfies the properties of a *diversity* [7], that is, $\delta(\mathcal{X}) \geq 0$, $\delta(\mathcal{X}) = 0$ if and only if $|\mathcal{X}| = 1$ and

$$\delta(\mathcal{X} \cup \mathcal{Z}) \leq \delta(\mathcal{X} \cup \mathcal{Y}) + \delta(\mathcal{Y} \cup \mathcal{Z})$$

whenever $\mathcal{Y} \neq \emptyset$. This gives us access to a small, but growing, set of tools and theorems to aid analysis and computation.

Our main observation here though is that Eqs. (2.1)–(2.4) translate quite elegantly to this new context. Instead of computing probabilities for pairs of regions, we compute $s(\mathcal{X}_t)$ for a set of regions $\mathcal{X}_t$, this being the probability that *all* regions in the set have the same type of allele at time t. It is now a straightforward matter to derive the differential equations for the probabilities $s(\mathcal{X}_t)$.

$$\frac{ds(\mathcal{X}_t)}{dt} = -|\mathcal{X}_t| r s(\mathcal{X}_t) \tag{2.5}$$

$$+ \big(1 - s(\mathcal{X}_t)\big) \sum_{\mathbf{X} \in \mathcal{X}} \sum_{\mathbf{Y} \in N_{\mathbf{X}}} a \frac{1}{k(\mathbf{X})} s\big(\big(\mathcal{X}_t \setminus \{\mathbf{X}\}\big) \cup \{\mathbf{Y}\}\big) \tag{2.6}$$

$$\times s(\mathcal{X}_t) \sum_{\mathbf{X} \in \mathcal{X}} \sum_{\mathbf{Y} \in N_{\mathbf{X}} - \mathcal{X}} a \frac{1}{k(\mathbf{X})} \big(1 - s\big(\big(\mathcal{X}_t \setminus \{\mathbf{X}\}\big) \cup \{\mathbf{Y}\}\big)\big). \tag{2.7}$$

Here, (2.5) captures the rate at which mutations occur within $\mathcal{X}_t$; (2.6) captures the rate by which a migration removes a dissimilar element from within $\mathcal{X}_t$, thereby making all regions have the same allele type; (2.7) captures the rate at which migrations from outside $\mathcal{X}_t$ introduce non-identical allele types.

A convenient property of (2.5)–(2.7) is that the equation for $\frac{ds(\mathcal{X})}{dt}$ involves only variables $s(\mathcal{X}')$ with $|\mathcal{X}'| \leq |\mathcal{X}|$. Extending Sankoff's approach to triples or quadruples of regions will not generate an exponential explosion in complexity.

2.4 Alternative Methods for Analysis

2.4.1 The Structured Coalescent

Given that 40 years have passed since Sankoff's model was published, we might expect dramatic progress in the tools we can bring to the analysis. Here we briefly

consider the range of modern approaches which we might use to carry out inference with the splitting model.

Perhaps the biggest methodological breakthrough in population genetics is due to *coalescent theory*, originally published by Kingman [17] but greatly advanced by a large number of mathematicians and statisticians. Let S denote the sequence data, G the geography (adjacency graph and splittings) and T the genealogical tree for a particular site. The tree is affected by both splittings and borrowings. Using the structured coalescent (e.g. [25]) one obtains a distribution $P(T|G)$ of the tree given the adjacency graph and splittings. Following a Bayesian analysis, one uses Monte Carlo algorithms to simulate values from the joint posterior distribution

$$\begin{aligned} P(G,T|S) &\propto P(S|G,T)P(G,T) \\ &= P(S|T)P(T|G)P(G). \end{aligned}$$

The posterior distribution $P(G|S)$ follows directly.

One open problem is the calculation of $P(T|G)$: given a splitting process and the resulting adjacency graph, what is the distribution for a genealogical tree. If the rate of borrowing or migration is low, then $P(T|G)$ will primarily reflect the tree formed from the splittings themselves. As the rate of borrowing increases, the distribution will become more diffuse.

It may be useful to combine the multiway similarity method of the previous section with the structured coalescent. Consider a subset $\mathcal{X}$ of regions and let $L(\mathcal{X})$ be the length of a gene tree with tips corresponding to these regions. The probability of all regions sharing the same trait is the probability of no mutation along the length of this tree, or $e^{L(\mathcal{X})}$. Hence for a random gene tree,

$$s(\mathcal{X}) = E\left[e^{L(\mathcal{X})}\right].$$

This connection could, for example, be used to check convergence when sampling gene trees.

2.4.2 Stochastic Diffusion Methods

The stochastic diffusion strategy [20, 21] is to rearrange the conditional probabilities, to give

$$P(G,T|S) \propto P(S|G,T)P(G,T) \tag{2.8}$$

$$= P(S|T)P(G|T)P(T). \tag{2.9}$$

The difference is that we now have to calculate $P(G|T)$ instead of $P(T|G)$. This method has been widely used, including several high profile applications (e.g. [4, 26]. The accompanying software produces beautiful graphics depicting phylogenies on maps.

In itself, (2.9) is, of course, completely correct, and is a simple consequence of conditional probabilities. The problem is the computation of $P(G|T)$. In [20] it is

assumed that computing $P(G|T)$ is a simple matter of adapting standard algorithms for phylogenetic characters. Kühnert et al. [18] dub these 'migration' models since they analyze migrations using models designed for mutations.

They, and a large number of earlier likelihood-based analysis, have made a critical error in their probability calculations. When they calculate the probability of the geographic locations on the tree, they assume that the same conditional independence underlying likelihood calculations for genetic data also applies for geographic characters. However, by conditioning on a tree they are making an implicit assumption that all lineages survive to the present.

If you consider an ancestral lineage, the rate of migration to a given island is one thing, the rate of migration *conditional on survival to the present* is another. If the islands (or regions) are small then any lineage is likely to be either lost or fixed. Hence if one lineage in the tree occupies a particular island it is highly unlikely that any other lineage will occupy the island at the same time. This conflict breaks down the conditional independence that is so critical for the calculation of phylogenetic likelihoods. Furthermore, ignoring these conflicts is essentially equivalent to ignoring the interplay between geographic structure and drift, assuming infinite effective population sizes within each region.

2.4.3 Approximate Bayesian Computation

Approximate Bayesian computation (ABC) [2, 8, 9] has grown immensely in popularity, partly because it (at first glance) does not require much specialist mathematical knowledge to implement and partly because it is possible to set up analyses for extremely complex models fairly simply.

The general idea is to use simulations in place of likelihood calculations, and it can be proven that, with sufficient iterations and sufficient summary statistics, the approach provably converges to the correct posterior distribution. The approach has a simplicity and transparency which makes it especially attractive. There are, unfortunately, serious issues in higher dimensional space, though recent techniques have a great deal of potential [1]. Until these issues are resolved, we suspect that we will not have the computational power to do more than infer, roughly, even the smallest splitting history.

2.5 Future Work

We see two principal methodological advances made in Sankoff's work. The first is the idea to model adjacencies not by a single graph, but by a sequence of vertex splittings generating a graph. The model uses an abstraction of the spatial component, in the sense that the vertices in the graph are not given specific geographic locations. The potential advantage of this could be tractable inferential methods which still capture aspects of geography essential to phylogenetics.

The second advance is the method for computing pairwise similarities. Bodmer and Cavalli-Sforza [3] had investigated this problem for a general migration matrix, but only derived approximate results. Sankoff's approach computes exact probabilities incorporating all regions simultaneously. We have shown that this extends beyond pairwise comparisons, and that the approach has not been superseded by modern developments, though many practical computational and statistical problems remain.

References

1. Beaumont, M.A., Nielsen, R., Robert, C., Hey, J., Gaggiotti, O., Knowles, L., Estoup, A., Panchal, M., Corander, J., Hickerson, M.: In defence of model-based inference in phylogeography. Mol. Ecol. **19**(3), 436–446 (2010)
2. Beaumont, M.A., Zhang, W., Balding, D.J.: Approximate Bayesian computation in population genetics. Genetics **162**(4), 2025–2035 (2002)
3. Bodmer, W.F., Cavalli-Sforza, L.L.: A migration matrix model for the study of random genetic drift. Genetics **59**(4), 565–592 (1968)
4. Bouckaert, R., Lemey, P., Dunn, M., Greenhill, S.J., Alekseyenko, A.V., Drummond, A.J., Gray, R.D., Suchard, M.A., Atkinson, Q.D.: Mapping the origins and expansion of the Indo-European language family. Science **337**(6097), 957–960 (2012)
5. Brenner, S.C., Scott, R.: The Mathematical Theory of Finite Element Methods. Springer, Berlin (2008)
6. Bryant, D.: Radiation and network breaking in Polynesian linguistics. In: Forster, P., Renfrew, C. (eds.) Phylogenetic Methods and the Prehistory of Languages, McDonald Institute for Archaeological Research, pp. 111–118. (2006)
7. Bryant, D., Tupper, P.F.: Hyperconvexity and tight-span theory for diversities. Adv. Math. **231**(6), 3172–3198 (2012)
8. Cornuet, J.-M., Santos, F., Beaumont, M.A., Robert, C.P., Marin, J.-M., Balding, D.J., Guillemaud, T., Estoup, A.: Inferring population history with DIY ABC: a user-friendly approach to approximate Bayesian computation. Bioinformatics **24**(23), 2713–2719 (2008)
9. Csilléry, K., Blum, M.G.B., Gaggiotti, O.E., François, O.: Approximate Bayesian computation (ABC) in practice. Trends Ecol. Evol. **25**(7), 410–418 (2010)
10. Dobrushin, R.L.: Gibbsian random fields for lattice systems with pairwise interactions. Funct. Anal. Appl. **2**(4), 292–301 (1968)
11. Embleton, S.: Lexicostatistical tree reconstruction incorporating borrowing. Toronto Working Papers in Linguistics, 2 (1981)
12. Felsenstein, J.: A pain in the torus: some difficulties with models of isolation by distance. Am. Nat. **109**(967), 359–368 (1975)
13. Guillot, G., Leblois, R., Coulon, A., Frantz, A.C.: Statistical methods in spatial genetics. Mol. Ecol. **18**(23), 4734–4756 (2009)
14. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Mesh Optimization (1993)
15. Kimura, M., Crow, J.F.: The number of alleles that can be maintained in a finite population. Genetics **49**(4), 725 (1964)
16. Kindermann, R., Snell, J.L.: Markov Random Fields and Their Applications. American Mathematical Society, Providence (1980)
17. Kingman, J.F.C.: The coalescent. Stoch. Process. Appl. **13**(3), 235–248 (1982)
18. Kühnert, D., Wu, C.-H., Drummond, A.J.: Phylogenetic and epidemic modeling of rapidly evolving infectious diseases. Infect. Genet. Evol. **11**(8), 1825–1841 (2011)
19. Kydd, J.: The effect of horizontal transfer and borrowing on phylogenetic signal. Honours dissertation, University of Auckland, Auckland (2007)

20. Lemey, P., Rambaut, A., Drummond, A.J., Suchard, M.A.: Bayesian phylogeography finds its roots. PLoS Comput. Biol. **5**(9), e1000520 (2009)
21. Lemey, P., Rambaut, A., Welch, J.J., Suchard, M.A.: Phylogeography takes a relaxed random walk in continuous space and time. Mol. Biol. Evol. **27**(8), 1877–1885 (2010)
22. De Loera, J.A., Rambau, J., Santos, F.: Triangulations: Structures for Algorithms and Applications. Springer, Berlin (2010)
23. Malécot, G.: The Mathematics of Heredity. Freeman, New York (1970)
24. Melax, S.: A simple, fast, and effective polygon reduction algorithm. Game Dev. **5**(11), 44–49 (1998)
25. Notohara, M.: The coalescent and the genealogical process in geographically structured population. J. Math. Biol. **29**(1), 59–75 (1990)
26. Okoro, C.K., Kingsley, R.A., Connor, T.R., Harris, S.R., Parry, C.M., Al-Mashhadani, M.N., Kariuki, S., Msefula, C.L., Gordon, M.A., de Pinna, E., Wain, J., Heyderman, R.S., Obaro, S., Alonso, P.L., Mandomando, I., MacLennan, C.A., Tapia, M.D., Levine, M.M., Tennant, S.M., Parkhill, J., Dougan, G.: Intracontinental spread of human invasive salmonella typhimurium pathovariants in sub-Saharan Africa. Nat. Genet. **44**(11), 1215–1221 (2012)
27. Robledo-Arnuncio, J.J., Rousset, F.: Isolation by distance in a continuous population under stochastic demographic fluctuations. J. Evol. Biol. **23**(1), 53–71 (2010)
28. Sankoff, D.: Reconstructing the history and geography of an evolutionary tree. Am. Math. Mon. **79**(6), 596–603 (1972)
29. Steel, M., McKenzie, A.: Properties of phylogenetic trees generated by Yule-type speciation models. Math. Biosci. **170**(1), 91 (2001)
30. Steinitz, E., Rademacher, H.: Vorlesungen über die Theorie der Polyeder. Springer, Berlin (1934)
31. Sumner, J.G., Jarvis, P.D.: Entanglement invariants and phylogenetic branching. J. Math. Biol. **51**(1), 18–36 (2005)
32. Wright, S.: Isolation by distance. Genetics **28**(2), 114–138 (1943)

Chapter 3
How to Infer Ancestral Genome Features by Parsimony: Dynamic Programming over an Evolutionary Tree

Miklós Csűrös

Abstract We review mathematical and algorithmic problems of reconstructing evolutionary features at ancestors in a known phylogeny. In particular, we revisit a generic framework for the problem that was introduced by Sankoff and Rousseau (Math. Program. 9:240–246, 1975).

3.1 Introduction

If extant organisms descended from a common ancestor through modifications [13], then their genomes carry clues about the extinct ancestors' genomes. This simple observation can lead to impressive insights when the descendants are sufficiently diverse. For example, by Blanchette et al.'s estimate [4], more than 95 % of an early mammalian genome can be inferred soundly from whole-genome sequences.

Linus Pauling proposed the reconstruction of ancestral molecular sequences as early as 1962 (as recounted in [41]). Pauling presented the idea by the example of 70 homologous sites in four human hemoglobin peptide chains (α, β, γ, δ) and three other related sequences available at the time. The alignment and the ancestral inference were done manually, using only amino acid identities. It took another decade to work out general computational procedures to do alignment and reconstruction. Dynamic programming, the key algorithmic technique for the task, was introduced into molecular biology by Needleman and Wunsch [40] with cursory mathematical exposition. Subsequent work in the early 1970s, including notable foundational contributions from David Sankoff, rigorously established the utility of the dynamic programming approach in problems related to sequence alignment, phylogeny and RNA secondary structure [44].

Phylogenetic reconstruction methods matured concomitantly with sequence alignment. Edwards and Cavalli-Sforza [17] proposed the idea of "minimal evolution" or *parsimony* [7]: the phylogeny should imply the least evolutionary change leading to the features observed in extant organisms. The principle, recast into prob-

M. Csűrös (✉)
Department of Computer Science and Operations Research, University of Montréal, Montreal, QC, Canada
e-mail: csuros@iro.umontreal.ca

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*, Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_3,

abilistic terms, leads to likelihood methods in phylogenetic inference [22]. Alternative ways of quantifying "evolutionary change" give rise to a number of parsimony varieties [23]. Efficient algorithms have been developed for many special cases of parsimony [7, 19, 20, 24, 28, 35, 49]. Sankoff and Rousseau [47] proposed an elegant method for parsimony inference that is general enough to apply to many specific variants and has been adapted often in contemporary approaches to ancestral reconstruction.

Here, I aim to revisit the power of the Sankoff–Rousseau algorithm and explore some modern applications.

3.2 Ancestral Reconstruction by Parsimony

We are interested in the problem of using homologous traits in extant organisms to reconstruct the states of the corresponding phylogenetic character at their ancestors. The corresponding mathematical problem, called here *parsimony labeling*, is introduced in Sect. 3.2.1. The discussed mathematical abstraction searches for an assignment of states to the nodes of a known evolutionary tree which minimizes a penalty imposed on state changes between parents and children. The penalization represents the "surprise value" associated with an assumed state change in the evolution of the studied phylogenetic character; searching for the least surprising evolutionary history is intended to ensure the reconstruction's biological plausibility. The most popular mathematical varieties of parsimony, involving different types of characters and penalties, are reviewed in Sect. 3.2.2. Sankoff and Rousseau's generic algorithm is presented in Sect. 3.2.3, along with its applications in many specific parsimony variants.

3.2.1 Parsimony Labeling

Consider a given *phylogeny* $\Psi = (\mathcal{L}, \mathcal{V}, \mathcal{E})$ over the terminal taxa $\mathcal{L}$, which is a tree with node set $\mathcal{V}$, leaves $\mathcal{L} \subseteq \mathcal{V}$, and edge set $\mathcal{E}$. The tree is rooted at a designated *root node* $\rho \in \mathcal{V}$: for every node $u \in \mathcal{V}$, exactly one path leads from ρ to u. A node $u \in \mathcal{V}$ and all its descendants form the *subtree* rooted at u, denoted by Ψ_u.

Every node $u \in \mathcal{V}$ is associated with a *label* $\xi[u] \in \mathcal{F}$ over some feature *alphabet* $\mathcal{F}$. The labels $\xi[x]$ represent the states of a homologous character at different nodes of the phylogeny. Labels are observed at the terminal nodes, but not at the other nodes, which represent hypothetical ancestors; see Fig. 3.1.

We state the problem of ancestral reconstruction in an optimization setting, where the label space is equipped with a *cost function* $d\colon \mathcal{F} \times \mathcal{F} \mapsto [0, \infty]$. Suppose we are given a fixed leaf labeling $\Phi\colon \mathcal{L} \to \mathcal{F}$. The goal is to extend it to a joint labeling $\xi\colon \mathcal{V} \mapsto \mathcal{F}$ with minimum total cost between parent-edge labels on the edges. In systematics, the labels are states of a phylogenetic character at the nodes. The cost function reflects the evolutionary processes at play. Formally, we are interested in the following optimization problem.

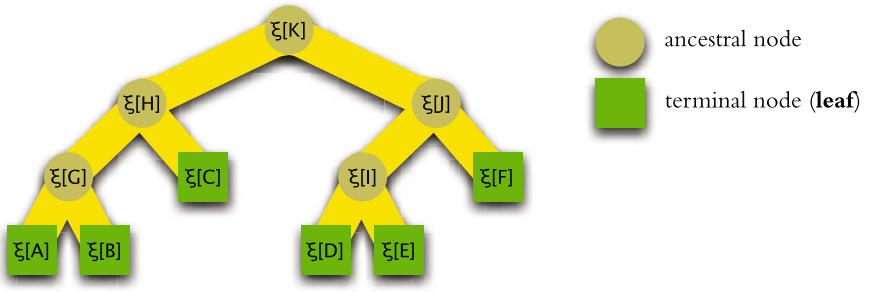


Fig. 3.1 Ancestral inference. Node labels $\xi[u]$ are observed at the leaves and need to be inferred at ancestral nodes

General Parsimony Labeling Problem *Consider a phylogeny* $\Psi = (\mathcal{L}, \mathcal{V}, \mathcal{E})$, *and a label space* $\mathcal{F}$ *equipped with a cost function* $d : \mathcal{F} \times \mathcal{F} \mapsto [0, \infty]$. *Find a joint labeling* $\xi : \mathcal{V} \mapsto \mathcal{F}$ *that extends a given leaf labeling* $\Phi : \mathcal{L} \to \mathcal{F}$ *and minimizes the total change*

$$f^* = \min_{\xi \supset \Phi} f(\xi) = \min_{\xi \supset \Phi} \sum_{uv \in \mathcal{E}} d\big(\xi[u], \xi[v]\big). \tag{3.1}$$

Parsimony labeling belongs to a large class of optimization problems related to Steiner trees [29]. In a Steiner-tree problem, the pair $(\mathcal{F}, d)$ forms a metric space, and the labels describe the placement of tree nodes in that space. Leaves have a fixed placement and need to be connected by a minimal tree through additional inner nodes, or so-called Steiner vertices. Classically, the placement is considered in k-dimensional Euclidean space with $\mathcal{F} = \mathbb{R}^k$, and d is the ordinary Euclidean distance. General parsimony labeling gives the optimal placement of Steiner vertices for a fixed topology.

The algorithmic difficulty of parsimony labeling depends primarily on the assumed cost function d. Finding the most parsimonious tree is NP-hard under all traditional parsimony variants [14–16], but computing the score of a phylogeny is not always difficult.

3.2.2 A Quick Tour of Parsimony Variants

The minimum total change of Eq. (3.1) measures the economy of the assumed phylogeny, or its *parsimony score* $\min_\xi f(\xi)$. Systematicists have been routinely constructing hypothetical phylogenies minimizing the parsimony score over some chosen phylogenetic characters [23]. Provided that the cost function *truly* reflects the economy of the implied evolutionary histories, parsimony is the phylogenetic equivalent of Occam's razor. Common parsimony variants use fairly simple abstractions about evolutionary processes. For instance, Dollo parsimony (Sect. 3.2.2.1) and Fitch parsimony (Sect. 3.2.2.1), use cost functions that penalize every state

change the same way. In the following tour, we briefly discuss classic parsimony variants for directed evolution (Sect. 3.2.2.1), numerical characters (Sect. 3.2.2.2) and molecular sequences (Sect. 3.2.2.3), along with some historical notes.

3.2.2.1 Directed Evolution

The earliest formalizations of parsimony [7, 34] framed phylogenetic inference for situations where evolutionary changes have a known (or assumed) directionality. In *Dollo–* and *Camin–Sokal* parsimony, the directionality constraints yield simple solutions to ancestral labeling.

Camin and Sokal [7] examine phylogenetic characters with some fixed ordering across possible states. The ordering is ensured by cost asymmetry. If $x \prec y$, then $0 \le d(x, y) < \infty$ and $d(y, x) = \infty$. The cost function is additive over successive states: for any three successive labels $x \prec y \prec z$, $d(x, z) = d(x, y) + d(y, z)$. In [7], this type of scoring is introduced for morphological characters—nine characters such as foot anatomy encoded by small integers—to establish a phylogeny for ten horse fossils. Ancestral labeling is straightforward: set $\xi[u]$ to the minimum label seen at leaves in u's subtree.

Dollo's law [34] applies to rare evolutionary gains (e.g., complex morphological structures) that subsequent lineages may lose. The principle translates into a binary labeling problem where only one $0 \to 1$ transition is allowed in the entire phylogeny, and the task is to minimize the number of $1 \to 0$ transitions. As first explained by Farris [20], the optimal labeling is directly determined by the principle. The lowest common ancestor w of leaves u with label $\xi[u] = 1$ is labeled as $\xi[w] = 1$. It is either the root node, or the point of the single gain $0 \to 1$ in the entire phylogeny. Outside w's subtree, all nodes are labeled with 0. Within Ψ_w, every ancestral node v is labeled by the maximum of the labels under it: if all its descendants are labeled with 0, so is $\xi[v] = 0$; otherwise, $\xi[v] = 1$.

3.2.2.2 Numerical Labels

Cavalli-Sforza and Edward [9] pose the problem of inferring a phylogeny of populations from gene allele frequencies. Allele frequencies and many other interesting characters are best captured by numerical labels as real-valued continuous variables. When $\mathcal{F} = \mathbb{R}$, the absolute and the squared distances are common choices for parsimony labeling.

Wagner parsimony [32] applies to a numerical label space (discrete or continuous) and uses the distance $d(x, y) = |x - y|$. Farris [19] describes a linear-time algorithm for labeling a binary tree, which was proven to be correct and adaptable to non-binary trees by Sankoff and Rousseau [47].

Squared parsimony [35] employs the squared distance $d(x, y) = (x - y)^2$ over a continuous label space and gives a linear-time algorithm to compute the ancestral labeling in linear time. Squared parsimony has an attractive probabilistic interpretation involving a Brownian motion model [35]. Suppose that changes

along each edge follow a Brownian motion, so child labels have a normal distribution centered around the parent's label with variance proportional to the edge length. If edge lengths are the same, then the ancestral labeling that maximizes the likelihood also minimizes the parsimony score with the cost function $d'(x, y) = -\log(\frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-y)^2}{2\sigma^2}))$, where σ is the common standard deviation of the child labels. After stripping away the constant term and common scaling factors, only the remaining squared distance $d(x, y) = (x - y)^2$ determines the labeling's optimality.

3.2.2.3 Molecular Sequences

For the purposes of phylogenetic inference and ancestral reconstruction from molecular sequences, parsimony scoring has to capture the peculiarities of homologous sequence evolution. One possibility is to consider a fixed multiple alignment and use parsimony with residues at aligned sites. Fitch parsimony [24], which applies to a finite label space such as the four-letter DNA alphabet, simply minimizes the number of different labels on tree edges. Much more challenging is parsimony with edit distance, first addressed by David Sankoff [43, 46], when the label space encompasses all possible sequences, and the scoring includes insertions and deletions. Parsimony labeling with edit distance is NP-hard, since it is equivalent to multiple alignment with a fixed guide tree, which is known to be NP-hard for any alignment scoring [18].

Originally, Kluge and Farris [32] employed Wagner parsimony to six binary characters derived from distinguishing body features in 12 frog families. When labels are binary ($\mathcal{F} = \{0, 1\}$) in Wagner parsimony, there are only two possible distances: $d(x, y) = 0$ if $x = y$ and $d(x, y) = 1$ if not. *Fitch parsimony* [24] generalizes the same scoring principle to any discrete alphabet:

$$d(x, y) = \{x \neq y\} = \begin{cases} 0 & \text{if } x = y; \\ 1 & \text{if } x \neq y \end{cases}$$

Computing the optimal labeling under this distance takes linear time in the tree size [24, 28]. Fitch parsimony, in contrast to Wagner parsimony, accommodates non-numerical phylogenetic characters, including amino acids and nucleotides. In an early application, Fitch and Farris [25] apply the scoring method to nucleotides in homologous positions in a multiple alignment in order to infer the most parsimonious RNA coding sequences from protein data.

3.2.3 The Sankoff–Rousseau Technique

The crucial insight of [47] is that general parsimony labeling has a recursive structure that calls for solutions by dynamic programming.

For every node $u \in \mathcal{V}$, define the *subtree cost* $f_u : \mathcal{F} \mapsto [0, \infty)$ as the minimum total change implied by u's label within its subtree:

$$f_u(x) = \min \sum_{vw \in \mathcal{E}_u ; \xi[u]=x} d\big(\xi[v], \xi[w]\big),$$

where $\mathcal{E}_u$ are the edges within the subtree Ψ_u. The minimum is taken across all ancestral node labelings with the same assignment x at u. By the principle of optimality, the following recursions hold:

$$f_u(x) = \begin{cases} 0 & \text{if } x = \Phi[u]; \\ \infty & \text{if } x \neq \Phi[u]; \end{cases} \qquad \{u \in \mathcal{L}\} \tag{3.2a}$$

$$f_u(x) = \sum_{uv \in \mathcal{E}} \min_{y \in \mathcal{F}} \big(d(x, y) + f_v(y)\big) \qquad \{u \in \mathcal{V} \setminus \mathcal{L}\} \tag{3.2b}$$

In particular, the parsimony score is retrieved by considering the minimum total penalty at different root labelings:

$$f^* = \min_{x \in \mathcal{F}} f_\rho(x).$$

The recursions of Eqs. (3.2a), (3.2b) suggest a general outline for computing the best ancestral labeling together with its score. First, compute all f_u in a postfix traversal, visiting parents after child nodes, as shown by the procedure ANCESTRAL below. Second, retrieve the best labeling in a prefix traversal that realizes the computed minimum f^* by standard backtracking, as shown by the procedure LABELING here.

```
   ANCESTRAL(u)                                          // (computes f_u for a node u)
A1 if u ∈ L then f_u(x) ← {x = Φ[u]}?0 : ∞                // (leaf)
A2 else
A3    for uv ∈ E do                                      // (for all children v)
A4       f_v ← ANCESTRAL(v)
A5       compute h_uv(x) ← min_{y∈F}(d(x, y) + f_v(y))
A6    set f_u(x) ← Σ_{uv∈E} h_uv(x)
A7 return f_u
```

(Line A1 uses $\{x = \Phi[u]\}?0 : \infty$ to denote the function for a forced labeling at a terminal node u, from Eq. (3.2a).) Line A5 computes the *stem cost* $h_{uv}(x)$ for a child of u, which is the minimal change along the edge and within the subtree of v, given that u is labeled with x. Line A6 sums the stem cost functions to construct f_u.

```
   LABELING(v)                                           // (computes the best labeling)
L1 if v is the root then ξ[v] = arg min_x f_v(x)
L2 else
L3    u ←parent of v; x ← ξ[u]
L4    ξ[v] ← arg min_{y∈F}(d(x, y) + f_v(y))
L5 for vw ∈ E do LABELING(w)                             // (for all children of w, if any)
```

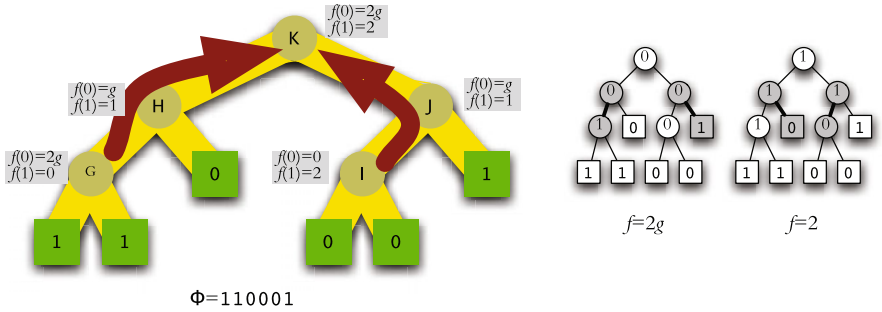


Fig. 3.2 Inference of ancestral labels by parsimony. This example uses a binary character (presence-absence) with profile Φ, a loss penalty 1 and gain penalty $g \geq 0$; i.e., the distance function over the feature space $\mathcal{F} = \{0, 1\}$ is defined as $d(x, x) = 0$ and $d(0, 1) = g$, $d(1, 0) = 1$. Depending on the gain penalty g, the optimal solution may imply two losses (score $f = 2$) or two gains ($f = 2g$). The dynamic programming proceeds from the leaves towards the root, computing the score $f_u(x)$ of the optimal reconstruction within each subtree Ψ_u, in the order indicated by the *arrows*

For a finite label space, the minimum in Line A5 is found by examining all possible values. At the same time, the best labeling y for each x is saved for backtracking in Lines L1 and L4. For an infinite space or very large label space, it is not immediately clear how the minimization can be done in practice. Luckily, it is possible to track f and h by other means than tabulation in many important cases.

3.2.3.1 Few Possible Labels

The general Sankoff–Rousseau outline immediately yields an algorithm when labels have only a few possible values. Figure 3.2 shows an example with absence-presence labels ($\mathcal{F} = \{0, 1\}$). The distance metric is defined by the gain penalty g and loss penalty 1, which apply on every edge uv to labelings $\xi[u] = 0$, $\xi[v] = 1$, and $\xi[u] = 1$, $\xi[v] = 0$, respectively.

A computer implementation can tabulate $f_u(x)$ for all nodes u and possible values x. With a table-based implementation, the running time grows linearly with tree size, but quadratically with possible labels. Note that the tables accommodate arbitrary cost functions, not only true distance metrics.

Theorem 1 *For a finite label space of size $r = |\mathcal{F}|$, and an evolutionary tree with m edges, algorithms* ANCESTRAL *and* LABELING *compute an optimal labeling in $O(mr^2)$ time and $O(mr)$ space.*

Proof A table stores $f_u(x)$ for $m + 1$ nodes and r possible labels. Line A1 is executed once for every terminal node. In Line A5, $\min_{y \in \mathcal{F}}$ is found in a loop over possible child labelings y in $O(r)$ time. Lines A5 and A6 are executed for each edge and label x, in $O(mr^2)$ total time.

In order to use with backtracking, the minimal y found in Line A5 is saved for every edge uv and label x, using $m \times r$ entries in a table. Line L4 takes $O(1)$ to retrieve the optimal labels on each edge, and Algorithm LABELING completes in $O(m)$ time. □

3.2.3.2 Molecular Sequences

Sankoff and Rousseau [47] generalize Sankoff's previously developed method for the ancestral reconstruction off-RNA sequences [43, 46], which is by far the most challenging case of ancestral parsimony. The recursions for multiple alignment and ancestral labeling can be combined to find an optimal solution, but the computation takes an exponentially long time in the number of nodes [43].

3.2.3.3 Squared Parsimony

Squared parsimony was first proposed [9, 42] as an appropriate cost function $d(x, y) = (x - y)^2$ for inference from allele frequencies $\xi \in [0, 1]$ observed in populations. Maddison [35] solved the parsimony labeling problem by directly employing the method of Sankoff and Rousseau [47]. Theorem 2 below restates the key result: subtree and stem costs are quadratic functions, for which the parameters can be computed recursively.

Theorem 2 *In the general parsimony problem with $\mathcal{F} = \mathbb{R}$ and $d(x, y) = (y - x)^2$, the subtree weight functions are quadratic. In other words, one can write the subtree cost function at each non-leaf node u as*

$$f_u(x) = \alpha_u (x - \mu_u)^2 + \phi_u, \tag{3.3}$$

with some parameters $\alpha_u, \phi_u, \mu_u \in \mathbb{R}$. The parameters α, μ satisfy the following recursions:

$$\alpha_u = \begin{cases} \textit{undefined} & \textit{if } u \textit{ is a leaf}; \\ \sum_{uv \in \mathcal{E}} \beta_v & \textit{otherwise}; \end{cases} \tag{3.4a}$$

$$\mu_u = \begin{cases} \xi[u] & \textit{if } u \textit{ is a leaf}; \\ \frac{\sum_{uv \in \mathcal{E}} \beta_v \mu_v}{\sum_{uv \in \mathcal{E}} \beta_v} & \textit{otherwise}; \end{cases} \tag{3.4b}$$

where β_v is defined for all $v \in \mathcal{V}$ by

$$\beta_v = \begin{cases} 1 & \textit{if } v \textit{ is a leaf}; \\ \frac{\alpha_v}{\alpha_v + 1} & \textit{otherwise}. \end{cases} \tag{3.4c}$$

By Theorem 2, ANCESTRAL can proceed by storing α and μ at every node.

Theorem 3 *For squared parsimony, algorithms* ANCESTRAL *and* LABELING *compute an optimal labeling in $O(m)$ time and $O(m)$ space.*

Proof Lines A5–A6 compute the subtree costs by the recursions of (3.4a)–(3.4c) in $O(m)$ total time. By Eq. (3.3), LABELING needs to set the root label in Line L1 as

$$\xi[\rho] = \arg\min_x \alpha_\rho (x - \mu_\rho)^2 + \phi_\rho = \mu_\rho \tag{3.5}$$

since $\alpha_u > 0$ at all nodes. The stem weight function $h_{uv}(x)$ for an inner node v is

$$h_{uv}(x) = \min_y \big((x-y)^2 + f_v(y)\big) = \frac{\alpha_v}{\alpha_v + 1}(x - \mu_v)^2 + \phi_v,$$

with minimum at

$$y^* = \arg\min_y \big((x-y)^2 + f_v(y)\big) = \frac{x + \alpha_v \mu_v}{\alpha_v + 1}. \tag{3.6}$$

Therefore, Line L4 sets the child labeling as

$$\xi[v] = \frac{\xi[u] + \alpha_v \mu_v}{\alpha_v + 1}.$$

LABELING thus spends $O(1)$ time on each node and finishes in $O(m)$ time. □

Squared parsimony can be generalized to k-dimensional features $\mathcal{F} = \mathbb{R}^k$. The optimal parsimony labeling with the squared Euclidean distance between labels $d(x, y) = \sum_{i=1}^k = (x_i - y_i)^2$ can be computed component-wise, since plugging it into (3.1) gives

$$\min_{\xi \supset \Phi} f(\xi) = \sum_{i=1}^{k} \underbrace{\min_{\xi_i \supset \Phi_i} \sum_{uv \in \mathcal{E}} \big(\xi_i[u] - \xi_i[v]\big)^2}_{\text{best labeling in coordinate } i}. \tag{3.7}$$

It suffices to apply the recursions of (3.4a)–(3.4c) in each coordinate separately. In an application where each label represents a distribution over k elements, it is not immediately clear that the coordinate-wise reconstruction yields valid ancestral distributions, i.e., that $\sum_{i=1}^k \xi_i[u] = 1$ is ensured everywhere. Fortunately, the optimal labeling formulas of (3.5) and (3.6) automatically ensure that the separate reconstructions always add up to proper distributions [12].

3.2.3.4 Wagner (Linear) Parsimony

Wagner parsimony considers the linear cost function $d(x, y) = |x - y|$ over a numerical label space like $\mathcal{F} = \mathbb{R}$. A consequence of the linear cost is that the subtree costs have a simple recursive structure [19, 47, 49], and the Sankoff–Rousseau

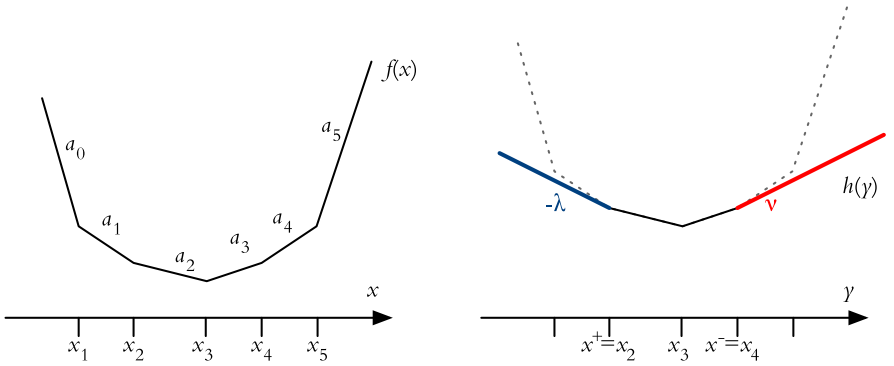


Fig. 3.3 Illustration of Theorem 4 about the shape of the cost functions. *Left*: for asymmetric Wagner parsimony, the subtree cost function f is always piecewise linear with slopes $a_0, \ldots, a_k$ ($k = 5$ here). *Right*: the stem cost function $h(y) = \min_x (d(y, x) + f(x))$ is obtained by "shaving off" the steep extremities of f and replacing them with slopes of $(-\lambda)$ and ν, respectively

method can be carried out by tracking simple intervals. An asymmetric linear cost function, examined by [12], leads to a similarly recursive structure. Namely, *asymmetric Wagner parsimony* uses a linear cost function of the form

$$d(x, y) = \begin{cases} \lambda(y - x) & \{y \geq x\} \\ \nu(x - y) & \{x > y\}, \end{cases}$$

with gain and loss penalties λ, ν. In asymmetric Wagner parsimony, the subtree cost functions are continuous, convex and piecewise linear. Consequently, they can be manipulated symbolically as vectors of slopes and breakpoints, see Fig. 3.3.

Theorem 4 *For every non-leaf node $u \in \mathcal{V} \setminus \mathcal{L}$, there exist $k \geq 1$, $\alpha_0 < \alpha_1 < \cdots < \alpha_k$ (slopes), $x_1 < x_2 < \cdots < x_k$ (breakpoints), and $\phi_0, \ldots, \phi_k \in \mathbb{R}$ that define f_u in the following manner*:

$$f_u(x) = \begin{cases} \phi_0 + \alpha_0 x & \text{if } x \leq x_1; \\ \phi_1 + \alpha_1 (x - x_1) & \text{if } x_1 < x \leq x_2; \\ \ldots & \\ \phi_{k-1} + \alpha_{k-1}(x - x_{k-1}) & \text{if } x_{k-1} < x \leq x_k; \\ \phi_k + \alpha_k (x - x_k) & \text{if } x_k < x, \end{cases} \tag{3.8}$$

where $\phi_1 = \phi_0 + \alpha_0 x_1$ and $\phi_{i+1} = \phi_i + \alpha_i (x_{i+1} - x_i)$ for all $0 < i < k$. Moreover, if u has d children, then $a_0 = -d\lambda$ and $a_k = d\nu$.

Figure 3.3 illustrates the proof of Theorem 4 from [12]. The Sankoff–Rousseau algorithm can be implemented by storing the breakpoints for the slopes between $(-\lambda)$ and ν. In classical Wagner parsimony, $\lambda = \nu = 1$, and the two stored

breakpoints define an interval of equivalently optimal labelings at every node, used in the original algorithm of [19].

Theorem 5 *Algorithms* ANCESTRAL *and* LABELING *find the optimal labeling by asymmetric Wagner parsimony in* $O(nh \log d_{\max})$ *time for a phylogeny of height* h *with* n *nodes and maximum node degree* $d_{\max}$. *For integer-valued penalties* λ, ν *with* $B = \lambda + \nu$, *the algorithms label the tree in* $O(nB)$ *time.*

Proof For general real-valued penalties λ and ν, the breakpoints and the slopes defining the subtree cost functions are stored by ordered lists. The symbolic summation of stem costs in Line A6 involves merging ordered lists, leading to a running-time bound of $O(nh \log d_{\max})$ [12].

For integer-valued penalties, it is enough to store the $B = (\lambda + \nu)$ possible breakpoints associated with slopes between $-\lambda$ and ν that can play a role in an optimal labeling. By storing the breakpoints for f_u in an array of length B, Line A6 sums the stem costs in $O(nB)$ time across all nodes, and an optimal labeling is found in $O(1)$ time per node. □

Wagner parsimony easily generalizes to the k-dimensional labels $\mathcal{F} = \mathbb{R}^k$ with the Manhattan distance $d(x, y) = \sum_{i=1}^{k} |x_i - y_i|$. Just like with squared parsimony, the optimal labeling can be decomposed by coordinates.

3.2.3.5 Multiple Reconstructions

The optimal ancestral labeling for Camin–Sokal and Dollo is always unique, due to the directionality of the parsimony cost function. Squared parsimony, as well, has a unique optimal solution by Theorem 2. Otherwise, the most parsimonious labeling of Eq. (3.1) is not necessarily unique. For example, the two ancestral labelings depicted in Fig. 3.2 are both minimal when gains and losses are penalized equally ($g = 1$): they entail either two loss events or two gain events. Even if multiple solutions are possible, the ancestral labeling algorithm can resolve the ties at will between ancestral labels that yield the same minimum subtree costs either at the root (Line L1 in LABELING) or on the edges (Line L4), following the normal order of the algorithm.

Theorem 4 shows an important property of Wagner parsimony, first recognized by Farris [19]. Namely, the minimum subtree score for an ancestral node is attained either at a single label, or within a closed interval where the cost function has slope 0. The ambiguity of optimal ancestral labelings can be characterized by computing the set of possible labels (a closed interval, possibly containing a single point) at each ancestral node in linear time [49]. When multiple ancestral labels are equally optimal, one of two heuristics are traditionally chosen to resolve ties. The first one, proposed in [19] and named ACCTRAN (for "accelerated transformation") by [49], chooses a reconstruction where label changes are placed closer to the root.

Mathematically, ACCTRAN is the unique optimal reconstruction in which all subtree scores are minimized; i.e., $\sum_{vw \in T_u} d(\xi[v], \xi[w])$ takes its minimum value in each subtree T_u among all reconstructions ξ with minimum parsimony score [39]. The other heuristic, called DELTRAN ("delayed transformation"), defers changes away from the root [49]. ACCTRAN is believed to give biologically more plausible reconstructions by minimizing parallel gains in different lineages, although closer scrutiny shows that ACCTRAN and DELTRAN do not always behave as expected [1].

3.3 Applications and Extensions

Many authors built on the Sankoff–Rousseau technique to develop related efficient algorithms. We sample a few algorithmic extensions in Sect. 3.3.1. A few biological applications in Sect. 3.3.2 illustrate the pertinence of parsimony-based reconstructions in contemporary studies of genome evolution.

3.3.1 Algorithmic Extensions

Tree-Additive Cost Functions A somewhat inconvenient property of the generic Sankoff–Rousseau algorithm is that it entails a quadratic dependence on the alphabet size (Theorem 1). Its simplicity, however, lends itself to efficient parallel implementation [31]. The quadratic factor can be avoided for certain cost functions with an additive structure, e.g., if $d(x, y)$ is an ultrametric distance [10].

Parsimony on a Phylogenetic Network Kannan and Wheeler [30] address the generalization of the Sankoff–Rousseau algorithm to a phylogenetic network. Specifically, they consider networks with some reticulate nodes having two incoming and one outgoing edges. The parsimony score sums the costs occurred along all the edges, including those connecting reticulate vertices. The optimal labeling can be computed by enumerating all possible joint labelings at reticulate nodes, in $O(mr^{k+2})$ time for m edges and k reticulate nodes over a r-letter alphabet.

Gain and Loss Edges After constructing an optimal ancestral labeling, it is trivial to collect the lineages with similar state transitions such as the edges on which some feature was lost (transition $1 \to 0$). In a binary labeling problem for absence-presence data, it is practicable to track such sets of edges by the single traversal of ANCESTOR [37]. Specifically, define the sets $L_{uv}(x)$ and $G_{uv}(x)$ for all edges uv as sets of loss and gain edges, respectively, affecting v's subtree when $\xi[u] = x$ in an optimal labeling ξ. For a terminal edge uv, $L_{uv}(0) = G_{uv}(1) = \emptyset$, $L_{uv}(1) = \{uv\}$ if $\xi[v] = 0$, and $G_{uv}(0) = \{uv\}$ if $\xi[v] = 1$. For all non-leaf nodes u, let $L_{u*}(x) =$

$\bigcup_{uv\in\mathcal{E}} L_{uv}(x)$ and $G_{u*}(x) = \bigcup_{uv\in\mathcal{E}} G_{uv}(x)$. The following recursions hold, depending on the event on the edge uv:

$$\big\langle L_{uv}(1), G_{uv}(1)\big\rangle = \begin{cases} \langle \{uv\} \cup L_{v*}(0), G_{v*}(0)\rangle & (\textit{loss on } uv) \\ \langle L_{v*}(1), G_{v*}(1)\rangle & (\textit{no loss on } uv) \end{cases}$$

$$\big\langle L_{uv}(0), G_{uv}(0)\big\rangle = \begin{cases} \langle L_{v*}(1), \{uv\} \cup G_{v*}(1)\rangle & (\textit{gain on } uv) \\ \langle L_{v*}(0), G_{v*}(0)\rangle & (\textit{no gain on } uv) \end{cases}$$

The stem cost for the edge h_{uv} counts the edges within the L_{uv} and G_{uv} sets. Using asymmetric gain–loss costs as in Sect. 3.2.3.4, $h_{uv}(x) = \lambda|G_{uv}(x)| + \nu|L_{uv}(x)|$. The choices between loss vs. no-loss and gain vs. no-gain are made to minimize the associated costs.

3.3.2 Applications

Parsimony's simple assumptions are appreciated even in contemporary studies of complex genome features. A case in point is Wagner parsimony that was recently used to study genome size evolution [6] and short sequence length polymorphisms [51]. Genome size and tandem repeat copy numbers as well as the other examples to follow are common in that they are difficult to address in probabilistic models, either for technical reasons or simply because the relevant evolutionary processes are still not understood well enough.

Phylogenetic Footprinting In phylogenetic footprinting, conserved short sequence motifs are discovered in a sample of unaligned sequences associated with the terminal nodes of a known phylogeny [5]. It is assumed that the sequences contain common regulatory signals with some level of conservation. The corresponding ancestral reconstruction problem (Substring Parsimony) labels the nodes with k-letter sequences $\mathcal{F} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k$ for a fixed k. Leaves must be labeled by a word that appears somewhere in the input sequence, and edge costs measure distance between parent and child labels. The algorithm of Sankoff and Rousseau can be readily modified to initialize a set of possible labels at the leaves with 0 cost. [5] propose practical algorithmic improvements for small k, but Substring Parsimony is NP-hard in general [18].

Gene Family Evolution A number of software packages implement asymmetric Wagner parsimony for the inference of ancestral gene family sizes [2, 8, 11]. Weighted parsimony has been used to study the frequency of gene loss and gain, and to estimate ancestral genome sizes [36, 37]. Pioneering large-scale studies on the phylogenetic distribution of gene families revealed a surprisingly gene-rich last universal common ancestor by ancestral reconstruction [33]. Genes tend to be lost

more often than gained in the course of evolution, and asymmetric gain-loss penalties can capture known discrepancies between the intensities of the two processes. Selecting the relative costs between loss and gain entails additional considerations such as the plausibility of the reconstructed ancestral genome size [33].

Han and Hahn [27] use k-dimensional linear parsimony to study gene duplications and losses concomitantly with transpositions between chromosomes. Homolog genes for a family are encoded in a k-dimensional integer vector by the copy numbers on k chromosomes. Homolog families over ten complete Drosophila reveal patterns of sex-specific gene movement to and from the X chromosome, overly active functional categories, and other idiosyncrasies that characterize fly genome evolution.

Splice Sites and Intron Length Gelfman and coauthors [26] resort to squared parsimony to infer ancestral intron length from homologous introns in 17 vertebrate genomes. The study links length constraints to splicing signal strength (the stronger the signal, the longer the intron can be) and shows that the correlations specifically pertain to vertebrates. In a related study, Schwartz et al. [48] infer ancestral splice site signals. Starting with aligned 5' splice sites and branch sites in different introns, the nucleotide frequencies in each motif position are compiled into probabilistic sequence motifs that label the leaf genomes. Ancestral nucleotide frequencies are reconstructed separately in each motif position by squared parsimony. The reconstruction implies that sites were degenerate in the earliest eukaryotes, hinting at the prevalence of alternative splicing in deep eukaryotic ancestors.

Gene Order Ancestral reconstruction of gene order was pioneered as a parsimony problem by Sankoff et al. [45]. In this context, nodes are labeled by gene orders, which are the permutations defined by the physical order of genes (or other genetic markers) along the chromosomes. Appropriate cost functions for such permutations can be defined using an edit distance penalizing various rearrangement events, or by counting conserved segments and breakpoints. Other contributions in this volume address the mathematically rich field of gene order comparisons in more detail (in particular, [38] discusses distances between gene orders): here, we mention only some recent connections to classic parsimony variants.

Wang and Tang [50] explored an encoding of adjacencies that are suitable to submit as phylogenetic characters to parsimony labeling. The reconstructions need to be corrected to yield valid gene orders (the inferred ancestral adjacencies may imply a circular chromosome)—the correction is shown to be NP-hard. Feijão and Meidanis [21] give an edit distance function for which parsimony labeling is feasible in polynomial time. They show in particular that by simply using adjacencies as binary phylogenetic characters, and applying Fitch parsimony (with some small algorithmic adjustments), one recovers a most parsimonious gene order history under the so-called single-cut-and-join distance. Bérard et al. [3] also use parsimony labeling for inferring ancestral adjacencies; the novelty of their approach is that it incorporates gene duplications and losses by carrying out Sankoff's dynamic programming method simultaneously along two gene phylogenies reconciled with a known species tree.

3.4 Conclusion

Parsimony is not as popular as it once was, mostly because today's large data sets contain enough statistical signal to employ sophisticated probabilistic models. Nevertheless, parsimony remains a viable choice in many contemporary applications, where parameter-rich stochastic models are not available or are impractical. Indeed, different scoring policies are available for the evolutionary analysis of "unconventional" genome features, including sequence motifs, copy numbers, genomic lengths and distributions. Surprisingly diverse policies are amenable to exact optimization by dynamic programming following the same basic recipe. Along with Felsenstein's seminal work on likelihood calculations [22], Sankoff's parsimony minimization [47] established fundamental algorithmic techniques for modeling evolutionary changes, which proved to be versatile enough to tackle new computational biology challenges for the past 40 years.

References

1. Agnarsson, I., Miller, J.A.: Is ACCTRAN better than DELTRAN? Cladistics **24**, 1–7 (2008)
2. Ames, R.M., Money, D., Ghatge, V.P., Whelan, S., Lovell, S.C.: Determining the evolutionary history of gene families. Bioinformatics **28**(1), 48–55 (2012)
3. Bérard, S., Gallien, C., Boussau, B., Szöllősi, G.J., Daubin, V., Tannier, E.: Evolution of gene neighborhoods within reconciled phylogenies. Bioinformatics **28**, i382–i388 (2012)
4. Blanchette, M., Green, E.D., Miller, W., Haussler, D.: Reconstructing large regions of an ancestral mammalian genome in silico. Genome Res. **12**, 2412–2423 (2004)
5. Blanchette, M., Schwikowski, B., Tompa, M.: Algorithms for phylogenetic footprinting. J. Comput. Biol. **9**(2), 211–223 (2002)
6. Caetano-Anollés, G.: Evolution of genome size in the grasses. Crop Sci. **45**, 1809–1816 (2005)
7. Camin, J.H., Sokal, R.R.: A method for deducing branching sequences in phylogeny. Evolution **19**, 311–326 (1965)
8. Capra, J.A., Williams, A.G., Pollard, K.S.: Proteinhistorian: tools for the comparative analysis of eukaryote protein origin. PLoS Comput. Biol. **8**(6), e1002567 (2011)
9. Cavalli-Sforza, L.L., Edwards, A.W.H.: Phylogenetic analysis models and estimation procedures. Am. J. Hum. Genet. **19**(3), 233–267 (1967)
10. Clemente, J.C., Ikeo, K., Valiente, G., Gojobori, T.: Optimized ancestral state reconstruction using Sankoff parsimony. BMC Bioinform. **10**, 51 (2009)
11. Csűrös, M.: Count: evolutionary analysis of phylogenetic profiles with parsimony and likelihood. Bioinformatics **26**(15), 1910–1912 (2010)
12. Csűrös, M.: Ancestral reconstruction by asymmetric Wagner parsimony over continuous characters and squared parsimony over distributions. In: Proc. Sixth RECOMB Comparative Genomics Satellite Workshop. Springer Lecture Notes in Bioinformatics, vol. 5267, pp. 72–86 (2008)
13. Darwin, C.: On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life. John Murray, London (1859)
14. Day, W.H.E.: Computationally difficult parsimony problems in phylogenetic systematics. J. Theor. Biol. **103**, 429–438 (1983)
15. Day, W.H.E., Johnson, D.S., Sankoff, D.: The computational complexity of inferring rooted phylogenies by parsimony. Math. Biosci. **81**, 33–42 (1986)

16. Day, W.H.E., Sankoff, D.: Computational complexity of inferring phylogenies by compatibility. Syst. Zool. **35**, 224–229 (1986)
17. Edwards, A.W.F., Cavalli-Sforza, L.L.: Reconstructing evolutionary trees. In: Heywood, V.H., McNeill, J. (eds.) Phenetic and Phylogenetic Classification, vol. 6, pp. 67–76. Systematics Association, London (1963)
18. Elias, I.: Settling the intractability of multiple alignment. J. Comput. Biol. **13**(7), 1323–1339 (2006)
19. Farris, J.S.: Methods for computing Wagner trees. Syst. Zool. **19**(1), 83–92 (1970)
20. Farris, J.S.: Phylogenetic analysis under Dollo's law. Syst. Zool. **26**(1), 77–88 (1977)
21. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–1329 (2011)
22. Felsenstein, J.: Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. Syst. Zool. **22**(3), 240–249 (1973)
23. Felsenstein, J.: Parsomony in systematics: biological and statistical issues. Annu. Rev. Ecol. Syst. **14**, 313–333 (1983)
24. Fitch, W.M.: Toward defining the course of evolution: minimum changes for a specific tree topology. Syst. Zool. **20**, 406–416 (1971)
25. Fitch, W.M., Farris, J.S.: Evolutionary trees with minimum nucleotide replacements from amino acid sequences. J. Mol. Evol. **3**(4), 263–278 (1974)
26. Gelfman, S., Burstein, D., Penn, O., Savchenko, A., Amit, M., Schwartz, S., Pupko, T., Ast, G.: Changes in exon–intron structure during vertebrate evolution affect the splicing pattern of exons. Genome Res. **22**(1), 35–50 (2012)
27. Han, M.V., Hahn, M.W.: Inferring the history of interchromosomal gene transposition in Drosophila using n-dimensional parsimony. Genetics **190**, 813–825 (2012)
28. Hartigan, J.: Minimum mutation fits to a given tree. Biometrics **29**, 53–65 (1973)
29. Hwang, F.K., Richards, D.S.: Steiner tree problems. Networks **22**, 55–89 (1992)
30. Kannan, L., Wheeler, W.C.: Maximum parsimony on phylogenetic networks. Algorithms Mol. Biol. **7**, 9 (2012)
31. Kasap, S., Benkrid, K.: High performance phylogenetic analysis with maximum parsimony on reconfigurable hardware. IEEE Trans. VLSI Syst. **19**(5) (2011)
32. Kluge, A.R., Farris, J.S.: Quantitative phyletics and the evolution of anurans. Syst. Zool. **18**, 1–32 (1969)
33. Koonin, E.V.: Comparative genomics, minimal gene sets and the last universal common ancestor. Nat. Rev. Microbiol. **1**, 127–136 (2003)
34. Le Quesne, W.J.: The uniquely evolved character concept and its cladistic application. Syst. Zool. **23**, 513–517 (1974)
35. Maddison, W.P.: Squared-change parsimony reconstructions of ancestral states for continuous-valued characters on a phylogenetic tree. Syst. Zool. **40**(3), 304–314 (1991)
36. Makarova, K.S., Sorokin, A.V., Novichkov, P.S., Wolf, Y.I., Koonin, E.V.: Clusters of orthologous genes for 41 archaeal genomes and implications for evolutionary genomics of archaea. Biol. Direct **2**, 33 (2007)
37. Mirkin, B.G., Fenner, T.I., Galperin, M.Y., Koonin, E.V.: Algorithms for computing evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes. BMC Evol. Biol. **3**, 2 (2003)
38. Moret, B.M.E., Lin, Y., Tang, J.: Rearrangements in phylogenetic inference: compare or encode? In: Chauve, C. et al. (eds.) Models and Algorithms for Genome Evolution. Computational Biology, vol. 19. Springer, Berlin (2014). In this volume
39. Narushima, H., Misheva, N.: On characteristics of ancestral-state reconstructions under the accelerated transformation optimization. Discrete Appl. Math. **122**, 195–209 (2002)
40. Needleman, S.B., Wunsch, C.B.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**, 443–453 (1970)
41. Pauling, L., Zuckerkandl, E.: Chemical paleogenetics: molecular "restoration studies" of extinct forms of life. Acta Chem. Scand. **17**, 9–16 (1963)
42. Rogers, J.S.: Deriving phylogenetic trees from allele frequencies. Syst. Zool., **52–63** (1984)

43. Sankoff, D.: Minimal mutation trees of sequences. SIAM J. Appl. Math. **28**(1) (1975)
44. Sankoff, D.: The early introduction of dynamic programming into computational biology. Bioinformatics **16**(1), 41–47 (2000)
45. 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**, 6575–6579 (1992)
46. Sankoff, D., Morel, C., Cedergren, R.J.: Evolution of 5S RNA and the non-randomness of base replacement. Nat., New Biol. **245**, 232–234 (1973)
47. Sankoff, D., Rousseau, P.: Locating the vertices of a Steiner tree in arbitrary metric space. Math. Program. **9**, 240–246 (1975)
48. Schwartz, S., Silva, J., Burstein, D., Pupko, T., Eyras, E., Ast, G.: Large-scale comparative analysis of splicing signals and their corresponding splicing factors in eukaryotes. Genome Res. **18**, 88–103 (2008)
49. Swofford, D.L., Maddison, W.P.: Reconstructing ancestral states using Wagner parsimony. Math. Biosci. **87**, 199–229 (1987)
50. Tang, J., Wang, L.-S.: Improving genome rearrangement phylogeny using sequence-style parsimony. In: Proceedings of the IEEE Fifth Symposium on Bioinformatics and Bioengineering (BIBE'05), pp. 137–144 (2005)
51. Witmer, P.D., Doheny, K.F., Adams, M.K., Boehm, C.D., Dizon, J.S., Goldstein, J.L., Templeton, T.M., Wheaton, A.M., Dong, P.N., Pugh, E.W., Nussbaum, R.L., Hunter, K., Kelmenson, J.A., Rowe, L.B., Brownstein, M.J.: The development of a highly informative mouse simple sequence length polymorphism (SSLP) marker set and construction of a mouse family tree using parsimony analysis. Genome Res. **13**, 485–491 (2003)

Chapter 4
Duplication, Rearrangement and Reconciliation: A Follow-Up 13 Years Later

Cedric Chauve, Nadia El-Mabrouk, Laurent Guéguen, Magali Semeria, and Eric Tannier

Abstract The evolution of genomes can be studied at least three different scales: the nucleotide level, accounting for substitutions and indels, the gene level, accounting for gains and losses, and the genome level, accounting for rearrangements of chromosome organization. While the nucleotide and gene levels are now often integrated in a single model using reconciled gene trees, very little work integrates the genome level as well, and considers gene trees and gene orders simultaneously. In a seminal book chapter published in 2000 and entitled "Duplication, Rearrangement and Reconciliation", Sankoff and El-Mabrouk outlined a general approach, making a step in that direction. This avenue has been poorly exploited by the community for over ten years, but recent developments allow the design of integrated methods where phylogeny informs the study of synteny and vice versa. We review these developments and show how this influence of synteny on gene tree construction can be implemented.

4.1 Introduction

Genomes evolve through a wide variety of mechanisms, not all of them well understood, or even known to us. These mechanisms range from small-scale events, such as point mutations or small insertions or deletions at the nucleotide level, to large-

C. Chauve
LaBRI, Université Bordeaux I, Talence, France

C. Chauve (✉)
Department of Mathematics, Simon Fraser University, Burnaby BC, Canada
e-mail: cedric.chauve@sfu.ca

N. El-Mabrouk
DIRO, Université de Montréal, Montréal QC, Canada

L. Guéguen · M. Semeria · E. Tannier
LBBE, Université Lyon I Claude Bernard, Lyon, France

E. Tannier
INRIA, Rhône-Alpes, France

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_4,

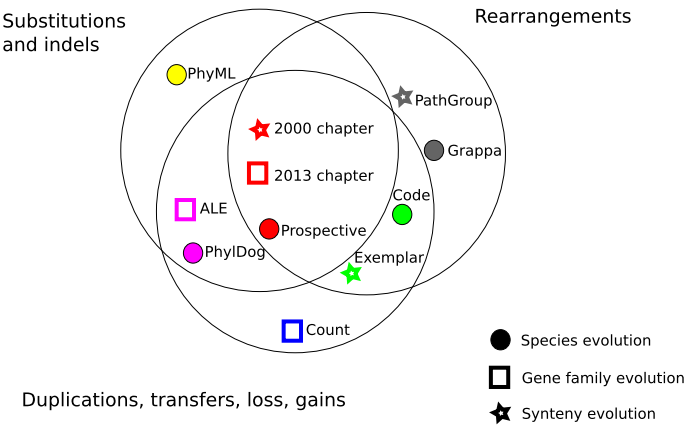


Fig. 4.1 Each of the three big sets represents one of the three kinds of mutations we are dealing with. *Dots*, *squares* and *stars* are models of genome evolution handling these kinds of mutations, respectively aimed at reconstructing phylogenies, gene content evolution and synteny evolution. If they lie in a set intersection, they integrate several kinds of mutations. Apart from the *red area*, the names aside the *dots*, *squares* and *stars* are examples of softwares or methods achieving the described goal (PhyML [2], Count [3], ODT [4], PhylDog [5], Exemplar [6], Pathgroup [7], Grappa [8], Code [9]). They are often chosen among a lot of other examples which would have been as relevant. The *red area* is the core of our chapter: the star refers to the 2000 Sankoff and El-Mabrouk book chapter we are celebrating, the *square* is achieved in the present chapter, and the *dot* is the still open problem toward which all integrative methods tend

scale cataclysmic events such as whole-genome duplications, through segmental duplications or deletions, inversions, transpositions, insertion of mobile elements, translocations, and chromosomes fusions and fissions [1]. While genome evolution is a joint process that combines all such mechanisms, evolutionary studies through computational and statistical methods are generally compartmentalized, as most of them focus on one or few kinds of evolutionary events. Nucleotide level mutations, inferred from alignments, are those considered by phylogenetic methods for gene and species tree constructions. Duplications and other *content-modifying operations* (gains, losses, transfers, ...) are considered for the inference of evolutionary histories of gene families. Inversions, transpositions, translocations and other gene order modifying *rearrangements* are the events considered in synteny evolution studies, which aim at reconstructing ancestral genome organizations or inferring rearrangement based phylogenies. Figure 4.1 attempts to represent some models for genome evolution according to the type of mutations they handle. For example, the blue and gray dots represent phylogenetic methods from nucleotide or genome level mutations. The gray star and blue square, respectively, stand for evolutionary studies of gene order and gene content accounting for rearrangement or gene gains and losses.

In this paper, we review the attempts to integrate this variety of multiple-scale events into a single framework (red region in Fig. 4.1). In addition we contribute to this integration by showing how reconciled gene trees can be improved using synteny information.

Sequence evolution (substitutions and indels) and chromosome evolution (rearrangement, gene order) are traditionally two separate domains of study. This can be traced back to the early stages of evolutionary studies based on molecular data. Usually, the first molecular phylogenies are dated back to the Pauling and Zuckerkandl series of papers in the early 1960s [10]. Nevertheless 30 years before, Surtevant and Dobzhansky were drawing *Drosophila* phylogenies comparing the structure of polytene chromosomes [11]. Even the mathematics of chromosome rearrangements were already investigated at that time, and computational problems were formally stated [12]. However, these pioneering works have not been followed, and mathematical and computational studies of genome rearrangements have been nearly absent for several decades. Instead methodologists did put a lot of effort into the modelization of the evolution of DNA or protein sequences. Advanced models and algorithms have been developed [13, 14] (see also Chap. 6 in this volume), integrating character substitutions and indels, reconstructing phylogenies and ancestral states by various statistical methods. It is only in the early 1980s that formal models of gene order evolution were investigated again, after a nearly 50 year hiatus [15], mainly following Sankoff's efforts [16, 17]. As for today, despite significant progress, the considered models for gene order evolution are still not reaching the sophistication of those for sequence evolution [18] (see also Chap. 7 in this volume).

For a long time, in most phylogenetic studies based on sequence, only genes with apparently simple histories, typically those present in a single copy in every genome, were considered [19]. This aspect has changed during the last 20 years, driven by the gene tree/species tree reconciliation studies pioneered by Goodman et al. [20]. Reconciliation gives a way of integrating gene family evolution into models of sequence evolution. Recently, many sophisticated methods for gene tree and/or species tree inference, integrating gene sequence evolution and gene insertion, duplication, loss or transfer, into a unified model have been developed [4, 5, 21–25]. This integration is represented by the purple square and dot in Fig. 4.1 (see also Chap. 12 in this volume).

In parallel, genome rearrangement studies were at first developed in a context where genomes were also assumed to have exactly the same set of genes, with exactly one copy per genome. Such an assumption is reasonable for specific data sets such as animal mitochondrial genomes [16], or in a more general context provided an appropriate pre-processing of genomes [26–30]. In this context of single copies, two kinds of models have been investigated: a "global model" where a genome is encoded by a unique object (permutation, string, or a variant) with a value space of size $\geq n!$ where n is the number of genes, and a "local model" in which a genome is decomposed into $O(n)$ characters as adjacencies evolving independently, each taking two possible values (present/absent). For global models it was shown that comparing two genomes can be done pretty efficiently [31, 32], while almost all attempts to compare more than two genomes lead to intractable problems (survey in [18]). The local model gave rise to easier problems [27–30, 33] (see also Chap. 7 in this volume), the drawback being that the independence hypotheses between adjacencies lead to ancestral states that are not necessarily compatible with linear or circular chromosomal structures, leading again to difficult linearization problems (although few exceptions exist [34, 35]).

Integrating duplications and more generally gene families with complex histories into the study of synteny evolution (the green star in Fig. 4.1) has been initiated by David Sankoff with the so-called "exemplar approach" [6], which consists in encoding genomes as strings instead of permutations, allowing for the representation of a single gene many times in a genome. In this spirit an insight on gene content evolution can be inferred from synteny by the detection of orthology relationships [36–38]. But this direction was hampered by ubiquitous intractability results even for the comparison of two genomes [18, 39, 40] (see also Chap. 9 of this volume). The local model has also allowed to overcome the computational complexity in that direction [9] (red dot in Fig. 4.1, see also Chap. 7 of this volume).

In 2000, a conference was held named *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families* [41], organized by Sankoff and Nadeau. The title of the conference, and of the companion book was already a manifest towards an integration of gene evolution and genome evolution. The present chapter, the volume it is included in, and the 2013 MAGE conference are also, in some ways, attempts to follow up on this event. In particular, the book published in 2000 contained the "Duplication, Rearrangement and Reconciliation" chapter by Sankoff and El-Mabrouk [42], which we revisit, 13 years later.

That chapter was one among several examples (see also Chap. 2 of the present volume for example) of a work in which David Sankoff has laid the basis of a research avenue several years before it was explored by the scientific community. We feel the exploration of this avenue really starts now, 13 years later. To advocate this, we first summarize the key concepts used by Sankoff and El-Mabrouk [42] (Sect. 4.2). Then we describe the two lines of research that have been built on these initial ideas: using phylogenetic information, by means of reconciliation, to study gene-order evolution (Sect. 4.3, red star in Fig. 4.1), and using gene-order information to study gene family evolution (Sect. 4.4, red square in Fig. 4.1). We give a contribution to the latter part by constructing an accurate method of synteny-aware gene tree correction. We conclude by some discussion points and perspectives on possible integration of phylogenies, syntenies and histories in a unified framework for studying genome evolution (Sect. 4.5, red dot in Fig. 4.1).

4.2 Duplication, Rearrangement and Reconciliation

In this section we revisit the 2000 Sankoff and El-Mabrouk chapter [42]. It is also the occasion to introduce concepts and objects related to reconciliations and rearrangements.

Evolution of Species and Genes Species evolve through *speciation*, which is the separation of one species into two distinct descendant species. The result of this evolution is a set Σ of n extant species. A *species tree* on Σ is a binary rooted tree whose leaves are in bijection with Σ, representing the evolutionary relationship

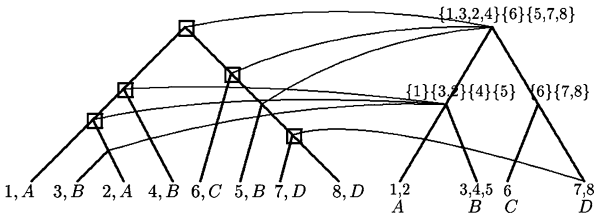


Fig. 4.2 (Copied from [42]). A reconciliation of a gene tree (*on the left*) with a species tree (*on the right*). The genome (*letter*) to which each gene (*number*) belongs is indicated in the label of the corresponding leaf in the gene tree. The mapping M is indicated by links between the two trees. The mapping E is indicated on the gene tree: *squares* are duplication nodes, while other internal nodes are speciations. In the species tree, ancestral nodes are labeled by their gene contents. Each set corresponds to a single ancestral gene

between the species of Σ: an internal node is an ancestral species at the moment of a speciation, and its two children are the descendent species.

Species are identified with their genomes, and a genome is a set of genes (plus a structure for gene order detailed later). Genes undergo speciation when the species to which they belong do. Within a species, genes can be *duplicated*, *lost* or *gained*. Various mechanisms lead to duplications of various sizes, ranging from one single gene (or segment of genes) to the whole genome [43]. Gene losses arise through the pseudogenization of previously functional genes or the outright deletion of chromosomal fragments. There are other gene level events like transfers, but here we stay with only duplication and losses as it was the context of the 2000 chapter [42].

A *gene tree*, representing the evolution of a *gene family*, is a binary rooted tree where each leaf is labeled by a gene, belonging to a species in Σ. Each internal node of a gene tree refers to an ancestral gene at the moment of an event (either speciation or duplication) resulting in two copies of this gene. The *lowest common ancestor* (LCA) of nodes x and y in a tree T, written $lca_T(x, y)$, is the internal node of T that is both an ancestor of x and y and is the farthest from the root of T.

In a gene tree, losses are invisible, and speciation and duplication events are not distinguishable, unless we *reconcile* it with a species tree.

Reconciliation A *reconciliation* of a gene tree T with a species tree S consists in assigning to each internal gene g of T a species $M(g) = s$, which is a node of S (either extant or ancestral), indicating that gene g belongs to species s, and an evolutionary event $E(g) \in \{\text{speciation, duplication}\}$. This is done in a way ensuring that the evolutionary history of the gene family described by the reconciliation is in agreement with the species evolution described by S. An example of reconciliation is shown in Fig. 4.2.

The reconciliation of T with S gives information about the gene family history. In particular, it defines the gene content of an ancestral species s at the time of speciation. A reconciliation also implies the orthology and paralogy relationships between genes: Two genes g and g' of T are said to be orthologous if $E(lca_T(g, g')) = Spec$; g and g' are paralogous if $E(lca_T(g, g')) = Dup$. They are said to be *ohnologous* if they are paralogous and the duplication event at $lca_T(g, g')$ is due to a whole-genome duplication.

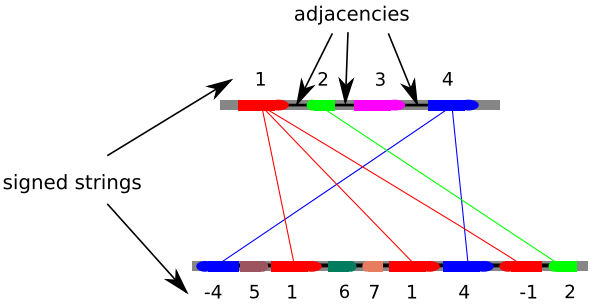


Fig. 4.3 Gene order comparison of two genomes with duplications. Each genome is a signed string on the gene family alphabet. The direction of each gene is written according to the relative orientation on the two genomes. Homology relationships are edges between genes of the two different genomes, and the comparison is achieved with matching problems on this bipartite graph. *Black links* indicate adjacencies which are another way to encode the strings. Focusing on one adjacency independently of the neighboring ones makes the comparison computations tractable, but the linear structure of the ancestral genomes might be lost

Since the work of Page [44, 45] in the beginning of the 1990s, and with an increasing interest in the last decade, several approaches have been developed to reconcile a gene tree with a species tree. The main guiding principle has been to optimize a given criterion such as parsimony in terms of duplications and/or losses or maximum likelihood (see [46] for a recent review). Recent methods aim at reconstructing gene trees and reconciliations simultaneously [22, 23, 25].

Gene Order and Genome Rearrangements We defined a genome as a set of genes and a structure on these genes. This structure can be for example a *signed permutation* which gives a total order to the genes, and a direction (± 1) to each gene. When two genomes have equal gene content (gene evolution is ignored), a rearrangement scenario is a sequence of operations on the permutation which transforms one genome into the other. In that case efficient algorithms exist for many genome rearrangement distances, often based on analyzing the structure of permutations and of their breakpoint graph. They hardly generalize to more than two genomes: *median* problems, considering three genomes, are often intractable, and hardly allow the exploration of solution spaces [18, 47].

When gene families may have several copies, a natural generalization of signed permutations is given by *signed strings* over the gene families alphabet (see Fig. 4.3). Each family is assigned an integer and each occurrence of this integer indicates an occurrence of a gene from the corresponding family. The comparison of two such strings can be achieved by finding an orthology assignment between gene copies from the same family. Keeping only ortholog pairs of genes transforms a signed string into a signed permutation, on which known algorithms apply. The orthology assignment is a matching in the bipartite graph over the string elements, in which there is an edge between two genes of the same family in different genomes (see Fig. 4.3). In his 1999 paper [6], Sankoff introduced the notion of *exemplar matching* that assumes all duplications are posterior to the speciation between the

two genomes. This corresponds to taking only one edge per family in the bipartite graph. But this also leads to hard problems: computing a parsimonious exemplar matching is hard, even to approximate for the simplest distances (see [39, 48] and references therein, as well as Chap. 9 in this volume). Other notions of matchings were introduced later, not assuming the precedence of speciation over all duplications [36], also leading to hard optimization problems [18, 49]. Although reasonably efficient exponential time algorithms have been developed [50], it is still an open question as to whether these approaches will scale efficiently to more than three genomes (see Chap. 13 in this volume).

Reconciliations and Rearrangements Interestingly, gene order provides formal methods for inferring "positional homology" [37], which can be applied to the detection of orthology [51] or ohnology [52, 53]. This places gene-order information as a concurrent of reconciliation for orthology or ohnology detection. This, among other reasons, calls for the use of reconciliation and gene order in the same framework, because both carry information on gene family evolution. Synteny and reconciled phylogenies have sometimes been used together to detect orthology or ohnology [52–54] but rarely in a whole-genome evolution model integrating duplications and rearrangements.

This is what was proposed by Sankoff and El-Mabrouk [42]. In their framework, an arbitrary number of genomes are given, along with a species tree describing their evolution. In addition, reconciled gene trees are given for all gene families and the genes at the leaves of these gene trees are ordered in the genomes. Then, as orthology is known from reconciliation, considerations on rearrangement distances between genomes can include duplications in the permutation model, tending to lower the additional algorithmic complexity. Nevertheless the general problem still contains two difficult problems, the median and the exemplar problems, as particular cases.

A solution was proposed in that paper to infer gene content and order at each internal node of a species tree, in a way that minimizes a total breakpoint distance on the species tree. Handling multicopy gene families was undertaken by integrating exemplar matching for the duplications that were identified by reconciliation to be posterior to speciations. A heuristic was proposed for solving the general problem. It was never applied on data, partly because it was developed before data was available in a usable form, and partly because the aim of this article was to open a research avenue more than to close a problem.

For several years this avenue has remained almost unexplored. But several recent publications have followed this research program, at least in some way, with updated genome rearrangement and reconciliation methods and some biological results.

4.3 Gene Tree Reconciliations Inform Synteny Evolution

Genome rearrangement studies with permutation or string models, i.e. global models, usually do not handle a large number of genomes or events. After the sem-

inal article by Sankoff and El-Mabrouk [42], we may mention a remarkable attempt of Ma et al. [55] to devise an integrated global model of genome evolution under certain restrictive conditions. But the computational complexity of global models usually restrains them to the study of small gene clusters, while paradoxically the histories of whole genomes are often inferred with local models of evolution. We describe the two possibilities, the first with a survey of existing literature and the second with a contribution to synteny-aware gene phylogeny construction.

4.3.1 Evolution of Gene Clusters with Global Models

A large fraction of duplications affecting genome organization consists of local duplications, mainly caused by unequal crossing-over during meiosis. As this phenomenon is favored by the presence of repetitive sequences, a single duplication can induce a chain reaction leading to further duplications, eventually creating large repetitive regions. When those regions contain genes, the result is a *Tandemly Arrayed Gene* (*TAG*) *cluster*: a group of paralogous genes that are adjacent on a chromosome. In the 1970s, Fitch [56] introduced the first evolutionary model for TAGs accounting for *tandem duplication*, in which the two descendent copies of a duplicated gene are adjacent. Since then, several studies have considered the problem of inferring an evolutionary history for TAG clusters [57–60]. These are essentially phylogenetic inference methods using the additional constraint that the resulting tree should be a *duplication tree*, i.e. induces a duplication history according to the given gene order. However, due to the occurrence of mechanisms other than tandem duplications (losses, rearrangements), it is often impossible to reconstruct a duplication tree [61].

In a series of papers [62–65], a solution is proposed to retrace the history of gene clusters subject to tandem duplications, losses and rearrangements. The latest developments allow us to study the evolution of orthologous TAG clusters in different species, subject to tandem duplications, inverted tandem duplications, inversions and deletions, each event involving one or a set of adjacent genes. Given the gene trees, the species tree, and the order of genes in TAG clusters, the method for inferring ancestral clusters combines reconciliation with gene-order information. First, ignoring gene order, reconciliation is used to infer ancestral gene contents. Second, ancestral gene orders are inferred that are consistent with minimizing the number of rearrangement events required to obtain a duplication tree. Due to the NP-hardness of the problem, exact approaches can hardly be envisaged, except for the special case of clusters in a single species subject to simple duplications (duplications of single genes) [62]. The DILTAG software [64, 65] developed for the most general case is a heuristic, showing good performance in practice for the inference of recent evolutionary events. Despite the uncertainty associated with the deeper parts of the reconstructed histories, it can be used to infer the duplication size distribution with some precision.

4.3.2 Evolution of Whole Genome with Adjacency Models

The structure of the genomes, as seen in Fig. 4.3, can be described by a set of adjacencies instead of signed permutations or strings. Adjacencies are edges linking gene extremities. If we wish to formalize the linear or circular structure of genomes, the set of adjacencies should be a matching over gene extremities. This models all types of genomes, from linear multichromosomal eukaryote genomes to circular bacterial genomes, possibly including circular or linear plasmids.

The switch to local models is achieved by comparing adjacencies instead of genomes. When extant genomes all have the same gene content, it is possible to rapidly and accurately reconstruct ancestral genomes [27, 30, 66] or even species phylogenies [67]. It is even possible to include no equal gene contents [9] for the same purposes (see Chap. 7 in this volume).

If gene families are described with reconciled trees with duplications, the software DupCAR [68] proposes the reconstruction of ancestral adjacencies. Nevertheless, its possible applications are rather limited as it does not handle losses and requires fully dated gene trees and species tree, in order to compute reconciliations that are compatible with the provided date information. Such precise information about gene trees is rare.

The joint use of adjacencies and reconciled gene trees has really been exploited in the last three years. Agora [66] or the method of El-Mabrouk and colleagues [69, 70] reconstruct ancestral adjacencies with a sort of Dollo parsimony principle, by pairwise comparisons of extant gene orders. Methods designed initially to handle equal gene content [27, 30] can also naturally be extended to follow this principle, by using orthology/paralogy information obtained from reconciliations. So adjacencies are reconstructed but no evolutionary scenario is proposed to explain them. In all such methods, linearization routines, based on the Traveling Salesman Problem [69, 70] or on path/cycles graph covering techniques [27, 35] are used to linearize ancestral genomes, that is, to remove a posteriori some of the proposed adjacencies so that every gene (or gene extremity) has at most two (or one) neighbor.

DeCo [71] is an algorithm and a software which models the evolution of adjacencies, and reconstructs ancestral adjacencies by minimizing the number of gains and losses of adjacencies (due to rearrangements) along the species tree. It is based on a generalization of the Sankoff-Rousseau algorithm (see Chap. 3 in this volume) adapted to the presence/absence of adjacencies and to reconciled gene trees. It has recently been extended to include transfers (Patterson et al., in preparation). But here again, the resulting ancestral adjacencies might not be compatible with a genome structure and require to be processed a posteriori.

4.4 Synteny Informs Gene Family Evolution

Synteny as a Control Synteny is usually a good way to infer orthologs [72]. As reconciled gene trees also yield orthology relationships, it is possible to compare the

results from both independent methods. This is what is often done to assess the quality of gene trees [73, 74]. Orthologs obtained from synteny are assumed to represent the truth, and can be compared with orthologs obtained from reconciliations.

An alternative idea is to use the structure of reconstructed ancestral genomes as a quality index. We have seen that in the extant genomes, each gene shares two adjacencies with other genes (except for chromosome extremities). Theoretically it should also be the same in the ancestral genomes. But due to errors in gene trees, in the species tree, or in the inference algorithms, in practice there are a lot of exceptions. And the number of exceptions should be correlated with the quality of gene trees. So the quality of gene trees can be measured by the number of ancestral genes with more or less than two adjacencies with other genees [5, 71].

These quality tests are first steps towards integrating synteny information into the construction of gene trees. Indeed, if the quality of gene trees can be measured with synteny, the next step is to integrate synteny into the objective function when computing gene trees.

Synteny as an Input Data for Gene Trees Very few studies use synteny information to construct gene trees. The only ones we are aware of are the ones of Wapinski et al. [75, 76]. In these papers, family clustering as well as gene trees are constructed with a "synteny score" as well as a "sequence score".

In the present book three different contributions to this direction are given. Chapter 13 deals with the construction of gene families with synteny information. Chapter 12 proposes a method to detect inconsistencies in gene trees based on synteny information.

In the present chapter we show how, by a simple procedure using available software, it is possible to guide the construction of gene trees with gene-order information. We retrieved all gene trees from the Ensembl database [73], version 70. Then we applied the DeCo software [71] to infer ancestral adjacencies in a mammalian species tree. As said before, DeCo does not guarantee that ancestral genes have at most two neighbors, as extant genes. This apparent weakness was turned into a strength as it was used for a quality control for gene trees. We show that it can be used to construct better gene trees.

Take an ancestral gene g with three neighbors, such that two of them, $g1$ and $g2$, are speciation nodes of the same gene tree, and are the two children of a duplication node d in this gene tree (see Fig. 4.4). Let then $g11$, $g12$ (resp $g21$ and $g22$) be the children of $g1$ (resp $g2$). Transform the subtree $((g11, g12), (g21, g22))$ to $((g11, g22), (g21, g12))$ whenever this switches d to a speciation node in the reconciliation (1519 trees out of 13132 Ensembl trees that contain mammalian genes can be modified this way). This transformation is illustrated in Fig. 4.4.

For all 1519 trees, we retrieved the gene family alignment from Ensembl. With PhyML [2] we computed the likelihood of two trees, before and after the transformation, given this alignment. These two likelihoods were compared with Consel [77]. For a majority of trees (773), the likelihood of the corrected tree is higher than the likelihood of the initial tree. And the correction is rejected (probability of the corrected tree <0.05) for only 281 of them (Fig. 4.5).

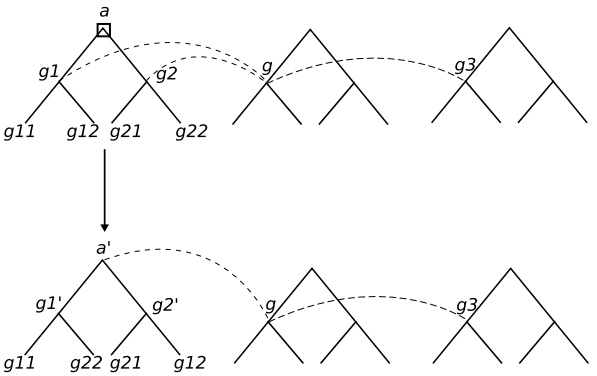


Fig. 4.4 Modifying a tree according to synteny information. The fact that gene g has three neighbors according to DeCo points to a possible error in gene trees (every gene should have at most two neighbors). In this situation we suspect the duplication d to be erroneous, as having only one gene instead of the two ($g1$ and $g2$) would decrease the number of neighbors of g. So we propose the correction in which there is only one gene, by rearranging the subtree rooted at d

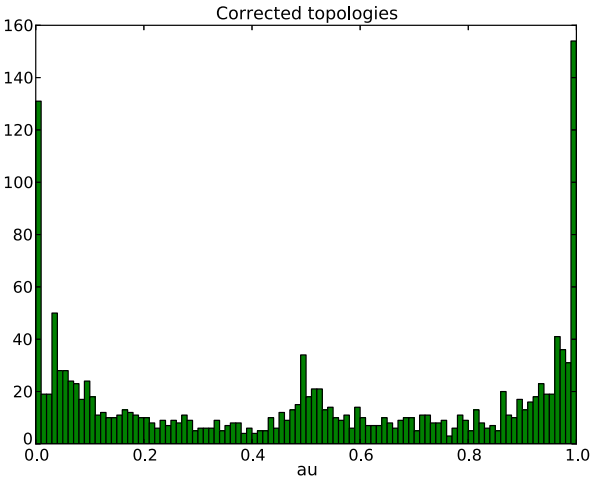


Fig. 4.5 Each *green bar* gives, for a given interval of p-values (x-axis), the number of gene families (y-axis), among the 1519 whom we corrected the tree, for which the corrected tree should be preferred with a significance in the p-value interval. The shape of this graph shows that Ensembl trees are in general not significantly preferred, showing the accuracy of most corrections

In conclusion, the synteny signal can be used to choose among the numerous trees that are statistically equivalent according to the sequence signal. This choice is sometimes in agreement with other reconciliation-based tree correction methods, but sometimes adds additional information. This provides a step towards synteny-aware gene tree construction methods (red square in Fig. 4.1).

4.5 Towards and Integrated Model

Boussau and Daubin [19] call for models of molecular evolution that would integrate all kinds of mutations and find likely ones according to a mixture of objectives. Because the species tree, gene trees and rearrangements all depend on each other, an iterative method, computing these objects one after another would not find an optimal solution. Integrated models of species and gene trees are already working [5, 25] (purple region in Fig. 4.1).

But despite the efforts we described here to mix gene trees and rearrangement in the same framework, the three-way influence is far from reached. Some attempts can be mentioned, as Ma et al.'s paper [55] gives a global algorithm under some very strict conditions (exact molecular clock, no convergent evolution, no breakpoint reuse, no gene loss).

Some other are less ambitious, like Kahn and colleagues who argued in a series of papers (see [78] and references therein) that reconciled trees cannot describe properly the evolutionary relationships and propose an extension to DAGs. They give insights into handling both evolutionary relationships and synteny, to trace the history of segmental duplications.

But 13 years after the paper of Sankoff and El-Mabrouk, which marked a first star in the three-way intersection of Fig. 4.1, and accounts for an increasing interest in this area over recent years, the existence of a phylogenetic method over all events is still an open question (red dot in Fig. 4.1).

Acknowledgements This work is funded by the Agence Nationale pour la Recherche, Ancestrome project ANR-10-BINF-01-01.

References

1. Graur, D., Li, W.H.: Fundamentals of Molecular Evolution, 2nd edn. Sinauer Associates, Sunderland (2000)
2. 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)
3. Csurös, M.: Count: evolutionary analysis of phylogenetic profiles with parsimony and likelihood. Bioinformatics **26**(15), 1910–1912 (2010)
4. Szöllosi, 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. USA **109**(43), 17513–17518 (2012)
5. Boussau, B., Szöllosi, 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)
6. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics **15**(11), 909–917 (1999)
7. Zheng, C.: Pathgroups, a dynamic data structure for genome reconstruction problems. Bioinformatics **26**(13), 1587–1594 (2010)
8. Tang, J., Moret, B.M.E.: Scaling up accurate phylogenetic reconstruction from gene-order data. Bioinformatics **19**(Suppl 1), i305–i312 (2003)

9. 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 (2013)
10. Zuckerkandl, E., Pauling, L.: Molecules as documents of evolutionary history. J. Theor. Biol. **8**(2), 357–366 (1965)
11. Sturtevant, A., 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**, 448–450 (1936)
12. Sturtevant, A., Novitski, E.: The homologies of chromosome elements in the genus drosophila. Genetics **26**, 517–541 (1941)
13. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.J.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge (1998)
14. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Sunderland (2004)
15. Watterson, G.A., Ewens, W.J., Hall, T.E., Morgan, A.: The chromosome inversion problem. J. Theor. Biol. **99**, 1–7 (1982)
16. 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)
17. Sankoff, D.: Edit distances for genome comparisons based on non-local operations. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) Proceedings: Combinatorial Pattern Matching, Third Annual Symposium, CPM 92, Tucson, Arizona, USA, 29 April–1 May 1992. Lecture Notes in Computer Science, vol. 644, pp. 121–135. Springer, Berlin (1992)
18. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press, Cambridge (2009)
19. Boussau, B., Daubin, V.: Genomes as documents of evolutionary history. Trends Ecol. Evol. **25**(4), 224–232 (2010)
20. 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)
21. Durand, D., Halldórsson, B.V., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. J. Comput. Biol. **13**(2), 320–335 (2006)
22. Akerborg, O., Sennblad, B., Arvestad, L., Lagergren, J.: Simultaneous Bayesian gene tree reconstruction and reconciliation analysis. Proc. Natl. Acad. Sci. USA **106**(14), 5714–5719 (2009)
23. Rasmussen, M.D., Kellis, M.: A Bayesian approach for fast and accurate gene tree reconstruction. Mol. Biol. Evol. **28**(1), 273–290 (2011)
24. 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)
25. Szöllosi, G.J., Rosikiewicz, W., Bousseau, B., Tannier, E., Daubin, V.: Efficient exploration of the space of reconciled gene trees. Syst. Biol. (2013). doi:10.1093/sysbio/syt054
26. Pevzner, P.A., Tesler, G.: Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. Genome Res. **13**(1), 37–45 (2003)
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. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. PLoS Comput. Biol. **4**(11), e1000234 (2008)
29. Chauve, C., Gavranovic, H., Ouangraoua, A., Tannier, E.: Yeast ancestral genome reconstructions: the possibilities of computational methods ii. J. Comput. Biol. **17**(9), 1097–1112 (2010)
30. Jones, B.R., Rajaraman, A., Tannier, E., Chauve, C.: Anges: reconstructing ancestral genomes maps. Bioinformatics **28**(18), 2388–2390 (2012)

31. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In: Thomson Leighton, F., Borodin, A. (eds.) Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, Las Vegas, Nevada, USA, 29 May–1 June 1995, pp. 178–189. ACM, New York (1995)
32. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: 36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23–25 October 1995, pp. 581–592. IEEE Computer Society, Los Alamitos (1995)
33. Zhang, Y., Hu, F., Tang, J.: A mixture framework for inferring ancestral gene orders. BMC Genomics **13**(Suppl 1), S7 (2012)
34. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–1329 (2011)
35. Maňuch, J., Patterson, M., Wittler, R., Chauve, C., Tannier, E.: Linearization of ancestral multichromosomal genomes. BMC Bioinform. **13**(Suppl 19), S11 (2012)
36. 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)
37. Dewey, C.N.: Positional orthology: putting genomic evolutionary relationships into context. Brief. Bioinform. **12**(5), 401–412 (2011)
38. Doerr, D., Thévenin, A., Stoye, J.: Gene family assignment-free comparative genomics. BMC Bioinform. **13**(Suppl 19), S3 (2012)
39. Zhu, B.: Approximability and fixed-parameter tractability for the exemplar genomic distance problems. In: Chen, J., Cooper, S.B. (eds.) Proceedings: Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009, Changsha, China, 18–22 May 2009. Lecture Notes in Computer Science, vol. 5532, pp. 71–80. Springer, Berlin (2009)
40. El-Mabrouk, N., Sankoff, D.: Analysis of gene order evolution beyond single-copy genes. Methods Mol. Biol. **855**, 397–429 (2012)
41. Sankoff, D., Nadeau, J. (eds.): Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families. Kluwer Academic, Dordrecht (2000)
42. Sankoff, D., El-Mabrouk, N.: Duplication, rearrangement and reconciliation. In: Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families. Computational Biology Series, vol. 1, pp. 537–550. Kluwer Academic, Dordrecht (2000)
43. Gregory, T.R. (ed.): The Evolution of the Genome. Elsevier/Academic Press, Amsterdam (2004)
44. Page, R.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. Syst. Biol. **43**, 58–77 (1994)
45. Page, R.: Genetree: comparing gene and species phylogenies using reconciled trees. Bioinformatics **14**, 819–820 (1998)
46. Doyon, J.P., Ranwez, V., Daubin, V., Berry, V.: Models, algorithms and programs for phylogeny reconciliation. Brief. Bioinform. **12**(5), 392–400 (2011)
47. Miklos, I., Tannier, E.: Approximating the number of double cut-and-join scenarios. Theor. Comput. Sci. **439**, 30–40 (2012)
48. Bulteau, L., Jiang, M.: Inapproximability of $(1, 2)$-exemplar distance. In: Bleris, L.G., Mandoiu, I.I., Schwartz, R., Wang, J. (eds.) Proceedings: Bioinformatics Research and Applications—8th International Symposium, ISBRA 2012, Dallas, TX, USA, 21–23 May 2012. Lecture Notes in Computer Science, vol. 7292, pp. 13–23. Springer, Berlin (2012)
49. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. **13**(1), 19–53 (2009)
50. 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), 1093–1115 (2008)

51. Goodstadt, L., Ponting, C.P.: Phylogenetic reconstruction of orthology, paralogy, and conserved synteny for dog and human. PLoS Comput. Biol. **2**(9), e133 (2006)
52. Ouangraoua, A., Tannier, E., Chauve, C.: Reconstructing the architecture of the ancestral amniote genome. Bioinformatics **27**(19), 2664–2671 (2011)
53. Makino, T., McLysaght, A.: Positionally biased gene loss after whole genome duplication: evidence from human, yeast, and plant. Genome Res. **22**(12), 2427–2435 (2012)
54. Cai, B., Yang, X., Tuskan, G.A., Cheng, Z.M.: Microsyn: a user friendly tool for detection of microsynteny in a gene family. BMC Bioinform. **12**, 79 (2011)
55. Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Miller, W., Haussler, D.: The infinite sites model of genome evolution. Proc. Natl. Acad. Sci. USA **105**(38), 14254–14261 (2008)
56. Fitch, W.: Phylogenies constrained by cross-over process as illustrated by human hemoglobins and a thirteen-cycle, eleven amino-acid repeat in human apolipoprotein A–I. Genetics **86**, 623–644 (1977)
57. Bertrand, D., Gascuel, O.: Topological rearrangements and local search method for tandem duplication trees. IEEE/ACM Trans. Comput. Biol. Bioinform. **2**, 15–28 (2005)
58. Elemento, O., Gascuel, O., Lefranc, M.P.: Reconstructing the duplication history of tandemly repeated genes. Mol. Biol. Evol. **19**(3), 278–288 (2002)
59. Tang, M., Waterman, M., Yooseph, S.: Zinc finger gene clusters and tandem gene duplication. In: Research in Molecular Biology (RECOMB 2001), pp. 297–304 (2001)
60. Zhang, L., Ma, B., Wang, L., Xu, Y.: Greedy method for inferring tandem duplication history. Bioinformatics **19**, 1497–1504 (2003)
61. Gascuel, O., Bertrand, D., Elemento, O.: Reconstructing the duplication history of tandemly repeated sequences. In: Gascuel, O. (ed.) Mathematics of Evolution and Phylogeny, pp. 205–235. Oxford University Press, Oxford (2005)
62. Lajoie, M., Bertrand, D., El-Mabrouk, N., Gascuel, O.: Duplication and inversion history of a tandemly repeated genes family. J. Comput. Biol. **14**(4), 462–478 (2007)
63. Bertrand, D., Lajoie, M., El-Mabrouk, N.: Inferring ancestral gene orders for a family of tandemly arrayed genes. J. Comput. Biol. **15**(8), 1063–1077 (2008)
64. Lajoie, M., Bertrand, D., El-Mabrouk, N.: Inferring the evolutionary history of gene clusters from phylogenetic and gene order data. Mol. Biol. Evol. **27**(4), 761–772 (2010)
65. Tremblay-Savard, O., Bertrand, D., El-Mabrouk, N.: Evolution of orthologous tandemly arrayed gene clusters. BMC Bioinform. **12**(Suppl 9), S2 (2011)
66. Muffato, M., Louis, A., Poisnel, C.E., Crollius, H.R.: Genomicus: a database and a browser to study gene synteny in modern and ancestral genomes. Bioinformatics **26**(8), 1119–1121 (2010)
67. Hu, F., Gao, N., Zhang, M., Tang, J.: Maximum likelihood phylogenetic reconstruction using gene order encodings. In: 8th Annual IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'11) (2011)
68. Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Zhang, L., Miller, W., Haussler, D.: Dupcar: reconstructing contiguous ancestral regions with duplications. J. Comput. Biol. **15**(8), 1007–1027 (2008)
69. Bertrand, D., Gagnon, Y., Blanchette, M., El-Mabrouk, N.: Reconstruction of ancestral genome subject to whole genome duplication, speciation, rearrangement and loss. In: Moulton, V., Singh, M. (eds.) Proceedings: Algorithms in Bioinformatics, 10th International Workshop, WABI 2010, Liverpool, UK, 6–8 September 2010. Lecture Notes in Computer Science, vol. 6293, pp. 78–89. Springer, Berlin (2010)
70. Gagnon, Y., Blanchette, M., El-Mabrouk, N.: A flexible ancestral genome reconstruction method based on gapped adjacencies. BMC Bioinform. **13**(Suppl 19), S4 (2012)
71. Bérard, S., Gallien, C., Boussau, B., Szöllősi, G.J., Daubin, V., Tannier, E.: Evolution of gene neighborhoods within reconciled phylogenies. Bioinformatics **28**(18), i382–i388 (2012)
72. Jun, J., Mandoiu, I.I., Nelson, C.E.: Identification of mammalian orthologs using local synteny. BMC Genomics **10**, 630 (2009)

73. Vilella, A., Severin, J., Ureta-Vidal, A., Heng, L., Birney, E.: EnsemblCompara GeneTrees: complete, duplication-aware phylogenetic trees in vertebrates. Genome Res. **19**(2), 327–335 (2009)
74. Wu, Y.C., Rasmussen, M.D., bansal, M.S., Kellis, M.: Treefix: statistically informed gene tree error correction using species trees. Syst. Biol. **62**(1), 110–120 (2013)
75. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Automatic genome-wide reconstruction of phylogenetic gene trees. Bioinformatics **23**(13), i549–i558 (2007)
76. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Natural history and evolutionary principles of gene duplication in fungi. Nature **449**, 54–61 (2007)
77. Shimodaira, H., Hasegawa, M.: Consel: for assessing the confidence of phylogenetic tree selection. Bioinformatics **17**(12), 1246–1247 (2001)
78. Kahn, C.L., Hristov, B.H., Raphael, B.J.: Parsimony and likelihood reconstruction of human segmental duplications. Bioinformatics **26**(18), i446–i452 (2010)

Chapter 5
The Genesis of the DCJ Formula

Anne Bergeron and Jens Stoye

Abstract The formula $N - (C + I/2)$ to compute the number of Double-Cut-and-Join operations needed to transform one genome into another is both simple and easy to prove. When it was published, in 2006, we omitted all details on how it was constructed. In this chapter, we will give an elementary treatment on the intuitions and methods underlying the formula, showing that simplicity is sometimes difficult to achieve. We will also prove that this formula is one among an infinite number of candidates, and that the techniques can be applied to other genomic distances.

5.1 Introduction

In May 2005, the authors attended the Recomb meeting in Boston, Mass. They had an accepted paper on a tamed variant of the genome rearrangement problem. Happily for them, the presenter was a young graduate student, Julia Mixtacki, and the authors had plenty of lounging time. On the sunny terraces, cafés and salons of the MIT campus, David Sankoff managed to introduce us to Sophia Yancopoulos, who had an original, thrilling, radical, but very informal view on genome rearrangements, presented on a poster at that conference. Her paper [10], written with O. Attie and R. Friedberg, appeared in the same month in Bioinformatics, but was a bit of a challenge to read.

The authors' team, including Julia who would play a determinant role in the sequel, felt that there should exist a more formal way of computing this distance. The road was bumpy. We first had to understand the real power of the Double-Cut-and-Join (DCJ) operation introduced by Yancopoulos et al. The original paper focused on the usefulness of the new concepts to explain known results, rather than exploring the consequences of the new definition. It was necessary to forget, for the

A. Bergeron (✉)
Lacim, Université du Québec à Montréal, Montréal, Canada
e-mail: bergeron.anne@uqam.ca

J. Stoye
Technische Fakultät, Universität Bielefeld, Bielefeld, Germany

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_5,

time being, the results of the former decade that explored rearrangement operations on linear genomes.

Immediately after Recomb, the authors and Julia spent three days together in Montréal and started trying to understand and formalize the ideas they had been introduced to. However, we completely failed, as we were too closely following the Yancopoulos 'recipe', instead of starting from scratch and re-phrase the DCJ model in our own language. Therefore, it was possibly a good idea to wait for eight months before continuing, so we could leave behind most of that early attempt.

The first breakthrough occurred in February 2006 in Lisbon, Portugal, when AB was giving a series of five lectures to graduate students at the Instituto Gulbenkian de Ciencia. The lectures were given in the morning, and the lecturer had the afternoons to herself to pursue her own research. Julia came from Bielefeld to Lisbon for a week. This move resulted in the definition of the adjacency graph, and in a deep grasp of the DCJ operations. While the shift of our underlying data structure, from the breakpoint graph to the adjacency graph, was rather formal, the understanding of the nature of the DCJ operations relied on a toy genome, made of black and white electrical chords, together with male and female connections that stood for double-stranded DNA, gene orientation, breaks and repairs. At one point, the two researchers 'executed' all the variants of a DCJ operation, using their four hands and the model genome. They were so concentrated on the 'proof' that it took them a certain time to realize that maintenance people were peering at them through the door's window. These results are described in Sects. 5.2 and 5.3.

The next, and crucial, development happened in Montréal, in Spring 2006, when JS came to visit as part of his sabbatical. At that point, the problem was not to develop one formula, but to cope with too many formulas! One of the frustrated authors decided, on one evening, to rely on a dirty mathematical trick, discussed in Sect. 5.4, to come up with the 'simplest' formula among those candidates. The result was suddenly quite simple, and the proof of the DCJ distance became elementary, which is discussed in Sect. 5.5.

In fact, the DCJ model is only one out of many where the same technique can be applied to quickly derive general and simple distance formulas, as we will show in Sect. 5.6 for the algebraic, the breakpoint and two single-cut distances.

5.2 Rearrangement Operations and the Adjacency Graph

Here we will briefly recall the notation we use to represent and manipulate genomes. While it may nowadays seem natural and many other authors have adopted the terminology, in 2005–2006 we spent probably more time on the development of this notation than on the derivation and proving of the DCJ distance formula and sorting algorithm.

A *gene* is a piece of DNA with two extremities, its *head* and its *tail*. For a gene a we denote its head by a^h and its tail by a^t. A *genome* for a given set of genes G is a set of *adjacencies*, consisting of pairs of gene extremities, where each extremity of each gene in G is contained in exactly one adjacency. One of the two gene extremi-

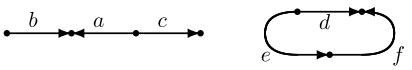
ties in an adjacency can be replaced by the *telomere marker* $\circ$, indicating the end of a linear chromosome. Such an adjacency is called a *telomere*.

Example 1 Consider the gene set $G = \{a, b, c, d, e, f\}$. Then the following set A is a genome for G:

$$A = \left\{\left\{\circ, b^t\right\}, \left\{b^h, a^h\right\}, \left\{a^t, c^t\right\}, \left\{c^h, \circ\right\}, \left\{e^t, d^t\right\}, \left\{d^h, f^h\right\}, \left\{f^t, e^h\right\}\right\}$$

A genome can be represented as a graph, called the *genome graph*, whose vertices are the adjacencies and whose edges connect for each gene the adjacency containing its head with the adjacency containing its tail. Clearly, each vertex of the genome graph has degree one or two, and therefore the connected components are either *cycles*, representing circular chromosomes, or *paths*, representing linear chromosomes.

Example 1 (cont'd) The genome graph of A looks as follows:



It is easy to see that A has two connected components, one of which is linear and the other one is circular.

We will also use a notation to represent the genome graph, in which a linear chromosome is written as the sequence of its genes from one of its telomeres to the other, where a gene is indicated by its name when it is read in tail-head direction, and by its overlined name when it is read in head-tail direction. A circular chromosome is represented similarly but, as it has no ends, spelling the genes can start anywhere, in any of the two possible directions, and all these representations are equivalent.

Example 1 (cont'd) In the linear notation, our genome looks as follows:

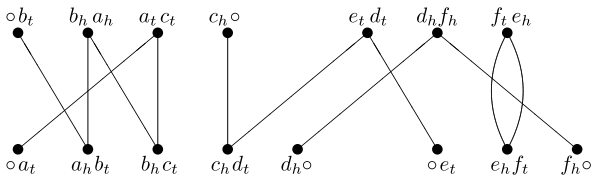
$$A = \left\{(\circ\ b\ \overline{a}\ c\ \circ)\ \ (d\ \overline{f}\ \overline{e})\right\}$$

Note that our genome model is very general in the sense that a genome can be a mix of circular and linear chromosomes. Other models have been considered, restricting genomes to contain only linear or only circular chromosomes. These constraints can be added at any time to the general model in order to reflect biological reality. However, as we will see in Sect. 5.3, rearrangement operations are independent of chromosome structure.

Another graph that will be very useful in the sequel is the *adjacency graph* for two genomes A and B containing the same genes. It will be the essential tool when calculating their rearrangement distance. The vertices of the adjacency graph are the adjacencies of the two genomes, and for each extremity of a gene from G we have an edge, connecting the two adjacencies (one from A and one from B) in which it is contained. Note that all vertices of the adjacency graph also have degree one or

two, thus its connected components are again paths or cycles. However, because the graph is bipartite, all cycles have even length.

Example 1 (cont'd) For A as above and genome $B = \{(\circ\ a\ b\ c\ d\ \circ)\ (\circ\ e\ f\ \circ)\}$, we have the following adjacency graph:



In the sequel we will consider various models of genome comparison, most of which realize some kind of edit distance, which in general can be phrased as follows.

Definition 1 (Genomic distance problem) Given two genomes A and B and a set of operations to manipulate them, what is the minimum number of operations to transform A into B?

If a corresponding sequence of operations realizing this number is actually reported, this is called the *genomic sorting problem*, but we will not discuss it further in this chapter.

5.3 An Illustrated Guide to the Double-Cut-and-Join Operation

In order to understand rearrangement operations it is necessary to have an idea of the mechanisms underlying them. When a double-stranded DNA sequence is broken, the cell is usually able to repair the damage by joining the two hanging ends together. However, as one wikipedian wrote in 2007 under the pseudonym *Amazinglarry* [9]:

> *Double-strand breaks, in which both strands in the double helix are severed, are particularly hazardous to the cell because they can lead to genome rearrangements.*

Indeed, when a genome breaks at two positions that are physically close, creating four hanging ends of double-strand DNA, the repair mechanisms may join the alternative ends together. This yields a deceptively simple definition of the Double-Cut-and-Join (DCJ) operation as: *the genome is cut in two places, and the pieces are joined in a different way*. This definition is correct, but it should be treated with the care deserved to informal definitions: the oriented nature of a double-stranded DNA sequence, and the fact that pieces may be lost or misplaced, introduce subtle constraints that need to be formalized.

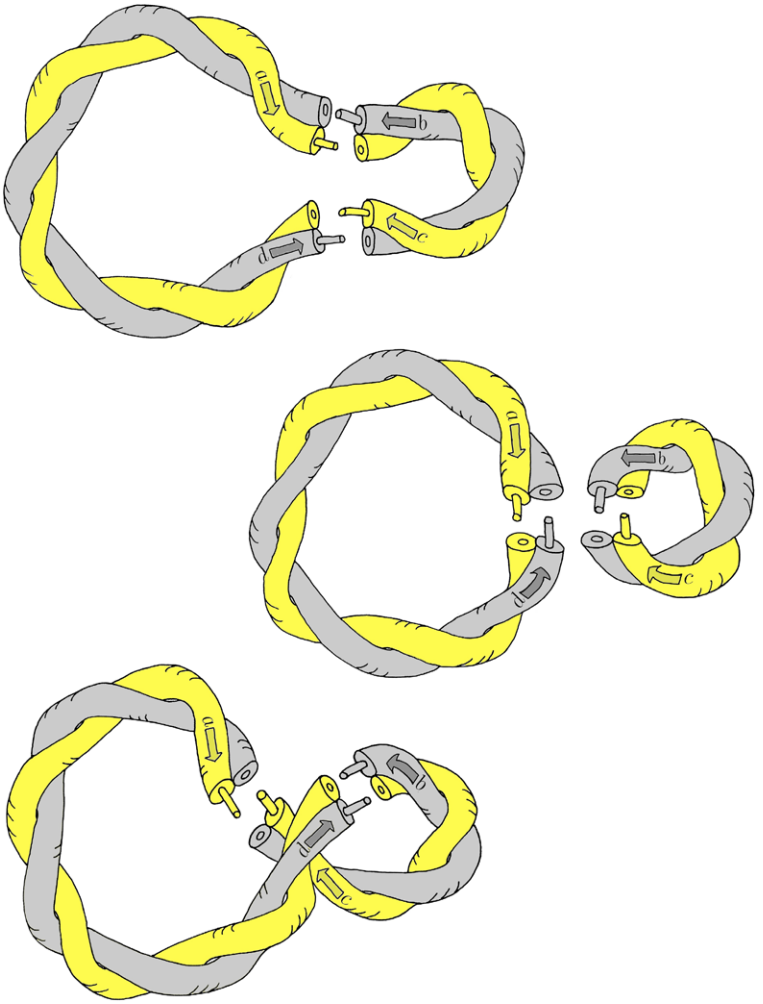


Fig. 5.1 The *top drawing* represents a genome with one double-stranded circular chromosome and four genes, a, b, c, and d. Genes are represented by *arrows*, and genes on opposite strands have opposite orientation. This genome can be represented as $(a\ \overline{b}\ c\ \overline{d})$. Suppose the chromosome is broken in two places, as illustrated, between genes a and b, and between genes c and d. The strands may be repaired in three different ways: the original arrangement $(a\ \overline{b}\ c\ \overline{d})$, the *middle* genome $(a\ \overline{d})\ (\overline{b}\ c)$ which is a *fission* of the top chromosome, and the *bottom* one $(a\ \overline{c}\ b\ \overline{d})$ that contains an *inversion* with respect to the original arrangement

The Basic DCJ Operation Let us begin with Fig. 5.1. The top genome is a circular chromosome broken at two physically close positions. Two genes are marked on each strand, a strand with genes a and c, and, in the opposite direction, a strand with genes b and d. Starting from gene a, and going around the chromosome, the

gene organization of this chromosome can be represented as

$$(a\ \overline{b}\ c\ \overline{d}).$$

The position of a break in the double-strand is described by the severed adjacency, which we say to be *cut*. In Fig. 5.1, the two adjacencies of the top genome:

$$\{a^h, b^h\} \quad \text{and} \quad \{c^h, d^h\}$$

are cut. If those two breaks are sufficiently close, the repair mechanisms, which have a very restricted understanding of the global situation, may *join* the a^h extremity with any of b^h, c^h, or d^h, whichever comes handy. The two remaining extremities are joined together, provided no pieces are lost. Thus there are three possible results of the repair, illustrated in Fig. 5.1:

(1) The original configuration, when b^h is chosen. The shape and gene order of the original genome are restored.
(2) An alternative configuration, when d^h is chosen. The circular chromosome is split in two circular chromosomes, in an event called a *fission*. The corresponding genome can be represented by $(a\ \overline{d})\ (\overline{b}\ c)$. The reverse event is called a *fusion*.
(3) An alternative configuration, when c^h is chosen. The chromosome is still circular, but the original strands are mixed: genes a and b are now on the same strand, opposite to genes c and d. This event is called an *inversion*. The new chromosome can be represented as $(a\ \overline{c}\ b\ \overline{d})$. Note the change in order and orientation of genes c and b with respect to the original genome.

This is it! The essence of the DCJ operation is contained in this example. The vast majority of identified rearrangement operations are based on this series of events, and the variations in terminology usually come from factors that are not directly related to the rearrangement operation itself.

A DCJ Within a Single Linear Chromosome Figure 5.2 is a reproduction of Fig. 5.1 in which the circular genome has been transformed into a linear genome by replacing a small segment of the double-stranded DNA with two telomeres. This modification has been done far from the breaks, and the rest of the picture is exactly the same. The genome organization would now be represented as

$$(\circ\ a\ \overline{b}\ c\ \overline{d}\ \circ),$$

to account for the new shape of the chromosome.

As in the circular case, the rearrangement operation between the top and bottom chromosome is called an *inversion*: the original strands are mixed, but the shape and gene content of the chromosome is the same. This type of rearrangements was first identified on fruit fly chromosomes, at the beginning of the last century [3], giving a founding example of rearranged genomes.

The middle genome of Fig. 5.2 is the only example in which linear and circular chromosomes are mixed. The DCJ operations that transform a linear chromosome

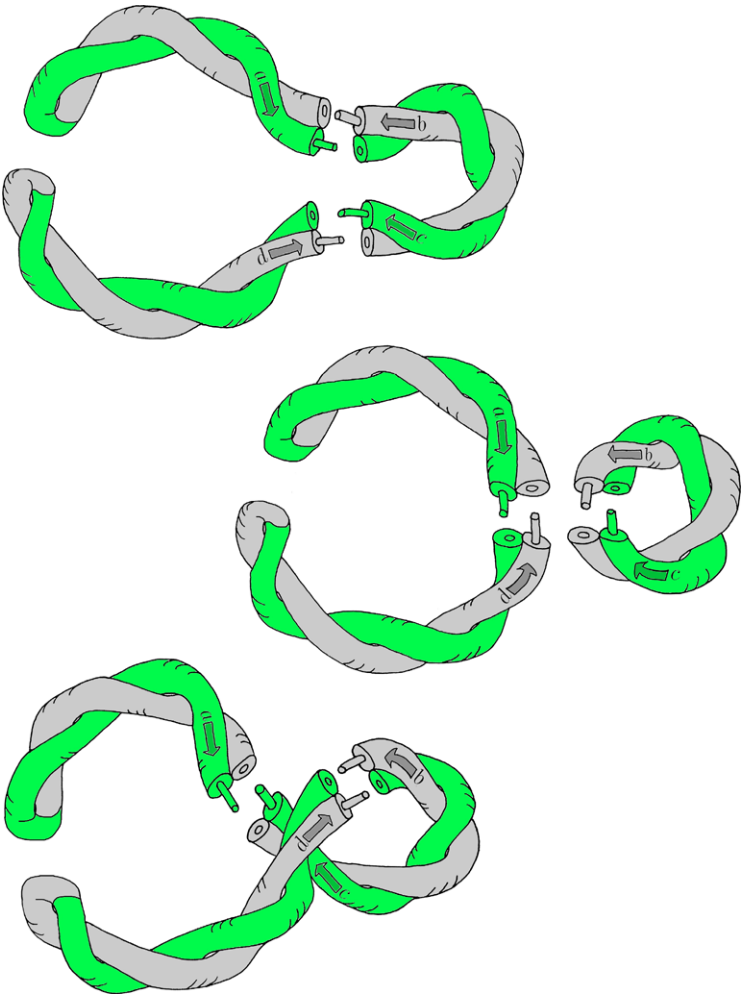


Fig. 5.2 In this figure, the *top drawing* represents a genome with one double-stranded linear chromosome. It was obtained by a slight modification of the genome in Fig. 5.1, consisting in removing a segment from the larger loop and capping the extremities with *telomeres*. This genome contains the same genes as the one in Fig. 5.1, and is now represented as $(\circ\ a\ \overline{b}\ c\ \overline{d}\ \circ)$ to account for the telomeres, but the breaks and repairs are exactly at the same positions. The *bottom* genome $(\circ\ a\ \overline{c}\ b\ \overline{d}\ \circ)$ contains an inversion with respect to the original arrangement. The *middle* genome has two chromosomes, one linear and one circular: $(\circ\ a\ \overline{d}\ \circ)\ (\overline{b}\ c)$

into such a genome is called a *circular excision*, and its reverse, a *reincorporation*. Many models of genome evolution explicitly forbid this type of rearrangement, arguing that, for example, the transformation of a genome consisting of linear chromosomes into a similar genome should not involve circular chromosomes. This is a quite natural requirement, but there is a tradeoff in the complexity of deriving the distance formula [6].

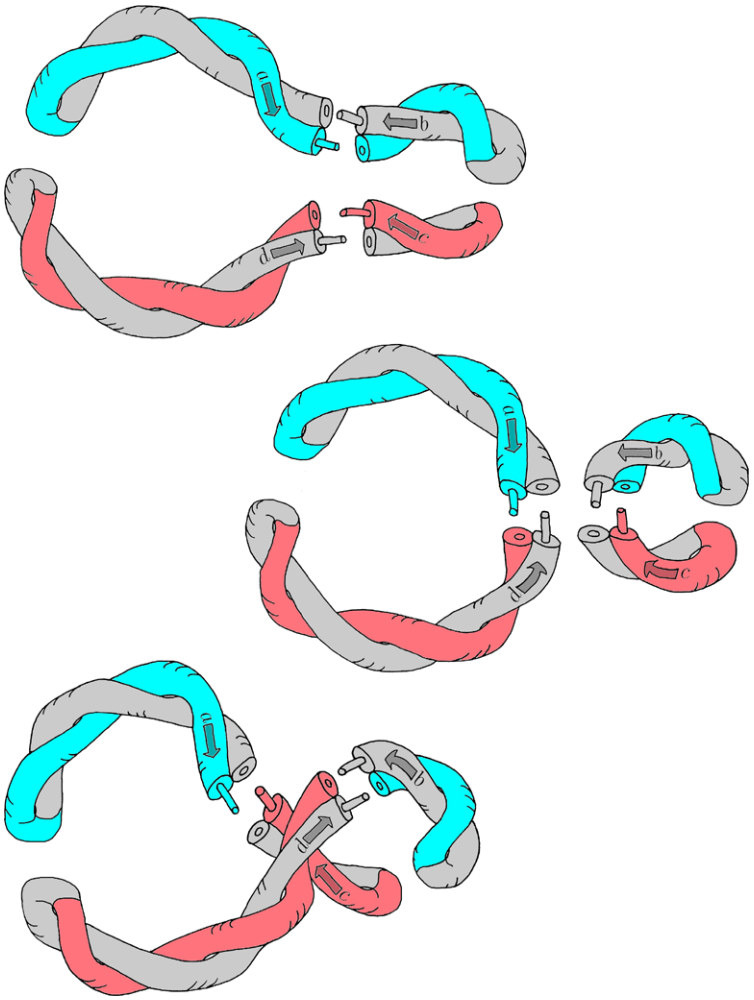


Fig. 5.3 This figure is another photoshopped version of Fig. 5.1 which produced two linear chromosomes out of the original circular ones. The two breaks and the gene labels are untouched. In this case, the operation that transforms one genome into any of the two others is called a *reciprocal translocation*. The *top* genome is represented by $(\circ\, a\, \overline{b}\, \circ)\ (\circ\, c\, \overline{d}\, \circ)$, the *middle* one by $(\circ\, a\, \overline{d}\, \circ)\ (\circ\, c\, \overline{b}\, \circ)$, and the *bottom* one by $(\circ\, a\, \overline{c}\, \circ)\ (\circ\, b\, \overline{d}\, \circ)$

A DCJ Between Two Linear Chromosomes In Fig. 5.3, telomeres are inserted in both ends of the circular genome of Fig. 5.1, resulting in a genome consisting of two linear chromosomes, represented by

$$(\circ\, a\, \overline{b}\, \circ)\ \ (\circ\, c\, \overline{d}\, \circ).$$

In this case, the DCJ operations are referred to as *reciprocal translocations*. Here again, the modifications have been done far from the breaks, and the rest of the

picture is the same, showing that the basic mechanics of DCJ cover a vast range of rearrangement operations.

Reciprocal translocations change the sets of genes associated with a particular chromosome, but do not modify the number of chromosomes in a genome. This can be annoying, since there are examples of really close species, with virtually the same set of genes, that have different number of chromosomes. This is the case, for example, with the human and chimpanzee genome, the latter having an extra chromosome. The DCJ model can be extended to cover this possibility as explained in the next paragraph.

Single Breaks and Lost Pieces As we have seen, the vast majority of genome rearrangements are caused by double breaks, but sometimes single breaks lead to genome modifications. With a single break, the repair mechanism usually restores the DNA strand but, in some rare instances, the break is never repaired. If this event occurs in a linear chromosome, we model the operation as

$$(\circ\, a \;\; b\, \circ) \longrightarrow (\circ\, a\, \circ)\, (\circ\, b\, \circ),$$

which is called a *fission*. When such an event occurs in a circular chromosome, it is called a *linearization*: the number of chromosomes is unchanged, but the genome is clearly modified. Despite involving only one break, these two operations are included in the DCJ model.

On the other hand, the reverse of these two operations, *fusion* of linear chromosomes and *circularization*, require two breaks and can be explained using the standard DCJ model and the loss of some hopefully redundant genetic material. Figure 5.3 contains many instances of fusions of chromosome segments belonging to different chromosomes: if all the necessary genetic information is contained in two fused segments, the remaining segments can be lost without consequences. As expected, we model this operation as the reverse of a fission:

$$(\circ\, a\, \circ)\, (\circ\, b\, \circ) \longrightarrow (\circ\, a \;\; b\, \circ),$$

where the telomere markers '$\circ$' stand in for the lost material. The circularization of a segment of a linear chromosome is central in Fig. 5.2: again, if all the necessary genetic information is contained in this circular segment, the two parts that contain telomeres can be lost.

These four rearrangement operations are often described as "standard" DCJ operations by introducing imaginary $\{\circ, \circ\}$ adjacencies: a DCJ operation applied to adjacencies $\{a^h, b^t\}$ and $\{\circ, \circ\}$ yields $\{a^h, \circ\}$ and $\{b^t, \circ\}$ and models fissions and linearizations. The reverse operation models fusions and circularizations.

5.4 Deriving the DCJ Formula

It was a clever observation by Sophia Yancopoulos that the DCJ operation subsumes the two operations that have most prominently been discussed in the genome rearrangement literature up to 2005: inversions and translocations. In their paper [10],

the authors also addressed the question of distance computation and gave the formula $D = b - c$ where $b := N - 1$ is the number of *initial breakpoints* between the N genes in the input genomes and c is a parameter closely related to the number of cycles in our adjacency graph. Nevertheless, their argument was rather informal. As said in the Introduction, it was our goal to formalize their approach and, if possible, simplify the argument and solution.

It was somehow clear to us that this should be possible, but even after almost a year of working on it, the exact way and the general DCJ formula still eluded our grasp. Therefore, in May 2005, we resorted to a 'dirty trick', based on just a few simple (and fortunately true) assumptions. Consider the following six parameters computed on the genomes and on the adjacency graph:

N: number of genes in each genome
C: number of cycles in the adjacency graph
I: number of odd paths in the adjacency graph
P: number of even paths in the adjacency graph
L: total number of linear chromosomes
R: total number of circular chromosomes

We begin with a well known mathematical technique called *guessing the solution*. Here, the educated guess is that the formula for the DCJ distance depends linearly on each of the above parameters, that is,

$$nN + cC + iI + pP + \ell L + rR = D,$$

where the coefficients n, c, i, p, ℓ and r are real numbers. Then we try to find the values of the coefficients.

These values are not necessarily independent. The most obvious relation is that the number L of linear chromosomes is related to the number of paths I and P by the equation

$$L = I + P.$$

This means that we can keep one of the coefficient as an arbitrary constant, which we choose to be ℓ in the sequel. We will wait until all the values of the other coefficients are known, and then choose a value of ℓ that will make the formula look 'simple'.

The next step is to consider a series of examples for which the DCJ distance is known. Each example will give a linear equation relating the values of the coefficients. The examples that we used in 2006 were lost in various paper baskets. This turns out to be a blessing, since recreating a suitable set of examples shows that only five very elementary examples are sufficient to determine the DCJ distance.

The Simplest Circular Chromosomes In these first two examples, we apply DCJ operations to a circular genome with two genes a and b. The graphs of Figs. 5.4 and 5.5 represent the two possible operations. Since a single DCJ has been applied in each case, the distance is $D = 1$.

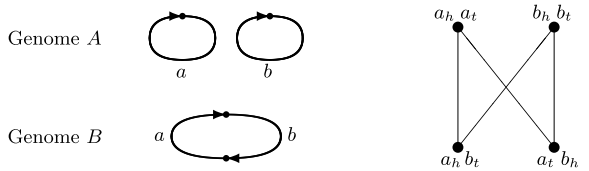


Fig. 5.4 A fusion of two circular chromosomes

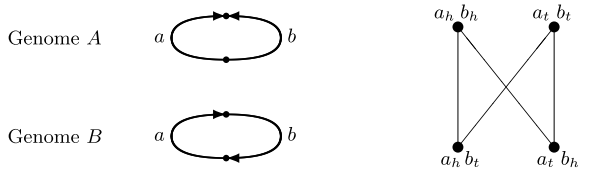


Fig. 5.5 An inversion within a circular chromosome

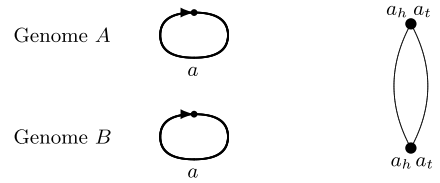


Fig. 5.6 Comparing a circular chromosome to itself

Example 2 Consider genomes $A = (a)\ (b)$ and $B = (a\ b)$, as in Fig. 5.4. The corresponding equation is

$$2n + c + 3r = 1.$$

Example 3 Consider genomes $A = (a\ \overline{b})$ and $B = (a\ b)$, as in Fig. 5.5, with the corresponding equation:

$$2n + c + 2r = 1.$$

From these equations we immediately conclude that $r = 0$, meaning that the distance is independent of the number of circular chromosomes. Setting r to its value yields the equation

$$2n + c = 1. \tag{5.1}$$

Equality with One Circular Chromosome The next equation is obtained by drawing the adjacency graph of a circular chromosome compared to itself. Obviously, the distance is $D = 0$ in this case, as in Fig. 5.6.

Example 4 Consider genomes $A = (a)$ and $B = (a)$. The corresponding equation is

$$n + c = 0. \tag{5.2}$$

Equations (5.1) and (5.2) yield $n = 1$ and $c = -1$.

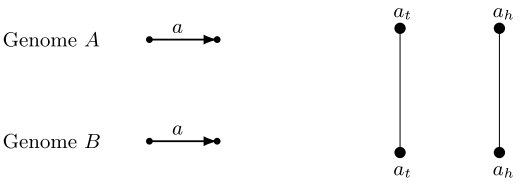


Fig. 5.7 Comparing a linear chromosome to itself

Equality with One Linear Chromosome When comparing a linear chromosome to itself, as in Fig. 5.7, we also get a distance of $D = 0$.

Example 5 Consider genomes $A = (\circ\ a\ \circ)$ and $B = (\circ\ a\ \circ)$. The corresponding equation is

$$n + 2i + 2\ell = 0. \tag{5.3}$$

Knowing that $n = 1$ and using ℓ as a constant, we get $i = -1/2 - \ell$.

Fusion/Fission of Linear Chromosomes The last example is given by the fusion of two linear chromosomes, and its dual operation, fission. In this case the distance is $D = 1$, and the adjacency graph is shown in Fig. 5.8.

Example 6 Consider genomes $A = (\circ\ a\ \circ)\ (\circ\ b\ \circ)$ and $B = (\circ\ a\ b\ \circ)$. The corresponding equation is

$$2n + 2i + p + 3\ell = 1, \tag{5.4}$$

which gives $p = -\ell$.

A Simple Formula Summing up the work thus far, we get the following family of distance formulas, indexed with ℓ:

$$D = N - C - (1/2 + \ell)I - \ell P + \ell L.$$

Clearly, the 'simplest' formula is obtained by setting $\ell = 0$.

The formula $D = N - C - I/2$ is simple in the sense that it has the fewest number of parameters. However, this simplicity does not impose an intrinsic value on the parameter I and we will also see, in the subsections of Sect. 5.6, that setting $\ell = 0$ does not always yield distance formulas with the least number of parameters.

5.5 Proving the DCJ Formula

Obtaining a formula based on a few examples does not mean that it computes the correct distance between arbitrary genomes: a general proof is still needed. In this section we discuss various topics associated with proving distance formulas, and we refer the reader to [2] for formal proofs.

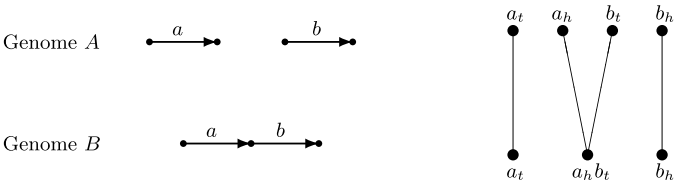


Fig. 5.8 Fusion/fission of linear chromosomes

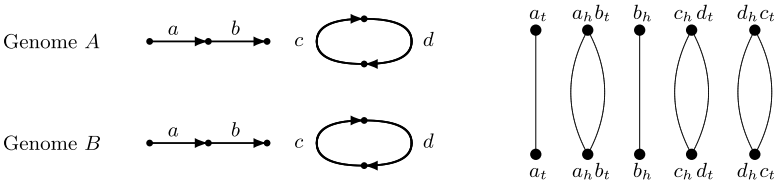


Fig. 5.9 The genome and adjacency graphs of two equal genomes

The formula $D = N - C - I/2$ of the preceding section is not only the simplest but, as a nice added benefit, provides a roadmap for the general proof. The first step is to prove that the distance between two genomes is 0 if and only if the two genomes are equal. Here, this statement translates as

Two genomes A and B are equal if and only if $N = C + I/2$.

The best informal justification of this result is to show the adjacency graph of two equal genomes with one circular and one linear chromosome, as in Fig. 5.9 with genomes $A = B = (a\ b)\ (\circ\ c\ d\ \circ)$. This figure also illustrates that the graph is a collection of cycles of length 2 and paths of length 1.

The next step in proving the distance formula is to show that $D \geq N - C - I/2$. This is done by considering the quantity $C + I/2$ in the adjacency graph of genomes A and B, and showing that it increases by at most 1 for *any* DCJ operation applied to genome A. A nice way to enumerate all the cases is to realize that a DCJ operation applied to genome A is also a DCJ operation applied to the adjacency graph. Thus, for example, the number of cycles can be increased either by extracting a cycle, or by creating a cycle from a path. In the first two cases, the number of paths is unchanged, and, in the third case, the length of the path must be even, since the lengths of all cycles of an adjacency graph are even. The reader is welcome to complete the details for the case of odd paths.

The final step in the proof is to show that $D \leq N - C - I/2$. The easiest way to prove this is to construct an algorithm that effectively sorts genome A into genome B in exactly $N - C - I/2$ steps, by showing that there always exists a DCJ operation on genome A that either increases the number of cycles by 1, or the number of odd paths by 2. In fact, *any* adjacency of genome B that is not an adjacency of genome A can be created in one DCJ operation on genome A: this operation creates a cycle of

length 2, and increases the number of cycles by 1. Once all adjacencies of genome B are created, the two genomes have the same adjacencies. If they are not equal, genome B has more telomeres than genome A, and a few fissions should solve the problem, each creating two paths of length 1.

5.6 Algebraic, Single-Cut and Breakpoint Distance Formulas

As mentioned in the Introduction, DCJ is just one of several rearrangement models by which genomes can be compared. Since many of the alternative models are closely related to DCJ, it is not surprising that their distance formulas are somewhat related as well. In this section we show in a systematic way how to derive the distance formulas for four other genome rearrangement models, using the techniques introduced in Sect. 5.4. In fact, we can re-use much of the derivation of the DCJ distance formula and only small modifications are necessary.

5.6.1 The Algebraic Distance

A line of research in genome rearrangement that has been introduced by Meidanis and Dias [7] uses algebraic operations acting on genomes represented as permutations. Several "traditional" results can similarly be derived in that formalism, and some new models have also been introduced, including the so-called *algebraic* (ALG) distance [5]. In its most general version including linear and circular chromosomes, it is identical to the DCJ distance, except that fissions and fusions weigh $1/2$ instead of 1.

Therefore, Eqs. (5.1), (5.2), and (5.3) are valid as well, the only difference is that Eq. (5.4) becomes

$$2n + 2i + p + 3\ell = 1/2.$$

The values of n, c and i are the same as in the DCJ model, but p becomes

$$p = -1/2 - \ell.$$

Thus the corresponding family of distance formulas is

$$D_{\mathrm{ALG}} = N - C - (1/2 + \ell)I - (1/2 + \ell)P + \ell L.$$

In this case, the 'simplest' formula is obtained by setting $\ell = -1/2$:

$$D_{\mathrm{ALG}} = N - C - L/2.$$

However, this formula mixes parameters from the genome graph, L, and from the adjacency graph, C. In Sect. 5.5, we saw that choosing both parameters from the adjacency graph gave us a big advantage in interpreting and proving the DCJ distance formula. It might be wise to do the same in this case.

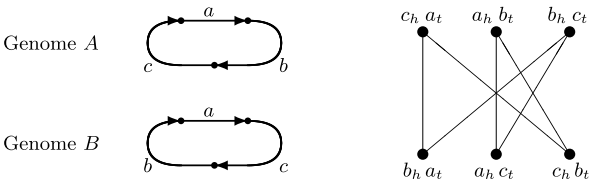


Fig. 5.10 A transposition within a circular chromosome

5.6.2 The Single-Cut-or-Join Distance

Another rearrangement distance, that is particularly charming because it allows efficient computational solutions to complicated multi-genome comparison, is the Single-Cut-or-Join (SCorJ) distance [4]. Here, any cut in a chromosome, and any join of two chromosome ends is considered as an individual operation, each of weight 1.

When deriving the SCorJ distance formula, we need one more pair of genomes in order to distinguish the roles of the long and short cycles, and we must rewrite the first four equations accordingly. The parameter C is split in two:

C_s: number of cycles of length 2
C_ℓ: number of long cycles

and the first four equations become

$$\begin{aligned} 2n + c_\ell &= 4 \\ n + c_s &= 0 \\ n + 2i + 2\ell &= 0 \\ 2n + 2i + p + 3\ell &= 1. \end{aligned}$$

The simplest rearrangement operation that gives the required new equation is a *transposition*, which exchanges two consecutive blocks of genes. In order to transpose blocks in a circular chromosome with Single-Cut-or-Join operations, it is necessary to cut three adjacencies, and join them at three different places.

Example 7 Consider the circular genomes $A = (a\ b\ c)$ and $B = (a\ c\ b)$, as in Fig. 5.10. The corresponding equation is

$$3n + c_\ell = 6. \tag{5.5}$$

The solution of the system is given by $n = 2$, $c_\ell = 0$, $c_s = -2$, $i = -1 - \ell$ and $p = -1 - \ell$, yielding the general formula

$$D_{\text{SCorJ}} = 2N - 2C_s - (1 + \ell)I - (1 + \ell)P + \ell L.$$

Here we would want to set $\ell = -1$, to get the 'simplest' distance formula:

$$D_{\text{SCorJ}} = 2N - 2C_s - L.$$

Again, here, this simplest formula might not be the wisest, since it mixes parameters from both the genome and the adjacency graphs.

5.6.3 The Single-Cut-and-Join Distance

Similar by name to SCorJ, but more closely related to the DCJ distance is the Single-Cut-and-Join (SCandJ) distance [1] where a single operation comprises at most one cut, followed by at most one join.

Again, we distinguish long and short cycles and get the following first four equations:

$$\begin{aligned} 2n + c_\ell &= 3 \\ n + c_s &= 0 \\ n + 2i + 2\ell &= 0 \\ 2n + 2i + p + 3\ell &= 1. \end{aligned}$$

The final equation can be derived from a transposition as in Example 7, which has distance[1] $D = 4$, yielding a fifth equation:

$$3n + c_\ell = 4.$$

Therefore the solution is $n = 1$, $c_\ell = 1$, $c_s = -1$, $2i + 2\ell = -1$, $2i + p + 3\ell = -1$, and we get the general formula:

$$D_{\text{SCandJ}} = N + C_\ell - C_s - (1/2 + \ell)I - \ell P + \ell L.$$

Setting $\ell = 0$, we find

$$D_{\text{SCandJ}} = N + C_\ell - C_s - I/2.$$

5.6.4 The Breakpoint Distance

Finally we consider the *breakpoint* (BRK) distance, which is possibly the most classical and simplest genomic distance. Different from the previous ones, the breakpoint distance is not an edit distance, but just defined as the number of adjacencies that are present in one, but not in the other of the two input genomes [8].

The breakpoint distance needs to distinguish the odd paths of length 1, that are shared telomeres between genomes, from the longer odd paths. Thus we need two more coefficients:

[1] Since Single-Cut-and-Join operations are sometimes less intuitive, here is a scenario that sorts genome $(a\ b\ c)$ to genome $(a\ c\ b)$ in four steps. Cuts are indicated by vertical bars: $(a\ b\ c\ |) \longrightarrow (\circ\ a\ |\ b\ c\ \circ) \longrightarrow (\circ\ a\ \circ)\ (b\ |\ c) \longrightarrow (\circ\ a\ c\ b\ \circ) \longrightarrow (a\ c\ b)$.

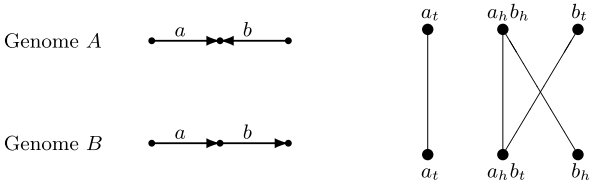


Fig. 5.11 An inversion inside a linear chromosome

I_s: number of paths of length 1
I_ℓ: number of long odd paths

We also need a revised set of equations, reflecting the new values on our pet examples for the breakpoint distance:

$$\begin{aligned} 2n + c_\ell &= 2 \\ n + c_s &= 0 \\ n + 2i_s + 2\ell &= 0 \\ 2n + 2i_s + p + 3\ell &= 1 \\ 3n + c_\ell &= 3. \end{aligned}$$

Finally, we need a last example to distinguish the roles of the short and long odd paths. The simplest one is the linear variant of Example 2.

Example 8 Consider genomes $A = (\circ\, a\, \overline{b}\, \circ)$ and $B = (\circ\, a\, b\, \circ)$ with the corresponding equation

$$2n + i_s + i_\ell + 2l = 3/2, \tag{5.6}$$

as in Fig. 5.11. The value 3/2 comes from the definition of the general breakpoint distance for mixed multichromosomal genomes given in [8]. The solution is given by $n = 1, c_\ell = 0, c_s = -1, i_s = -1/2 - \ell, i_\ell = -\ell, p = -\ell$ with the general formula:

$$D_{\text{BRK}} = N - C_s - (1/2 + \ell)I_s - \ell I_\ell - \ell P + \ell L.$$

Setting $\ell = 0$, as in the DCJ distance, we get the breakpoint distance formula of David Sankoff and co-authors [8]:

$$D_{\text{BRK}} = N - C_s - I_s/2.$$

5.7 Conclusion

In this paper, we showed that many standard formulas for computing the rearrangement distance between genomes can be obtained using a handful of very simple genomes and a little linear algebra. Table 5.1 summarizes the principal results. It

Table 5.1 Distance parameters for various genomes and rearrangement models

Genomes	Equation	DCJ	ALG	SCorJ	SCandJ	BRK
$(a)\ (b)$ & $(a\ b)$	$2n + c_\ell$	1	1	4	3	2
(a) & (a)	$n + c_s$	0	0	0	0	0
$(\circ\ a\ \circ)$ & $(\circ\ a\ \circ)$	$n + 2i_s + 2\ell$	0	0	0	0	0
$(\circ\ a\ \circ)\ (\circ\ b\ \circ)$ & $(\circ\ a\ b\ \circ)$	$2n + 2i_s + p + 3\ell$	1	1/2	1	1	1
$(a\ b\ c)$ & $(a\ c\ b)$	$3n + c_\ell$	2	2	6	4	3
$(\circ\ a\ \overline{b}\ \circ)$ & $(\circ\ a\ b\ \circ)$	$2n + i_s + i_\ell + 2\ell$	1	1	2	4	3/2

Parameter	Coefficient	DCJ	ALG	SCorJ	SCandJ	BRK
Genes	n	1	1	2	1	1
Long cycles	c_ℓ	-1	-1	0	1	0
Short cycles	c_s	-1	-1	-2	-1	-1
Long odd paths	i_ℓ	$-1/2-\ell$	$-1/2-\ell$	$-1-\ell$	0	$-\ell$
Short odd paths	i_s	$-1/2-\ell$	$-1/2-\ell$	$-1-\ell$	0	$-1/2-\ell$
Even paths	p	$-\ell$	$-1/2-\ell$	$-2-2\ell$	$-\ell$	-2ℓ
Linear chromosomes	ℓ	ℓ	ℓ	ℓ	ℓ	ℓ

should be noted, though, that the formulas must be treated as conjectures. As for the DCJ formula, independent correctness proofs are needed, and are available in the literature.

These distance formulas are similar, in the sense that they all depend linearly on the same set of parameters. However, in this setting, each model yields an infinite number of formulas: finding the simplest, or one that depends on given parameters, is just a mathematical trick. The important point is that the parameters should be chosen for their practical implications on the formulas, such as ease of interpretation or propensity toward elegant proofs.

References

1. Bergeron, A., Medvedev, P., Stoye, J.: Rearrangement models and single-cut operations. J. Comput. Biol. **17**(9), 1213–1225 (2010)
2. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proceedings of WABI 2006. LNBI, vol. 4175, pp. 163–173 (2006)
3. Dobzhansky, T., Sturtevant, A.H.: Inversions in the chromosomes of drosophila pseudoobscura. Genetics **23**(1), 28–64 (1938)
4. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–1329 (2011)
5. Feijão, P., Meidanis, J.: Extending the algebraic formalism for genome rearrangements to include linear chromosomes. In: Proceedings of BSB 2012. LNBI, vol. 7409, pp. 13–24 (2012)
6. Hannenhalli, S., Pevzner, P.A.: Transforming men into mice (polynomial algorithm for genomic distance problem). In: Proceedings of FOCS 1995, pp. 581–592 (1995)

7. Meidanis, J., Dias, Z.: An alternative algebraic formalism for genome rearrangements. In: Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families, pp. 213–223. Kluwer Academic, Dordrecht (2000)
8. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinform. **10**, 120 (2009)
9. Wikipedia: http://en.wikipedia.org/wiki/DNA_repair
10. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)

Part II
New Lights on Current Paradigms

Chapter 6
Large-Scale Multiple Sequence Alignment and Phylogeny Estimation

Tandy Warnow

Abstract With the advent of next generation sequencing technologies, alignment and phylogeny estimation of datasets with thousands of sequences is being attempted. To address these challenges, new algorithmic approaches have been developed that have been able to provide substantial improvements over standard methods. This paper focuses on new approaches for ultra-large tree estimation, including methods for co-estimation of alignments and trees, estimating trees without needing a full sequence alignment, and phylogenetic placement. While the main focus is on methods with empirical performance advantages, we also discuss the theoretical guarantees of methods under Markov models of evolution. Finally, we include a discussion of the future of large-scale phylogenetic analysis.

6.1 Introduction

Evolution is a unifying principle for biology, as has been noted by Dobzhansky, de Chardin, and others.[1] Phylogenies are mathematical models of evolution, and therefore enable insights into the evolutionary relationships between organisms, genes, and even networks. Indeed, phylogeny estimation is a major part of much biological research, including the inference of protein structure and function, of trait evolution, etc. [3].

Phylogeny estimation from molecular sequences generally operates as follows: first the sequences are aligned through the insertion of spaces between letters (nucleotides or amino-acids) in the sequences, and then a tree is estimated using the alignment. This "two-phase" approach to phylogeny estimation can produce highly accurate estimations of the tree for small to moderate-sized datasets that are fairly closely related. However, datasets that contain sequences that are quite different

[1]The famous quote by Dobzhansky "Nothing in biology makes sense except in the light of evolution" [1] reflects the less known quote by the Jesuit priest Pierre Teilhard de Chardin [2], who wrote "Evolution is a light which illuminates all facts, a curve that all lines must follow."

T. Warnow (✉)
Department of Computer Science, University of Texas at Austin, Austin TX, USA
e-mail: tandy@cs.utexas.edu

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_6,

from each other (especially if the datasets are very large), can be very difficult to align—and even considered "un-alignable", and trees based on poor alignments can have high error [4–11].

Although the estimation of both alignments and phylogenies is challenging for large, highly divergent datasets, there is substantial evidence that phylogenetic analyses of large datasets may result in more accurate estimations of evolutionary histories, due to improved taxonomic sampling [12–15]. Thus, although not all datasets will be improved through the addition of taxa, some phylogenetic questions—particularly the inference of deep evolutionary events—seem likely to require large datasets.

In this chapter, we discuss the challenges involved in estimating large alignments and phylogenies, and present some of the new approaches for large alignment and tree estimation. Thus, this chapter does not attempt to survey alignment estimation methods or tree estimation methods, each of which is an enormous task and discussed in depth elsewhere; see [16–20] for phylogeny estimation and [11, 21–27] for alignment estimation.

We begin with the basics of alignment and phylogeny estimation in Sect. 6.2, including Markov models of sequence evolution and statistical performance criteria; this section also discusses the class of "absolute fast converging methods" and presents one of these methods. Section 6.3 discusses some methods that co-estimate alignments and trees (rather than operating in two phases). Section 6.4 presents methods that estimate trees without needing a full multiple sequence alignment. We close with a discussion about the future of large-scale alignment and phylogenetic tree estimation in Sect. 6.5.

6.2 Two-Phase Alignment and Phylogeny Estimation

This section contains the basic material for this book chapter. We begin with a description of a phylogenomic pipeline (estimating the species history from a set of genes), and discuss the issues involved in resolving incongruence between different gene trees. We then discuss general issues for multiple sequence alignment estimation and evaluation, including how alignments can be used for different purposes, and hence evaluation metrics can differ. We describe certain algorithmic techniques for multiple sequence alignment that have been used to enable large-scale analyses, including template-based methods, divide-and-conquer, and progressive alignment, and we discuss some alignment methods that have been used on very large datasets.

We then turn to tree estimation, beginning with stochastic models of sequence evolution and statistical performance criteria for phylogeny estimation methods. We provide a brief background in the theoretical guarantees of different phylogeny estimation methods (e.g., which methods are statistically consistent under the basic sequence evolution models and the sequence-length requirements of methods), as well as some discussion about their empirical performance on large datasets. We discuss gap treatment methods and their performance guarantees, and the empiri-

Table 6.1 Table of methods for large-scale multiple sequence alignment estimation. We show methods that have published results on datasets with at least 25,000 sequences, showing the type of data (DNA, RNA, amino-acid, or all), the largest number of sequences in published dataset analyses, publications for the method, and techniques used. The methods listed in the table for co-estimation of alignments and trees can also be considered as alignment estimation methods, but are not listed here. Finally, the largest dataset size we note here for each method may not be the largest performed, but is the largest we were able to find and document. It is also possible that there are publications we are not aware of that present analyses of datasets of this size using methods not listed here

Alignment method	Data	Largest dataset	Publications	Techniques
`MAFFT-PartTree`	All	93,681 [28]	[29]	Progressive
`Clustal-Quicktree`	All	27,643 [30]	[31]	Progressive
`Kalign-2`	All	50,175 [28]	[32]	Progressive
`Clustal-Omega`	Amino-acid	93,681 [28]	[28]	Progressive HMMs
`Neuwald's method`	Amino-acid	400,000 (approx.) [33]	[33]	Template

Table 6.2 Table of methods for large-scale phylogeny estimation. We show methods that have published results on datasets with at least 25,000 sequences, showing the type of data (DNA, RNA, amino-acid, or all), the largest number of sequences in published dataset analyses, publications for the method, and techniques used. We do not show results for distance-based methods, although these tend to be able to run (efficiently) on very large datasets. With the exception of DACTAL, these methods require an input alignment. The methods listed in the table for co-estimation of alignments and trees can also be considered phylogeny estimation methods, but are not listed here. Finally, the largest dataset size we note here for each method may not be the largest performed, but is the largest we were able to find and document. It is also possible that there are publications we are not aware of that present analyses of datasets of this size using methods not listed here

Phylogeny method	Data	Largest dataset	Publications	Techniques
`FastTree-2`	All	1.06 million (approx.)	[34]	Maximum likelihood
`RAxML`	All	55,473[35]	[36]	Maximum likelihood
`TNT`	All	73,060 [37]	[38]	Maximum parsimony
`DACTAL` (almost alignment-free)	All	27,643 [30]	[30]	Iteration divide-and-conquer supertree

cal impact of these methods on phylogenetic analyses. We then discuss the analysis of datasets that contain short sequences (i.e., fragments of full-length sequences), including phylogenetic placement methods that insert short sequences into pre-computed trees, and how these methods can be used in metagenomic analysis.

Although this chapter provides some discussion about many methods—both for alignment estimation and phylogeny estimation—the focus is on those methods that can analyze large datasets. The main effort, therefore, is to describe just a few methods, and to try to identify the algorithmic techniques that make them able to analyze large datasets.

Table 6.3 Table of methods for large-scale co-estimation of phylogenies and alignments. We show methods that have published results on datasets with at least 25,000 sequences, showing the type of data (DNA, RNA, amino-acid, or all), the largest number of sequences in published dataset analyses, publications for the method, and techniques used. Finally, the largest dataset size we note here for each method may not be the largest performed, but is the largest we were able to find and document. It is also possible that there are publications we are not aware of that present analyses of datasets of this size using methods not listed here

Phylogeny method	Data	Largest dataset	Publications	Techniques
`SATé` co-estimates alignments and trees	All	27,643 [39]	[10, 39]	Iteration, progressive divide-and-conquer maximum likelihood
`Mega-phylogeny` co-estimates alignments and trees	All	55,473 [35]	[35]	Divide-and-conquer maximum likelihood

6.2.1 Standard Phylogenomic Analysis Pipelines

The focus of this paper is on the estimation of alignments and trees for single genes, which normally follows a two-phase process: first the sequences are aligned, and then a tree is estimated on the alignment. However, a description of how a species tree is estimated can help put these methods into a larger context.

Because gene trees can differ from species trees due to biological causes (such as incomplete lineage sorting, gene duplication and loss, and horizontal gene transfer [40]), species tree estimations are based on multiple genes rather than any single gene. At the simplest level, this can involve just a handful of genes, but increasingly "phylogenomic" analyses (involving genes from throughout the genome) are being performed [41–44], followed by biological discoveries based on these phylogenomic analyses [45]. The following approaches are the dominant methods used to estimate species trees from multiple genes:

1. Markers are selected, and homologous regions within the genomes are identified across the species; these homologous regions are sometimes limited to the orthologous parts, so that gene duplication and loss does not need to be considered in estimating the species history.
2. Multiple sequence alignments are estimated on each marker.
3. At this point, the standard analysis pipeline continues in one of the following ways:
   (a) the gene sequence alignments can be concatenated, and a tree estimated on the "supermatrix",
   (b) gene trees can be estimated, and then combined together into a species tree using supertree methods [46–54] or methods that explicitly take biological causes (e.g., incomplete lineage sorting and gene duplication and loss) for gene tree incongruence into account [40, 42, 55–70], or
   (c) the species tree can be estimated directly from the set of sequence alignments, taking biological causes for gene tree incongruence into account (an

example is *BEAST, which co-estimates the gene trees and species tree directly from the input set of alignments [71]).

The first approach is distinctly different in flavor from the last two approaches, and is called the "supermatrix" or "combined analysis" approach. The relative merits of these approaches with respect to accuracy are debated, but for statistical reasons, it makes sense to use methods that consider biological causes for incongruence when estimating species trees from multiple markers. The development and understanding of methods for estimating species trees given multiple genes, taking incomplete lineage sorting (ILS) and gene duplication and loss into account, is an active research area; see, for example, papers on this subject in the session on Phylogenomics and Population Genomics at the 2013 Pacific Symposium on Biocomputing [72–75].

6.2.2 Multiple Sequence Alignment

Introduction Alignment methods vary in type of data (DNA, RNA, or amino-acid) they can handle, and also, to some extent, the objectives of the alignment method. Thus, some methods are designed exclusively for proteins [33, 76–81], some exclusively for RNAs [82–87], but many alignment methods can analyze both protein and nucleotide datasets. We refer to methods that can analyze all types of molecular sequences as "generic" methods.

Alignment methods are used to predict function and structure, to determine whether a sequence belongs to a particular gene family or superfamily, to recognize homology in the 'twilight zone' (where sequence similarity is so low that homology is difficult to detect), to infer selection, etc. [11]. On the other hand, alignment methods are also used in order to estimate a phylogeny. As we shall see, the design of alignment methods and how they are evaluated depend on the purpose they are being used for.

MSA Evaluation Criteria The standard criteria used to evaluate alignments for accuracy are based on shared homologies between the true and the estimated alignment, with the SP-score [88, 89] (sum-of-pairs score) measuring the fraction of the true pairwise homologies correctly recovered, and the TC-score ("total column" score) measuring the number of identical columns. Variants on these criteria include the true metrics suggested by Blackburne et al. [90] and the consideration of different types of alignment error (i.e., both false positive and false negative) rather than one overall measure of "alignment accuracy" [89]. Other criteria, such as the identification and correct alignment of specific regions within a protein or rRNA, have also been used [33, 91]. Furthermore, because sequence alignment has the potential to impact phylogeny estimation, a third way of evaluating a multiple sequence alignment method is via its impact on phylogeny estimation [11].

Thus, there are at least three different ways of assessing alignment accuracy: the first type uses standard criteria and their variants (e.g., SP, TC, Cline Shift scores, and the methods suggested in [89, 90]) that focus on shared homologies and treat

them all identically; the second type reports accuracy with respect to only those sites with functional or structural significance; and the third type focuses on phylogenetic accuracy. These criteria are clearly related, but improved performance with respect to one criterion may not imply improved performance with respect to another! An example of this is given in [10], where some estimated alignments differed substantially in terms of their SP-scores, and yet maximum likelihood trees on these alignments had the same accuracy. Similarly, when the objective is protein structure and function prediction, mistakes in alignments that are not structurally or functionally important may not impact these predictions, and so two alignments could yield the same inferences for protein structure and function and yet have very different scores with respect to standard alignment evaluation criteria. Furthermore, the algorithmic techniques that lead to improved results for one purpose may not lead to improved performance for another, and benchmark datasets used to evaluate methods may also not be identical.

Benchmark Datasets Many studies (see [21, 24, 27, 87, 92, 93] for examples) have evaluated MSA methods using biological data for which structurally informed alignments are available. The best known of these benchmark datasets is probably BAliBASE [94], but others are also used [95–98]. The choice of benchmarks and how they are used has a large impact on the result of the evaluation, and so has been discussed in several papers [24, 78, 99–101].

However, the use of structurally defined benchmarks has also been criticized [11, 100, 102] as being inappropriate for evaluating alignments whose purpose is phylogenetic estimation. The main criticism is the observation that structural or functional "homology" and "positional homology" (whereby two residues are positionally homologous if and only if they descend from a residue in their common ancestor by substitutions alone [103]) are different concepts, and that molecules with residues that are functionally or structurally homologous due to convergent evolution without being positionally homologous have been found [11, 102, 104]. Thus, alignments that are correct with respect to functional or structural homology may be incorrect with respect to positional homology, and therefore violate the assumptions used in phylogenetic estimation. However, even reputed benchmark alignments have errors, as discussed in [11, 100], making the use of these benchmarks even for detecting structural motifs questionable in some cases.

The use of benchmarks in general, and specifically structural benchmarks, is discussed at length by Iantoro et al. [100]. From the perspective of phylogeny estimation, one of the most important of their observations is that structurally defined benchmarks often omit the highly variable parts of the molecule, including introns. Thus, an alignment can be considered completely correct as a structural alignment if it aligns the conserved regions, even if it fails to correctly align the variable regions. The problem with this criterion (as they point out) is that the highly variable portions are often the sites that are of most use for phylogeny estimation, whereas sites that change slowly have little phylogenetic signal.

An obvious response to the concerns about the potential disagreement between positional homology and structural homology is that while they two concepts are

not identical, structural features tend to change slowly and so there is a close relationship between the two concepts [105]. Thus, for many datasets, and perhaps even most, these definitions may be identical, and so the use of structural benchmarks is acceptable (and advisable) for most cases. However, the criticism raise by Iantoro et al. regarding the elimination of the highly variable regions in the benchmark is more difficult to counter, except, perhaps, by saying that correct structural alignments of the variable regions are much more difficult to establish.

This is one of the reasons that simulated data are also used to evaluate MSA methods: the true alignment is known with certainty, including the alignment of the hyper-variable regions. Another advantage of simulated datasets is that they enable the exploration of a larger range of conditions, whereas only a few biological datasets are used as alignment benchmarks. Finally, simulated datasets, when simulated on model trees under an evolutionary process, also provide a true tree to which estimated trees can be compared. Thus, when the purpose of the alignment is to estimate the phylogeny, simulations of sequence evolution down model trees present definite advantages over structural benchmarks, and have become the standard technique for evaluating alignments.

Relative Performance of MSA Methods While most studies have evaluated alignment methods in terms of standard criteria (notably, SP and TC scores) on biological benchmarks, some studies have explored alignment estimation for phylogeny estimation purposes. As commented on earlier, many studies have shown that alignment estimation impacts phylogenetic estimation, and that alignment and tree error increase with the rate of evolution. Also, on very large datasets, due to computational limitations, only a few alignment methods can even be run (and typically not the most accurate ones), which results in increased alignment error [6]. On the other hand, on large trees with rates of evolution that are sufficiently low, alignment estimation methods can differ substantially in terms of SP-score without impacting the accuracy of the phylogenetic tree estimated using the alignment [10]. More generally, standard alignment metrics may be only poorly correlated with tree accuracy in some conditions.

Not all alignment methods have been tested for their impact on phylogenetic accuracy; however, among those that have been tested, MAFFT [106] (when run in its most accurate settings) is among the best performing methods [4, 6, 10, 39] on both proteins and nucleotides, especially on datasets with many sequences. For small nucleotide datasets, especially those with relatively low rates of evolution, other methods (e.g., Probcons [107] and Prank [108]) can give excellent results [109, 110].

Progressive Aligners, and the Impact of Guide Trees Many alignment methods use progressive alignment on a guide tree to estimate the alignment; thus the choice of the guide tree and its impact on alignment and phylogeny estimation is also of interest [109, 111–113]. Nelesen et al. [109] studied the impact of the guide tree on alignment methods, and showed that improved phylogenetic accuracy can be obtained by first estimating a tree from the input using a good two-phase method (RAxML [36] on a MAFFT alignment). They noted particular benefits in using

Probcons with this guide tree, and called the resultant method "Probtree". Prank has also been observed to be very sensitive to guide trees [113], and to give improved results by the use of carefully computed guide trees (maximum likelihood on good alignments) [10, 112]. Another study showed that even when the alignment score does not change, the alignment itself can change in important ways when guide trees are changed [111]. Finally, Capela-Gutierrez and Gabaldon [113] found that the placement of gaps in an alignment results from the choice of the guide tree, and hence the gaps are *not* phylogenetically informative. Based on these observations, Capela-Gutierrez and Gabaldon recommended that alignment estimation methods should use the true tree (if possible), or else use an iterative co-estimation method that infers both the tree and the alignment.

Template-Based Methods Some alignment methods use a very different type of algorithmic structure, which is referred to as being "template-based" [24]. Instead of using progressive alignment on a guide tree, these methods use models (either profiles, templates, or Hidden Markov Models) for the gene of interest, and align each sequence to the model in order to produce the final multiple sequence alignment, as follows. First, a relatively small set of sequences from the family is assembled, and an alignment estimated for the set. Then, some kind of model (e.g., a template or a Hidden Markov Model) is constructed from this "seed" alignment. This model can be relatively simple or quite complex, typically depending on whether the model provides structural information. Once the model is estimated, the remaining sequences are added to the growing alignment. The model is used to align each sequence to the seed alignment (which does not change during the process), and then inserted into the growing alignment. Since the remaining sequences are only compared to the seed alignment, homologies between the remaining sequences can only be inferred through their homologies to the seed alignment. Thus, the choice of sequences in the seed alignment and how it is estimated can have a big impact on the resultant alignment accuracy. By design, once the seed alignment and the model are computed, the running time scales linearly with the number of sequences, and the algorithm is trivially parallelizable. Thus, these methods, which we will refer to jointly as "template-based methods", can scale to very large datasets with hundreds of thousands of sequences.

There are several examples of methods that use this approach [33, 85–87, 114–116] (see pp. 526–529 in [11]). Some of these methods use curated seed alignments based on structure and function of well-characterized proteins or rRNAs; for example, the protein alignment method by Neuwald [33] and the rRNA sequence alignment method by Gardner et al. [87] use curated alignments. Constraint-based methods, such as COBALT [117], 3DCoffee [79] and PROMALS [76], similarly use external information like structure and function, but then use progressive alignment techniques (or other such methods) to produce the final alignment. Clustal-Omega also has a version, called "External Profile Alignment", which uses external information (in the form of alignments) to improve the alignment step.

Finally, PAGAN [116] is another member of this class of methods; however, it has some specific methodological differences to the others. First, unlike several of

the others, it does not use external biological information (about structure, function, etc.) to define its seed alignment. Second, while the others tend to use either HMMs, profiles, or templates as a model to define the alignment of the remaining sequences, PAGAN estimates a tree on its seed alignment, and estimates sequences for the internal nodes. These sequences are then used to define the incorporation of the remaining sequences to the seed alignment. This technique is very similar to the technique used in PaPaRa [118], which was developed for the phylogenetic placement problem (see Sect. 6.2.4). Thus, PAGAN is one of the "phylogeny-aware" alignment methods, a technique that is atypical of these template-based methods, but shared by progressive aligners. PAGAN was compared to an HMM-based method (using HMMER on the reference alignment to build an HMM, and then using HMMALIGN to align the sequences to the HMM) on several datasets [116]. The comparison showed that PAGAN had very good accuracy, better than HMMALIGN, under low rates of evolution, and that both methods had reduced accuracy under high rates of evolution. They also noted that PAGAN failed to align some sequences under model conditions with high rates of evolution, while HMMER aligned all sequences; however, the sequences that both HMMER and PAGAN aligned were aligned more accurately using PAGAN.

Several studies [21, 33, 76, 80, 81, 87, 119] have shown that alignment methods that use high quality external knowledge can surpass the accuracy of some of the best purely sequence-based alignment methods. However, none of these template-based and constraint-based alignment estimation methods (whether or not based upon external biological knowledge) have been tested for their impact on phylogenetic estimation; instead, they have only been tested with respect to standard alignment criteria (e.g., SP-score), identification of functional or structural residues, or membership in a gene family. Thus, we do not know whether the improvements obtained with respect to traditional alignment accuracy metrics will translate to improvements in phylogeny estimation.

Methods that Use Divide-and-Conquer on the Taxon Set Some alignment methods use a divide-and-conquer strategy in which the taxon set is divided into subsets (rather than the sites) in order to estimate the alignment; these include the mega-phylogeny method developed by Smith et al. [120], SATé [10, 39], SATCHMO-JS [78], PROMALS [76], and the method by Neuwald [33]. (The SATé and SATCHMO-JS methods co-estimate alignments and trees, and so are not strictly speaking just alignment methods.) Neuwald's method is a bit of an outlier in this set, because the user provides the dataset decomposition, but we include it here for comparative purposes.

While the methods differ in some details, they use similar strategies to estimate alignments. Most estimate an initial tree, and then use the tree to divide the dataset into subsets. The method to compute the initial trees differs, with SATCHMO-JS using a neighbor joining [121] (NJ) tree on a MAFFT alignment, SATé using a maximum likelihood tree on a MAFFT alignment, PROMALS using a UPGMA tree on k-mer distances, and mega-phylogeny using a reference tree and estimated alignment. (See the description of mega-phylogeny provided by Roquet et al. [122] for more details.)

The subsequent division into subsets is performed in two ways. In the case of mega-phylogeny, SATCHMO-JS, and PROMALS, the division into subsets is performed by breaking the starting tree into clades so as to limit the maximum dissimilarity between pairs of sequences in each set. In contrast, SATé-2 [39] removes centroid edges from the unrooted tree, recursively, until each subset is small enough (below 200 sequences). Thus, the sets produced by the SATé-2 decomposition do not form clades in the tree, unlike the other decompositions. Furthermore, the sets produced by the SATé-2 decomposition are guaranteed to be small (at most 200 taxa) but are not constrained to have low pairwise dissimilarities between sequences.

Alignments are then produced on each subset, with PROMALS, SATé, and mega-phylogeny estimating alignments on each subset, and SATCHMO-JS using the alignment induced on the subset by the initial MAFFT alignment.

These alignments are then merged together into an alignment on the full set, but the methods use different techniques. PROMALS and mega-phylogeny use template-based methods to merge the alignments together, while SATCHMO-JS and SATé use progressive alignment techniques. PROMALS also uses external knowledge about protein structure to guide the template-based merger of the alignments together. PROMALS, SATCHMO-JS, and mega-phylogeny use sophisticated methods to merge subset-alignments, but SATé uses a very simple method (Muscle) to merge subset-alignments.

Neuwald's method [33] shares many features with these four methods, but has some unique features that are worth pointing out. First, like SATCHMO-JS and PROMALS, Neuwald's method can only be used on proteins (mega-phylogeny and SATé can be used on both nucleotides and protein sequences). Neuwald's method requires the user to provide a dataset decomposition and also a manually curated seed alignment reflecting structural and functional features of the protein family. The algorithm operates by estimating alignments on the subsets using simple methods, and then uses the seed alignment to merge the subset-alignments together.

Note that Neuwald's method, PROMALS, and mega-phylogeny are essentially template-based methods, and as such are very scalable once their templates are computed (this first step, however, can be very labor-intensive, if it depends on expert curation). Mega-phylogeny has been used to analyze a dataset with more than 50,000 nucleotide sequences [35], and Neuwald's method has been used to analyze a dataset with more than 400,000 protein sequences. By contrast, because SATCHMO-JS and SATé both rely upon progressive alignment, their running times are longer. Furthermore, SATé uses iteration to obtain improved results (even though the first iteration gives the most improvement), and although most runs finish in just a few iterations, this also adds to the running time. SATé has been used to analyze a dataset with approximately 28,000 nucleotide sequences, but has not been tested on larger datasets.

In terms of performance evaluations, Neuwald's method, SATCHMO-JS, and PROMALS, have been assessed using protein alignment benchmarks, and shown to give excellent results over standard methods. Ortuno et al. [21] explored the conditions in which PROMALS gave improvements over the other methods, and showed that the conditions in which the improvements were substantial were when the se-

quences were close to the 'twilight zone' (i.e., almost random with respect to each other), which is where sequence homology is difficult to detect, and information about structure is the most helpful.

The accuracy of SATé has been assessed using nucleotide alignments, and shown to be very good, both for standard alignment criteria and for phylogenetic accuracy [39]. A recent study [119] evaluated protein alignment methods (including SATé) on large datasets with respect to the TC (total column) score. They found substantial differences in running time between the template-based methods (which had the best speed) and other methods, including SATé, and so only ran the fastest methods on the largest datasets, which had 50,000 protein sequences. To the best of our knowledge, the mega-phylogeny method has not been compared to other methods on benchmark datasets with curated alignments or trees.

Very Large-Scale Alignment When the datasets are very large, containing many thousands of sequences, only a few alignment estimation methods are able to run. As noted, the template-based methods (including PROMALS and mega-phylogeny) scale linearly with the number of taxa, and so can be used with very large datasets. SATé and SATCHMO-JS are not quite as scalable; however, SATé has been able to analyze nucleotide datasets with about 28,000 sequences. Other methods that have been shown to run on very large datasets include Clustal-Omega [28], MAFFT-PartTree [29], and Kalign-2 [32], but many methods fail to run on datasets with tens of thousands of sequences [6]. Of these, Clustal-Omega is only designed for protein sequences, but MAFFT-PartTree and Kalign-2 can analyze both nucleotide and amino-acid sequences.

SATé is computationally limited by its use of progressive alignment and maximum likelihood method (RAxML or FastTree-2 [34]) in each iteration; both impact the running time and—in the case of large numbers of long sequences—memory usage. However, although limited to datasets with perhaps only 30,000 sequences (or so), on fast-evolving datasets with 1000 or more sequences, SATé provides improvements in phylogenetic accuracy relative to competing methods [6, 39].

6.2.3 Tree Estimation

Most phylogeny estimation methods are designed to be used with sequence alignments, and so presume that the alignment step is already completed. These methods are generally studied with respect to their performance under Markov models of evolution in which sequences evolve only with substitutions. Therefore, we begin with a discussion about site substitution models, and about statistical performance guarantees under these models.

6.2.3.1 Stochastic Models of Sequence Evolution

We begin with a description of the simplest stochastic models of DNA sequence evolution, and then discuss amino-acid sequence evolution models and codon evolution

models. The simplest models of DNA sequence evolution treat the sites within the sequences independently. Thus, a model of DNA sequence evolution must describe the probability distribution of the four states, A, C, T, G, at the root, the evolution of a random site (i.e., position within the DNA sequence) and how the evolution differs across the sites. Typically the probability distribution at the root is uniform (so that all sequences of a fixed length are equally likely). The evolution of a single site is modeled through the use of "stochastic substitution matrices," 4×4 matrices (one for each tree edge) in which every row sums to 1. A stochastic model of how a single site evolves can thus have up to 12 free parameters. The simplest such model is the Jukes–Cantor model, with one free parameter, and the most complex is the General Markov model, with all 12 parameters [123]:

Definition 1 The General Markov (GM) model of single-site evolution is defined as follows.

1. The nucleotide in a random site at the root is drawn from a known distribution, in which each nucleotide has positive probability.
2. The probability of each site substitution on an edge e of the tree is given by a 4×4 stochastic substitution matrix $M(e)$ in which $det(M(e))$ is not 0, 1, or -1.

This model is generally used in a context where all sites evolve identically and independently (the i.i.d. assumption), with rates of evolution drawn typically drawn from a gamma distribution. (Note that the distribution of the rates across sites has an impact on phylogeny estimation and dating, as discussed by Evans and Warnow [124].) In what follows, we will address the simplest version of the GM model so that all sites have the same rate of evolution.

We denote a model tree in the GM model as a pair, $(T, \{M(e) \colon e \in E(T)\})$, or more simply as (T, M). For each edge $e \in E(T)$, we define the length of the edge $\lambda(e)$ to be $-\log |det(M(e))|$. This allows us to define the matrix of leaf-to-leaf distances, $\{\lambda_{ij}\}$, where $\lambda_{ij} = \sum_{e \in P_{ij}} \lambda(e)$, and where P_{ij} is the path in T between leaves i and j. A matrix defined by path distances in a tree with edge weights is called "additive", and it is a well-known fact that given any additive matrix, it is easy to recover the underlying leaf-labeled tree T for that matrix in polynomial time.

This general model of site evolution subsumes the great majority of other models examined in the phylogenetic literature, including the popular General Time Reversible (GTR) model [125], which requires only that $M(e) = M(e')$ for all edges e and e'. Further constraints on the matrix $M(e)$ produce the Hasegawa–Kishino–Yang (HKY) model, the Kimura 2-parameter model (K2P), the Kimura 3-ST model (K3ST), the Jukes–Cantor model (JC), etc. These models are all special cases of the General Markov model, because they place restrictions on the form of the stochastic substitution matrices. The standard model used for nucleotide phylogeny estimation is GTR+gamma, i.e., the General Time Reversible (GTR) model of site substitution, equipped with a gamma distribution of rates across sites.

Protein Models Just as with DNA sequence evolution models, there are Markov models of evolution for amino-acid sequences, and also for coding DNA sequences. These models are described in the same way—a substitution matrix that governs the tree, and then branch lengths. While the GTR model can be extended to amino-acids (to produce a 20×20 matrix) or to codon-based models (to produce a 64×64 matrix), both of which must be estimated from the data, in practice these models use fixed matrices, each of which was estimated from external biological data. The most well known protein model is the Dayhoff model [126], but improved models have been developed in recent years [127–133]. Similarly, codon-based models have also been based on fixed 64×64 matrices (e.g., [134–136]). In practice, the selection of a protein model for a given dataset is often done using a statistical test, such as ProtTest [137], and then fixed. In the subsequent tree estimation performed under that model, only the tree and its branch lengths need to be estimated.

More General Site Evolution Models The models that are typically used in phylogenetic estimation tend to be fairly simple, and have come under serious criticism as a result, especially when used with proteins [138, 139]. For example, these models fail to account for GC content variation across the tree, rates of evolution that are not drawn from the gamma distribution (or similarly simple models), or heterotachy (where the substitution matrix depends on the edge and the site [140–143]). Studies of gene family evolution have also shown that the neutral model of evolution is unrealistic [144]. More general models of site evolution have been proposed, including the non-stationary, non-homogeneous model of Galtier and Guoy [145].

6.2.3.2 Phylogeny Estimation Methods

There are many different phylogeny estimation methods, too numerous to mention here. However, the major ones can be classified into the following types:

- distance-based methods, which first compute a pairwise distance matrix (usually based on a statistical model) and then compute the tree from the matrix [17],
- maximum parsimony and its variants [146], which seek a tree with a total minimum number of changes (as defined by edit distances between sequences at the endpoints on the edges of the tree),
- maximum likelihood [147], which seeks the model tree that optimizes likelihood under the given Markov model, and
- Bayesian MCMC methods, which return a distribution on trees rather than a single tree, and also use likelihood to evaluate a model tree.

6.2.3.3 Statistical Performance Criteria

We discuss three concepts here: *identifiability*, *statistical consistency*, and *sequence-length requirements*.

Identifiability A statistical model or one of its parameters is said to be "identifiable" if it is uniquely determined by the probability distribution defined by the model. Thus, in the context of phylogeny estimation, the unrooted model tree topology is identifiable if it is determined by the probability distribution (defined by the model tree, which includes the numeric parameters) on the patterns of nucleotides at the leaves of the tree. In the case of nucleotide models, the state at each leaf can be A, C, T, or G, and so there are 4^n possible patterns in a tree with n leaves (similarly, there are 20^n possible patterns for amino-acid models). It is well known that the unrooted tree topology is identifiable under the General Markov model [123], and recent work has extended this to other models [148–150].

Statistical Consistency We say that a method Φ is "statistically consistent" for estimating the topology of the model tree (T, θ) if the trees estimated by Φ *converges* to the unrooted version of T (denoted by T^u) as the number of sites increases. (Note that under this definition, we are not concerned with estimating the numeric parameters.) Equivalently, for all $\varepsilon > 0$ there is a sequence length K so that if a set S of sequences of length $k \geq K$ are generated by (T, θ), then the probability that $\Phi(S) = T^u$ is at least $1 - \varepsilon$. We say that a method is statistically consistent under the GM model if it is statistically consistent for all model trees in the GM model. Similarly, we say a method is statistically consistent under the GTR model if it is statistically consistent under all model trees in the GTR model.

Many phylogenetic methods are statistically consistent under the GM model, and hence also under its submodels (e.g., the GTR model). For example, maximum likelihood, neighbor joining (and other distance-based methods) for properly computed pairwise "distances", and Bayesian MCMC methods, are all statistically consistent [17, 151–153]. On the other hand, maximum parsimony and maximum compatibility are not statistically consistent under the GM model [154]. In addition, it is well known that maximum likelihood can be inconsistent if the generative model is different from the model assumed by maximum likelihood, but maximum likelihood can even be inconsistent when its assumptions match the generative model, if the generative model is too complex! For example, Tuffley and Steel showed that maximum likelihood is equivalent to maximum parsimony under a very general "no-common-mechanism" model [142], and so is inconsistent under this model. In this case, the model itself is not identifiable, and this is why maximum likelihood is not consistent [155–157]. However, there are identifiable models for which ML is not consistent, as observed by Steel [143].

Sequence-Length Requirement Clearly, statistical consistency under a model is a desirable property. However, statistical consistency does not address how well a method will work on finite data. Here, we address the "sequence-length requirement" of a phylogeny estimation method Φ, which is the number of sites that Φ needs to return the (unrooted version of the) true tree with probability at least $1 - \epsilon$ given sequences that evolve down a given model tree (T, θ). Clearly, the number of sites that suffices for accuracy with probability at least $1 - \epsilon$ will depend on Φ and ϵ, but it also depends on both T and θ.

We describe this concept in terms of the Jukes–Cantor model, since this is the simplest of the DNA sequence evolution models, and the ideas are easiest to understand for this model. However, the same concepts can be applied to the more general models, and the theoretical results that have been established regarding sequence-length requirements extend to the GM (General Markov) model, which contains the GTR model and all its submodels.

In the Jukes–Cantor (JC) model, all substitutions are equally likely, and all nucleotides have equal probability for the root state. Thus, a Jukes–Cantor model tree is completely defined by the rooted tree T and the branch lengths $\lambda(e)$, where $\lambda(e)$ is the expected number of changes for a random site on the edge e. It is intuitively obvious that as the minimum branch length shrinks, the number of sites that are needed to reconstruct the tree will grow, since a branch on which no changes occur cannot be recovered with high probability (the branch will appear in an estimated tree with probability at most one-third, since at best it can result from a random resolution of a node of degree at least 4). It is also intuitively obvious that as the maximum branch length increases, the number of sites that are needed will increase, since the two sides of the long branch will seem random with respect to each other. Thus, the sequence-length requirement for a given method to be accurate with probability at least $1 - \epsilon$ will be impacted by the shortest branch length f and the longest branch length g. It is also intuitively obvious that the sequence-length requirement will depend on the number of taxa in the tree.

Expressing the sequence-length requirement for the method Φ as a function of these parameters (f, g, n and ϵ) enables a different—and finer—evaluation of the method's performance guarantees under the statistical model. Hence, we consider f, g, and ϵ as fixed but arbitrary, and we let $JC_{f,g}$ denote all Jukes–Cantor model trees with $0 < f \leq \lambda(e) \leq g < \infty$ for all edges e. This lets us bound the sequence-length requirement of a method as a function only of n, the number of leaves in the tree.

The definition of "absolute fast convergence" under the Jukes–Cantor model is formulated as an upper bound on the sequence-length requirement, as follows:

Definition 2 A phylogenetic reconstruction method Φ is *absolute fast-converging* (afc) for the Jukes–Cantor (JC) model if, for all positive f, g, and ε, there is a polynomial $p(n)$ such that, for all (T, θ) in $JC_{f,g}$, on set S of n sequences of length at least $p(n)$ generated on T, we have $\Pr[\Phi(S) = T^u] > 1 - \varepsilon$.

Note also that this statement only refers to the estimation of the unrooted tree topology T^u and not the numeric parameters θ. Also, note that the method Φ operates without any knowledge of parameters f or g—or indeed any function of f and g. Thus, although the polynomial p depends upon both f and g, the method itself will not. Finally, this is an upper bound on the sequence-length requirement, and the actual sequence-length requirement could be much lower.

The function $p(n)$ can be replaced by a function $f(n)$ that is not polynomial to provide an upper bound on the sequence-length requirement for methods that are not proven to be absolute fast converging.

In a sequence of papers, Erdős et al. [158–160] presented the first absolute fast converging methods for the GM model, and presented techniques for establishing the sequence length requirements of distance-based methods. Following this, the sequence-length requirement of neighbor joining (NJ) was studied, and lower bounds and upper bounds that are exponential in n were established [151, 161]. These papers were followed by a number of other studies presenting other afc methods (some with even better theoretical performance than the first afc methods) or evaluating the sequence-length requirements of known methods [162–175].

6.2.3.4 Empirical Performance

So far, these discussions have focused on theoretical guarantees under a model, and have addressed whether a method will converge to the true tree given long enough sequences (i.e., statistical consistency), and if so, then how long the sequences need to be (sequence-length requirements). However, these issues are purely theoretical, and do not address how accurate the trees estimated by methods are in practice (i.e., on data). In addition, the computational performance (time and memory usage) of phylogeny estimation methods is also important, since a method that is highly accurate but will use several years of compute time will not generally be useful in most analyses.

Phylogenetic tree accuracy can be computed in various ways, and there are substantive debates on the "right" way to calculate accuracy [176, 177]; however, although disputed, the Robinson-Foulds [178] (RF) distance, also called the "bipartition distance", is the most commonly used metric on phylogenetic trees. We describe this metric here.

Given a phylogenetic tree T on n taxa, each edge can be associated with the bipartition it induces on the leaf set; hence, the tree itself can be identified with the set of leaf-bipartitions defined by the edges in the tree. Therefore, two trees on the same set of taxa can be compared with respect to their bipartition sets. The RF distance between two trees is the size of the symmetric difference of these two sets, i.e., it is the number of bipartitions that are in one tree's dataset but not both. This number can be divided by $2(n-3)$ (where n is the number of taxa) to obtain the "RF rate." In the context of evaluating phylogeny estimation methods, the RF distance is sometimes divided into false negatives and false positives, where the false negatives (also called "missing branches") are branches in the true tree that are not present in the estimated tree, and the false positives are the branches in the estimated tree that are not present in the true tree. This distinction between false positives and false negatives enables a more detailed comparison between trees that are not binary.

Many studies have evaluated phylogeny estimation methods on simulated data, varying the rate of evolution, the branch lengths, the number of sites, etc. These studies have been enormously informative about the differences between methods, and have helped biologists make informed decisions regarding methods for their phylogenetic analyses. Some of the early simulation studies explored performance on very small trees, including the fairly exhaustive study by Huelsenbeck and Hillis

on 4-leaf trees [179], but studies since then have explored larger datasets [4, 10, 180, 181] and more complex questions. For example, studies have explored the impact of taxon sampling on phylogenetic inference [12, 180, 182, 183], the impact of missing data on phylogenetic inference [184–187], and the number of sites needed for accuracy with high probability [188]. In fact, simulation studies have become, perhaps, the main way to explore phylogenetic estimation.

Distance-Based Methods Distance-based methods operate by first computing a matrix of distances (typically using a statistically defined technique, to correct for unseen changes) between every pair of sequences, and then construct the tree based on this matrix. Most, but not all, distance-based methods are statistically consistent, and so will be correct with high probability, given long enough sequences. In general, distance-based methods are polynomial time, and so have been popular for large-scale phylogeny estimation. While the best known distance-based method is probably neighbor joining [121], there are many others, and many are faster and/or more accurate [189–194].

One of the interesting properties about distance-based methods is that although they are typically guaranteed to be statistically consistent, not all distance-based methods have good empirical performance! A prime example of this lesson is the Naive Quartet Method, a method that estimates a tree for every set of four leaves using the Four-Point Method (a statistically consistent distance method) and then returns the tree that is consistent with all the quartets *if it exists* [17]. It is easy to show that the Naive Quartet Method runs in polynomial time and is statistically consistent under the General Markov model; however, because it requires that every quartet be accurately estimated, it has terrible empirical performance! Thus, while statistical consistency is desirable, in many cases statistically inconsistent methods can outperform consistent ones [195, 196].

Maximum Parsimony Maximum parsimony (MP) is NP-hard [146], and so the methods for MP use heuristics (most without any performance guarantees). The most efficient and accurate maximum parsimony software for very large datasets is probably TNT [38], but PAUP* [197] is also popular and effective on datasets that are not extremely large. TNT is a particularly effective parsimony heuristic for large trees [54], and has been able to analyze a multi-marker sequence dataset with more than 73,000 sequences [37].

Maximum Likelihood Maximum likelihood (ML) is also NP-hard [198], and so attempts to solve ML are also made using heuristics. While the heuristics for MP used to be computationally more efficient than the heuristics for ML, the current set of methods for ML are quite effective at "solving" large datasets. (Here the quotes indicate that there is no guarantee, but reasonably good results do seem to be obtained using the current best software.)

The leading methods for large-scale ML estimation under the GTR+Gamma model include RAxML [36], FastTree-2 [34], PhyML [199], and GARLI [200]. Of these four methods, RAxML is clearly the most frequently used ML method, in part

because of its excellent parallel implementations. However, a recent study [201] showed that trees estimated by FastTree-2 were almost as accurate as those estimated by RAxML, and that FastTree-2 finished in a fraction of the time; for example, FastTree-2 was able to analyze an alignment with almost 28,000 rRNA sequences in about 5 hours, but RAxML took much longer. Furthermore, FastTree-2 has been used to analyze larger datasets (ones with more sequences) than RAxML: the largest dataset published with a RAxML analysis had 55,000 nucleotide sequences [35], but FastTree has analyzed larger datasets. For example, FastTree-2 has analyzed a dataset with more than 1 million nucleotide sequences [202], and another with 330,556 sequences [203]. The reported running time for these analyses are 203 hours for the million-taxon dataset, and 13 hours (with 4 threads) for the 330 K-taxon dataset.[2] By comparison, the RAxML analysis of 55,000 nucleotide sequences took between 100,000 and 300,000 CPU hours.[3] The difference in running time is substantial, but we should note two things: the RAxML analysis was a multi-marker analysis, and so the sequences were much longer (which impacts running time), and because RAxML is highly parallelized, the impact of the increased running time is not as significant (if one has enough processors). Nevertheless, for maximum likelihood analysis of alignments with large numbers of sequences, FastTree-2 provides distinct speed advantages over RAxML.

There are a few important limitations for FastTree-2, compared to RAxML. First, FastTree-2 obtains its speed by somewhat reducing the accuracy of the search; thus, the trees returned by FastTree-2 may not produce maximum likelihood scores that are quite as good as those produced by RAxML. Second, FastTree-2 does not handle very long alignments with hundreds of thousands of sites very well, while RAxML has a new implementation that is designed specifically for long alignments. Third, FastTree-2 has a smaller set of models for amino-acid analyses than RAxML. Therefore, in some cases (e.g., for wide alignments, and perhaps for amino-acid alignments), RAxML may be the preferred method.

However, the ML methods discussed above estimate trees under the GTR+Gamma model, which has simplifying assumptions that are known to be violated in biological data. The nhPhyml [204] method is a maximum likelihood method for estimating trees under the non-stationary, non-homogeneous model of Galtier and Guoy [145], and hence provides an analytical advantage in that it can be robust to some violations of the GTR+Gamma model assumptions. However, nhPhyml seems to be able to give reliably good analyses only on relatively small datasets (i.e., with at most a few hundred sequences, or fewer sequences if they are very long). The explanation is computational—it uses NNI (nearest neighbor interchanges, see below) to search tree space, but NNI is relatively ineffective [205, 206], which means that it is likely to get stuck in local optima. This is unfortunate, since large datasets spanning substantial evolutionary distances are most likely to exhibit an increased incidence in model violations. Therefore, highly accurate phylogeny estimation of

[2] Morgan Price, personal communication, 1 May 2013.

[3] Alexis Stamatakis, personal communication, 1 May 2013.

large datasets may require the use of new methods that are based upon more realistic, and more general, models of sequence evolution.

Bayesian MCMC Methods Bayesian methods are similar to maximum likelihood methods in that the likelihood of a model tree with respect to the input sequence alignment is computed during the analysis; the main difference is that maximum likelihood selects the model tree (both topology and numeric parameters) that optimizes the likelihood, while a Bayesian method outputs a distribution on trees. However, once the distribution is computed, it can be used to compute a single point estimate of the true tree using various techniques (e.g., a consensus tree can be computed, or the model tree topology with the maximum total probability can be returned).

The standard technique used to estimate this distribution is a random walk through the model tree space, and the distribution is produced after the walk has converged to the stationary distribution.

There are many different Bayesian methods (e.g., MrBayes [207], BEAST [208], PhyloBayes [209], Foster [210], and BayesPhylogenies [211]), differing in terms of the techniques used to perform the random walk, and the model under which the likelihood is computed; however, MrBayes [212] is the most popular of the methods. Bayesian methods provide theoretical advantages compared to maximum likelihood methods [213–216]. However, the proper use of a Bayesian MCMC method requires that it run to convergence, and this can take a very long time on large datasets [61]. Thus, from a purely empirical standpoint, Bayesian methods do not yet have the scalability of the best maximum likelihood methods, and they are generally not used on very large datasets.

Comparisons Between Methods Simulation studies have shown some interesting differences between methods. For example, the comparison between neighbor joining and maximum parsimony reveals that the relative performance may depend on the number of taxa and the rate of evolution, with maximum parsimony sometimes performing better on large trees with high rates of evolution [195], even though the reverse generally holds for smaller trees [179].

More generally, most simulation studies have shown that maximum likelihood and Bayesian methods (when they can be run properly) outperform maximum parsimony and distance-based methods in many biologically realistic conditions (see Wang et al. [4] for one such study).

Heuristics for Exploring Tree Space Since both maximum likelihood and maximum parsimony are NP-hard, methods for "solving" these problems use heuristics to explore the space of different tree topologies. These heuristics differ by the techniques they use to score a candidate tree (with the best ones typically using information from previous trees that have already been scored), and how they move within tree space.

The standard techniques for exploring tree space (i.e., for changing the unrooted topology of the current tree) use either NNI (nearest neighbor interchanges),

SPR (subtree prune-and-regraft) or TBR (tree-bisection-and-reconnection) moves. All these moves modify unrooted trees, as follows. In an NNI move, an edge in the tree is identified, and two subtrees (one on each side of the edge) are swapped. In an SPR move, a rooted subtree of the tree is deleted from the tree, and then reattached. In a TBR move, an edge in the tree is deleted, thus creating two separate (unrooted) trees, and then the two trees are attached through the addition of an edge. Another type of move, called p-ECR (p-edge-contract-and-refine) [217, 218], has also been suggested. In this move, p different edges are contracted, thus creating one or more high degree nodes; the resultant unresolved tree is then either randomly refined, or refined optimally with respect to the criterion of interest (see, for example, [218, 219] for results regarding maximum parsimony). By definition, an NNI move is an SPR move, and an SPR move is a TBR move; thus, the TBR move is the most general of these three moves. However, p-ECR moves generate different neighborhoods than these moves, although an NNI move is a 1-ECR move. Software that only use NNI moves (e.g., nhPhyml [204]) has the advantage of being faster, since they will reach local optima more quickly; however, they also have a tendency to get stuck in local optima more frequently. The TNT software for maximum parsimony uses more complicated techniques, including sectorial-search, for exploring tree space [38]. Theoretical evaluations of these techniques for exploring tree space have been made [205, 206, 217, 218] that help explain the trade-offs between search strategies.

Because local optima are a problem for heuristic searches for NP-hard problems, randomization is often used to move out of local optima. An example of a technique that uses randomness effectively is the parsimony ratchet [220], which was also implemented for maximum likelihood [221]. In the parsimony ratchet, the search alternates between heuristic searches based on the original alignment, and searches based on stochastically modified versions of the alignment; thus, the tree found during the search for the stochastically modified alignment is used to initiate a search based on the original alignment, etc. This technique thus uses randomness to modify the alignment, rather than to move to a random point in tree space; thus, randomness is a general technique that can be used to improve heuristic searches.

6.2.3.5 DCM_{NJ}: A Fast Converging Method with Good Empirical Performance

In Sect. 6.2.3.3, we discussed absolute fast converging methods, which are methods that provably reconstruct the true tree with high probability from sequences of lengths that grow only polynomially in the number of taxa. As stated, this is a mathematical property rather than an empirical property. Here we describe one of the early absolute fast converging methods, called DCM-neighbor joining (DCM_{NJ}) [222, 223].

The input to DCM_{NJ} is a distance matrix $[D_{ij}]$ for a set of n species, where the distance matrix is defined appropriately for the model (e.g., the use of the logdet [123] distances for the GTR model). DCM-neighbor joining has two phases.

In the first phase, it computes a set X of $O(n^2)$ trees (one for each entry in the distance matrix), and in the second phase, it selects a tree from the set X based on a "true tree selection criterion". To obtain a theoretical guarantee of absolute fast convergence, both phases must have some statistical guarantees, which we now describe in the context of the Jukes–Cantor model.

Phase 1 Property: Given JC model tree (T, θ) with branch lengths satisfying $0 < f \leq \lambda(e) \leq g < \infty$ and given $\epsilon > 0$, there is a polynomial $p(n)$ (which can depend on f, g and ϵ) so that given sequences of length at most $p(n)$, then the set X of trees produced in Phase 1 will contains the unrooted true tree T^u with probability at least $1 - \epsilon$.

Phase 2 Property: Let C be the criterion used for Phase 2. Then the desired property for Phase 2 is defined as follows. Given JC model tree (T, θ) with branch lengths satisfying $0 < f \leq \lambda(e) \leq g < \infty$ and given $\epsilon > 0$, there is a polynomial $p(n)$ (which can depend on f, g, and ϵ) so that given sequences of length at most $p(n)$, and given a set X of trees on taxon set S that contains the T^u, then $T^{\text{opt}} = T^u$ with probability at least $1 - \epsilon$, where T^{opt} is the tree in X that optimizes criterion C.

We now describe these two phases.

Phase 1 of *DCM*$_{NJ}$ For each entry q in the distance matrix $[D_{ij}]$, DCM_{NJ} computes a tree T_q, as follows. First, a "threshold graph" is computed based on $[D_{ij}]$ and the threshold q, so that there is a vertex for every taxon, and an edge between two vertices v_i and v_j if and only if $D_{ij} \leq q$. If the distance matrix is additive (meaning that it equals the path distance in some edge-weighted tree [17]), the threshold graph will be triangulated (also called "chordal"), which means that either the graph is acyclic, or that every induced simple cycle in the graph is of size 3. Chordal graphs have special properties, including that the set of maximal cliques can be enumerated in polynomial time; thus, we can compute the set of (at most) n maximal cliques in the threshold graph [224]. Otherwise, we add edges to the threshold graph (minimizing the maximum "weight" of any added edge) to create a triangulated graph, and then continue. Each maximal clique thus defines a subset of the input sequence set, in the obvious way.

We use the "base method" (here, neighbor joining) to construct a tree on each of these subsets, and we combine these subset trees into a tree on the entire dataset using a particular supertree method called the Strict Consensus Merger [225] (the first phase of SuperFine [53]). For threshold values q that are very small, the set of neighbor joining trees will be insufficient to define the full tree (because of failure to overlap sufficiently). For very large threshold values, there will be sufficient overlap, but the neighbor joining trees on the subsets may have errors. However, for intermediate threshold values, given polynomial length sequences, the neighbor joining subtrees will be correct with high probability and sufficient to define the full tree [222]. Under these conditions, the Strict Consensus Merger will produce the true tree (Lemma 6.2 in [222]). Hence, from polynomial length sequences, with high probability, the first phase will produce the true tree as *one of* the trees in the set of trees it produces (one for each threshold value).

Phase 2 Each tree in the set of trees produced in Phase I is scored using the desired "True Tree Selection" (TTS) criterion, and the tree with the best score is returned. Examples of criteria that have been considered are the maximum likelihood (ML) score, the maximum parsimony (MP) score, and the "short quartet support" (SQS) score. Of these, the MP and SQS scores can be computed exactly and in polynomial time, but the ML score can only be estimated heuristically [226]. Of these criteria, the SQS score is guaranteed to satisfy the required property (described above), but the use of the ML and MP scores gives somewhat more accurate trees.

To summarize, DCM-NJ uses a two-phase process, in which the first phase produces a set of trees, and the second phase selects a tree from the set. Furthermore, each tree computed in the first phase is obtained by using a graph-theoretic technique to decompose the dataset into small overlapping subsets, neighbor joining is used to construct trees on each subset, and then these subset trees are combined together using a supertree method. The result is a method that reconstructs the true tree with high probability from polynomial length sequences, even though the base method (NJ) has a sequence-length requirement that is exponential [161]. Thus, DCM-NJ is a technique that estimates a tree on the full set of taxa and that "boosts" the performance of NJ. A similar but simpler method [164] was designed to boost another distance-based method called the "Buneman Tree" (named after Peter Buneman) to produce the DCM-Buneman method, also proven to be afc.

Figure 6.1 evaluates two variants of DCM-NJ, differing by the "true tree selection" criterion in the second phase, and compares them to neighbor joining (NJ) and to $HGT + FP$, another absolute fast converging method [162]. $DCM_{NJ} + SQS$ uses SQS for the true tree selection criterion, while $DCM_{NJ} + MP$ uses maximum parsimony. By design, SQS has the desired theoretical property, but MP does not. Instead, MP is used for its empirical performance, as the figure shows.

These methods are evaluated on simulated 1000-site datasets generated down K2P model trees, each with the same branch length distribution, but with varying numbers of taxa. Thus, as the number of taxa increases, the overall amount of evolution increases, and the dataset becomes more difficult to analyze (especially since the sequence length remains fixed at 1000 sites). As shown in Fig. 6.1, the error rate of neighbor joining (using corrected distances to reflect the model of evolution) begins low but increases, so that at 1600 taxa, it is above 40 %. By contrast, $DCM_{NJ} + SQS$, $DCM_{NJ} + MP$ and $HGT + FP$ all have fairly low error rates throughout the range of datasets we tested. Note also that $DCM_{NJ} + MP$ is slightly more accurate than $DCM_{NJ} + SQS$, even though it has no guarantees. Finally, $DCM_{NJ} + SQS$, $DCM_{NJ} + MP$ and $HGT + FP$ seem to have error rates that do not increase with the numbers of taxa; this is obviously impossible, and so for large enough numbers of taxa, the error rates will eventually increase, and this trend cannot continue indefinitely.

The development of methods with good sequence-length requirements is an area of active research, and newer methods with even better theoretical performance have been developed. Because afc methods are designed to extract phylogenetic signal from small numbers of sites, this means that performance on very large datasets (with many taxa) might be improved using these methods, even without needing to

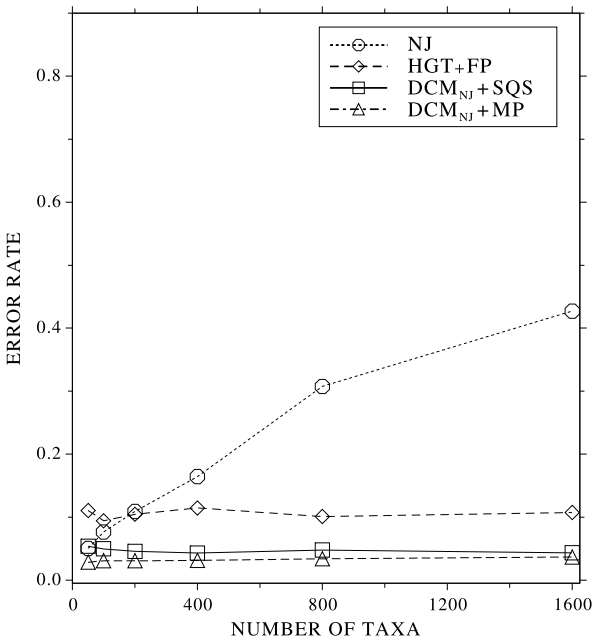


Fig. 6.1 The performance of $DCM1_{NJ}$ with different techniques used in Phase II, compared to NJ and to another absolute fast converging method, HGT+FP [162], as a function of the number of taxa. In this experiment we simulated evolution of sequences with 1000 sites down K2P model trees with topologies drawn from the uniform distribution and with branch lengths drawn from a fixed range. K2P distances were used as inputs to each method (This figure appeared in Nakhleh et al. [223])

use huge numbers of sites. These methods are not used in practice (and DCM_{NJ} is not publicly distributed), and so these methods are mostly of theoretical interest rather than practical.

Clearly there is the potential for these methods to give highly accurate trees for very large datasets; however, the methods and theorems described here assume that the sequences evolve without any indels, and under the General Markov model. While the results could be extended to other identifiable models (including ones with indels) for which statistically consistent distant estimation techniques are available, they would still require that the true alignment be known. Thus, none of this theory applies to more realistic conditions—sequences that evolve with indels, for which the true alignment is *not* known.

6.2.3.6 Gap Treatment in Phylogeny Estimation Methods

Until now, the entire discussion about phylogeny estimation methods and their guarantees under Markov models of evolution has ignored the fact that sequence alignments often have gaps and that indels are part of sequence evolution. Instead, the Markov models we have discussed are entirely indel-free, and the methods were

described as though the sequences were also indel-free. Obviously, since phylogeny estimation methods have been applied to real data, this means that modifications to the data or to the methods have been made to enable them to be used with sequence alignments that have gaps. The purpose of this section is to describe these modifications, and present some discussion about the pros and cons of each modification.

Given a sequence alignment containing gaps, the following approaches are the main ones used in practice for estimating phylogenies:

1. Remove all sites in which any indel appears;
2. Assign an additional state for each dash (thus, for nucleotides, this would result in a 5-state model);
3. Code all the gaps (contiguous segments of dashes) in the alignment, and treat the presence or absence of a gap as a binary character (complementing the original sequence alignment character data); and
4. Treat the gaps as missing data. In parsimony analyses, this is often treated by finding the best nucleotide to replace the gap, but in likelihood-based analyses, this is often treated by summing the likelihood over all possible nucleotides for each gap.

Note that the first three approaches specifically modify the data, and that with the exception of the first approach, all techniques change the input in such a way that the method used to estimate a tree on the alignment must also be changed. Thus, for approach #2, the method must be able to handle 5-state data (for DNA) or 21-state data (for proteins). For approach #3, the method has to be able to handle binary data. In the case of parsimony or likelihood, the challenge is whether changes from presence to absence are treated the same as from absence to presence, and also whether the Markov assumption still makes sense. There are arguments in favor and against each of these gap treatments, especially with respect to statistical consistency under a stochastic model that includes indels as well as substitutions [227].

The first approach of removing all sites with gaps has the advantage of being statistically consistent for stochastic models with indel events in which the substitution process and the mechanism producing insertions and deletions are independent. However, it removes data, and in practice, especially on datasets with many taxa, it could result in phylogenetically uninformative sequence alignments. (A less extreme version of removing all sites with gaps is called "masking", whereby only some of the sites with gaps are removed. The benefits of using masking are debated, but some recent studies suggest that masking may not be desirable [228].)

The second and third approaches do not reduce the amount of data (which is good) and there are many different gap-coding techniques [229–231]. Simulation studies evaluating some of these methods have shown improvements in some cases for tree estimation obtained through gap-coding over treating gaps as missing data [232–234], but others have found differently [228, 235].

However, the use of gap-coding is controversial [232], in part because of the very substantive challenges in creating a statistically appropriate treatment (consider the meaning of positional homology [103]). Instead, the most frequently used option, and the default for most software, is to treat gaps as missing data. The simulation

studies presented later in this paper are all based on analyses of data, treating gaps as missing data.

6.2.3.7 Theoretical Guarantees for Standard Phylogeny Estimation Methods on Alignments with Gaps

Are any of the phylogeny estimation methods we have discussed guaranteed to be statistically consistent when treating gaps as missing data? This is one of the interesting open questions in phylogenetics, for which we give a partial answer.

To address this problem, we defined "monotypic" alignments to be ones in which each site has only one type of nucleotide (all As, all Cs, all Ts, or all Gs) and we proved the following [236]:

Theorem *When the true alignment is monotypic and gaps are treated as missing data, then all trees are optimal for the true alignment under Jukes–Cantor maximum likelihood. Therefore, if the model tree allows indels but not substitutions, then all trees are optimal for Jukes–Cantor maximum likelihood, when gaps are treated as missing data.*

At first glance, this theorem might seem to be the result of monotypic alignments not having phylogenetic signal, but this is not the case! In fact, monotypic alignments have sufficient signal to enable accurate trees [237, 238]. Thus, there is phylogenetic signal in an alignment that contains gaps even for the case of monotypic alignments, and this signal can be used to estimate the true tree, provided that appropriate methods are used. In other words, the indels within an alignment can be phylogenetically informative, and indels can even be sufficient to define the tree topology. However, gap treatments can result in loss of information, or lead to erroneous trees (as in the case of ML, treating gaps as missing data, when handling monotypic alignments).

Note that this result does not imply that ML, treating gaps as missing data, is inconsistent under models with positive probabilities of substitutions, and it seems very likely that for most biologically realistic conditions, treating gaps as missing data will not lead to meaningless results. Furthermore, the simulations we and others have performed suggest that ML methods, treating gaps as missing data, do produce reasonably accurate trees. Even so, the potential for reductions in accuracy due to inappropriate handling of gaps is clearly present, and it raises the real possibility that the methods that are known to be statistically consistent under standard substitution-only models, such as GTR, may not be statistically consistent (even on the true alignment!) when sequences evolve with both substitutions and indels.

6.2.4 Handling Fragmentary Data: Phylogenetic Placement

Multiple sequence alignment methods are generally studied in the context of full-length sequences, and little is known about how well methods work when some

of the sequences are very fragmentary. Furthermore, phylogenetic estimation in the context of fragmentary sequences is unreliable, even if the alignments of the fragmentary sequences are accurate [184, 239].

One approach to handling fragmentary sequences is phylogenetic placement: in the first step, an alignment and tree is estimated for the full length sequences for the same gene (these are called the "backbone alignment" and "backbone tree" [240]); in the second step, the fragmentary sequences are added into the backbone alignment to create an "extended alignment"; and finally in the third step, the fragments are then placed in the tree using the extended alignment. This is called the "phylogenetic placement problem". The first methods for this problem were pplacer [241] and Evolutionary Placement Algorithm (EPA) [242]; both use HMMALIGN [243, 244] to insert the fragments into the alignment of the full-length sequences and then place the fragments into the tree using maximum likelihood for this extended alignment. The initial studies showed that EPA and pplacer exhibited little difference in accuracy or computational requirements [242]. An alternative method, PaPaRa [118], uses a very different technique to align the sequences to the backbone alignment: it infers ancestral state vectors in the phylogeny, and uses these ancestral state vectors to align the fragmentary sequences to the backbone alignment. PaPaRa can give improved accuracy over HMMER when the rate of evolution is slow enough, but otherwise HMMER gives more accurate results [240].

New methods showing improvements over EPA and pplacer have also been developed [240, 245]. Brown and Truskowski [245] use hashing to speed up the method, while SEPP [240] uses a divide-and-conquer technique to speed up the method and improve the accuracy.

Phylogenetic placement can be used in metagenomic analyses [246, 247], in order to estimate the taxonomic identity (what species it is, what genus, etc.) of the short reads produced in shotgun sequencing of a metagenomic sample. When all the reads are drawn from the same gene, then phylogenetic placement can be used to identify the species for the read as described above: first, full-length sequences for the gene are obtained, then an alignment is estimated for the full-length sequences, and finally the reads are inserted into a taxonomy for the species, using the estimated alignment and the phylogenetic placement method. However, since the reads are not drawn from the same gene, then the metagenomic sample must first be processed so that the reads are assigned to genes (or else left unassigned), and then each "bin" of reads for a given gene can be analyzed as described. Thus, phylogenetic placement can be used in a pipeline as a taxon identification method, but the process is substantially more complicated.

6.3 Co-estimation Methods

Co-estimation of trees and alignments is an obvious approach for several reasons. First, an alignment, like a tree, is a hypothesis about the evolutionary history of the given data. To separate the two hypotheses prevents them from being mutually informative. Thus, rather than first estimating the alignment, treating it as fixed, and

then estimating the tree, a co-estimation procedure would try to find the tree and alignment at the same time. The challenge, of course, is how to do this.

In this section, we begin with a discussion of a co-estimation approach that seeks the tree with the minimum total edit distance, but where indels also count towards the total cost. We then continue with a discussion of co-estimation methods that use likelihood calculations on trees, under stochastic models that include indels as well as substitutions. Finally, we discuss SATCHMO-JS, SATé, and mega-phylogeny, methods that return an alignment and a tree from unaligned sequences; these methods were discussed earlier in the section on sequence alignment methods, and here we discuss them in the context of phylogeny estimation.

6.3.1 Treelength, or "Direct Optimization"

Probably the most commonly used approach to estimating the tree at the same time as estimating the alignment is the "Treelength" approach, also called "Direct Optimization" [248]. This is a natural extension of maximum parsimony to allow it to handle sequences that evolve with indels and so have different lengths.

In order to understand the approach, we begin by noting that a pairwise alignment between two sequences defines one or more edit transformations, each consisting of operations—some substitutions and some indels—that transform the first sequence into the second. Each indel event may be of a single letter (either a nucleotide or an amino-acid), or could be of a string of letters. The "cost" of the pairwise alignment is then the minimum cost of any transformation that is consistent with the alignment. Note that implicit in this definition is the limitation of the operations to just substitutions and indels; therefore, no more complicated operations (such as tandem repeats, inversions, etc.) are considered.

Similarly, each edit transformation that is based on indels and substitutions defines a pairwise alignment. In fact, pairwise alignments are typically computed using dynamic programming algorithms that explicitly compute the minimum cost edit transformation. Thus, there is a close relationship between edit transformations based on indels and substitutions and pairwise alignments.

We now define the treelength problem, where the tree is fixed, and each leaf is labeled by a sequence. The "length" of the tree would be computed by producing sequences at the internal nodes, and then calculating the edit distance between sequences on each edge, using the best possible labeling of the internal nodes (so as to minimize the output length). Since pairwise distances can be computed in these conditions, the length of a tree can be defined and computed, once the sequences at the internal nodes are provided. The treelength problem is then to find the best sequences for the internal nodes so that the total length is minimized.

Given this, we formalize the *Tree Alignment* problem as follows:

Definition 3 Given a rooted tree T on n leaves which is leaf-labeled by a set $S = \{s_1, s_2, \ldots, s_n\}$ of sequences over Σ (for any fixed alphabet Σ) and an edit cost function $c(\cdot, \cdot)$ for comparing any two sequences over Σ, find sequences to label

the internal nodes of T so as to minimize $\text{cost}(T) = \sum_{(v,w)\in E(T)} c(l_v, l_w)$, where l_x is the sequence labeling node x in T.

Note that given sequences at the internal nodes, then for each edge e there is a pairwise alignment of the sequences l_v and l_w labeling the endpoints of e whose cost is identical to $c(l_v, l_w)$. By taking the *transitive closure* of these pairwise alignments we obtain a multiple sequence alignment of the entire dataset whose total cost is $\text{cost}(T)$. Thus, the output of the Tree Alignment problem can be either considered to be the sequences at the internal nodes, or also the MSA that it defines on the sequences at the leaves of the tree.

The Tree Alignment problem has a rich literature, beginning with [249–251]. The Tree Alignment problem is NP-hard, even for simple gap penalty functions [252], but solutions with guaranteed approximation ratios can be obtained [253–255]. This contrasts with the maximum parsimony problem, which is polynomial time when the tree is fixed (i.e., the optimal sequences for the internal nodes of a given tree can be found in polynomial time using dynamic programming). Thus the treelength problem is *harder* than the maximum parsimony problem.

A generalization of this problem, named after David Sankoff due to his contributions [250, 251], is as follows:

Definition 4 (The Generalized Sankoff Problem (GSP) [From Liu and Warnow [256]]) The input is a set S of unaligned sequences and a function $c(x, y)$ for the edit cost between two sequences x and y. The output is a tree $T = (V, E)$ with leaves labeled by S and internal nodes labeled with additional sequences such that the treelength $\sum_{(v,w)\in E} c(l_v, l_w)$ is minimized, where l_x is the sequence labeling vertex x.

Not surprisingly, GSP is NP-hard, since the case in which the edit distance function forbids gaps (by setting the cost for a gap to be infinite) is the NP-hard Maximum Parsimony (MP) problem [146].

6.3.1.1 POY

The standard method for "solving" the GSP problem is POY [248, 257]. Note that in the literature discussing POY, the treelength problem is called "Direct Optimization" (or "DO" for short). POY handles only certain types of edit distances, and in particular, it only enables affine gap penalties. Thus, the cost of a gap of length L is given by $\text{cost}(L) = c_0 + c_1 L$, where c_0 is the gap open cost and c_1 is the gap extend cost. When $c_0 = 0$ the gap cost is said to be "simple" (a special case of affine) and when $c_0 > 0$ the gap cost is said to be "affine". POY also enables different costs for transitions and transversions. Thus, the input to POY is a set of unaligned sequences, values for c_0 and c_1, and the cost of transitions and transversions.

The use of treelength optimization to find good trees (and/or alignments) is a matter of substantial controversy in phylogenetics [98, 102, 227, 248, 258–261]. Most of

the studies that have examined the accuracy of POY trees and alignments explored performance under simple gap penalties, and found that POY did not produce trees and alignments of comparable accuracy to maximum parsimony on the ClustalW alignment, denoted MP(Clustal) [259, 262]. A later study examined how the gap penalty affected the accuracy of trees and/or alignments computed by POY [263], and evaluated POY under affine gap penalties (where the gap open cost is non-zero). They found a particular affine gap penalty, which they called "Affine", for which POY produced very good results, and in fact was competitive with MP(Clustal). This study seemed to suggest that POY, and hence the treelength optimization approach to estimating trees and alignments, could be used to find highly accurate trees provided that the right edit distances were used. However, it was also observed that POY was not always effective at finding the best solutions to the treelength problem, and hence the accuracy of POY's trees might not indicate any value in optimizing treelength.

6.3.1.2 BeeTLe: Better TreeLength

A subsequent study [256] revisited this question, by focusing on whether finding good solutions to treelength criteria would yield improved alignments and trees. In order to understand the impact of the treelength criterion, they developed a new technique for treelength optimization, called "BeeTLe" (Better TreeLength), which is guaranteed to find solutions that are at least as good as those found by POY. BeeTLe runs a collection of methods, including POY, to produce a set of trees on a given input set of unaligned sequences, uses POY to compute the treelength of each tree, and then returns the tree that had the shortest treelength. Thus, BeeTLe is guaranteed to find trees at least as short as those found using POY, and thus enables us to evaluate the impact of using treelength to find trees.

Here we present some results from Liu and Warnow [256], in which BeeTLe was compared to various two-phase methods on simulated 100-taxon datasets, in which sequences evolve with substitutions and indels. We show results for alignments estimated using BeeTLe, MAFFT and ClustalW, and MP and ML trees on the MAFFT, ClustalW, and true alignments. We present alignment error rates (SP-FN, the fraction of true homologies missing from the estimated alignment) and tree topology error rates (the missing branch rate, which is the fraction of edges in the true tree that are not in the estimated tree). We study BeeTLe under three different treelength criteria, each of which has unit cost for substitutions: "Simple-1", which sets the cost of every indel to 1; "Simple-2" (the treelength criterion studied by Ogden and Rosenberg [259] that they found to produce more accurate trees than any other treelength criterion they considered), which assigns cost 2 to indels and transversions and cost 1 to transitions; and "Affine" (the treelength criterion studied in Liu et al. [263] that produced more accurate trees than Simple-1 or Simple-2), which sets the cost of a gap of length L to $4 + L$.

In Fig. 6.2, we see that BeeTLe-Affine (BeeTLe using this affine gap penalty) produces the most accurate trees of all BeeTLe variants. We also see that BeeTLe-Affine improves on MP on ClustalW alignments, and matches MP on MAFFT

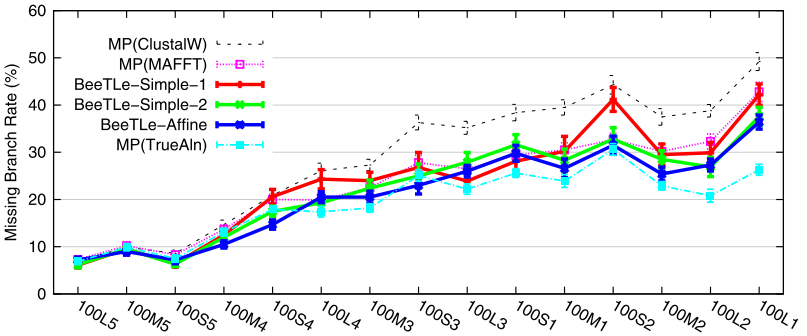


Fig. 6.2 Comparing BeeTLe to MP-based analyses. We report missing branch rates on 100-taxon model conditions for BeeTLe (under three gap penalty treatments) in comparison to maximum parsimony on the ClustalW, MAFFT, and true alignment (TrueAln). Averages and standard error bars are shown; $n = 20$ for each reported value (This figure appeared in Liu and Warnow [256])

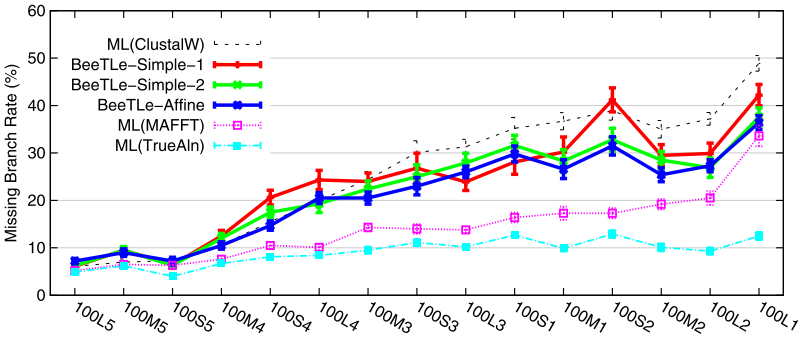


Fig. 6.3 Comparing BeeTLe to ML-based analyses. We report missing branch rates on 100-taxon model conditions for BeeTLe (under three gap penalty treatments) in comparison to maximum likelihood on ClustalW, MAFFT, and the true alignment (TrueAln). Averages and standard error bars are shown; $n = 20$ for each reported value (This figure appeared in Liu and Warnow [256])

alignments. It also is fairly close to MP on true alignments, except for the hardest 100-taxon model conditions. In Fig. 6.3, we see a comparison of BeeTLe to ML trees computed on ClustalW, MAFFT, and the true alignment. Note how BeeTLe-Affine often produces more accurate trees than ML(ClustalW), but (with the exception of the very easiest model conditions, where there is very little difference between methods), also produces substantially less accurate trees than ML(MAFFT) and ML(TrueAln).

An evaluation of the alignment error on the same datasets (Fig. 6.4) shows that BeeTLe alignments generally have very high alignment SP-FN error, and that BeeTLe-Affine has lower SP-FN error than the other BeeTLe variants. The comparison to ClustalW shows that BeeTLe-Affine is less accurate on some models and more accurate on others; however, neither ClustalW nor BeeTLe-Affine comes close to the accuracy of MAFFT, except on the easiest 100-taxon models.

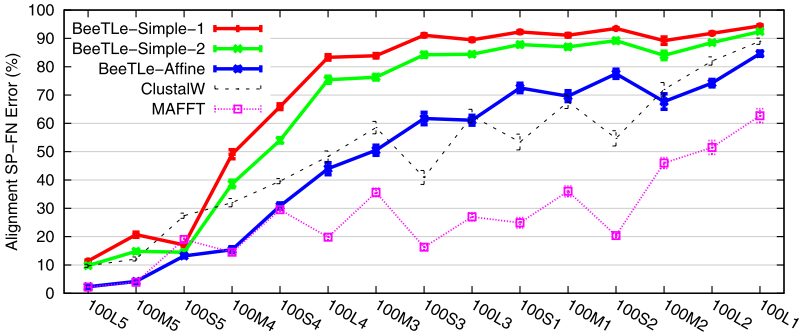


Fig. 6.4 A comparison of alignment error for BeeTLe to other methods. We show alignment SP-FN error of BeeTLe in comparison to MAFFT and ClustalW on 100-taxon model conditions. Averages and standard error bars are shown; $n = 20$ for each reported value (This figure appeared in Liu and Warnow [256])

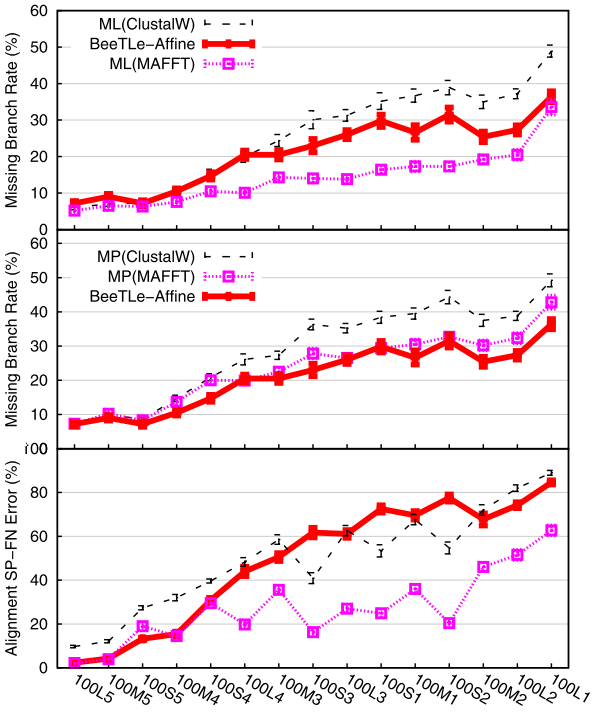


Fig. 6.5 Performance of BeeTLe-Affine. We compare the BeeTLe-Affine trees to MP and ML trees on ClustalW and MAFFT alignments, and we also compare the alignments obtained by BeeTLe-Affine, ClustalW, and MAFFT, on 100-taxon model conditions. Averages and standard error bars are shown; $n = 20$ for each reported value. (This figure appeared in Liu and Warnow [256])

Figure 6.5 shows the direct comparison between BeeTLe-Affine and alignments and trees estimated using ClustalW and MAFFT. Since Fig. 6.4 already gave the comparison with respect to alignment error, we focus only on tree estimation. Note that BeeTLe-Affine generally gives more accurate trees than MP(Clustal) and MP(MAFFT), except on the easiest models where they are all equally accurate.

However, when compared to ML-based trees, the best results are clearly obtained using ML(MAFFT), with BeeTLe-Affine in second place and ML(Clustal) in last place.

6.3.1.3 Summary Regarding the Treelength Problem

Recall that BeeTLe is guaranteed to produce solutions to treelength optimization that are at least as good as POY, and in fact BeeTLe generally produces shorter trees than POY, as shown in Liu and Warnow [256]. Therefore, the performance of BeeTLe with respect to tree and alignment accuracy is a better indication of the consequences of using treelength for estimating alignments and trees than POY. As shown here, however, although improvements can be obtained by using this particular affine gap penalty (compared to the simple gap penalties that were examined), the alignments and trees are not as accurate as those produced using the better alignment methods (e.g., MAFFT) followed by maximum likelihood.

We note that the use of affine gap penalty treatments, although more general (and hence better) than simple gap penalties, are not necessarily sufficiently general for alignment estimation [264–268]; therefore, better trees might be obtained by optimizing treelength under other gap penalty treatments. In the meantime, the evidence suggests that optimizing treelength using the range of gap penalty treatments in common use (simple or affine penalties) is unlikely to yield the high quality alignments and trees that are needed for the best phylogenetic analyses.

6.3.2 Statistical Co-estimation Methods

Methods that co-estimate alignments and trees based upon statistical models of evolution that incorporate indels have also been developed. The simplest of these models are TKF1 [269] and TKF2 [270, 271], but more complex models have also been developed [272–278]. Many statistical methods (some which estimate trees from fixed alignments, and some which co-estimate alignments and trees) have been developed based on these models [273, 275, 276, 279–286], but only BAli-Phy [276] has been shown to be able to co-estimate alignments and trees on datasets with 100 sequences; the others are limited to much smaller datasets [287]. A recent technique [288] may be able to speed up calculations of likelihood under models that include indels and substitutions, but to date, none of the co-estimation methods has been able to run on datasets with more than about 200 sequences.

6.3.3 Other Co-estimation Methods

In addition to the statistical co-estimation methods described above, several methods are designed to return trees and alignments given unaligned sequences as in-

put. Here we discuss three of these methods, SATé, SATCHMO-JS, and mega-phylogeny, each of which was discussed in the earlier section on alignment estimation. Of these three methods, SATé and mega-phylogeny are methods that compute an alignment and a maximum likelihood tree on the alignment, where mega-phylogeny uses RAxML and SATé uses either RAxML or FastTree-2 (depending on the user's preference). Although mega-phylogeny has been used on empirical datasets, to our knowledge it has not been tested on benchmark datasets, and so its performance (in terms of alignment and/or tree accuracy) is more difficult to assess. Therefore, we focus the rest of the discussion here on SATé and SATCHMO-JS.

6.3.3.1 SATé

SATé ("Simultaneous Alignment and Tree Estimation") [10, 39] is a method that was designed to estimate alignments and trees on very large datasets. SATé was discussed earlier in the section on multiple sequence alignment; here we focus on its performance as a method for estimating trees.

Unlike the statistical methods discussed earlier that explicitly consider Markov models of evolution that include indels and thus can have performance guarantees under such models, SATé has no such guarantees. Instead, the design of SATé is guided by the empirical objective of improving the accuracy of alignment and phylogeny estimations on very large datasets.

The observations that led to SATé come from studies that showed that existing nucleotide alignment methods had poor accuracy on large datasets that evolve down trees with high rates of substitutions and indels, and that some of the most accurate alignment estimation methods (e.g., MAFFT) have computational requirements (sometimes due to memory usage) that makes them unable to be run in their most accurate setting on datasets above a relatively small number of sequences. Therefore, while small datasets can be aligned with the best alignment methods, larger datasets must be aligned with less accurate methods. These observations together guided the design of SATé, which we now describe.

SATé uses an iterative process, in which each iteration begins with the tree from the previous iteration, and uses it (within a divide-and-conquer framework) to realign the sequence dataset. Then a maximum likelihood tree is estimated on the new alignment, to produce a new tree. The first iteration begins with a fast two-phase method (for example, FastTree-2 on a MAFFT-PartTree alignment). The first few iterations provide the most improvement, and then the improvements level off. The user can provide a stopping rule based upon the number of iterations, the total amount of clock time, or stopping when the maximum likelihood score fails to improve.

Although iteration is an important aspect of SATé's algorithm design, the divide-and-conquer strategy used by SATé is equally important, and the strategy has changed since its initial version. In its first version [10], SATé divided the dataset into subsets by taking a centroid branch in the tree (which divides the dataset roughly into two equal parts) and branching out until the desired number of subsets

is produced (32 by default, but this value could change, based on the dataset size). Each subset was then aligned using MAFFT in a highly accurate setting(-linsi), and the subset alignments were merged together using Muscle. Then, an ML tree was estimated on the resultant alignment using RAxML. SATé iterated until 24 hours had elapsed (finishing its final iteration if it began before the 24 hour deadline). SATé returns the tree/alignment pair with the best ML score.

This approach led to very good results, as shown in Fig. 6.6, where we compare SATé to two-phase methods on 1000-taxon model trees. We include RAxML on ClustalW, Muscle, Prank+GT (Prank with a RAxML(MAFFT) guide tree), and the true alignment (TrueAln). The first two panels show the tree and alignment error for these methods. Note that on the easiest model conditions all methods produce the same level of accuracy as RAxML on the true alignment, although the estimated alignments have errors. However, on the harder models the phylogenetic estimations have different error rates, and SATé produces much more accurate trees and alignments than the other methods. Furthermore, SATé comes close to RAxML on the true alignment for all but the hardest models. The third panel gives the empirical statistics for the different models, and shows that factors that lead to datasets that are hard to align include the percent of the true alignment matrix that is gapped ("Percent indels") and the average p-distance[4] between pairs of sequences (again, based on the true alignment). Thus, alignments can be quite easy to estimate if the rate of substitutions is low enough (as reflected in the average p-distance), even if there are many indels, and it is only when the substitution rate is high enough and there are at least a moderate number of indels that alignments become difficult to estimate.

Recall that SATé generates a sequence of alignments and trees, and that the tree and alignment it outputs is the pair that has the best ML score. The fourth panel shows the results of a correlation analysis we performed to see if the ML score was correlated with either alignment accuracy or tree accuracy. What we observed is that tree error (measured using SP-FN) and ML score are very weakly correlated on the easier models, but then highly correlated on the harder models, while alignment error and ML score are highly correlated on all models. This correlation analysis suggests that using the ML score to select an alignment *might* be a good idea. It also suggests that when the conditions are such that improving the alignment would improve the tree (which seems to be the case for harder models rather than easier models), using the ML score to select the tree might also be a good idea. In other words, it suggested the following optimization problem:

Definition 5 The *ML_{GTR} Tree/Alignment Search Problem*: Given a set S of unaligned sequences, find a tree T and an alignment A of S so as to maximize likelihood under GTR, treating gaps as missing data.

[4]The p-distance between two aligned sequences is the number of positions in which the two sequences differ, and then normalized to give a number between 0 and 1.

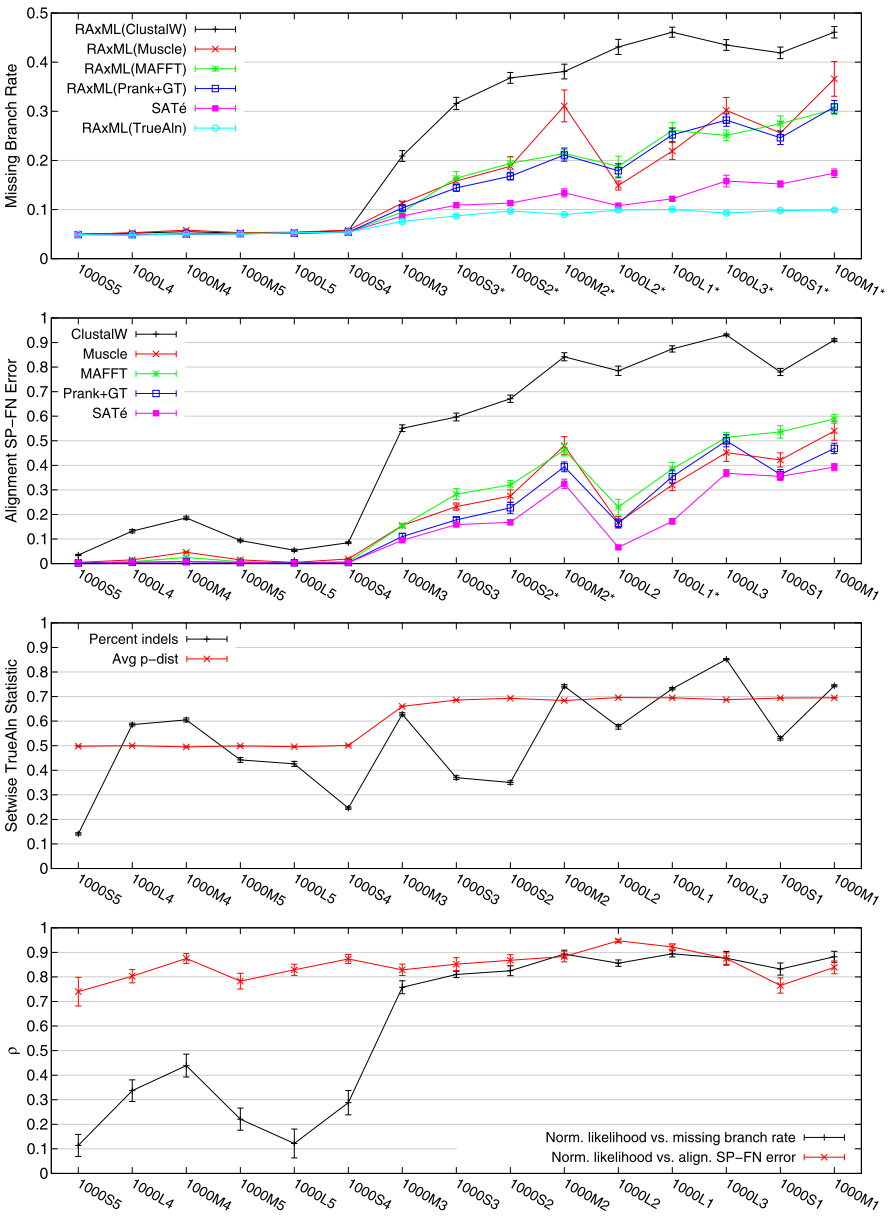


Fig. 6.6 Performance of SATé on 1000-taxon model results. X-axes have the 15 1000-taxon models roughly sorted with respect to the phylogenetic estimation error, based on missing branch rates. The *bottom two panels* show true alignment (TrueAln) setwise statistics and Spearman rank correlation coefficients (ρ). All data points include standard error bars. For the *top two panels*, models on the x-axis followed by an *asterisk* indicate that SATé's performance was significantly better than the nearest two-phase method (paired t-tests, setwise $\alpha = 0.05$, $n = 40$ for each test) (This figure appeared in Liu et al. [10])

However, the Jukes–Cantor version of this optimization problem is *not* a good way of trying to optimize alignments or trees, as we now show. Recalling the definition of "monotypic" alignments (see Sect. 6.2.3.6), we proved:

Theorem (From Liu et al. [39]) *If alignments are allowed to vary arbitrarily, then for all input sets of sequences, the alignment that gives the best Jukes–Cantor ML score (treating gaps as missing data) is monotypic, and every tree is an optimal solution for this alignment.*

Thus, optimizing the ML score while allowing the alignments to change arbitrarily is not helpful. Given that we observed a correlation between the ML score and alignment and tree accuracy generated during a SATé analysis, how does this make sense? The explanation between these two seemingly contradictory statements is that the set of tree/alignment pairs in which we observed the correlation is not arbitrary. Instead, the set of alignments computed during a SATé run is not random at all, nor are the alignments and trees in that set explicitly modified to optimize ML. Instead, the alignments are computed using a divide-and-conquer strategy where MAFFT is used to align subsets and Muscle is used to merge these subset alignments. Thus, although allowing alignments to change arbitrarily is definitely not desirable (and will lead to an optimal ML score but poor trees), using ML to select among the alignments and trees produced during the SATé process *may* be beneficial.

SATé-II Subsequent studies revealed that for some datasets, the SATé analysis was very slow, and this turned out to be due to some subset sizes being very large—too large, in fact, for MAFFT to comfortably analyze the dataset using its most accurate setting (-l -insi). A careful analysis revealed that these large subsets came about as a result of the specific decomposition we used (consider, for example, the result of using the SATé-decomposition on a caterpillar tree). We then changed the decomposition technique, as follows. The new decomposition keeps removing centroid edges in the subtrees that are created until every subtree has no more than a maximum number of leaves (200 by default). As a result, all the subsets are "small", and MAFFT -l -insi can comfortably analyze each subset. Note also that these subsets are not necessarily clades in the tree! The resultant version of SATé, called SATé-II, produces trees and alignments that are even more accurate than SATé, and is also faster. This is the version of SATé that is in the public distribution.

6.3.3.2 SATCHMO-JS

SATCHMO-JS [78] is another co-estimation method designed for empirical performance, and specifically for protein sequence alignment. SATCHMO stands for "Simultaneous Alignment and Tree Construction using Hidden Markov models", and SATCHMO-JS stands for "jump-start SATCHMO". The basic approach here has three steps. First, SATCHMO-JS computes a neighbor joining tree on a MAFFT

alignment, and uses this tree to divide the dataset into clades, each of which has a maximum pairwise p-distance that is below some threshold. For each such subset, a tree is computed on the alignment using the SciPhy method [289]. These subtrees are then passed to the SATCHMO [77] method, which then completes the task of computing an overall alignment and tree. Finally, the branch lengths on the tree are optimized using maximum likelihood.

Thus, SATCHMO-JS is a hybrid between MAFFT and SATCHMO that combines the best of the two methods—MAFFT to align closely related sequences, and SATCHMO to align distantly related sequences. Since a tree is estimated at the same time as the alignment is estimated, the output is both a tree and an alignment—hence the name of the method.

As shown by Hagopian et al. [78], SATCHMO-JS produced more accurate alignments than MAFFT, SATCHMO, ClustalW and Muscle on several protein benchmarks. Furthermore, although by design SATCHMO-JS is slower than MAFFT (since it runs MAFFT to obtain its initial decomposition into subsets), it is much faster than SATCHMO, and was able to analyze a dataset with 500 protein sequences of moderate length (392 aa) in about 18 minutes.

It is worth noting that the way SATCHMO determines the tree cannot be described as finishing the alignment estimation and then computing a tree on that alignment; this distinguishes SATCHMO-JS from SATé and mega-phylogeny, which always return a maximum likelihood tree on the output alignment.

6.4 Tree Estimation Without Full Alignments

Because multiple sequence alignment estimation tends to be computationally intensive and inaccurate on large datasets that evolve with high rates of evolution [4, 6], estimating trees without any multiple sequence alignment step has obvious appeal. In this section, we first discuss alignment-free estimation, and then estimation methods that use multiple sequence alignment estimation on subsets but not on the full set of taxa.

6.4.1 Alignment-Free Estimation

Potential Benefits of Alignment-Free Estimation There are several reasons that alignment-free estimation has been considered promising, which we briefly discuss here (see [290, 291] for longer and more detailed discussions). First, recall that standard multiple sequence alignments insert gaps between letters within sequences in order to "align" them, and hence only model homologies that result from substitutions and indels. However, genome-scale evolution is very complex, involving recombination, rearrangements, duplications, and horizontal gene transfers, none of which is easily handled in standard multiple sequence alignments. Thus, one of

the points in favor of alignment-free estimation methods is that they may be more robust to these genome-scale events (rearrangements, recombination, duplications, etc.) than alignment-based methods. Another point in favor is that alignment-free methods are able to avoid the need to do each step of the standard analysis pipeline, and hence can be robust (possibly) to the difficulties in identifying orthology groups, aligning sequences, estimating gene trees, and combining gene trees and/or alignments. Thus, alignment-free estimation may be robust to some of the methodological challenges inherent in the standard phylogenetic pipeline.

Alignment-free estimation also have another distinct advantage, in that they enable the use of all the nucleotides in the genomes, not just the ones that fall into the regions identified for the phylogenomic analysis. Thus, alignment-free methods have the potential to utilize more of the genomic data, and this could enable more accurate trees.

Finally, as we shall see, alignment-free estimation is in general very fast, especially for datasets that involve many markers and/or many taxa. This is one of the big advantages over the standard analysis pipeline.

History of Alignment-Free Estimation The first method for alignment-free phylogeny estimation was developed in 1986 [292], and new methods continue to be developed [290, 291, 293–295].

Alignment-free methods are based upon computing distances (often, but not always, by computing k-mer distributions) between unaligned sequences. Once these distances are computed, trees can be computed on the resultant distance matrix, using distance-based methods (e.g., the well-known neighbor joining [121] method, but there are many others). Because both steps can be quite fast, these alignment-free methods can be applied to very large genomes.

While some of the alignment-free techniques for estimating pairwise distances are fairly heuristic, others use sophisticated statistical techniques, and some have provable performance under Markov models of evolution. A particularly exciting result is by Daskalakis and Roch [237], who gave a polynomial time distance-based alignment-free method and proved that it is statistically consistent under the TFK1 model [269].

These methods have shown some promise, as some simulation studies evaluating trees based on these distances have shown that these can be more accurate than trees based on distances calculated using estimated multiple sequence alignments [296]. Furthermore, plausible phylogenetic trees have been estimated on biological datasets using these methods [290, 291].

Comparison to Two-Phase Methods Despite all the potential benefits of alignment-free estimation, there has not been any comparison of alignment-free methods to the most accurate ways of computing large trees, e.g., Bayesian or maximum likelihood analyses on good alignments. Instead, the only comparisons have been to distance-based phylogeny estimation, and in some cases only to distances computed using other alignment-free methods. Therefore, while there is distinct potential for alignment-free estimation to provide improved species tree estimations, this possibility has not been properly evaluated.

However, as noted before, alignment-free estimation does provide a distinct computational advantage over the standard pipelines, and it can enable the use of *all* of the genomic data. Thus, even if alignment-free estimation is not as accurate as maximum likelihood on alignments, the ability to use more data may offset the possible reduction in accuracy. The question might then become *whether more data analyzed using a less accurate method is as good as less data analyzed using a more accurate method*!

6.4.2 DACTAL

DACTAL [30], which stands for "Divide-And-Conquer Trees (almost) without ALignments", is a method that estimates trees without requiring an alignment on the full dataset. Unlike the methods discussed in the previous subsection, DACTAL is not truly alignment-free. Instead, DACTAL uses an iterative divide-and-conquer strategy, and estimates alignments and trees on small (and carefully selected) subsets of taxa. By ensuring that the subsets have sufficient overlap, this makes it possible to produce a tree on the full set of taxa. The key to making this work well is ensuring that the taxon sampling in each subset is favorable to tree and alignment estimation, the subsets are small enough that the best alignment and tree estimation methods can be used on them, and that the overlap patterns enable a highly accurate supertree to be estimated [297].

A DACTAL analysis can be initiated in one of several ways. The simplest way is to obtain a starting tree for the dataset (e.g., a taxonomy for the dataset, or an estimated tree obtained using a fast two-phase method). This starting tree is then used to decompose the dataset into small overlapping subsets of sequences that are close together in the starting tree, using the "PRD" technique [30, 298]. Alternatively, this decomposition into small overlapping subsets of similar sequences can be obtained using one of several BLAST-based decompositions [298].

Once this decomposition is computed, alignments and trees are computed on each subset, and the subtrees are merged into a tree on the full set of taxa using SuperFine [53, 299], a new supertree method that can produce more accurate trees on large datasets than other supertree methods [54, 300]. The resultant tree is then used to start the next iteration, which continues with a decomposition of the taxa into small, overlapping subsets, the estimation of alignments and trees on each subset, and the merger of the subset-trees into a tree on the full dataset. Finally, this iterative process (shown in Fig. 6.7) continues for a user-defined maximum number of iterations.

DACTAL has comparable accuracy to SATé-I (the initial implementation of SATé as described by Liu et al. [10]), and substantially improved accuracy compared to the leading two-phase methods. on datasets with 1000 or more sequences. In addition, DACTAL is faster than SATé-I, and gives very good accuracy on large biological datasets.

Figure 6.8 [30] shows the results for five iterations of DACTAL on three large rRNA datasets with up to 27,643 sequences, in comparison to maximum likeli-

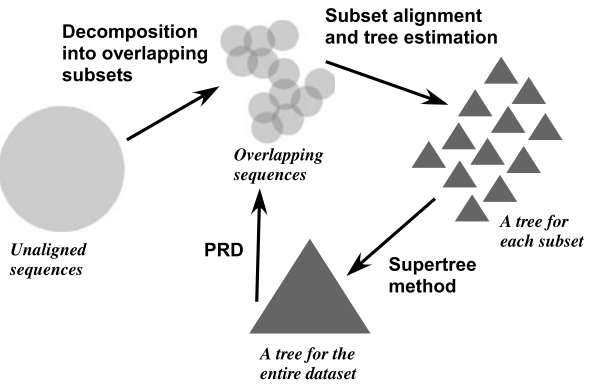


Fig. 6.7 DACTAL algorithmic design. DACTAL can begin with an initial tree (*bottom triangle*), or through a technique that divides the unaligned sequence dataset into overlapping subsets. Each subsequent DACTAL iteration uses a decomposition strategy called "PRD" (padded recursive decomposition) to divide the dataset into small, overlapping subsets, estimates trees on each subset, and merges the small trees into a tree on the entire dataset (This figure appeared in Nelesen et al. [30])

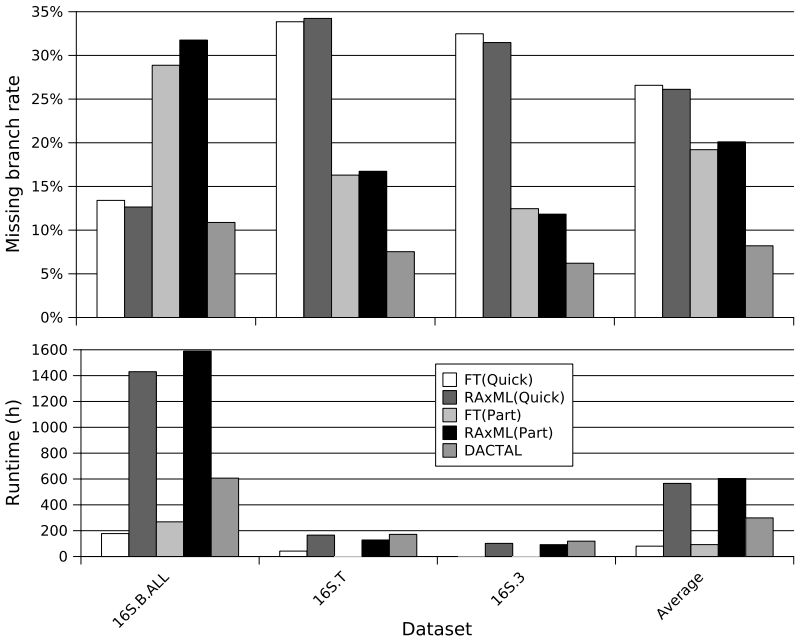


Fig. 6.8 DACTAL (based upon five iterations) compared to ML trees computed on alignments of three large biological datasets with 6,323 to 27,643 sequences. We used FastTree-2 (FT) and RAxML to estimate ML trees on the MAFFT-PartTree (Part) and ClustalW-Quicktree (Quick) alignments. The starting tree for DACTAL on each dataset is FT(Part) (This figure appeared in Nelesen et al. [30])

hood trees (computed using FastTree-2 and RAxML) on two alignments (Clustal-Quicktree and MAFFT-PartTree). DACTAL was run using a subset decomposition size of 200 and RAxML(MAFFT) to estimate trees on the subsets. Each of these datasets has a reliable curated alignment based on rRNA structure [301]. The reference trees for this study were obtained by estimating maximum likelihood trees (using RAxML with bootstrapping) on the curated alignment, and then collapsing all branches with bootstrap support below 75 %. Note the substantial reduction in tree error obtained using DACTAL in comparison to these two-phase methods. Furthermore, other analyses show that DACTAL is highly robust to its starting tree, and that even a single iteration produces a large improvement [30].

6.5 Lessons Learned and Future Directions

6.5.1 Basic Observations

This chapter has introduced several new methods and techniques for both alignment and phylogeny estimation, and shown that highly accurate large-scale estimation is beginning to be possible. However, there are several recurring themes in this chapter, which point to the limitations of the current research, and the need for future work. These are:

1. Only a few techniques have been developed that are capable of large-scale alignment or phylogeny estimation, and very few methods have been even tested on large datasets,
2. The standard criteria used to evaluate alignment estimation methods (e.g., the SP- and TC-scores) are not suited to predicting accuracy with respect to tree estimation, especially for large datasets,
3. Many promising alignment estimation methods have not been formally tested on simulated or biological benchmark datasets for their impact on phylogeny estimation,
4. The relative performance of phylogeny estimation methods (and perhaps of alignment estimation methods) can change with the number of taxa (e.g., compare HGT+FP and NJ in Fig. 6.1), and so performance studies that only examine small datasets are not predictive of performance on larger datasets, and
5. In general, the simulation studies used to evaluate alignment or phylogeny estimation methods have been based on very simple models of evolution, and so it is not clear how well these methods will perform under more realistic conditions.

These limitations are due to a number of factors, one being that many (though not all) of the alignment estimation methods were developed for use by the structural biologists, and hence were tested with respect to the ability to identify structural and functional features. Since structural and functional homology may not be identical to positional homology, this contributes to the issues raised above. Furthermore,

the two communities—phylogenetics and structural biology—are still fairly disconnected, and alignment methods continue to be tested almost exclusively on structural benchmarks, typically based on protein datasets. Bridging the gaps between the various disparate communities, including phylogenetics, structural biology, and functional genomics, may be necessary in order to change this practice.

Many of these issues also reflect the challenges in evaluating phylogeny estimation methods, especially on large datasets. For example, although we know a great deal about the performance of methods in terms of statistical consistency under the General Markov model and simpler models of sequence evolution, much less is known mathematically about performance on finite data, and even less about the performance under more complex models.

As a whole, these observations highlight the importance of benchmark datasets (whether biological or simulated), since mathematical results are typically limited to statistical consistency or model identifiability. However, biological datasets that are appropriate for evaluating phylogeny estimation methods are rare, since the true tree is rarely known. In our own studies, we have used biological datasets with curated alignments, and used maximum likelihood bootstrap trees estimated on these curated alignments as the benchmark trees. This approach has the advantage of being phylogenetically based, but has two disadvantages: the alignment itself (however well curated) may not be correct, and even if it is correct, the tree estimated on the alignment may also not be correct. Thus, more—and better—biological benchmarks are needed.

As noted, simulation studies are a standard technique used to evaluate phylogenetic estimation methods, but designing good simulation studies and extracting general principles from them is not easy. Among the many challenges, the first is that the models available in most (but not all) simulator software are too simple, generally no more complex than the models available in the phylogeny estimation software; thus, data generated by most simulation software do not exhibit the properties of biological data. Another challenge is that the relative performance of methods can change with the model condition, and exploring the parameter space is computationally infeasible. Perhaps because of this, many studies have focused on small trees, where more thorough exploration of the parameter space is feasible. However, computational challenges in analyzing large datasets also explains why few studies have examined performance on large datasets, and relatively little is known about performance on large datasets. All these issues add to the challenge in developing good benchmarks, and thus of understanding phylogeny estimation methods—especially on large datasets.

6.5.2 Future Research Directions

As noted, substantial progress towards developing methods that can estimate alignments and trees on very large datasets has been made. For example, we have presented new approaches for estimating alignments and trees, for co-estimating align-

ments and trees, and for phylogenetic placement, each of which has provided substantial improvements in accuracy (and in some cases scalability) over previous methods. Here we discuss additional research questions that remain, acknowledging that these are just a small sample of the open problems in this area.

6.5.2.1 Theoretical Performance of Phylogeny Estimation Methods Under Long Indel Models

While much is known about the theoretical performance of phylogeny estimation under indel-free models of evolution, much less is known about the statistical guarantees of methods when sequences evolve with indels; even the result by Daskalakis and Roch [237] establishing statistical consistency under a model with indels only allows indels of single nucleotides. Open questions here include the statistical guarantees (both statistical consistency and sequence-length requirements) for methods under models with long indels given the true alignment, and when alignments must be estimated.

6.5.2.2 Sequence-Length Requirements for Phylogeny Estimation

While much is known about the sequence-length requirements under the General Markov model for many methods, several questions remain. For example, the best published upper bound on the sequence-length requirement for maximum likelihood is no better than that of neighbor joining [166], but this bound is likely to be loose, as suggested by Roch [302]. Thus, a basic question is whether maximum likelihood is an absolute fast converging method? Other questions of this sort include the sequence-length requirements of methods to recover some fixed percentage of the model tree bipartitions, or to estimate the model tree under more complex models than the General Markov model. Finally, from an empirical standpoint, it would be good to have implementations of absolute fast converging methods so that these methods can be compared to other methods, such as maximum likelihood.

6.5.2.3 Genome Rearrangements and Duplications

Genomes evolve with duplications, rearrangements (inversions, transpositions, transversions), fissions and fusions, and alignment and phylogeny estimation in the presence of these events is very complicated. While some work has been done on the problem of estimating whole-genome alignments [303–311], much still needs to be done.

The problem of estimating trees in this case is similarly challenging, and is the subject of a chapter in this volume by Bernard Moret et al. A basic open problem here is whether genome-scale events and sequence evolution events can be analyzed together, since they are likely to be complementary.

6.5.2.4 Evolutionary Networks

The objective of finding a "Tree of Life" presents a very basic challenge, since not all evolution is tree-like (e.g., horizontal gene transfer [312–317] and hybridization [318–320]); thus, the phrase "Tree of Life" is in a sense a misnomer, as discussed by Mindell [321].

The failure of trees to completely represent the evolutionary history is most obvious in the case of hybridization, where two species hybridize to make a new species; clearly, this history cannot be represented by a tree, and instead requires a network (i.e., a graph that has cycles). However, in the case of horizontal gene transfer, the underlying species history may still be reasonably represented by a tree [45], and edges representing the HGT events can be added to the tree to create a network. (This is how the phylogenetic network for language evolution is obtained, where horizontal edges represent "borrowing" between languages [322, 323].) Thus, in the event of hybridization or horizontal gene transfer, the evolutionary history is best represented by a network, though the specific representation (and meaning of edges) can differ between these networks.

In the literature, the term "phylogenetic network" [324–326] has been used to describe these graphical models, but, as Morrison points out [325], this term is used for more than one purpose. That is, there are two types of networks that have been proposed: one type is suited for exploratory data analysis (EDA) and another that is suited for a hypothesis of evolutionary history. Morrison suggests that networks that are best suited for EDA of phylogenetic data should be referred to as "Data-Display Networks", and that networks that are graphical representations of a reticulate evolutionary history should be referred to as "Evolutionary Networks"; in accordance with his suggestions, we will use these terms here.

This distinction is important, since a data-display network does not provide any direct information about the evolutionary history. As Morrison says [325, p. 47],

> The basic issue, of course, is the simple fact that data-display networks and evolutionary networks can *look* the same. That is, they both contain reticulations even if they represent different things… Many people seem to have confused the two types of network, usually by trying to interpret a data-display network as an evolutionary network… The distinction between the two types of network has frequently been noted in the literature, so it is hardly an original point for me to make here. Interestingly, a number of authors have explicitly noted the role of display networks in exploratory data analysis and then proceeded to treat them as genealogies anyway. It is perhaps not surprising, then, that non-experts repeatedly make the same mistake.

Both types of networks serve valuable purposes (as noted also by Morrison), but the purposes are different. However, there are many more methods that produce data-display networks than evolutionary networks, and the estimation of evolutionary networks is much more complicated and challenging. For examples of some of the few evolutionary network methods, see [327–333].

Note that while the species history is not tree-like, the evolution of individual genes may still be treelike. However, since reticulate events can result in genes with different trees, the detection of differences, sometimes strongly supported differences, between gene trees can indicate that some kind of reticulation may have

occurred. However, gene trees can also be incongruent under other conditions, including cases where the species history is still treelike; a prime example of this is incomplete lineage sorting [40, 42, 57].

Many challenges remain for estimating evolutionary histories in the case of these events. One obvious challenge is the estimation of the underlying species tree from either gene sequence alignments or trees, for the case where genes evolve with horizontal gene transfer but without hybridization. Lapierre et al. [334] performed a simulation study to evaluate supertree and supermatrix methods for estimating the species tree, and found that supermatrix methods gave better results when there was only small amounts of horizontal gene transfer, and supertree methods were better when there were many horizontal gene transfers. A new method for estimating the underlying species tree, and a probabilistic analysis of this method, has also been developed by Roch and Snir [335], but has not been evaluated in simulation or on real data. New methods, and evaluations of these methods, are clearly needed.

In addition, since there are very few methods for estimating evolutionary networks, more work in this area needs to be done. However, all evolutionary networks—especially those that operate by combining estimated gene trees—need to be able to distinguish incongruence between estimated gene trees that is due to true reticulation (as in lateral gene transfer or hybridization), incongruence that is due to incomplete lineage sorting or gene duplication and loss (for which a species tree still makes sense), and incongruence due to estimation error. Recent progress has been made in developing statistical methods for distinguishing between different causes for incongruence, and for estimating evolutionary histories given a mixture of different events [333, 336, 337]. While these are still in their infancy, the potential for substantial advances in phylogenomic analysis could be very high.

6.5.2.5 Incorporating Biological Knowledge into Alignment and Phylogeny Estimation

One of the most interesting developments in both alignment and phylogeny estimation is the attempt to utilize biological knowledge, especially about structure, into the process [102]. Although structurally based alignments may not always reflect positional homology [100], the use of external knowledge has greatly impacted the alignment of sets of protein sequences that are close to the twilight zone, in which sequence similarity can be close to random while structural properties may still be conserved. A similar effort is occurring on the phylogeny estimation side, so that the Markov models used in phylogeny estimation (especially those involving protein analyses) are becoming more realistic [138, 139]. For example, Liberles et al. [139] say:

> At the interface of protein structure, protein biophysics, and molecular evolution there is a set of fundamental processes that generate protein sequences, structures and functions. A better understanding of these processes requires both biologically realistic models that bring structural and functional considerations into evolutionary analysis, and similarly incorporation of evolutionary and population genetic approaches into the analysis of protein

> structure and underlying protein biophysics... The potential benefits of the synergy between biophysical and evolutionary approaches can hardly be overestimated. Their integration allows us not only to incorporate structural constraints into improved evolutionary models, but also to investigate how natural selection interacts with biophysics and thus explain how both physical and evolutionary laws have shaped the properties of extant macromolecules.

But, as Claus Wilke said in "Bringing Molecules Back into Molecular Evolution" [138]:

> A side effect of the strong emphasis on developing sophisticated methods for sequence analysis has been that the underlying biophysical objects represented by the sequences, DNA molecules, RNA molecules, and proteins, have taken a back-seat in much computational molecular evolution work. The vast majority of algorithms for sequence analysis, for example, incorporate no knowledge of biology or biochemistry besides that DNA and RNA sequences use an alphabet of four letters, protein sequences use an alphabet of 20, and the genetic code converts one into the other. The choice to treat DNA, RNA, and proteins simply as strings of letters was certainly reasonable in the late 20th century. Computational power was limited and many basic aspects of sequence analysis were still relatively poorly understood. However, in 2012 we have extremely powerful computers and a large array of highly sophisticated algorithms that can analyze strings of letters. It is now time to bring the molecules back into molecular evolution.

Both Wilke and Liberles et al. point out excellent work that is being done to add more biological realism into existing models and methods, and are passionate about the potential benefits to science that could result. However, the challenges to using more realistic models in phylogenetic estimation are enormous. As Wilke's article suggests, the added complexity in these models will lead to increased computational challenges (e.g., more parameters to estimate for maximum likelihood, or longer MCMC runs to reach convergence for the Bayesian methods). However, there are other challenges as well: as discussed earlier, increased model complexity can lead to non-identifiable models, making inferences under the model less reliable and interpretable. Furthermore, it is not always the case that adding complexity (even if realistic) to a model will improve inferences under the model, and it may be that phylogeny estimation under simpler, not necessarily as realistic models, may give the most accurate results.

However, even this question depends on what one is trying to estimate. Is it just the gene tree, or also the numerical parameters on the tree? If more than just the topology, then which parameters? Or is the tree itself just a nuisance parameter, and the objective something else? Indeed, it is possible that more parameter rich and biologically realistic models may not have a substantial impact on phylogeny estimation, but they may improve the detection of selection, the estimation of dates at internal nodes in the tree, and the inference of protein function and structure. That said, the prospects for improved accuracy in biological understanding through these new approaches that integrate biological knowledge with statistical methods may be large, and are worth investigating.

A final point about biologically realistic models: even if they turn out not to be particularly useful for estimation, they are certainly useful for simulation! That is, simulating under more realistic models and then estimating under simpler models

allows us to determine the robustness of the method to model violations, and to predict performance under more realistic conditions. Therefore, the development of improved statistical models of evolution may be important for testing methods, even if the use of these models for inference turns out to be impractical.

6.6 Conclusions

This chapter set out to survey large-scale phylogeny and alignment estimation. As we have seen, large-scale alignment and phylogeny estimation (whether of species or of individual genes) is a complicated problem. Despite the multitude of methods for each step, the estimation of very large sequence alignments or gene trees is still quite difficult, and the estimation of species phylogenies (whether trees or networks!), even more so. Furthermore, the challenges in large-scale estimation are not computational feasibility (running time and memory), as inferential methods can differ in their theoretical and empirical performance.

The estimation of very large alignments and trees, containing tens of thousands of sequences, is now feasible; see Tables 6.1 and 6.2 for a summary of the methods that have been demonstrated to be able to estimate alignments and trees, respectively, on at least 25,000 sequences. Even co-estimation of alignments and trees of this size is now feasible, as shown in Table 6.3. However, improvements in scalability of many alignment and tree estimation methods is likely, and some of the methods not listed in these tables may also be able to analyze datasets of this size.

However, very substantial progress has been made for large-scale estimation, and the field seems poised to move very dramatically forward. For example, there are new algorithmic approaches being used that have the ability to improve the performance of base methods for alignment and phylogeny estimation, including iteration, divide-and-conquer, probabilistic models, and the incorporation of external biological knowledge. At the same time, there is a definite move to incorporate more biological realism and knowledge into statistical estimation methods. The combination of these approaches is likely to be a very powerful tool towards substantial improvements in accuracy and, potentially, scalability.

Acknowledgements During the time I wrote the paper, I was a Program Director at the National Science Foundation working on the BigData program; however, the research discussed in this paper took place over a span of many years. This research was therefore supported by U.S. National Science Foundation, Microsoft New England, the Guggenheim Foundation, the David and Lucile Packard Foundation, the Radcliffe Institute for Advanced Study, the Program for Evolutionary Dynamics at Harvard, the David Bruton Jr. Centennial Professorship in Computer Sciences at U.T. Austin, and two Faculty Research Assignments from the University of Texas at Austin.

It makes sense now to tell how some of the work in this paper came about. I was working with Randy Linder (UT-Austin Integrative Biology) on various problems, including large-scale alignment and phylogeny estimation. During our initial attempts to design a fast and accurate co-estimation method, we began by trying to come up with a better solution to the Treelength optimization problem. Our interest in treelength optimization convinced a colleague, Vijaya Ramachandran (UT-Austin Computer Science), to develop a fast exact median calculator [338], which

led to an improved treelength estimator; however our subsequent studies [263] suggested that improving the treelength would not lead to improved alignments and trees. This led us to look for other approaches to obtain more accurate alignments and trees from large datasets. Our next attempts considered the impact of guide trees, which gave a small benefit [109], but even iterating in this manner also did not lead to substantial improvements. Finally, we developed SATé, the co-estimation method described earlier. In a very real sense, therefore, much of the work in this chapter was inspired by David Sankoff, since he introduced the treelength optimization problem. And so, I end by thanking David Sankoff for this, as well as many other things.

References

1. Dobzhansky, T.: Nothing in biology makes sense except in the light of evolution. Am. Biol. Teach. **35**, 125–129 (1973)
2. de Chardin, P.T.: Le Phénomene Humain. Harper Perennial, New York (1959)
3. Eisen, J.A.: Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. Genome Res. **8**, 163–167 (1998)
4. Wang, L.-S., Leebens-Mack, J., Wall, K., Beckmann, K., de Pamphilis, C., et al.: The impact of protein multiple sequence alignment on phylogeny estimation. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**, 1108–1119 (2011)
5. Simmons, M., Freudenstein, J.: The effects of increasing genetic distance on alignment of, and tree construction from, rDNA internal transcribed spacer sequences. Mol. Phylogenet. Evol. **26**, 444–451 (2003)
6. Liu, K., Linder, C.R., Warnow, T.: Multiple sequence alignment: a major challenge to large-scale phylogenetics. PLoS Currents: Tree of Life (2010)
7. Hall, B.G.: Comparison of the accuracies of several phylogenetic methods using protein and DNA sequences. Mol. Evol. Biol. **22**, 792–802 (2005)
8. Kumar, S., Filipski, A.: Multiple sequence alignment: in pursuit of homologous DNA positions. Genome Res. **17**, 127–135 (2007)
9. Ogden, T., Rosenberg, M.: Multiple sequence alignment accuracy and phylogenetic inference. Syst. Biol. **55**, 314–328 (2006)
10. 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**, 1561–1564 (2009)
11. Morrison, D.: Multiple sequence alignment for phylogenetic purposes. Aust. Syst. Bot. **19**, 479–539 (2006)
12. Graybeal, A.: Is it better to add taxa or characters to a difficult phylogenetic problem? Syst. Biol. **47**, 9–17 (1998)
13. Pollock, D., Zwickl, D., McGuire, J., Hillis, D.: Increased taxon sampling is advantageous for phylogenetic inference. Syst. Biol. **51**, 664–671 (2002)
14. Zwickl, D., Hillis, D.: Increased taxon sampling greatly reduces phylogenetic error. Syst. Biol. **51**, 588–598 (2002)
15. Hillis, D.: Inferring complex phylogenies. Nature **383**, 130–131 (1996)
16. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Sunderland (2003)
17. Kim, J., Warnow, T.: Tutorial on phylogenetic tree estimation. Presented at the ISMB 1999 Conference (1999). Available on-line at http://www.cs.utexas.edu/users/tandy/tutorial.ps
18. Linder, C.R., Warnow, T.: An overview of phylogeny reconstruction. In: Aluru, S. (ed.) Handbook of Computational Molecular Biology. Chapman and Hall/CRC Computer and Information Science Series, vol. 9. CRC Press, Boca Raton (2005)
19. Semple, C., Steel, M.: Phylogenetics. Oxford University Press, London (2003)
20. Hillis, D., Moritz, C., Mable, B. (eds.): Molecular Systematics. Sinauer Associates, Sunderland (1996)

21. Ortuno, F., Valenzuela, O., Pomares, H., Rojas, F., Florido, J., et al.: Predicting the accuracy of multiple sequence alignment algorithms by using computational intelligent techniques. Nucleic Acids Res. **41** (2013)
22. Whelan, S., Lin, P., Goldman, N.: Molecular phylogenetics: state-of-the-art methods for looking into the past. Trends Genet. **17**, 262–272 (2001)
23. Goldman, N., Yang, Z.: Introduction: statistical and computational challenges in molecular phylogenetics and evolution. Philos. Trans. R. Soc. Lond. B, Biol. Sci. **363**, 3889–3892 (2008)
24. Kemena, C., Notredame, C.: Upcoming challenges for multiple sequence alignment methods in the high-throughput era. Bioinformatics **25**, 2455–2465 (2009)
25. Do, C., Katoh, K.: Protein multiple sequence alignment. In: Methods in Molecular Biology: Functional Proteomics, Methods and Protocols, vol. 484, pp. 379–413. Humana Press, Clifton (2008)
26. Mokaddem, A., Elloumi, M.: Algorithms for the alignment of biological sequences. In: Elloumi, M., Zomaya, A. (eds.) Algorithms in Computational Molecular Biology. Wiley, New York (2011). doi:10.1002/9780470892107.ch12
27. Pei, J.: Multiple protein sequence alignment. Curr. Opin. Struct. Biol. **18**, 382–386 (2008)
28. Sievers, F., Wilm, A., Dineen, D., Gibson, T., Karplus, K., et al.: Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. Mol. Syst. Biol. **7** (2011)
29. Katoh, K., Toh, H.: PartTree: an algorithm to build an approximate tree from a large number of unaligned sequences. Bioinformatics **23**(3), 372–374 (2007)
30. 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)
31. Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., Mcgettigan, P.A., et al.: ClustalW and ClustalX version 2.0. Bioinformatics **23**, 2947–2948 (2007)
32. Lassmann, T., Frings, O., Sonnhammer, E.: Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. Nucleic Acids Res. **37**, 858–865 (2009)
33. Neuwald, A.: Rapid detection, classification, and accurate alignment of up to a million or more related protein sequences. Bioinformatics **25**, 1869–1875 (2009)
34. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree-2—approximately maximum-likelihood trees for large alignments. PLoS ONE **5**, e9490 (2010). 10.1371/journal.pone.0009490
35. Smith, S., Beaulieu, J., Stamatakis, A., Donoghue, M.: Understanding angiosperm diversification using small and large phylogenetic trees. Am. J. Bot. **98**, 404–414 (2011)
36. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**, 2688–2690 (2006)
37. Goloboff, P.A., Catalano, S.A., Mirande, J.M., Szumik, C.A., Arias, J.S., et al.: Phylogenetic analysis of 73,060 taxa corroborates major eukaryotic groups. Cladistics **25**, 211–230 (2009)
38. Goloboff, P., Farris, J., Nixon, K.: TNT, a free program for phylogenetic analysis. Cladistics **24**, 774–786 (2008)
39. Liu, K., Warnow, T., Holder, M., Nelesen, S., Yu, J., et al.: SATé-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Syst. Biol. **61**, 90–106 (2011)
40. Maddison, W.: Gene trees in species trees. Syst. Biol. **46**, 523–536 (1997)
41. Delsuc, F., Brinkmann, H., Philippe, H.: Phylogenomics and the reconstruction of the tree of life. Nat. Rev. Genet. **6**, 361–375 (2005)
42. Edwards, S.V.: Is a new and general theory of molecular systematics emerging? Evolution **63**, 1–19 (2009)
43. Dunn, C.W., Hejnol, A., Matus, D.Q., Pang, K., Browne, W.E., et al.: Broad phylogenomic sampling improves resolution of the animal tree of life. Nature **452**, 745–749 (2008)
44. Wu, D., Hugenholtz, P., Mavromatis, K., Pukall, R., Dalin, E., et al.: A phylogeny-driven genomic encyclopedia of bacteria and archaea. Nature **462**, 1056–1060 (2009)

45. Eisen, J., Fraser, C.: Phylogenomics: intersection of evolution and genomics. Science **300**, 1706–1707 (2003)
46. Bininda-Emonds, O. (ed.): Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. Kluwer Academic, Dordrecht (2004)
47. 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 (2004)
48. 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)
49. Bininda-Emonds, O.R.P.: The evolution of supertrees. Trends Ecol. Evol. **19**, 315–322 (2004)
50. Snir, S., Rao, S.: Quartets MaxCut: a divide and conquer quartets algorithm. IEEE/ACM Trans. Comput. Biol. Bioinform. **7**, 704–718 (2010)
51. Steel, M., Rodrigo, A.: Maximum likelihood supertrees. Syst. Biol. **57**, 243–250 (2008)
52. Swenson, M., Suri, R., Linder, C., Warnow, T.: An experimental study of quartets MaxCut and other supertree methods. Algorithms Mol. Biol. **6**(1), 7 (2011)
53. Swenson, M., Suri, R., Linder, C., Warnow, T.: SuperFine: fast and accurate supertree estimation. Syst. Biol. **61**, 214–227 (2012)
54. Nguyen, N., Mirarab, S., Warnow, T.: MRL and SuperFine+MRL: new supertree methods. Algorithms Mol. Biol. **7**(3) (2012)
55. Than, C.V., Nakhleh, L.: Species tree inference by minimizing deep coalescences. PLoS Comput. Biol. **5** (2009)
56. Boussau, B., Szollosi, G., Duret, L., Gouy, M., Tannier, E., et al.: Genome-scale coestimation of species and gene trees. Genome Res. **23**(2), 323–330 (2013)
57. Degnan, J.H., Rosenberg, N.A.: Gene tree discordance, phylogenetic inference and the multispecies coalescent. Trends Ecol. Evol. **26**, 332–340 (2009)
58. 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)
59. Larget, B., Kotha, S.K., Dewey, C.N., Ané, C.: BUCKy: gene tree/species tree reconciliation with the Bayesian concordance analysis. Bioinformatics **26**, 2910–2911 (2010)
60. 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)
61. Yang, J., Warnow, T.: Fast and accurate methods for phylogenomic analyses. BMC Bioinform. **12**(Suppl 9), S4 (2011). doi:10.1186/1471-2105-12-S9-S4
62. 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)
63. Chauve, C., Doyon, J.P., El-Mabrouk, N.: Gene family evolution by duplication, speciation, and loss. J. Comput. Biol. **15**, 1043–1062 (2008)
64. Hallett, M.T., Lagergren, J.: New algorithms for the duplication-loss model. In: Proceedings RECOMB 2000, pp. 138–146. ACM Press, New York (2000)
65. Doyon, J.P., Chauve, C.: Branch-and-bound approach for parsimonious inference of a species tree from a set of gene family trees. Adv. Exp. Med. Biol. **696**, 287–295 (2011)
66. Ma, B., Li, M., Zhang, L.: From gene trees to species trees. SIAM J. Comput. **30**, 729–752 (2000)
67. Zhang, L.: From gene trees to species trees II: species tree inference by minimizing deep coalescence events. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**, 1685–1691 (2011)
68. Arvestad, L., Berglung, A.C., Lagergren, J., Sennblad, B.: Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In: Bininda-Emonds, O. (ed.) Proc. RECOMB 2004, pp. 238–252 (2004)
69. Sennblad, B., Lagergren, J.: Probabilistic orthology analysis. Syst. Biol. **58**, 411–424 (2009)
70. Edwards, S., Liu, L., Pearl, D.: High-resolution species trees without concatenation. Proc. Natl. Acad. Sci. USA **104**, 5936–5941 (2007)

71. Heled, J., Drummond, A.J.: Bayesian inference of species trees from multilocus data. Mol. Biol. Evol. **27**, 570–580 (2010)
72. Roch, S.: An analytical comparison of multilocus methods under the multispecies coalescent: the three-taxon case. In: Proc. Pacific Symposium on Biocomputing, vol. 18, pp. 297–306 (2013)
73. Kopelman, N.M., Stone, L., Gascuel, O., Rosenberg, N.A.: The behavior of admixed populations in neighbor-joining inference of population trees. In: Proc. Pacific Symposium on Biocomputing, vol. 18 (2013)
74. Degnan, J.H.: Evaluating variations on the STAR algorithm for relative efficiency and sample sizes needed to reconstruct species trees. In: Proc. Pacific Symposium on Biocomputing, vol. 18, pp. 262–272 (2013)
75. Bayzid, M., Mirarab, S., Warnow, T.: Inferring optimal species trees under gene duplication and loss. In: Proc. Pacific Symposium on Biocomputing, vol. 18, pp. 250–261 (2013)
76. Pei, J., Grishin, N.: PROMALS: towards accurate multiple sequence alignments of distantly related proteins. Bioinformatics **23**, 802–808 (2007)
77. Edgar, R.C., Sjölander, K.: SATCHMO: sequence alignment and tree construction using hidden Markov models. Bioinformatics **19**, 1404–1411 (2003)
78. Hagopian, R., Davidson, J., Datta, R., Jarvis, G., Sjölander, K.: SATCHMO-JS: a webserver for simultaneous protein multiple sequence alignment and phylogenetic tree construction. Nucleic Acids Res. **38**(Web Server Issue), W29–W34 (2010)
79. O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D., Notredame, C.: 3DCoffee: combining protein sequences and structure within multiple sequence alignments. J. Mol. Biol. **340**, 385–395 (2004)
80. Zhou, H., Zhou, Y.: SPEM: improving multiple sequence alignment with sequence profiles and predicted secondary structures. Bioinformatics **21**, 3615–3621 (2005)
81. Deng, X., Cheng, J.: MSACompro: protein multiple sequence alignment using predicted secondary structure, solvent accessibility, and residue-residue contacts. BMC Bioinform. **12**, 472 (2011)
82. Roshan, U., Livesay, D.R.: Probalign: multiple sequence alignment using partition function posterior probabilities. Bioinformatics **22**, 2715–2721 (2006)
83. Roshan, U., Chikkagoudar, S., Livesay, D.R.: Searching for RNA homologs within large genomic sequences using partition function posterior probabilities. BMC Bioinform. **9**, 61 (2008)
84. Do, C., Mahabhashyam, M., Brudno, M., Batzoglou, S.: PROBCONS: probabilistic consistency-based multiple sequence alignment of amino acid sequences. Software available at http://probcons.stanford.edu/download.html (2006)
85. Nawrocki, E.P., Kolbe, D.L., Eddy, S.R.: Infernal 1.0: inference of RNA alignments. Bioinformatics **25**, 1335–1337 (2009)
86. Nawrocki, E.P.: Structural RNA homology search and alignment using covariance models. Ph.D. thesis, Washington University in Saint Louis, School of Medicine (2009)
87. Gardner, D., Xu, W., Miranker, D., Ozer, S., Cannonne, J., et al.: An accurate scalable template-based alignment algorithm. In: Proc. International Conference on Bioinformatics and Biomedicine, 2012, pp. 237–243 (2012)
88. Edgar, R.C.: MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinform. **5**, 113 (2004)
89. Mirarab, S., Warnow, T.: FastSP: linear-time calculation of alignment accuracy. Bioinformatics **27**, 3250–3258 (2011)
90. Blackburne, B., Whelan, S.: Measuring the distance between multiple sequence alignments. Bioinformatics **28**, 495–502 (2012)
91. Stojanovic, N., Florea, L., Riemer, C., Gumucio, D., Slightom, J., et al.: Comparison of five methods for finding conserved sequences in multiple alignments of gene regulatory regions. Nucleic Acids Res. **27**, 3899–3910 (1999)
92. Edgar, R.: Quality measures for protein alignment benchmarks. Nucleic Acids Res. **7**, 2145–2153 (2010)

93. Thompson, J.D., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Res. **27**, 2682–2690 (1999)
94. Thompson, J., Plewniak, F., Poch, O.: BAliBASE: a benchmark alignments database for the evaluation of multiple sequence alignment programs. Bioinformatics **15**, 87–88 (1999)
95. Raghava, G., Searle, S.M., Audley, P.C., Barber, J.D., Barton, G.J.: Oxbench: a benchmark for evaluation of protein multiple sequence alignment accuracy. BMC Bioinform. **4**, 47 (2003)
96. Gardner, P., Wilm, A., Washietl, S.: A benchmark of multiple sequence alignment programs upon structural RNAs. Nucleic Acids Res. **33**, 2433–2439 (2005)
97. Walle, I.L.V., Wyns, L.: SABmark-a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics **21**, 1267–1268 (2005)
98. Carroll, H., Beckstead, W., O'Connor, T., Ebbert, M., Clement, M., et al.: DNA reference alignment benchmarks based on tertiary structure of encoded proteins. Bioinformatics **23**, 2648–2649 (2007)
99. Blazewicz, J., Formanowicz, P., Wojciechowski, P.: Some remarks on evaluating the quality of the multiple sequence alignment based on the BAliBASE benchmark. Int. J. Appl. Math. Comput. Sci. **19**, 675–678 (2009)
100. Iantomo, S., Gori, K., Goldman, N., Gil, M., Dessimoz, C.: Who watches the watchmen? An appraisal of benchmarks for multiple sequence alignment. arXiv:1211.2160 [q-bio.QM] (2012)
101. Aniba, M., Poch, O., Thompson, J.D.: Issues in bioinformatics benchmarking: the case study of multiple sequence alignment. Nucleic Acids Res. **38**, 7353–7363 (2010)
102. Morrison, D.A.: Why would phylogeneticists ignore computerized sequence alignment? Syst. Biol. **58**, 150–158 (2009)
103. Reeck, G., de Haen, C., Teller, D., Doolitte, R., Fitch, W., et al.: "Homology" in proteins and nucleic acids: a terminology muddle and a way out of it. Cell **50**, 667 (1987)
104. Galperin, M., Koonin, E.: Divergence and convergence in enzyme evolution. J. Biol. Chem. **287**, 21–28 (2012)
105. Sjolander, K.: Getting started in structural phylogenomics. PLoS Comput. Biol. **6**, e1000621 (2010)
106. Katoh, K., Kuma, K., Miyata, T., Toh, H.: Improvement in the accuracy of multiple sequence alignment MAFFT. Genome Inf. **16**, 22–33 (2005)
107. Do, C., Mahabhashyam, M., Brudno, M., Batzoglou, S.: PROBCONS: probabilistic consistency-based multiple sequence alignment. Genome Res. **15**, 330–340 (2005)
108. Loytynoja, A., Goldman, N.: An algorithm for progressive multiple alignment of sequences with insertions. Proc. Natl. Acad. Sci. **102**, 10557–10562 (2005)
109. Nelesen, S., Liu, K., Zhao, D., Linder, C.R., Warnow, T.: The effect of the guide tree on multiple sequence alignments and subsequent phylogenetic analyses. In: Proc. Pacific Symposium on Biocomputing, vol. 13, pp. 15–24 (2008)
110. Fletcher, W., Yang, Z.: The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection. Mol. Biol. Evol. **27**, 2257–2267 (2010)
111. Penn, O., Privman, E., Landan, G., Graur, D., Pupko, T.: An alignment confidence score capturing robustness to guide tree uncertainty. Mol. Biol. Evol. **27**, 1759–1767 (2010)
112. Toth, A., Hausknecht, A., Krisai-Greilhuber, I., Papp, T., Vagvolgyi, C., et al.: Iteratively refined guide trees help improving alignment and phylogenetic inference in the mushroom family *bolbitiaceae*. PLoS ONE **8**, e56143 (2013)
113. Capella-Gutiérrez, S., Gabaldón, T.: Measuring guide-tree dependency of inferred gaps for progressive aligners. Bioinformatics **29**(8), 1011–1017 (2013)
114. Preusse, E., Quast, C., Knittel, K., Fuchs, B., Ludwig, W., et al.: SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. Nucleic Acids Res. **35**, 718–796 (2007)
115. DeSantis, T., Hugenholtz, P., Keller, K., Brodie, E., Larsen, N., et al.: NAST: a multiple sequence alignment server for comparative analysis of 16S rRNA genes. Nucleic Acids Res. **34**, W394–W399 (2006)

116. Löytynoja, A., Vilella, A.J., Goldman, N.: Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. Bioinformatics **28**, 1685–1691 (2012)
117. Papadopoulos, J.S., Agarwala, R.: COBALT: constraint-based alignment tool for multiple protein sequences. Bioinformatics **23**, 1073–1079 (2007)
118. Berger, S.A., Stamatakis, A.: Aligning short reads to reference alignments and trees. Bioinformatics **27**, 2068–2075 (2011)
119. Sievers, F., Dineen, D., Wilm, A., Higgins, D.G.: Making automated multiple alignments of very large numbers of protein sequences. Bioinformatics **29**(8), 989–995 (2013)
120. Smith, S., Beaulieu, J., Donoghue, M.: Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. BMC Evol. Biol. **9**, 37 (2009)
121. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
122. Roquet, C., Thuiller, W., Lavergne, S.: Building megaphylogenies for macroecology: taking up the challenge. Ecography **36**, 013–026 (2013)
123. Steel, M.A.: Recovering a tree from the leaf colourations it generates under a Markov model. Appl. Math. Lett. **7**, 19–24 (1994)
124. Evans, S., Warnow, T.: Unidentifiable divergence times in rates-across-sites models. IEEE/ACM Trans. Comput. Biol. Bioinform. **1**, 130–134 (2005)
125. Tavaré, S.: Some probabilistic and statistical problems in the analysis of DNA sequences. In: Lectures on Mathematics in the Life Sciences, vol. 17, pp. 57–86 (1986)
126. Dayhoff, M., Schwartz, R., Orcutt, B.: A model of evolutionary change in proteins. In: Dayhoff, M. (ed.) Atlas of Protein Sequence and Structure. National Biomedical Research Foundation, pp. 345–352 (1978)
127. Lakner, C., Holder, M., Goldman, N., Naylor, G.: What's in a likelihood? Simple models of protein evolution and the contribution of structurally viable reconstructions to the likelihood. Syst. Biol. **60**, 161–174 (2011)
128. Le, S., Gascuel, O.: An improved general amino acid replacement matrix. Mol. Biol. Evol. **25**, 1307–1320 (2008)
129. 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**, 691–699 (2001)
130. Kosiol, C., Goldman, N.: Different versions of the Dayhoff rate matrix. Mol. Biol. Evol. **22**, 193–199 (2005)
131. Thorne, J.: Models of protein sequence evolution and their applications. Curr. Opin. Genet. Dev. **10**, 602–605 (2000)
132. Thorne, J., Goldman, N.: Probabilistic models for the study of protein evolution. In: Balding, D., Bishop, M., Cannings, C. (eds.) Handbook of Statistical Genetics, pp. 209–226. Wiley, New York (2003)
133. Adachi, J., Hasegawa, M.: Model of amino acid substitution in proteins encoded by mitochondrial DNA. J. Mol. Evol. **42**, 459–468 (1996)
134. Goldman, N., Yang, Z.: A codon-based model of nucleotide substitution for protein-coding DNA sequences. Mol. Biol. Evol. **11**, 725–736 (1994)
135. Scherrer, M., Meyer, A., Wilke, C.: Modeling coding-sequence evolution within the context of residue solvent accessibility. BMC Evol. Biol. **12**, 179 (2012)
136. Mayrose, I., Doron-Faigenbom, A., Bacharach, E., Pupko, T.: Towards realistic codon models: among site variability and dependency of synonymous and non-synonymous rates. Bioinformatics **23**, i319–i327 (2007)
137. Abascal, F., Zardoya, R., Posada, D.: ProtTest: selection of best-fit models of protein evolution. Bioinformatics **21**, 2104–2105 (2005)
138. Wilke, C.: Bringing molecules back into molecular evolution. PLoS Comput. Biol. **8**, e1002572 (2012)
139. Liberles, D., Teichmann, S., et al.: The inference of protein structure, protein biophysics, and molecular evolution. Protein Sci. **21**, 769–785 (2012)

140. Lopez, P., Casane, D., Philippe, H.: Heterotachy, an important process of protein evolution. Mol. Biol. Evol. **19**, 1–7 (2002)
141. Whelan, S.: Spatial and temporal heterogeneity in nucleotide sequence evolution. Mol. Biol. Evol. **25**, 1683–1694 (2008)
142. Tuffley, C., Steel, M.: Links between maximum likelihood and maximum parsimony under a simple model of site substitution. Bull. Math. Biol. **59**, 581–607 (1997)
143. Steel, M.A.: Can we avoid 'SIN' in the house of 'No Common Mechanism'? Syst. Biol. **60**, 96–109 (2011)
144. Lobkovsky, A., Wolf, Y., Koonin, E.: Gene frequency distributions reject a neutral model of genome evolution. Genome Biol. Evol. **5**, 233–242 (2013)
145. Galtier, N., Gouy, M.: Inferring pattern and process: maximum-likelihood implementation of a nonhomogeneous model of DNA sequence evolution for phylogenetic analysis. Mol. Biol. Evol. **15**, 871–879 (1998)
146. Foulds, L.R., Graham, R.L.: The Steiner problem in phylogeny is NP-complete. Adv. Appl. Math. **3**, 43–49 (1982)
147. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**, 368–376 (1981)
148. Allman, E.S., Ané, C., Rhodes, J.: Identifiability of a Markovian model of molecular evolution with gamma-distributed rates. Adv. Appl. Probab. **40**, 229–249 (2008)
149. Allman, E.S., Rhodes, J.: Identifying evolutionary trees and substitution parameters for the general Markov model with invariable sites. Math. Biosci. **211**, 18–33 (2008)
150. Allman, E.S., Rhodes, J.A.: The identifiability of tree topology for phylogenetic models, including covariant and mixture models. J. Comput. Biol. **13**, 1101–1113 (2006)
151. Atteson, K.: The performance of neighbor-joining methods of phylogenetic reconstruction. Algorithmica **25**, 251–278 (1999)
152. Chang, J.: Full reconstruction of Markov models on evolutionary trees: identifiability and consistency. Math. Biosci. **137**, 51–73 (1996)
153. Steel, M.A.: Consistency of Bayesian inference of resolved phylogenetic trees. arXiv: 1001.2864 [q-bioPE] (2010)
154. Felsenstein, J.: Cases in which parsimony or compatibility methods will be positively misleading. Syst. Zool. **27**, 401–410 (1978)
155. Chang, J.T.: Inconsistency of evolutionary tree topology reconstruction methods when substitution rates vary across characters. Math. Biosci. **134**, 189–215 (1996)
156. Matsen, F., Steel, M.: Phylogenetic mixtures on a single tree can mimic a tree of another topology. Syst. Biol. **56**, 767–775 (2007)
157. Allman, E., Rhodes, J., Sullivant, S.: When do phylogenetic mixture models mimic other phylogenetic models? Syst. Biol. **61**, 1049–1059 (2012)
158. Erdos, P., Steel, M., Szekely, L., Warnow, T.: Local quartet splits of a binary tree infer all quartet splits via one dyadic inference rule. Comput. Artif. Intell. **16**, 217–227 (1997)
159. Erdos, P., Steel, M., Szekely, L., Warnow, T.: A few logs suffice to build (almost) all trees (i). Random Struct. Algorithms **14**, 153–184 (1999)
160. Erdos, P., Steel, M., Szekely, L., Warnow, T.: A few logs suffice to build (almost) all trees (ii). Theor. Comput. Sci. **221**, 77–118 (1999)
161. 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**, 188–215 (2006)
162. Csürős, M., Kao, M.Y.: Recovering evolutionary trees through harmonic greedy triplets. Proc. SODA **99**, 261–270 (1999)
163. Csurös, M.: Fast recovery of evolutionary trees with thousands of nodes. J. Comput. Biol. **9**, 277–297 (2002)
164. Huson, D., Nettles, S., Warnow, T.: Disk-covering, a fast converging method for phylogenetic tree reconstruction. J. Comput. Biol. **6**, 369–386 (1999)
165. Steel, M.A., Székely, L.A.: Inverting random functions. Ann. Comb. **3**, 103–113 (1999)
166. Steel, M.A., Székely, L.A.: Inverting random functions—II: explicit bounds for discrete maximum likelihood estimation, with applications. SIAM J. Discrete Math. **15**, 562–575 (2002)

167. King, V., Zhang, L., Zhou, Y.: On the complexity of distance-based evolutionary tree reconstruction. In: SODA: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, pp. 444–453 (2003)
168. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden Markov models. In: Proc. 37th Symp. on the Theory of Computing (STOC'05), pp. 366–376 (2005)
169. Mossel, E., Roch, S.: Learning nonsingular phylogenies and hidden Markov models. Ann. Appl. Probab. **16**, 538–614 (2006)
170. Daskalakis, C., Mossel, E., Roch, S.: Optimal phylogenetic reconstruction. In: STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 159–168 (2006)
171. Daskalakis, C., Hill, C., Jaffe, A., Mihaescu, R., Mossel, E., et al.: Maximal accurate forests from distance matrices. In: RECOMB, pp. 281–295 (2006)
172. Mossel, E.: Distorted metrics on trees and phylogenetic forests. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**, 108–116 (2007)
173. Gronau, I., Moran, S., Snir, S.: Fast and reliable reconstruction of phylogenetic trees with very short edges. In: SODA (ACM/SIAM Symp. Disc. Alg), pp. 379–388 (2008)
174. Roch, S.: Sequence-length requirement for distance-based phylogeny reconstruction: breaking the polynomial barrier. In: FOCS (Foundations of Computer Science), pp. 729–738 (2008)
175. Daskalakis, C., Mossel, E., Roch, S.: Phylogenies without branch bounds: contracting the short, pruning the deep. In: RECOMB, pp. 451–465 (2009)
176. Lin, Y., Rajan, V., Moret, B.: A metric for phylogenetic trees based on matching. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**, 1014–1022 (2012)
177. Rannala, B., Huelsenbeck, J., Yang, Z., Nielsen, R.: Taxon sampling and the accuracy of large phylogenies. Syst. Biol. **47**, 702–710 (1998)
178. Robinson, D., Foulds, L.: Comparison of phylogenetic trees. Math. Biosci. **53**, 131–147 (1981)
179. Huelsenbeck, J., Hillis, D.: Success of phylogenetic methods in the four-taxon case. Syst. Biol. **42**, 247–265 (1993)
180. Hillis, D.: Taxonomic sampling, phylogenetic accuracy, and investigator bias. Syst. Biol. **47**, 3–8 (1998)
181. Nakhleh, L., Moret, B., Roshan, U., St John, K., Sun, J., et al.: The accuracy of fast phylogenetic methods for large datasets. In: Proc. 7th Pacific Symposium on BioComputing, pp. 211–222. World Scientific, Singapore (2002)
182. Zwickl, D.J., Hillis, D.M.: Increased taxon sampling greatly reduces phylogenetic error. Syst. Biol. **51**, 588–598 (2002)
183. Pollock, D.D., Zwickl, D.J., McGuire, J.A., Hillis, D.M.: Increased taxon sampling is advantageous for phylogenetic inference. Syst. Biol. **51**, 664–671 (2002)
184. Wiens, J.: Missing data and the design of phylogenetic analyses. J. Biomed. Inform. **39**, 36–42 (2006)
185. Lemmon, A., Brown, J., Stanger-Hall, K., Lemmon, E.: The effect of ambiguous data on phylogenetic estimates obtained by maximum-likelihood and Bayesian inference. Syst. Biol. **58**, 130–145 (2009)
186. Wiens, J., Morrill, M.: Missing data in phylogenetic analysis: reconciling results from simulations and empirical data. Syst. Biol. **60**, 719–731 (2011)
187. Simmons, M.: Misleading results of likelihood-based phylogenetic analyses in the presence of missing data. Cladistics **28**, 208–222 (2012)
188. Moret, B., Roshan, U., Warnow, T.: Sequence-length requirements for phylogenetic methods. In: Guigo, R., Gusfield, D. (eds.) Proc. 2nd International Workshop on Algorithms in Bioinformatics. Lecture Notes in Computer Science, vol. 2452, pp. 343–356. Springer, Berlin (2002)
189. 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)

190. Bruno, W.J., Socci, N.D., Halpern, A.L.: Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. Mol. Biol. Evol. **17**, 189–197 (2000)
191. Wheeler, T.: Large-scale neighbor-joining with NINJA. In: Proc. Workshop Algorithms in Bioinformatics (WABI), vol. 5724, pp. 375–389 (2009)
192. Desper, R., Gascuel, O.: Fast and accurate phylogeny reconstruction algorithm based on the minimum-evolution principle. J. Comput. Biol. **9**, 687–705 (2002)
193. Price, M., Dehal, P., Arkin, A.: FastTree: computing large minimum evolution trees with profiles instead of a distance matrix. Mol. Biol. Evol. **7**, 1641–1650 (2009)
194. Brown, D., Truszkowski, J.: Towards a practical $O(n \log n)$ phylogeny algorithm. In: Proc. Workshop Algorithms in Bioinformatics (WABI), pp. 14–25 (2011)
195. Rice, K., Warnow, T.: Parsimony is hard to beat! In: Jiang, T., Lee, D. (eds.) Proceedings, Third Annual International Conference of Computing and Combinatorics (COCOON), pp. 124–133 (1997)
196. Hillis, D., Huelsenbeck, J., Swofford, D.: Hobgoblin of phylogenetics. Nature **369**, 363–364 (1994)
197. Swofford, D.: PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods), Version 4.0. Sinauer Associates, Sunderland (1996)
198. Roch, S.: A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**, 92–94 (2006)
199. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. **52**, 696–704 (2003)
200. Zwickl, D.: 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)
201. Liu, K., Linder, C., Warnow, T.: RAxML and FastTree: comparing two methods for large-scale maximum likelihood phylogeny estimation PLoS ONE **6**, e27731 (2012).
202. Claesson, M.J., Cusack, S., O'Sullivan, O., Greene-Diniz, R., de Weerd, H., et al.: Composition, variability, and temporal stability of the intestinal microbiota of the elderly. Proc. Natl. Acad. Sci. **108**, 4586–4591 (2011)
203. McDonald, D., Price, M.N., Goodrich, J., Nawrocki, E.P., DeSantis, T.Z., et al.: An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. ISME J. **6**, 610–618 (2012)
204. Boussau, B., Guoy, M.: Efficient likelihood computations with non-reversible models of evolution. Syst. Biol. **55**, 756–768 (2006)
205. Whelan, S., Money, D.: The prevalence of multifurcations in tree-space and their implications for tree-search. Mol. Biol. Evol. **27**, 2674–2677 (2010)
206. Whelan, S., Money, D.: Characterizing the phylogenetic tree-search problem. Syst. Biol. **61**, 228–239 (2012)
207. Ronquist, F., Huelsenbeck, J.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**, 1572–1574 (2003)
208. Drummond, A., Rambaut, A.: BEAST: Bayesian evolutionary analysis by sampling trees. BMC Evol. Biol. **7**, 214 (2007)
209. Lartillot, N., Philippe, H.: A Bayesian mixture model for across-site heterogeneities in the amino acid replacement process. Mol. Biol. Evol. **21** (2004)
210. Foster, P.: Modeling compositional heterogeneity. Syst. Biol. **53**, 485–495 (2004)
211. Pagel, M., Meade, A.: A phylogenetic mixture model for detecting pattern heterogeneity in gene sequence or character state data. Syst. Biol. **53**, 571–581 (2004)
212. Huelsenbeck, J., Ronquist, R.: MrBayes: Bayesian inference of phylogeny. Bioinformatics **17**, 754–755 (2001)
213. Ronquist, F., Deans, A.: Bayesian phylogenetics and its influence on insect systematics. Annu. Rev. Entomol. **55**, 189–206 (2010)
214. Huelsenbeck, J.P., Ronquist, F., Nielsen, R., Bollback, J.P.: Bayesian inference of phylogeny and its impact on evolutionary biology. Science **294**, 2310–2314 (2001)

215. Holder, M., Lewis, P.: Phylogeny estimation: traditional and Bayesian approaches. Nat. Rev. Genet. **4**, 275–284 (2003)
216. Lewis, P., Holder, M., Holsinger, K.: Polytomies and Bayesian phylogenetic inference. Syst. Biol. **54**, 241–253 (2005)
217. Ganapathy, G., Ramachandran, V., Warnow, T.: On contract-and-refine-transformations between phylogenetic trees. In: ACM/SIAM Symposium on Discrete Algorithms (SODA'04), pp. 893–902. SIAM Press, Philadelphia (2004)
218. Ganapathy, G., Ramachandran, V., Warnow, T.: Better hill-climbing searches for parsimony. In: Proceedings of the Third International Workshop on Algorithms in Bioinformatics (WABI), pp. 245–258 (2003)
219. Bonet, M., Steel, M., Warnow, T., Yooseph, S.: Faster algorithms for solving parsimony and compatibility. J. Comput. Biol. **5**, 409–422 (1999)
220. Nixon, K.C.: The parsimony ratchet, a new method for rapid parsimony analysis. Cladistics **15**, 407–414 (1999)
221. Vos, R.: Accelerated likelihood surface exploration: the likelihood ratchet. Syst. Biol. **52**, 368–373 (2003)
222. Warnow, T., Moret, B.M.E., St John, K.: Absolute phylogeny: true trees from short sequences. In: Proc. 12th Ann. ACM/SIAM Symp. on Discr. Algs., SODA01, pp. 186–195. SIAM Press, Philadelphia (2001)
223. Nakhleh, L., Roshan, U., St John, K., Sun, J., Warnow, T.: Designing fast converging phylogenetic methods. Bioinformatics **17**, 190–198 (2001)
224. Warnow, T.: Large-scale phylogenetic reconstruction. In: Aluru, S. (ed.) Handbook of Computational Molecular Biology. Chapman and Hall/CRC Computer and Information Science Series, vol. 9. CRC Press, Boca Raton (2005)
225. Roshan, U., Moret, B., Williams, T., Warnow, T.: Rec-I-DCM3: a fast algorithmic technique for reconstructing large phylogenetic trees. In: Proc. 3rd Computational Systems Biology Conf. (CSB'05). Proceedings of the IEEE, pp. 98–109 (2004)
226. Steel, M.: The maximum likelihood point for a phylogenetic tree is not unique. Syst. Biol. **43**, 560–564 (1994)
227. Blair, C., Murphy, R.: Recent trends in molecular phylogenetic analysis: where to next? J. Heredity **102**, 130–138 (2011)
228. Nagy, L., Kocsube, S., Csanadi, Z., Kovacs, G., Petkovits, T., et al.: Re-mind the gap! Insertion and deletion data reveal neglected phylogenetic potential of the nuclear ribosomal internal transcribed spacer (its) of fungi. PLoS ONE **7**, e49794 (2012).
229. Barriel, V.: Molecular phylogenies and nucleotide insertion-deletions. C. R. Acad. Sci. III **7**, 693–701 (1994)
230. Young, N., Healy, J.: GapCoder automates the use of indel characters in phylogenetic analysis. BMC Bioinform. **4** (2003)
231. Muller, K.: Incorporating information from length-mutational events into phylogenetic analysis. Mol. Phylogenet. Evol. **38**, 667–676 (2006)
232. Ogden, T., Rosenberg, M.: How should gaps be treated in parsimony? A comparison of approaches using simulation. Mol. Phylogenet. Evol. **42**, 817–826 (2007)
233. Dwivedi, B., Gadagkar, S.: Phylogenetic inference under varying proportions of indel-induced alignment gaps. BMC Evol. Biol. **9**, 211 (2009)
234. Dessimoz, C., Gil, M.: Phylogenetic assessment of alignments reveals neglected tree signal in gaps. Genome Biol. **11**, R37 (2010)
235. Yuri, T., Kimball, R.T., Harshman, J., Bowie, R.C.K., Braun, M.J., et al.: Parsimony and model-based analyses of indel in avian nuclear genes reveal congruent and incongruent phylogenetic signals. Biology **2**, 419–444 (2013)
236. Warnow, T.: Standard maximum likelihood analyses of alignments with gaps can be statistically inconsistent. PLoS Currents Tree of Life (2012)
237. Daskalakis, C., Roch, S.: Alignment-free phylogenetic reconstruction. In: Berger, B. (ed.) Proc. RECOMB 2010. Lecture Notes in Computer Science, vol. 6044, pp. 123–137. Springer, Berlin (2010). http://dx.doi.org/10.1007/978-3-642-12683-3_9

238. Thatte, B.: Invertibility of the TKF model of sequence evolution. Math. Biosci. **200**, 58–75 (2006)
239. Hartmann, S., Vision, T.: Using ESTs for phylogenomics: can one accurately infer a phylogenetic tree from a Gappy alignment? BMC Evol. Biol. **8**, 95 (2008)
240. Mirarab, S., Nguyen, N., Warnow, T.: SEPP: SATé-enabled phylogenetic placement. In: Pacific Symposium on Biocomputing, pp. 247–258 (2012)
241. Matsen, F.A., Kodner, R.B., Armbrust, E.V.: pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinform. **11**, 538 (2010)
242. Berger, S.A., Krompass, D., Stamatakis, A.: Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. Syst. Biol. **60**, 291–302 (2011)
243. Eddy, S.: A new generation of homology search tools based on probabilistic inference. Genome Inform. **23**, 205–211 (2009)
244. Finn, R., Clements, J., Eddy, S.: HMMER web server: interactive sequence similarity searching. Nucleic Acids Res. **39**, W29–W37 (2011)
245. Brown, D.G., Truskowski, J.: LSHPlace: fast phylogenetic placement using locality-sensitive hashing. In: Pacific Symposium on Biocomputing, vol. 18, pp. 310–319 (2013)
246. Stark, M., Berger, S., Stamatakis, A., von Mering, C.: MLTreeMap—accurate maximum likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. BMC Genomics **11**, 461 (2010)
247. Droge, J., McHardy, A.: Taxonomic binning of metagenome samples generated by next-generation sequencing technologies. Brief. Bioinform. (2012)
248. Giribet, G.: Exploring the behavior of POY, a program for direct optimization of molecular data. Cladistics **17**, S60–S70 (2001)
249. Hartigan, J.: Minimum mutation fits to a given tree. Biometrics **29**, 53–65 (1973)
250. Sankoff, D.: Minimal mutation trees of sequences. SIAM J. Appl. Math. **28**, 35–42 (1975)
251. Sankoff, D., Cedergren, R.J.: Simultaneous comparison of three or more sequences related by a tree. In: Sankoff, D., Kruskall, J.B. (eds.) Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, pp. 253–263. Addison Wesley, New York (1993)
252. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. J. Comput. Biol. **1**, 337–348 (1994)
253. Wang, L., Jiang, T., Lawler, E.: Approximation algorithms for tree alignment with a given phylogeny. Algorithmica **16**, 302–315 (1996)
254. Wang, L., Gusfield, D.: Improved approximation algorithms for tree alignment. J. Algorithms **25**(2), 255–273 (1997)
255. Wang, L., Jiang, T., Gusfield, D.: A more efficient approximation scheme for tree alignment. SIAM J. Comput. **30**(1), 283–299 (2000)
256. Liu, K., Warnow, T.: Treelength optimization for phylogeny estimation. PLoS ONE **7**, e33104 (2012)
257. Varón, A., Vinh, L., Bomash, I., Wheeler, W.: POY software. Documentation by Varon, A., Vinh, L.S., Bomash, I., Wheeler, W., Pickett, K., Temkin, I., Faivovich, J., Grant, T., Smith, W.L. Available for download at http://research.amnh.org/scicomp/projects/poy.php (2007)
258. Kjer, K., Gillespie, J., Ober, K.: Opinions on multiple sequence alignment, and an empirical comparison on repeatability and accuracy between POY and structural alignment. Syst. Biol. **56**, 133–146 (2007)
259. Ogden, T.H., Rosenberg, M.: Alignment and topological accuracy of the direct optimization approach via POY and traditional phylogenetics via ClustalW+PAUP*. Syst. Biol. **56**, 182–193 (2007)
260. Yoshizawa, K.: Direct optimization overly optimizes data. Syst. Entomol. **35**, 199–206 (2010)

261. Wheeler, W., Giribet, G.: Phylogenetic hypotheses and the utility of multiple sequence alignment. In: Rosenberg, M. (ed.) Sequence Alignment: Methods, Models, Concepts and Strategies, pp. 95–104. University of California Press, Berkeley (2009)
262. Lehtonen, S.: Phylogeny estimation and alignment via POY versus clustal + PAUP*: a response to Ogden and Rosenberg. Syst. Biol. **57**, 653–657 (2008)
263. Liu, K., Nelesen, S., Raghavan, S., Linder, C., Warnow, T.: Barking up the wrong treelength: the impact of gap penalty on alignment and tree accuracy. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**, 7–21 (2009)
264. Gu, X., Li, W.H.: The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. J. Mol. Evol. **40**, 464–473 (1995)
265. Altschul, S.F.: Generalized affine gap costs for protein sequence alignment. Proteins, Struct. Funct. Genomics **32**, 88–96 (1998)
266. Gill, O., Zhou, Y., Mishra, B.: Aligning sequences with non-affine gap penalty: PLAINS algorithm, a practical implementation, and its biological applications in comparative genomics. In: Proc. ICBA 2004 (2004)
267. Qian, B., Goldstein, R.: Distribution of indel lengths. Proteins **45**, 102–104 (2001)
268. Chang, M., Benner, S.: Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. J. Mol. Biol. **341**, 617–631 (2004)
269. Thorne, J.L., Kishino, H., Felsenstein, J.: An evolutionary model for maximum likelihood alignment of DNA sequences. J. Mol. Evol. **33**, 114–124 (1991)
270. Thorne, J.L., Kishino, H., Felsenstein, J.: Inching toward reality: an improved likelihood model of sequence evolution. J. Mol. Evol. **34**, 3–16 (1992)
271. Thorne, J.L., Kishino, H., Felsenstein, J.: Erratum, an evolutionary model for maximum likelihood alignment of DNA sequences. J. Mol. Evol. **34**, 91–92 (1992)
272. Rivas, E.: Evolutionary models for insertions and deletions in a probabilistic modeling framework. BMC Bioinform. **6**, 30 (2005)
273. Rivas, E., Eddy, S.: Probabilistic phylogenetic inference with insertions and deletions. PLoS Comput. Biol. **4**, e1000172 (2008)
274. Holmes, I., Bruno, W.J.: Evolutionary HMMs: a Bayesian approach to multiple alignment. Bioinformatics **17**, 803–820 (2001)
275. Miklós, I., Lunter, G.A., Holmes, I.: A "long indel model" for evolutionary sequence alignment. Mol. Biol. Evol. **21**, 529–540 (2004)
276. Redelings, B., Suchard, M.: Joint Bayesian estimation of alignment and phylogeny. Syst. Biol. **54**, 401–418 (2005)
277. Suchard, M.A., Redelings, B.D.: BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics **22**, 2047–2048 (2006)
278. Redelings, B., Suchard, M.: Incorporating indel information into phylogeny estimation for rapidly emerging pathogens. BMC Evol. Biol. **7**, 40 (2007)
279. Fleissner, R., Metzler, D., von Haeseler, A.: Simultaneous statistical multiple alignment and phylogeny reconstruction. Syst. Biol. **54**, 548–561 (2005)
280. Novák, A., Miklós, I., Lyngso, R., Hein, J.: StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. Bioinformatics **24**, 2403–2404 (2008)
281. Lunter, G.A., Miklos, I., Song, Y.S., Hein, J.: An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees. J. Comput. Biol. **10**, 869–889 (2003)
282. Lunter, G., Miklós, I., Drummond, A., Jensen, J.L., Hein, J.: Bayesian phylogenetic inference under a statistical indel model. In: Benson, G., Page, R. (eds.) Third International Workshop (WABI 2003). Lecture Notes in Bioinformatics vol. 2812, pp. 228–244. Springer, Berlin (2003)
283. Lunter, G., Drummond, A., Miklós, I., Hein, J.: Statistical alignment: recent progress, new applications, and challenges. In: Nielsen, R. (ed.) Statistical Methods in Molecular Evolution (Statistics for Biology and Health), pp. 375–406. Springer, Berlin (2005)

284. Metzler, D.: Statistical alignment based on fragment insertion and deletion models. Bioinformatics **19**, 490–499 (2003)
285. Miklós, I.: Algorithm for statistical alignment of sequences derived from a Poisson sequence length distribution. Discrete Appl. Math. **127**, 79–84 (2003)
286. Arunapuram, P., Edvardsson, I., Golden, M., Anderson, J., Novak, A., et al.: StatAlign 2.0: combining statistical alignment with RNA secondary structure prediction. Bioinformatics **29**(5), 654–655 (2013)
287. Lunter, G., Miklós, I., Drummond, A., Jensen, J.L., Hein, J.: Bayesian coestimation of phylogeny and sequence alignment. BMC Bioinform. **6**, 83 (2005)
288. Bouchard-Côté, A., Jordan, M.I.: Evolutionary inference via the Poisson indel process. Proc. Natl. Acad. Sci. **110**, 1160–1166 (2013)
289. Brown, D., Krishnamurthy, N., Sjolander, K.: Automated protein subfamily identification and classification. PLoS Comput. Biol. **3**, e160 (2007)
290. Vinga, S., Almeida, J.: Alignment-free sequence comparison—a review. Bioinformatics **19**, 513–523 (2003)
291. Chan, C., Ragan, M.: Next-generation phylogenomics. Biol. Direct **8** (2013)
292. Blaisdell, B.: A measure of the similarity of sets of sequences not requiring sequence alignment. Proc. Natl. Acad. Sci. USA **83**, 5155–5159 (1986)
293. Sims, G., Jun, S.R., Wu, G., Kim, S.H.: Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. Proc. Natl. Acad. Sci. USA **106**, 2677–2682 (2009)
294. Jun, S.R., Sims, G., Wu, G., Kim, S.H.: Whole-proteome phylogeny of prokaryotes by feature frequency profiles: an alignment-free method with optimal feature resolution. Proc. Natl. Acad. Sci. USA **107**, 133–138 (2010)
295. Liu, X., Wan, L., Li, J., Reinert, G., Waterman, M., et al.: New powerful statistics for alignment-free sequence comparison under a pattern transfer model. J. Theor. Biol. **284**, 106–116 (2011)
296. Yang, K., Zhang, L.: Performance comparison between k-tuple distance and four model-based distances in phylogenetic tree reconstruction. Nucleic Acids Res. **36**, e33 (2008)
297. 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 (2004)
298. Nelesen, S.: Improved methods for phylogenetics. Ph.D. thesis, The University of Texas at Austin (2009)
299. Swenson, M.: Phylogenetic supertree methods. Ph.D. thesis, The University of Texas at Austin (2008)
300. Neves, D., Warnow, T., Sobral, J., Pingali, K.: Parallelizing SuperFine. In: 27th Symposium on Applied Computing (ACM-SAC) (2012)
301. Cannone, J., Subramanian, S., Schnare, M., Collett, J., D'Souza, L., et al.: The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron and other RNAs. BMC Bioinform. **3** (2002)
302. Roch, S.: Towards extracting all phylogenetic information from matrices of evolutionary distances. Science **327**, 1376–1379 (2010)
303. Darling, A., Mau, B., Blatter, F., Perna, N.: Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome Res. **14**, 1394–1403 (2004)
304. Darling, A., Mau, B., Perna, N.: progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. PLoS ONE **5**, e11147 (2010)
305. Raphael, B., Zhi, D., Tang, H., Pevzner, P.: A novel method for multiple alignment of sequences with repeated and shuffled elements. Genome Res. **14**, 2336–2346 (2004)
306. Dubchak, I., Poliakov, A., Kislyuk, A., Brudno, M.: Multiple whole-genome alignments without a reference organism. Genome Res. **19**, 682–689 (2009)
307. Brudno, M., Do, C., Cooper, G., Kim, M., Davydov, E., et al.: LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. Genome Res. **13**, 721–731 (2003)

308. Phuong, T., Do, C., Edgar, R., Batzoglou, S.: Multiple alignment of protein sequences with repeats and rearrangements. Nucleic Acids Res. **34**, 5932–5942 (2006)
309. Paten, B., Earl, D., Nguyen, N., Diekhans, M., Zerbino, D., et al.: Cactus: algorithms for genome multiple sequence alignment. Genome Res. **21**, 1512–1528 (2011)
310. Angiuoli, S., Salzberg, S.: Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics (2011). 10.1093/bioinformatics/btq665
311. Agren, J., Sundstrom, A., Hafstrom, T., Segerman, B.: Gegenees: fragmented alignment of multiple genomes for determining phylogenomic distances and genetic signatures unique for specified target groups. PLoS ONE **7**, e39107 (2012)
312. Gogarten, J., Doolittle, W., Lawrence, J.: Prokaryotic evolution in light of gene transfer. Mol. Biol. Evol. **19**, 2226–2238 (2002)
313. Gogarten, J., Townsend, J.: Horizontal gene transfer, genome innovation and evolution. Nat. Rev. Microbiol. **3**, 679–687 (2005)
314. Bergthorsson, U., Richardson, A., Young, G., Goertzen, L., Palmer, J.: Massive horizontal transfer of mitochondrial genes from diverse land plant donors to basal angiosperm Amborella. Proc. Natl. Acad. Sci. USA **101**, 17,747–17,752 (2004)
315. Bergthorsson, U., Adams, K., Thomason, B., Palmer, J.: Widespread horizontal transfer of mitochondrial genes in flowering plants. Nature **424**, 197–201 (2003)
316. Wolf, Y., Rogozin, I., Grishin, N., Koonin, E.: Genome trees and the tree of life. Trends Genet. **18**, 472–478 (2002)
317. Koonin, E., Makarova, K., Aravind, L.: Horizontal gene transfer in prokaryotes: quantification and classification. Annu. Rev. Microbiol. **55**, 709–742 (2001)
318. Linder, C., Rieseberg, L.: Reconstructing patterns of reticulate evolution in plants. Am. J. Bot. **91**, 1700–1708 (2004)
319. Sessa, E., Zimmer, E., Givnish, T.: Reticulate evolution on a global scale: a nuclear phylogeny for New World *Dryopteris* (Dryopteridaceae). Mol. Phylogenet. Evol. **64**, 563–581 (2012)
320. Moody, M., Rieseberg, L.: Sorting through the chaff, nDNA gene trees for phylogenetic inference and hybrid identification of annual sunflowers *Helianthus*. Mol. Phylogenet. Evol. **64**, 145–155 (2012) (sect. *Helianthus*)
321. Mindell, D.: The tree of life: metaphor, model, and heuristic device. Syst. Biol. **62**(3), 479–489 (2013)
322. Warnow, T., Evans, S., Ringe, D., Nakhleh, L.: A stochastic model of language evolution that incorporates homoplasy and borrowing. In: Phylogenetic Methods and the Prehistory of Languages, pp. 75–90. Cambridge University Press, Cambridge (2006)
323. Nakhleh, L., Ringe, D.A., Warnow, T.: Perfect phylogenetic networks: a new methodology for reconstructing the evolutionary history of natural languages. Language **81**, 382–420 (2005)
324. Huson, D., Rupp, R., Scornovacca, C.: Phylogenetic Networks: Concepts, Algorithms and Applications. Cambridge University Press, Cambridge (2010)
325. Morrison, D.: Introduction to Phylogenetic Networks. RJR Productions, Uppsala (2011)
326. Nakhleh, L.: Evolutionary phylogenetic networks: models and issues. In: Problem Solving Handbook in Computational Biology and Bioinformatics, pp. 125–158. Springer, Berlin (2011)
327. 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**, i124–i131 (2010)
328. Wu, Y.: An algorithm for constructing parsimonious hybridization networks with multiple phylogenetic trees. In: Proc. RECOMB (2013)
329. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Maximum likelihood of phylogenetic networks. Bioinformatics **22**, 2604–2611 (2006)
330. Jin, G., Nakhleh, L., Snir, S., Tuller, T.: Inferring phylogenetic networks by the maximum parsimony criterion: a case study. Mol. Biol. Evol. **24**, 324–337 (2007)

331. Nakhleh, L., Warnow, T., Linder, C.: Reconstructing reticulate evolution in species—theory and practice. In: Proc. 8th Conf. Comput. Mol. Biol. (RECOMB'04), pp. 337–346. ACM Press, New York (2004)
332. Nakhleh, L., Ruths, D., Wang, L.S.: RIATA-HGT: a fast and accurate heuristic for reconstructing horizontal gene transfer. In: Proc. 11th Conf. Computing and Combinatorics (COCOON'05). Lecture Notes in Computer Science. Springer, Berlin (2005)
333. Yu, Y., Than, C., Degnan, J., Nakhleh, L.: Coalescent histories on phylogenetic networks and detection of hybridization despite incomplete lineage sorting. Syst. Biol. **60**, 138–149 (2011)
334. Lapierre, P., Lasek-Nesselquist, E., Gogarten, J.: The impact of HGT on phylogenomic reconstruction methods. Brief. Bioinform. (2012). 10.1093/bib/bbs050
335. Roch, S., Snir, S.: Recovering the tree-like trend of evolution despite extensive lateral genetic transfer: a probabilistic analysis. In: Proceedings RECOMB 2012 (2012)
336. Gerard, D., Gibbs, H., Kubatko, L.: Estimating hybridization in the presence of coalescence using phylogenetic intraspecific sampling. BMC Evol. Biol. **11**, 291 (2011)
337. 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**, e1002660 (2012)
338. Chowdhury, R., Ramachandran, V.: Cache-oblivious dynamic programming. In: Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 591–600 (2006)

Chapter 7
Rearrangements in Phylogenetic Inference: Compare, Model, or Encode?

Bernard M.E. Moret, Yu Lin, and Jijun Tang

Abstract We survey phylogenetic inference from rearrangement data, as viewed through the lens of the work of our group in this area, in tribute to David Sankoff, pioneer and mentor.

Genomic rearrangements were first used for phylogenetic analysis in the late 1920s, but it was not until the 1990s that this approach was revived, with the advent of genome sequencing. G. Watterson et al. proposed to measure the inversion distance between two genomes, J. Palmer et al. studied the evolution of mitochondrial and chloroplast genomes, and D. Sankoff and W. Day published the first algorithmic paper on phylogenetic inference from rearrangement data, giving rise to a fertile field of mathematical, algorithmic, and biological research.

Distance measures for sequence data are simple to define, but those based on rearrangements proved to be complex mathematical objects. The first approaches for phylogenetic inference from rearrangement data, due to D. Sankoff, used model-free distances, such as synteny (colocation on a chromosome) or breakpoints (disrupted adjacencies). The development of algorithms for distance and median computations led to modeling approaches based on biological mechanisms. However, the multiplicity of such mechanisms pose serious challenges. A unifying framework, proposed by S. Yancopoulos et al. and popularized by D. Sankoff, has supported major advances, such as precise distance corrections and efficient algorithms for median estimation, thereby enabling phylogenetic inference using both distance and maximum-parsimony methods.

B.M.E. Moret (✉) · Y. Lin
Laboratory for Computational Biology and Bioinformatics, EPFL, EPFL-IC-LCBB INJ230, Station 14, 1015 Lausanne, Switzerland
e-mail: bernard.moret@epfl.ch

Y. Lin
e-mail: yu.lin@epfl.ch

J. Tang
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA
e-mail: jtang@cse.sc.edu

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*, Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_7,

Likelihood-based methods outperform distance and maximum-parsimony methods, but using such methods with rearrangements has proved problematic. Thus we have returned to an approach we first proposed 12 years ago: encoding the genome structure into sequences and using likelihood methods on these sequences. With a suitable a bias in the ground probabilities, we attain levels of performance comparable to the best sequence-based methods. Unsurprisingly, the idea of injecting such a bias was first proposed by D. Sankoff.

7.1 Introduction

Rearrangement data bypass multiple sequence alignment—an often troublesome step in sequence analysis; they represent the outcome of events much rarer than simple nucleotide mutations and thus hold the promise of high accuracy and of a reach extending back to the very distant past; and they are more closely tied to function (and through it to evolutionary selection) than point mutations. These attractive characteristics are mitigated by our limited understanding of the mechanisms causing large-scale structural changes in genomes.

The use of genomic rearrangements in phylogeny dates back to the early days of genetics, with a series of papers in the 1930s from A. Sturtevant and T. Dobzhansky on inversions in the genome of Drosophila pseudoobscura [21, 87]. However, this early foray had no successor until the 1980s, when G. Watterson proposed to build phylogenies from pairwise distances between circular genomes under inversions [100], J. Palmer and various coauthors published a series of papers on the structure and evolution of mitochondrial and chloroplast genomes in plants [37, 61], including studies of inversions in these genomes and their use in phylogenetic inference [22, 62], and D. Sankoff and W. Day published the first algorithmic paper on phylogenetic inference from rearrangement data [17]. Many hundreds of papers have been published since then on the use of rearrangement data in phylogenetic inference, by biologists, mathematicians, and computer scientists. A first major conference was organized by D. Sankoff and J. Nadeau in 2000 [78], followed by the yearly RECOMB Workshop on Comparative Genomics, started by J. Lagergren, B. Moret, and D. Sankoff.

Algorithms for phylogenetic inference (from any type of data) fall into three main categories. Simplest are the distance-based methods, which reduce the input data to a matrix of pairwise (evolutionary) distances. Next come the Maximum Parsimony (MP) methods, which attempt to find a tree that minimizes the total number of changes required to explain the data. Most complex are the probabilistic methods, either Maximum Likelihood (ML) or Bayesian, which attempt to find a tree (or a population of trees) that maximizes the conditional or posterior probability of observing the data.

In sequence-based phylogenetic inference, distance measures have a long history; all are simple and all have corresponding "distance corrections," that is, maximum-likelihood estimators that yield an estimate of the true (as opposed to observed) pairwise distance. Their simplicity derives in good part from the fact that they are

based on observed results (rather than proposed mechanisms) and on local models of change. Distances based on rearrangements, however, turned out to be complex concepts: it remains possible to define a mechanism-free distance, as D. Sankoff did in 1992 [72, 81], but the model cannot be local, as a single rearrangement can alter the location of almost every gene in the genome. Other distance measures based on mechanisms, whether biological or mathematical (inversion distance, transposition distance, DCJ distance, etc.), have proved yet harder to characterize, with work still ongoing. The absence of localization of changes also means that the fundamental assumption of parsimony- and likelihood-based approaches, independence between different regions of the sequence, does not hold for rearrangement data, thus creating enormous algorithmic difficulties.

Thus the first approaches for phylogenetic inference from rearrangement data used simple distances [6, 74], such as the observed number of nonconserved adjacencies; attempts were also made to encode observed adjacencies in order to use sequence-based inference methods [15, 97]. The development of better algorithms for distance and median computations led to a period during which most approaches were based on documented biological mechanisms for rearrangements, such as inversions, transpositions, translocations, etc. The ability to compute some of these distances efficiently also led researchers to devise suitable distance corrections, some of which resulted in substantial improvements in the quality of inference [54, 55]. The multiplicity of biological mechanisms posed a serious problem, however, since there were no data to quantify their relative preponderance during the course of evolution. Fortunately, a unifying framework was proposed by S. Yancopoulos et al. in 2005 [105] and popularized by D. Sankoff; since then nearly all research on the algorithmics of rearrangements have used this model, leading to very precise distance corrections [41] and efficient and accurate algorithms for median estimation [101] and tree scoring [103]. The precise distance estimates overcame, to a large extent, the weakness of distance-based methods, so that large-scale phylogenetic inference from high-resolution genomes became possible [69].

With sequence data, likelihood-based methods outperform distance-based and MP methods [99]—the two classes of methods used with rearrangement data to date. Direct use of likelihood-based methods was attempted once using a Bayesian approach [39], but the complex mechanisms of rearrangement created insurmountable convergence problems. In the latest step in the evolution of methods for phylogenetic inference from rearrangement data, we have returned to the idea of encoding the genome structure into binary sequences [45] that we first proposed a dozen years ago [15, 97]. This time, however, we use an ML method for inference and inject a bias in its ground probabilities to reflect our better understanding of the evolution of genomic structure; and we have attained levels of performance, in terms of both speed and accuracy, that compare favorably to the best sequence-based methods. Unsurprisingly, the idea of a bias in ground probabilities was first proposed by D. Sankoff in 1998 [75]. Today, then, after over 15 years of research by dozens of groups, phylogenetic inference from rearrangement data is best carried out using a mechanism-free approach and a simple statistical bias in the one operation allowed under the model (transitions between the 0 and the 1 state for each character), much as D. Sankoff advocated from the beginning.

What we present below is a survey of phylogenetic inference from rearrangement data, as viewed through the lens of the work of our group in this area, in tribute to David Sankoff, pioneer and mentor, who more than anyone is responsible for the blossoming of research, unfolding over the last 30 years, on models and algorithms for the evolution of genome structure through rearrangements, duplications, and losses.

7.2 Background

7.2.1 Genome Representations

Each chromosome of the genome is represented by an ordered list of identifiers, each identifier referring to a syntenic block or, more commonly, to a member of a gene family. (In the following, we shall use the word "gene" in a broader sense to denote elements of such orderings and refer to such orderings as "gene orders.") A gene is a stranded sequence of DNA that starts with a tail and ends with a head. The tail of a gene a is denoted by a^t and its head by a^h. We are interested, not in the strand of one single gene, but in the connection of two consecutive genes in one chromosome. Due to different strandedness, two consecutive genes b and c can be connected by one *adjacency* of the following four types: $\{b^t, c^t\}$, $\{b^h, c^t\}$, $\{b^t, c^h\}$ and $\{b^h, c^h\}$. If gene d lies at one end of a linear chromosome, then we have a singleton set, $\{d^t\}$ or $\{d^h\}$, called *telomere*. (These definitions use the notation codified by Bergeron et al. [4].) Given a reference set of n genes $\{g_1, g_2, \ldots, g_n\}$, a genome can be represented by an *ordering* of some multi-subset of these genes. Each gene is given a sign to denote its orientation (strandedness). A genome can be *linear* or *circular*. A linear genome is simply a permutation on the multi-subset, while a circular genome can be represented in the same way under the implicit assumption that the permutation closes back on itself. A multiple-chromosome genome can be represented in the same manner, with telomeres indicating the start and end of a chromosome.

7.2.2 Evolutionary Events

Let G be the genome with signed ordering of $g_1, g_2, \ldots, g_n$. An *inversion* between indices i and j $(i \leq j)$, produces the genome with linear ordering

$$g_1, g_2, \ldots, g_{i-1}, -g_j, -g_{j-1}, \ldots, -g_i, g_{j+1}, \ldots, g_n$$

A *transposition* on genome G acts on three indices i, j, k, with $i \leq j$ and $k \notin [i, j]$, picking up the interval $g_i, g_{i+1}, \ldots, g_j$ and inserting it immediately after g_k. Thus genome G is replaced by (assume $k > j$):

$$g_1, \ldots, g_{i-1}, g_{j+1}, \ldots, g_k, g_i, g_{i+1}, \ldots, g_j, g_{k+1}, \ldots, g_n$$

An *insertion* is the addition of one gene or a segment of genes, and a *deletion* is the loss of same. A section of the chromosome can be *duplicated*, through *tandem duplication*, in which the copied section is inserted immediately after the original, or through *transposed duplication*, in which the copied section is inserted somewhere else. With multichromosomal genomes, additional operations that involve two chromosomes come into play: *translocation* (one end segment in one chromosome is exchanged with one end segment in the other chromosome), *fission* (one chromosome splits and becomes two), and *fusion* (two chromosomes combine to become one).

7.2.3 Distance Computation and Pairwise Genome Comparison

Given two genomes G and G' on the same set of genes, a breakpoint in G is defined as an ordered pair of genes (g_i, g_j) that forms an adjacency in G but not in G'. The *breakpoint distance* [74] is simply the number of breakpoints in G relative to G'.

We define an *edit distance* as the minimum number of events required to transform one genome into the other. (The breakpoint distance is not an edit distance.) We can apply edit distances to any collection of evolutionary operations. Thus we can consider distances under inversions, insertions, and deletions; we can also consider transpositions (within the same chromosome) and translocations (across chromosome), as well as duplications, whether of just one gene, a large segment of a chromosome, or, in the extreme case, the entire genome. The range of possible evolutionary events also include chromosome fusion (merging two chromosomes into one) and fission (the reverse operation), as well as chromosome linearization (turning a circular chromosome into a linear one) and circularization (the reverse event).

When we compute edit distances, we find the minimum number of events required to transform one genome into another, yet evolution need not have proceeded along the shortest path to the current genomes. What we would like to recover is the *true evolutionary distance*, that is, the mean number of evolutionary events on the paths to the two genomes from their last common ancestor. By computing edit distances, we may significantly underestimate the true distance—a problem that also arises with pairwise distances between two sequences or, indeed, between any two objects evolving through some given operation. In sequence analysis, *distance corrections* were devised to provide maximum-likelihood estimators of the true distance given the edit distance. The same can be done, formally or empirically, for rearrangement distances.

7.2.4 Phylogenetic Reconstruction and Ancestral Genome Estimation

Working with genome rearrangement data is computationally much harder than with sequence data. For example, given a fixed phylogeny, the minimum number of evolutionary events can be found in linear time if the leaves of the phylogeny are labeled

with DNA or protein sequences, whereas such a task for rearrangement data is NP-hard, even when the phylogeny has only three leaves [10, 63].

Methods to reconstruct phylogenies from genome rearrangement data include distance-based methods (such as neighbor-joining [70] and minimum-evolution [19]), maximum parsimony and likelihood methods based on encodings [15, 35, 45, 97], and optimization methods, usually based on median computations.

Distance-based methods transform the input genomes into a matrix of pairwise distances, from which the phylogeny is then inferred. Any of the distances introduced above can be used, but inference using estimates of true distances generally outperforms inference based on edit distances [55]; similarly, any distance-based inference method can be used, but FastME [19, 20] appears to outperform most others.

Optimization methods to date have been based on maximum parsimony—they seek the tree and associated ancestral data that minimizes the total number of events. Because this minimization is hard even for a fixed tree of just three leaves, a heuristic first proposed by D. Sankoff in another context [71] and then reused by him for breakpoints (see, e.g., [73]), is widely used. The heuristic uses an iterative improvement approach, based on the recomputation of medians: given three genomes, find a single genome that minimizes the sum of the pairwise distances between itself and each of the three given genomes.

7.3 Comparing: Distance Computations

The first pairwise distance measure used to compare two genomes was a measure of conservation of synteny due to D. Sankoff and J. Nadeau [24, 77]. Inversion and DCJ distances are now the two most commonly used genomic distances. Neither inversions nor DCJ events affect the gene content of a genome: they are pure rearrangements. Little by little, events that do affect gene content, such as deletions, insertions, and duplications, were included in distance computations—a necessity with real genomes. Although various methods have been proposed to combine rearrangements and duplications and losses [14, 84], the problem remains poorly solved.

7.3.1 Inversion Distance

D. Sankoff [72] formulated the fundamental computational problem about inversions: given two signed permutations on the same index set, what is their edit distance under inversions? The breakthrough came in 1995, when S. Hannenhalli and P. Pevzner provided a polynomial-time algorithm to solve this problem [32]. Their algorithm is based on the *breakpoint graph* (see Fig. 7.1). Assume we are given two genomes, G_1 and G_2, with the same n genes and assume that G_2 is the identity. Add two extra "genes" (mathematical markers), gene 0 on the left of the

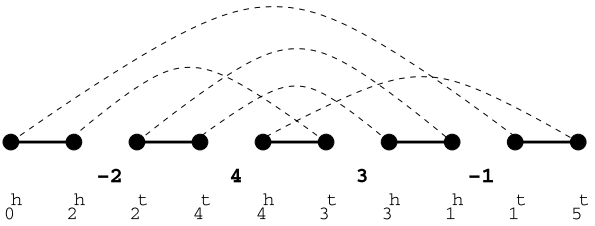


Fig. 7.1 Breakpoint graph between genome $(-2\ 4\ 3\ -1)$ and the identity genome $(1\ 2\ 3\ 4)$

genome and gene $n+1$ on the right. For each gene i in G_1, the breakpoint graph contains two vertices i^h and i^t, connected with two (colored) undirected edges, one for each genome. The *desire edges* (also called gray edges) connect i^h and $(i+1)^t$ for all $0 \leq i \leq n$ and represent the identity genome G_2; they are shown with dashed arcs in Fig. 7.1. The *reality edges* (also called black edges) represent the rearranged genome, G_1; for each adjacency (i, j) in G_1, a reality edge begins at vertex i^h if gene i is positive or at vertex i^t if i is negative, and ends at vertex j^t if j is positive or at vertex j^h if j is negative. Reality edge are shown as solid lines in Fig. 7.1. These edges form cycles of even length that alternate between reality and desire edges; denote by $c(G_1, G_2)$ the number of these cycles. Overlapping cycles in certain configurations create structures known as *hurdles*; denote by $h(G_1, G_2)$ the number of these hurdles. Finally, a very unlikely [89] configuration of hurdles can form a *fortress*. S. Hannenhalli and P. Pevzner [32] proved that the inversion distance between two signed permutations of n genes is given by $n - c(G_1, G_2) + h(G_1, G_2) + (1$ if a fortress is present, 0 otherwise).

D. Bader, M. Yan, and B. Moret [2] later showed that this edit distance can be computed in linear time. Extending this distance to multichromosomal genomes can be done through a reduction to the unichromosomal case using "capping," a subtle process that required several iterations before it was done right [3, 33, 38, 93]. The various operations supported under this multichromosomal model (for which see the next section), all of which keep the gene content intact, give rise to what we shall call the HP-distance. The *transposition distance* is known to be NP-hard to compute [8]; attempts at defining distances combining transpositions and inversions have so far proved unsuccessful.

Moving to distances between genomes of unequal gene content has proved very challenging. N. El-Mabrouk [25] first extended the results of S. Hannenhalli and P. Pevzner to the computation of edit distances for inversions and deletions and gave a heuristic for inversions, deletions, and non-duplicating insertions. The distance computation is NP-hard when duplications and inversions are present [12]. M. Marron et al. [48] gave a guaranteed approximation for edit distances under arbitrary operations (including duplications and deletions).

7.3.2 DCJ Distance

S. Yancopoulos, O. Attie, and R. Friedberg [105] proposed a double-cut-and-join (DCJ) operation that accounts for inversions, translocations, fissions, and fusions,

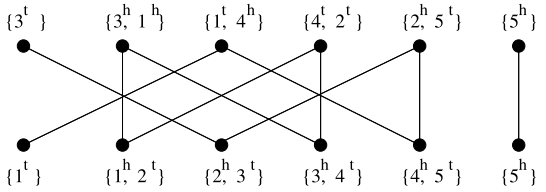


Fig. 7.2 Adjacency graph and DCJ distance of two genomes $G_1 = (3, -1, -4, 2, 5)$ and $G_2 = (1, 2, 3, 4, 5)$. The number of cycles C is 1, the number of odd paths I is 2, the DCJ distance is $N - (C + I/2) = 3$

yielding in a new genomic distance that can be computed in linear time. As its name indicates, a DCJ operation makes a pair of cuts in the chromosomes and reglues the four cut ends into two adjacencies or telomeres, giving rise to four cases:

- A pair of adjacencies $\{i^u, j^v\}$ and $\{p^x, q^y\}$ can be replaced by the pair $\{i^u, p^x\}$ and $\{j^v, q^y\}$ or by the pair $\{i^u, q^y\}$ and $\{j^v, p^x\}$.
- An adjacency $\{i^u, j^v\}$ and a telomere $\{p^x\}$ can be replaced by the adjacency $\{i^u, p^x\}$ and telomere $\{j^v\}$ or by the adjacency $\{j^v, p^x\}$ and telomere $\{i^u\}$.
- A pair of telomeres $\{i^u\}$ and $\{j^v\}$ can be replaced by the adjacency $\{i^u, j^v\}$.
- An adjacency $\{i^u, j^v\}$ can be replaced by the pair of telomeres $\{i^u\}$ and $\{j^v\}$.

Given two genomes G_1 and G_2, their DCJ distance can be computed using the adjacency graph $AG(G_1, G_2)$. The adjacency graph has a vertex for each adjacency and each telomere of G_1 and G_2 and, for each $u \in G_1$ and $v \in G_2$, has $|u \cap v|$ edges between u and v (see Fig. 7.2). S. Yancopoulos et al. [105] and, for the multichromosomal formulation, A. Bergeron et al. [4], showed that the DCJ distance between G_1 and G_2 is just $d_{DCJ}(G_1, G_2) = n - (C + I/2)$, where C is the number of cycles and I the number of odd paths in the adjacency graph. While there is no single biological mechanism to mirror the DCJ operation, researchers everywhere have adopted the DCJ model in their work because of its mathematical simplicity and because of its observed robustness in practice.

The DCJ operator, like the operator in the multichromosomal rearrangement model of S. Hannenhalli and P. Pevzner, preserves gene content. The DCJ model has been extended, with mixed results, to handle insertions and deletions [14, 84], duplications [1], and all three [82]. Inversion distances, HP distances, and DCJ distances are all implemented in the UniMoG software [34].

7.3.3 Estimating True Distances

Edit distances underestimate the true number of events, particularly so when the two genomes are distant. Estimating the true distance through a maximum-likelihood approach requires an evolutionary model. Since models are the subject of the next section, we focus here on the estimators themselves. The IEBP estimator [95] (and

its improved version [94]) uses a Markov chain formulation to provide an exact formula for the expected number of events (inversions, transpositions, and inverted transpositions—the events in the Nadeau–Taylor model) as a function of the breakpoint (edit) distance. The EDE estimator [51] uses curve-fitting to approximate the most likely number of evolutionary events, derived by simulating known numbers of inversions and comparing that number against the computed inversion (edit) distance. The formula thus takes an inversion (edit) distance and returns an estimate of the true number of inversions. Although EDE was designed just for inversions, experience shows that it works well even with a significant number of transpositions and that its use significantly improves the accuracy of distance-based phylogenetic reconstruction [96].

K. Swenson et al. [88] proposed a heuristic to approximate the true evolutionary distance under inversions, duplications, insertions, and deletions for unichromosomal genomes. Y. Lin and B. Moret [41] developed a true distance estimator for the DCJ model based on the DCJ distance; later Y. Lin et al. [43] gave an estimator for the true number of events in the DCJ model based directly on the lists of gene adjacencies—a useful generalization as it can be used even for lists of adjacencies that do not define a "real" genome. Given genome G, for any genome G^*, we can divide the adjacencies and telomeres of G^* into four sets, $SA(G^*)$, $ST(G^*)$, $DA(G^*)$ and $DT(G^*)$, where $SA(G^*)$ is the set of adjacencies of G^* that also appear in G, $ST(G^*)$ is the set of telomeres of G^* that also appear in G, $DA(G^*)$ is the set of adjacencies of G^* that do not appear in G, and $DT(G^*)$ is the set of telomeres of G^* that do not appear in G. Then we can calculate a vector $V_G(G^*) = (SA^*, ST^*, DA^*, DT^*)$ to represent the genome G^* in terms of G, where SA^*, ST^*, DA^* and DT^* are the cardinalities of the sets $SA(G^*)$, $ST(G^*)$, $DA(G^*)$ and $DT(G^*)$, respectively. Obviously, we have $2n = 2SA^* + ST^* + 2DA^* + DT^*$. Let G^k be the genome obtained from $G = G^0$ by applying k randomly selected DCJ operations. The $(i+1)$st DCJ operation is selected from a uniform distribution of all possible DCJ operations on the current genome G^i. We can compute the vector $V_G(G^k) = (SA^k, ST^k, DA^k, DT^k)$ to represent the genome G^k with respect to G. For any integer $k > 0$, we can also produce the estimate $\widetilde{E}(V_G(G^k)) = (\widetilde{SA}^k, \widetilde{ST}^k, \widetilde{DA}^k, \widetilde{DT}^k)$ for the expected vector $E(V_G(G^k))$. We then use $\widetilde{SA}^k$ to approximate the expected number of adjacencies present in both G and G^k and compute SA^F from G and G^F. The estimated true number of evolutionary events is then the integer k that minimizes the difference $|SA^F - \widetilde{SA}^k|$. This estimator is quite robust and achieves better performance than the EDE estimator; it was later extended [43] to include gene losses and duplications.

7.4 Modeling Genomic Evolution

Models for genomic rearrangements have been studied intensely over the last 30 years by biologists, computer scientists, and mathematicians—for an overview of the work of the latter two, see, e.g., [29, 50, 56]. We briefly review the main models used in phylogenetic inference.

7.4.1 Inversions

Of the various genomic rearrangements studied, perhaps the simplest and best documented is the *inversion* (also called reversal), through which a segment of a chromosome is inverted in place. In 1989, D. Sankoff and M. Goldstein formalized a probabilistic model of genomic inversions in which a chromosome is represented as a permutation of gene indices [76]; in this framework, an inversion acts on an interval of the permutation by reversing the order in which the indices appear within the interval. A year later, D. Sankoff et al. [80] also proposed to apply this model to the assessment of evolutionary relationships among a number of bacterial genomes using genetic maps. Two years later, D. Sankoff used gene-order data from 16 mitochondrial genomes from fungi and other eukaryotes to infer a species phylogeny; the study used nearly complete genomic sequences for the organelles [81].

7.4.2 The Generalized Nadeau–Taylor Model

The generalized Nadeau–Taylor (NT) model [57] deals with inversions, transpositions, and inverted transpositions; it assumes that the number of each of those three events obeys a Poisson distribution on each edge, that the relative probabilities of each type of event are fixed across the tree, and that events of a given type have the same probability, regardless of the location and size of the affected region. The model thus has two main parameters—two of the three relative probabilities of occurrence of the three types of events. The generalized Nadeau–Taylor model can be extended to include insertions, deletions, or duplications in the triplet, by choosing the relative probabilities for all events. Nevertheless, the generalized NT model remains far from realistic, and many features of genome evolution are lacking in this model, such as hot spots (a much debated feature [18, 64, 65, 79]), operons, as well as fission and fusion events.

7.4.3 The HP model

The first model of multichromosomal rearrangements was given by S. Hannenhalli and P. Pevzner [33]. This HP model includes inversions, translocations, fusions, and fissions (see Fig. 7.3).

Inversion: Given a linear chromosome $A = (a_1 \dots a_i a_{i+1} \dots a_n)$, reverse all genes between a_i and a_j yields $(a_1 \dots a_{i-1} - a_j \dots - a_i a_{j+1} \dots a_n)$. Two adjacencies, $\{a^h_{i-1}, a^t_i\}$ and $\{a^h_j, a^t_{j+1}\}$, are replaced by two others, $\{a^h_{i-1}, a^h_j\}$ and $\{a^t_i, a^t_{j+1}\}$.

Translocation: Given two linear chromosomes $A = (a_1 \dots a_i a_{i+1} \dots a_n)$ and $B = (b_1 \dots b_j b_{j+1} \dots b_m)$, exchange two segments between these two chromosomes. There are two possible outcomes, $(a_1 \dots a_i b_{j+1} \dots b_m)$ and $(b_1 \dots b_j a_{i+1} \dots a_n)$

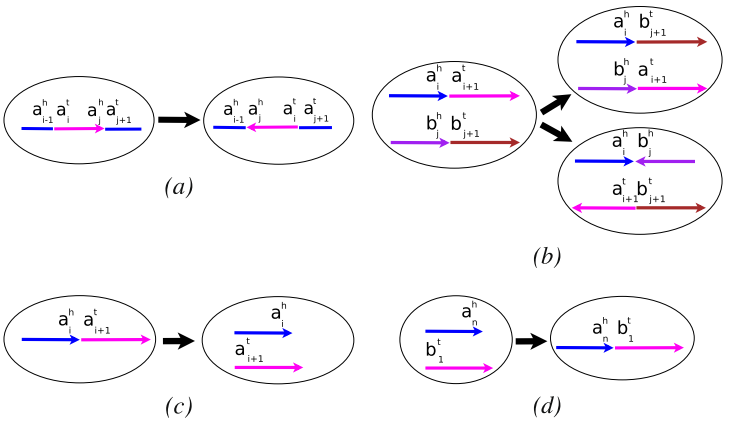


Fig. 7.3 Possible rearrangements in the HP model: (**a**) inversion, (**b**) translocation, (**c**) fission, and (**d**) fusion

or $(a_1 \ldots a_i - b_j \ldots - b_1)$ and $(-b_n \ldots - b_{j+1} a_{i+1} \ldots a_n)$. Two adjacencies, $\{a_i^h, a_{i+1}^t\}$ and $\{b_j^h, b_{j+1}^t\}$, are replaced by $\{a_i^h, b_{j+1}^h\}$ and $\{a_{i+1}^t, b_j^t\}$ or $\{a_i^h, b_j^h\}$ and $\{a_{i+1}^t, b_{j+1}^t\}$.

Fission: Given a linear chromosome $A = (a_1 \ldots a_i a_{i+1} \ldots a_n)$, split A into two new linear chromosomes, $(a_1 \ldots a_i)$ and $(a_{i+1} \ldots a_n)$. The adjacency $\{a_i^h, a_{i+1}^t\}$ is replaced by two telomeres $\{a_i^h\}$ and $\{a_{i+1}^t\}$.

Fusion: Given two linear chromosomes $A = (a_1 \ldots a_n)$ and $B = (b_1 \ldots b_m)$, concatenate these two linear chromosomes into a single new chromosome $(a_1 \ldots a_n b_1 \ldots b_m)$. Two telomeres, $\{a_n^h\}$ and $\{b_1^t\}$, are replaced by one adjacency $\{a_n^h, b_1^t\}$.

In 2009, A. Bergeron et al. [5] gave an optimal linear-time algorithm to compute the HP distance with a simple formula.

7.4.4 The DCJ Model

The HP model cannot mix linear and circular chromosomes. In such a mix, additional operations come into play: linearization and circularization, which transform circular chromosomes into linear ones and vice versa. The DCJ operation, on the other hand, does model such a mix and supports every simple rearrangement: inversions, transpositions, block exchanges, circularizations, and linearizations, all of which act on a single chromosome, and translocations, fusions, and fissions, which act on a pair of chromosomes. The DCJ model is less well motivated than the HP model in terms of biology: whereas inversions and translocations are well documented, the additional rearrangements possible in the DCJ model may not correspond to actual evolutionary events. For example, if the two cuts are in the same

linear chromosome, one of the two nontrivial rejoinings causes a fission, excising a portion of the original chromosome and packaging that portion as a new circular chromosome—something usually called a "circular intermediate," the name itself denoting the opinion that such structures are at best ephemeral. (In the best tradition of biology, where "anything that can happen already has," the existence of circular intermediates has been inferred in vertebrates [23, 31].) A simple modification to the DCJ model (forbidding the least realistic operation) can lead genomes into the two stable structures (single circular chromosome or multiple linear chromosomes) found in the vast majority of prokaryotes and eukaryotes, respectively [42]. The main argument for the DCJ model is its mathematical simplicity: it is much easier to reason about and devise algorithms for a model with a single operation (however multifaceted) than for one with a "zoo" of operations.

7.4.5 Models for Rearrangements, Duplications, and Losses

Gene (or segment) duplications and losses have long been studied by geneticists and molecular biologists. A particularly spectacular version of duplication is *whole genome doubling (WGD)*, the duplication of the entire genome—a very rare event, but one often viewed as responsible for much the diversity of life forms. WGD has been of particular interest to evolutionary biologists for many years. D. Sankoff et al. integrated WGD with rearrangements, and introduced the Genome Halving Problem [26–28]: from the present-day doubled and rearranged genome, recover the pre-doubling ancestral genome using a criterion of maximum parsimony. Since the ancestral solutions are often numerous, D. Sankoff et al. proposed to take into account external reference genomes as outgroups, to guide and narrow down the search [92, 107].

Segmental duplications and gene losses do not affect just gene content: they can mimic the effect of rearrangements; the most obvious example is transposition: instead of moving a segment from one location to another, one can envision deleting that segment from its original location and inserting it at its new location. N. El-Mabrouk [25] gave an exact algorithm to compute edit distances for inversions and losses and also a heuristic to approximate edit distances for inversions, losses, and non-duplicating insertions. Her work was then extended and refined by M. Marron et al. [48]. In 2008, S. Yancopoulos and R. Friedberg [104] gave an algorithm to compute edit distances under deletions, insertions, duplications, and DCJ operations, under the constraint that each deletion can only remove a single gene. These and other approaches targeted the edit distance, not the true evolutionary distance. K. Swenson et al. [88] gave a first heuristic to approximate the true evolutionary distance under inversions, duplications, and losses; more recently, Y. Lin et al. [43] gave an algorithm to estimate the true evolutionary distance under deletions, insertions, duplications, and inversions. Rearrangements, duplications, and losses have also been addressed in the framework of ancestral reconstruction [47, 60], a topic of rapidly increasing interest.

7.4.6 Inferring Phylogenies Using Models

In phylogenetic inference, the main use of models is to guide inference phrased as an optimization problem, using maximum parsimony or maximum likelihood criteria. The first program used for phylogenetic inference from rearrangement data, D. Sankoff's `BPAnalysis`, used the breakpoint model and median-based heuristics aimed at obtained a tree of maximum parsimony. The approach proposed by D. Sankoff is based on scoring each possible tree separately. Since scoring a single tree is NP-hard [63], the scoring procedure is a heuristic, using iterative improvement, that D. Sankoff himself had proposed much earlier for multiple sequence alignment on trees [71]. First, each internal node of the tree is initialized with some "ancestral genome." Then, and until no changes can be made to any internal node, each internal node is examined in turn (according to some chosen scheduling): the median of its immediate neighbors' "genomes" is computed and, if better (giving a lower parsimony score) than the current genome at the node in question, is used to replace that genome. Computing the median of three or more genomes is itself NP-hard [10, 63]—indeed, it is just the simplest case of parsimony scoring, for a tree of diameter 2. Thus the overall procedure nests some unknown number of instances of an NP-hard problem within an exponential loop: obviously such an approach cannot scale up very far.

D. Sankoff and M. Blanchette [74] provided the first software package for the problem, `BPAnalysis`, subsequently improved by our group with `GRAPPA` [16, 52]. `BPAnalysis` handled 8 genomes of 40 genes each; `GRAPPA` [52] scaled to 15 genomes and a few hundred genes and supported both breakpoint and inversion models (with both edit and estimated true distances). Reusing the `GRAPPA` code for inversion distance computation, P. Pevzner's group produced `MGR` [7], which could handle multichromosomal genomes and, rather than scoring every tree, used a construction heuristic to build a single tree in incremental fashion. In doing the latter, `MGR` came closer to sequence-based MP (or ML) algorithms, none of which, naturally, scores every tree; but the lack of tight bounding for subtrees (an indirect consequence of the complexity of rearrangement operations) meant that the construction heuristic gave poor results—in direct comparisons, the exhaustive approach of `BPAnalysis` and `GRAPPA` inferred better trees. Further developments of `GRAPPA`, such as very tight bounds on entire trees (but not on subtrees) [91], reduced the space to be explored by orders of magnitude, while high-performance methods [53] further increased its speed, yet scaling such an algorithm requires an entirely different algorithmic approach, such as the Disk-Covering Methods developed by T. Warnow et al. [36], used with some success in combination with `GRAPPA` [90].

D. Sankoff [73] had shown that seeking a median that minimizes the breakpoint distance can be transformed into a special instance of the well-studied Traveling Salesperson Problem and thus can be solved relatively efficiently (as shown in `GRAPPA`). In practice, however, computing breakpoint medians yields poor solutions; a better approach is to use the inversion median [54], although exact solvers for this problem scale poorly [11, 83, 106]. Thanks to the relative simplicity of the

DCJ model, DCJ median solvers are easier to design and perform better than inversion solvers, so that parsimony methods using DCJ median solvers outperform other methods in terms of speed and accuracy [68]. Among existing DCJ median solvers, the best to date appears to be `ASMedian` [102], due to W. Xu and D. Sankoff, and our extension `GASTS` [103], used to produce the parsimony score of a given tree.

As the algorithmic community expended considerable energy on improving the initialization, the computation of medians, and the exploration of the search space, it became clear that scoring through iterative improvement was going to cause serious problems, due to a combination of two factors. First, the number of local minima for parsimony scoring is huge, so that the number of times a median gets improved tends to one as the instance gets larger, with consequent poor results. Secondly, the error accumulates quickly as the distance from the leaves increases: what works reasonably well on very small trees of 10–20 leaves (in which most internal nodes are within 1–2 edges of a leaf) fails more and more (and worse and worse) as the trees became larger. Better median computation helps reduce the second problem, as better initialization helps reduce the first: both are deployed in the `GASTS` scoring software [103], but at a significant expense of computational time.

The difficulty of deriving bounds for the completion of partial trees means that the standard approaches used in sequence-based MP and ML inference cannot be used today with rearrangement data. There has been one attempt to use a probabilistic approach, using Bayesian inference (the `BADGER` tool) [39], but it could not achieve reasonable scaling and kept suffering from convergence problems since it was not clear what steps should be used in the construction of the Markov chains. Until new results in mathematics and algorithms are obtained to provide bounds on partial trees, model-based inference of phylogenies under MP or ML remains restricted to small instances.

7.5 Encodings

Distance-based methods suffer from the problem of *saturation*: the observed changes may be only a small piece of the history of changes and any attempt at estimating the true number of changes from a large observed number of changes will suffer from a very large variance. In good part, this problem stems from the pairwise approach of these methods: if two leaves in the tree have the root as their last common ancestor, then the pairwise distance is an unreliable predictor of the length of the tree path connecting these two leaves. Methods that score trees rather than pairs can take advantage of the smaller evolutionary steps represented by tree edges: they do not compute any pairwise distances between leaves, focusing instead on pairwise comparisons between the endpoints of a tree edge. Unfortunately the optimization problem for rearrangement data solved by such methods (maximum parsimony or maximum likelihood) is NP-hard even on a fixed tree. For sequence data, however, scoring a single tree under parsimony can be done in linear time, while fast and well tested packages exist to compute the MP or ML tree. Thus a natural approach is to

produce sequences from the input permutations, use a sequence-based phylogenetic inference package, then analyze the resulting tree in terms of rearrangements.

7.5.1 Parsimonious Methods

The idea of encoding the genome structure into sequences was introduced a dozen years ago [15, 97], based on an earlier characterization approach of D. Sankoff and M. Blanchette [74]. Two encoding methods were proposed:

- Maximum Parsimony on Binary Encodings (MPBE): each genome is translated into a binary sequence, where each site from the binary sequence corresponds to an adjacency present in at least one of the input genomes. Uninformative sites are discarded.
- Maximum Parsimony on Multistate Encodings (MPME): each genome of n genes is translated into a sequence of length $2n$, where site i takes the index of the gene immediately following gene i and site $n + i$ takes the index of the gene immediately following gene $-i$, for $1 \leq i \leq n$.

Results show that an encoding based on adjacencies preserves useful phylogenetic information that a parsimonious search can put to good use. However, both MPBE and MPME proved too computationally expensive compared to distance-based methods, while MPME was ill suited for use with existing MP and ML inference packages because of its large number of character states. Moreover, accuracy with MP inference was not significantly better than accuracy using distance-based methods.

7.5.2 Likelihood-Based Approaches

In the last few years, likelihood-based inference packages such as `RAxML` [86] and `FastTree` [67] have largely overcome computational limitations and allowed reconstructions of large trees (with tens of thousands of taxa) and the use of long sequences (to several hundred thousand characters). In 2011, F. Hu et al. [35] applied likelihood-based inference to an unusual encoding scheme, in which the same adjacency could be translated into multiple character positions. Results on bacterial genomes were promising, but difficult to explain, while the method appeared too time-consuming to handle eukaryotic genomes.

In 2013, we described MLWD (ML on Whole-genome Data) [45], a new approach that encodes genomic structure into binary sequences using both gene adjacencies and gene content, estimates the transition parameters for the resulting binary sequence data, and finally uses sequence-based ML reconstruction to infer the tree. For each adjacency or telomere within the entire collection of genomes, there exists exactly one position in the sequence, with 1 indicating presence of this adjacency in

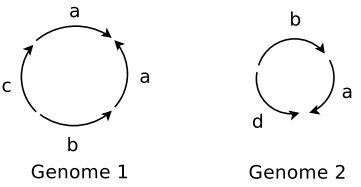


Fig. 7.4 Two toy genomes

a genome and 0 indicating absence. If the total number of distinct genes among the input genomes is n, then the total number of distinct adjacencies and telomeres cannot exceed $\binom{2n+2}{2}$, but the actual number is typically far smaller—it is usually linear in n rather than quadratic. For each gene family within the collection of genomes, there is exactly one position, with the same meaning attributed to the Boolean values. For the two toy genomes of Fig. 7.4, the resulting binary sequences and their derivation are shown in Table 7.1.

Since the encodings are binary sequences, the parameters of the model are simply the transition probability from presence (1) to absence (0) and that from absence (0) to presence (1). In rearrangements, every DCJ operation will select two adjacencies (or telomeres) uniformly at random, and, if adjacencies, break them to create two new adjacencies. Each genome has $n + O(1)$ adjacencies and telomeres ($O(1)$ is the number of linear chromosomes in the genome, viewed as a small constant). Thus the transition probability from 1 to 0 under one DCJ operation at some fixed index in the sequence is $\frac{2}{n+O(1)}$. Since there are up to $\binom{2n+2}{2}$ possible adjacencies and telomeres, the transition probability from 0 to 1 is $\frac{2}{2n^2+O(n)}$. Thus the transition from 0 to 1 is roughly $2n$ times less likely than that from 1 to 0. Despite the restrictive assumption that all DCJ operations are equally likely, this result is in line with general opinion about the probability of eventually breaking an ancestral adjacency (high) vs. that of creating a particular adjacency along several lineages (low)—a version of homoplasy for adjacencies. The probability of losing a gene independently along several lineages is high, whereas the probability of gaining the same gene independently along several lineages (the standard homoplasy) is low. Unsurprisingly, D. Sankoff first observed and studied such a bias in transitions of adjacencies in 1998 [75].

Figure 7.5 shows the Robinson–Foulds (RF) error rates of three different approaches, MLWD, MLWD* and TIBA. (The RF rate measures the difference between two trees as the ratio of the number edges present in one tree but not in the other to the total number of edges and is the most commonly used measure of phylogenetic accuracy.) MLWD*, used as a control for bias, follows the same pro-

Table 7.1 The binary encodings for the two genomes of Fig. 7.4

	Adjacency information						Content information			
	$\{a^h, a^h\}$	$\{a^t, b^h\}$	$\{a^t, c^h\}$	$\{b^t, c^t\}$	$\{a^h, d^h\}$	$\{b^t, d^t\}$	a	b	c	d
Genome 1	1	1	1	1	0	0	1	1	1	0
Genome 2	0	1	0	0	1	1	1	1	0	1

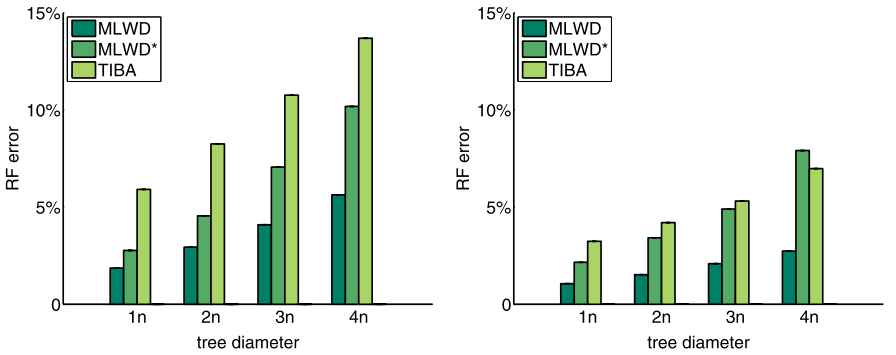


Fig. 7.5 RF error rates for different approaches for trees with 100 species, with genomes of 1,000 and 5,000 genes and tree diameters from one to four times the number of genes, under the rearrangement model

cedure as MLWD, but without setting the bias explicitly, while TIBA [44] is a fast distance-based tool to reconstruct phylogenies from rearrangement data, combining a pairwise distance estimator [41] and the FastME [19] distance-based reconstruction method. These simulations show that MLWD can reconstruct much more accurate phylogenies from rearrangement data than the distance-based approach TIBA, in line with experience in sequence-based reconstruction. MLWD also outperforms MLWD*, underlining the importance of estimating and setting the transition biases before applying the sequence-based maximum-likelihood method.

Figure 7.6 (from [45]) shows the MLWD-inferred phylogeny for 68 eukaryotic genomes from the eGOB (Eukaryotic Gene Order Browser) database [46]. The database contains the order information of orthologous genes (identified by OrthoMCL [13]) of 74 different eukaryotic species; the total number of different gene markers in eGOB is around 100,000. We selected 68 genomes with from $3k$ to $42k$ gene markers—the remaining six genomes in the database have too few adjacencies (fewer than 3,000). We encoded the adjacency and gene content information of all 68 genomes into 68 binary sequences of length 652,000. Inferring this phylogeny (using RAxML and setting the bias ratio to 100) took under 3 hours on a desktop computer, showing that MLWD can easily handle high-resolution genomic data.

As shown in Fig. 7.6, all major groups in those 68 eukaryotic genomes are correctly identified, with the exception of Amoebozoa. Those incorrect branches with respect to Amoebozoa receive extremely low bootstrap values (0 and 2), indicating that they are very likely to be wrong. For the phylogeny of Metazoa, the tree is well supported from existing studies [66, 85]. For the phylogeny of model fish species (D. rerio, G. aculeatus, O. latipes, T. rubripes, and T. nigroviridis), two conflicting phylogenies have been published, using different choices of alignment tools and reconstruction methods for sequence data [58]. Our result supports the second phylogeny, which is considered as the correct one by the authors in their discussion [58]. For the phylogeny of Fungi, our results agree with most branches for common

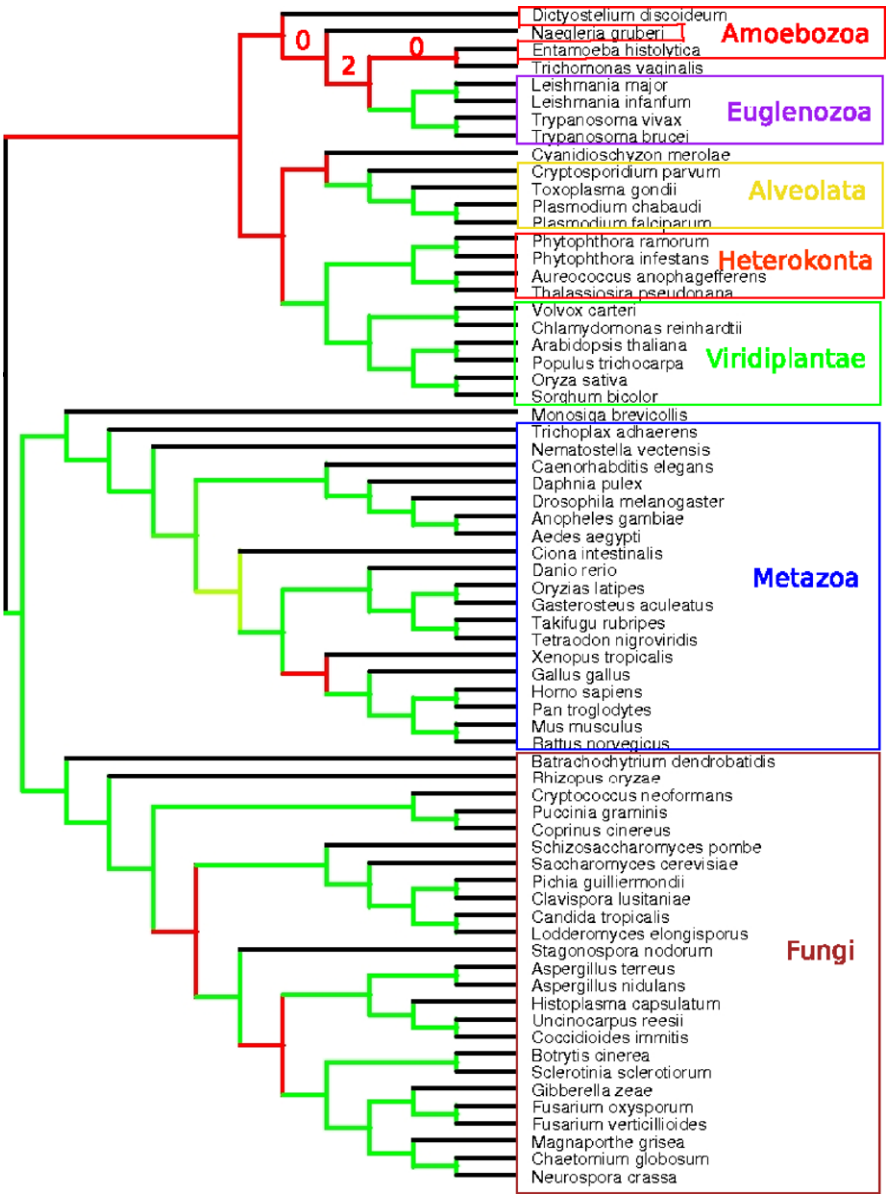


Fig. 7.6 The inferred phylogeny of 68 eukaryotic genomes, drawn with iTOL [40]. Internal branches are colored *green*, *yellow*, and *red*, to indicate, respectively, strong (bootstrap value >90), medium (bootstrap value between 60 and 90), and weak support (bootstrap value <60)

species in recent studies [30, 98]. It is worth mentioning that among three Chytridiomycota species C. cinereus, P. gramnis, and C. neoformans, our phylogeny shows that C. cinereus and P. gramnis are more closely related, which conflicts with the

placement of C. cinereus and C. neoformans as sister taxa, but with very low support value (bootstrapping score 35) [98]. The phylogenetic placement of C. merolae, a primitive red algae, has been the topic of a long-running debate [59]. Our result suggests that C. merolae is closer to Alveolata than to Viridiplantae, in agreement with a recent finding obtained by sequencing and comparing expressed sequence tags from different genomes [9].

This approach opens the way to widespread use of whole-genome data in phylogenetic analysis, as it uses a fairly general model of genomic evolution (rearrangements plus duplications, insertions, and losses of genomic regions), is very accurate, scales as well as sequence-based approaches, and, importantly, supports standard bootstrapping methods. In addition, the nature of the encoding makes it robust against typical errors in genome assembly or in the identification of genes or syntenic blocks, as a few erroneous entries in a sequence of some hundreds of thousands of characters have little impact on the outcome. Moreover, the encoding can be modified to increase robustness by coding for proximity rather than just adjacency; such an encoding could use degrees of proximity to maintain discrimination among local rearrangements or deliberately treat all neighbors in the same manner to create invariance with respect to local rearrangements—a useful property when dealing with bacterial operons.

7.6 Conclusions

The shifting emphasis on simple comparisons, model-based distance computations, and encoding of features into sequences reflects both our increased understanding of rearrangements and duplication in genomes and the well known superiority (under most circumstances) of likelihood-based approaches in phylogenetic inference. Starting with very simple measures (the number of breakpoints) and with attempts at encoding the genomic structure into sequence data (in both cases because anything else remained unsolved), we have moved to computing model-based edit distances, then to estimate model-based true distances, then to use these as tools in median heuristics for a parsimonious approach. Most recently, we have returned to encodings, not for lack of alternatives, but because our deeper understanding of duplications and rearrangements in terms of adjacencies has led us to such a step. Yet the encoding is at least partly motivated by the impossibility, at this time, of using a direct approach to ML inference, of the style used for sequence data: a direct approach would require some parameterized model of genomic evolution under rearrangements and duplications and all models to date are both overly simplistic in terms of biology and far too complex for a Bayesian or ML inference strategy. (Even were such a model to emerge, a direct approach would remain a formidable algorithmic challenge, because of the lack of bounding methods for partial trees.) Thus the encoding of Y. Lin et al. is to evolutionary genomics much what binary character encoding is to morphological evolution: a way to take very rich and complex data produced through poorly understood events and to reduce them to a simple formulation that can be handled with today's phylogenetic inference tools.

From completing with some difficulty the inference of a phylogeny for fewer than 10 species with mitochondrial data featuring fewer than 50 common, single-copy genes using a rather inaccurate heuristic for maximum parsimony (the original `BPAnalysis`), we now have moved, thanks to this latest encoding approach, to easy handling of datasets of hundreds of species with tens of thousands of genes, many of them duplicated or missing in many of the species, using standard tools from sequence-based analysis. Yet it is clearly not the final word on phylogenetic reconstruction from rearrangement data: this area of research is little more than 15 years old and sufficient data to support it have been available for less than half that time. Challenges range from modeling and algorithmic questions to implementation and assessment [49]. As new data accumulate at a frenetic pace and our understanding of the genome deepens with the daily additions to the research literature, we expect further insights, better models, refined methodology, and some breakthroughs—and, as has now been the case for nearly 30 years, these next developments are likely to be inspired by some past or forthcoming publication or remark of David Sankoff's.

References

1. Bader, M.: Genome rearrangements with duplications. BMC Bioinform. **11**(Suppl. 1), S27 (2010)
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. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. In: Proc. 9th Ann. Int'l Conf. on Research in Computational Molecular Biology (RECOMB'05). Lecture Notes in Comp. Sci., vol. 3500, pp. 615–629 (2005)
4. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proc. 6th Workshop Algs. in Bioinf. (WABI'06). Lecture Notes in Comp. Sci., vol. 4175, pp. 163–173. Springer, Berlin (2006)
5. 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)
6. Blanchette, M., Bourque, G., Sankoff, D.: Breakpoint phylogenies. In: Miyano, S., Takagi, T. (eds.) Genome Informatics, pp. 25–34. Univ. Academy Press, Tokyo (1997)
7. Bourque, G., Pevzner, P.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res. **12**, 26–36 (2002)
8. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. In: Proc. 38th Int'l Colloq. on Automata, Languages, and Programming (ICALP 2011). Lecture Notes in Comp. Sci., vol. 6756. Springer, Berlin (2011)
9. Burki, F., et al.: Phylogenomics reshuffles the eukaryotic supergroups. PLoS ONE **2**(8), e790 (2007)
10. Caprara, A.: Formulations and hardness of multiple sorting by reversals. In: Proc. 3rd Int'l Conf. Comput. Mol. Biol. (RECOMB'99), pp. 84–93. ACM Press, New York (1999)
11. Caprara, A.: On the practical solution of the reversal median problem. In: Proc. 1st Workshop Algs. in Bioinf. (WABI'01). Lecture Notes in Comp. Sci., vol. 2149, pp. 238–251. Springer, Berlin (2001)
12. Chen, X., Zheng, J., Fu, Z., Nan, P., Zhong, Y., Lonardi, S., Jiang, T.: Computing the assignment of orthologous genes via genome rearrangement. In: Proc. 3rd Asia Pacific Bioinf. Conf. (APBC'05), pp. 363–378. Imperial College Press, London (2005)

13. Chen, F., Mackey, A., Vermunt, J., Roos, D.: Assessing performance of orthology detection strategies applied to eukaryotic genomes. PLoS ONE **2**(4), e383 (2007)
14. Compeau, P.: A simplified view of DCJ-Indel distance. In: Proc. 12th Workshop Algs. in Bioinf. (WABI'12). Lecture Notes in Comp. Sci., vol. 7534, pp. 365–377. Springer, Berlin (2012)
15. Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L., Warnow, T., Wyman, S.: An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In: Sankoff, D., Nadeau, J. (eds.) Comparative Genomics, pp. 99–122. Kluwer Academic, Dordrecht (2000)
16. Cosner, M., Jansen, R., Moret, B., Raubeson, L., Wang, L., Warnow, T., Wyman, S.: A new fast heuristic for computing the breakpoint phylogeny and experimental phylogenetic analyses of real and synthetic data. In: Proc. 8th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB'00), pp. 104–115 (2000)
17. Day, W., Sankoff, D.: The computational complexity of inferring phylogenies from chromosome inversion data. J. Theor. Biol. **127**, 213–218 (1987)
18. Demongeot, J., et al.: Hot spots in chromosomal breakage: from description to etiology. In: Sankoff, D., Nadeau, J. (eds.) Comparative Genomics. Computational Biology, vol. 1, pp. 71–83. Springer, Berlin (2000)
19. Desper, R., Gascuel, O.: Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. J. Comput. Biol. **9**(5), 687–705 (2002)
20. 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 (2003)
21. Dobzhansky, T., Sturtevant, A.: Inversions in the chromosomes of Drosophila pseudoobscura. Genetics **23**(1), 28–64 (1938)
22. Downie, S.R., Palmer, J.D.: Use of chloroplast DNA rearrangements in reconstructing plant phylogeny. In: Soltis, D., Soltis, P., Doyle, J. (eds.) Molecular Systematics of Plants, pp. 14–35. Chapman and Hall, New York (1992)
23. Durkin, K., et al.: Serial translocation by means of circular intermediates underlies colour sidedness in cattle. Nature **482**(7383), 81–84 (2012)
24. Ehrlich, J., Sankoff, D., Nadeau, J.: Synteny conservation and chromosome rearrangements during mammalian evolution. Genetics **147**, 289–296 (1997)
25. El-Mabrouk, N.: Genome rearrangement by reversals and insertions/deletions of contiguous segments. In: Proc. 11th Ann. Symp. Combin. Pattern Matching (CPM'00). Lecture Notes in Comp. Sci., vol. 1848, pp. 222–234. Springer, Berlin (2000)
26. El-Mabrouk, N., Sankoff, D.: The reconstruction of doubled genomes. SIAM J. Comput. **32**(3), 754–792 (2003)
27. El-Mabrouk, N., Nadeau, J., Sankoff, D.: Genome halving. In: Proc. 9th Ann. Symp. Combin. Pattern Matching (CPM'98). Lecture Notes in Comp. Sci., pp. 235–250. Springer, Berlin (1998)
28. El-Mabrouk, N., Bryant, D., Sankoff, D.: Reconstructing the pre-doubling genome. In: Proc. 3rd Int'l Conf. Comput. Mol. Biol. RECOMB'99, pp. 154–163. ACM Press, New York (1999)
29. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press, Cambridge (2009)
30. Fitzpatrick, D., Logue, M., Stajich, J., Butler, G.: A fungal phylogeny based on 42 complete genomes derived from supertree and combined gene analysis. BMC Evol. Biol. **6**(1), 99 (2006)
31. Fujimura, K., Conte, M., Kocher, T.: Circular DNA intermediate in the duplication of Nile Tilapia vasa genes. PLoS ONE **6**(12), e29477 (2011)
32. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In: Proc. 27th Ann. ACM Symp. Theory of Comput. (STOC'95), pp. 178–189. ACM Press, New York (1995)

33. Hannenhalli, S., Pevzner, P.: Transforming mice into men (polynomial algorithm for genomic distance problems). In: Proc. 36th Ann. IEEE Symp. Foundations of Comput. Sci. (FOCS'95), pp. 581–592. IEEE Press, Piscataway (1995)
34. Hilker, R., Sickinger, C., Pedersen, C., Stoye, J.: UniMoG—a unifying framework for genomic distance calculation and sorting based on DCJ. Bioinformatics **28**(19), 2509–2511 (2012)
35. Hu, F., Gao, N., Zhang, M., Tang, J.: Maximum likelihood phylogenetic reconstruction using gene order encodings. In: Proc. 2011 IEEE Symp. Comput. Intell. in Bioinf. & Comput. Biol. (CIBCB'11), pp. 117–122. IEEE Press, Piscataway (2011)
36. Huson, D., Nettles, S., Warnow, T.: Disk-covering, a fast converging method for phylogenetic tree reconstruction. J. Comput. Biol. **6**(3), 369–386 (1999)
37. Jansen, R., Palmer, J.: A chloroplast DNA inversion marks an ancient evolutionary split in the sunflower family (Asteraceae). Proc. Natl. Acad. Sci. USA **84**, 5818–5822 (1987)
38. Jean, G., Nikolski, M.: Genome rearrangements: a correct algorithm for optimal capping. Inf. Process. Lett. **104**(1), 14–20 (2007)
39. Larget, B., Simon, D., Kadane, J.: Bayesian phylogenetic inference from animal mitochondrial genome arrangements. J. R. Stat. Soc. B **64**(4), 681–694 (2002)
40. Letunic, I., Bork, P.: Interactive tree of life v2: online annotation and display of phylogenetic trees made easy. Nucleic Acids Res. **39**(S2), W475–W478 (2011)
41. Lin, Y., Moret, B.: Estimating true evolutionary distances under the DCJ model. In: Proc. 16th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB'08). Bioinformatics, vol. 24(13), pp. i114–i122 (2008)
42. Lin, Y., Moret, B.: A new genomic evolutionary model for rearrangements, duplications, and losses that applies across eukaryotes and prokaryotes. J. Comput. Biol. **18**(9), 1055–1064 (2011)
43. Lin, Y., Rajan, V., Swenson, K., Moret, B.: Estimating true evolutionary distances under rearrangements, duplications, and losses. In: Proc. 8th Asia Pacific Bioinf. Conf. (APBC'10). BMC Bioinformatics, vol. 11(Suppl. 1), pp. S54 (2010)
44. Lin, Y., Rajan, V., Moret, B.: Fast and accurate phylogenetic reconstruction from high-resolution whole-genome data and a novel robustness estimator. J. Comput. Biol. **18**(9), 1130–1139 (2011)
45. 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: Proc. 18th Pacific Symp. on Biocomputing (PSB'13), pp. 285–296. World Scientific, Singapore (2013)
46. López, M., Samuelsson, T.: eGOB: eukaryotic Gene Order Browser. Bioinformatics (2011)
47. Ma, J., Ratan, A., Raney, B., Suh, B., Miller, W., Haussler, D.: The infinite sites model of genome evolution. Proc. Natl. Acad. Sci. USA **105**(38), 14254–14261 (2008)
48. Marron, M., Swenson, K., Moret, B.: Genomic distances under deletions and insertions. Theor. Comput. Sci. **325**(3), 347–360 (2004)
49. 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 Comp. Sci., vol. 2547, pp. 163–180. Springer, Berlin (2002)
50. Moret, B., Warnow, T.: Advances in phylogeny reconstruction from gene order and content data. In: Zimmer, E., Roalson, E. (eds.) Molecular Evolution: Producing the Biochemical Data, Part B. Methods in Enzymology, vol. 395, pp. 673–700. Elsevier, Amsterdam (2005)
51. Moret, B., Wang, L.S., Warnow, T., Wyman, S.: New approaches for reconstructing phylogenies from gene-order data. In: Proc. 9th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB'01). Bioinformatics, vol. 17, pp. S165–S173 (2001)
52. Moret, B., Wyman, S., Bader, D., Warnow, T., Yan, M.: A new implementation and detailed study of breakpoint analysis. In: Proc. 6th Pacific Symp. on Biocomputing (PSB'01), pp. 583–594. World Scientific, Singapore (2001)
53. Moret, B., Bader, D., Warnow, T.: High-performance algorithm engineering for computational phylogenetics. J. Supercomput. **22**, 99–111 (2002)

54. Moret, B., Siepel, A., Tang, J., Liu, T.: Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In: Proc. 2nd Workshop Algs. in Bioinf. (WABI'02). Lecture Notes in Comp. Sci., vol. 2452, pp. 521–536. Springer, Berlin (2002)
55. 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)
56. Moret, B., Tang, J., Warnow, T.: Reconstructing phylogenies from gene-content and gene-order data. In: Gascuel, O. (ed.) Mathematics of Evolution and Phylogeny, pp. 321–352. Oxford Univ. Press, London (2005)
57. Nadeau, J., Taylor, B.: Lengths of chromosome segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. USA **81**, 814–818 (1984)
58. Negrisolo, E., Kuhl, H., Forcato, C., Vitulo, N., Reinhardt, R., Patarnello, T., Bargelloni, L.: Different phylogenomic approaches to resolve the evolutionary relationships among model fish species. Mol. Biol. Evol. **27**(12), 2757–2774 (2010)
59. Nozaki, H., et al.: The phylogenetic position of red algae revealed by multiple nuclear genes from mitochondria-containing eukaryotes and an alternative hypothesis on the origin of plastids. J. Mol. Evol. **56**(4), 485–497 (2003)
60. Ouangraoua, A., Boyer, F., McPherson, A., Tannier, E., Chauve, C.: Prediction of contiguous regions in the amniote ancestral genome. In: Proc. 5th Int'l Symp. Bioinformatics Research & Appls (ISBRA'09). Lecture Notes in Comp. Sci., vol. 5542, pp. 173–185. Springer, Berlin (2009)
61. Palmer, J.D., Herbon, L.A.: Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. J. Mol. Evol. **27**, 87–97 (1988)
62. Palmer, J.: Chloroplast and mitochondrial genome evolution in land plants. In: Herrmann, R. (ed.) Cell Organelles, pp. 99–133. Springer, Berlin (1992)
63. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Elec. Colloq. on Comput. Complexity **71** (1998)
64. Peng, Q., Pevzner, P., Tesler, G.: The fragile breakage versus random breakage models of chromosome evolution. PLoS Comput. Biol. **2**(2), e14 (2006)
65. Pevzner, P., Tesler, G.: Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. Proc. Natl. Acad. Sci. USA **100**(13), 7672–7677 (2003)
66. Ponting, C.: The functional repertoires of metazoan genomes. Nat. Rev. Genet. **9**(9), 689–698 (2008)
67. Price, M., Dehal, P., Arkin, A.: Fasttree: computing large minimum-evolution trees with profiles instead of a distance matrix. Mol. Biol. Evol. **26**, 1641–1650 (2009)
68. Rajan, V., Xu, A., Lin, Y., Swenson, K., Moret, B.: Heuristics for the inversion median problem. Proc. 8th Asia Pacific Bioinf. Conf. (APBC'10). BMC Bioinform. **11**(Suppl. 1), S30 (2010)
69. Rajan, V., Lin, Y., Moret, B.: TIBA: a tool for phylogeny inference from rearrangement data with bootstrap analysis. Bioinformatics **28**(24), 3324–3325 (2012)
70. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
71. Sankoff, D.: Minimal mutation trees of sequences. SIAM J. Appl. Math. **28**(1), 35–42 (1975)
72. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Proc. 3rd Ann. Symp. Combin. Pattern Matching (CPM'92). Lecture Notes in Comp. Sci., vol. 644, pp. 121–135. Springer, Berlin (1992)
73. Sankoff, D., Blanchette, M.: The median problem for breakpoints in comparative genomics. In: Proc. 3rd Conf. Computing and Combinatorics (COCOON'97). Lecture Notes in Comp. Sci., vol. 1276, pp. 251–264. Springer, Berlin (1997)
74. Sankoff, D., Blanchette, M.: Multiple genome rearrangement and breakpoint phylogeny. J. Comput. Biol. **5**, 555–570 (1998)
75. Sankoff, D., Blanchette, M.: Phylogenetic invariants for metazoan mitochondrial genome evolution. In: Miyano, S., Takagi, T. (eds.) Genome Informatics, pp. 22–31. Univ. Academy Press, Tokyo (1998)

76. Sankoff, D., Goldstein, M.: Probabilistic models for genome shuffling. Bull. Math. Biol. **51**, 117–124 (1989)
77. Sankoff, D., Nadeau, J.: Conserved synteny as a measure of genomic distance. Discrete Appl. Math. **71**(1–3), 247–257 (1996)
78. Sankoff, D., Nadeau, J. (eds.): Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families. Kluwer Academic, Dordrecht (2000)
79. Sankoff, D., Trinh, P.: Chromosomal breakpoint re-use in genome sequence rearrangement. In: Proc. 9th Int'l Conf. Comput. Mol. Biol. (RECOMB'05). Lecture Notes in Comp. Sci., vol. 3388, pp. 30–35. Springer, Berlin (2005)
80. Sankoff, D., Cedergren, R., Abel, Y.: Genomic divergence through gene rearrangement. In: Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences. Methods in Enzymology, vol. 183, pp. 428–438. Academic Press, San Diego (1990)
81. Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B., Cedergren, R.: Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. Proc. Natl. Acad. Sci. USA **89**(14), 6575–6579 (1992)
82. 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)
83. Siepel, A., Moret, B.: Finding an optimal inversion median: experimental results. In: Proc. 1st Workshop Algs. in Bioinf. (WABI'01). Lecture Notes in Comp. Sci., vol. 2149, pp. 189–203. Springer, Berlin (2001)
84. da Silva, P.H., Braga, M.D.V., Machado, R., Dantas, S.: DCJ-indel distance with distinct operation costs. In: Proc. 12th Workshop Algs. in Bioinf. (WABI'12). Lecture Notes in Comp. Sci., vol. 7534, pp. 378–390. Springer, Berlin (2012)
85. Srivastava, M., et al.: The functional repertoires of metazoan genomes. Nature **454**(7207), 955–960 (2008)
86. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21), 2688–2690 (2006)
87. Sturtevant, A., 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**, 448–450 (1936)
88. Swenson, K., Marron, M., Earnest-DeYoung, J., Moret, B.: Approximating the true evolutionary distance between two genomes. ACM J. Experimental Algorithmics **12** (2008)
89. Swenson, K., Lin, Y., Rajan, V., Moret, B.: Hurdles and sorting by inversions: combinatorial, statistical, and experimental results. J. Comput. Biol. **16**(10), 1339–1351 (2009)
90. Tang, J., Moret, B.: Scaling up accurate phylogenetic reconstruction from gene-order data. In: Proc. 11th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB'03). Bioinformatics, vol. 19, pp. i305–i312. Oxford Univ. Press, London (2003)
91. Tang, J., Moret, B.: Linear programming for phylogenetic reconstruction based on gene rearrangements. In: Proc. 16th Ann. Symp. Combin. Pattern Matching (CPM'05). Lecture Notes in Comp. Sci., vol. 3537, pp. 406–416. Springer, Berlin (2005)
92. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal genome median and halving problems. In: Proc. 8th Workshop Algs. in Bioinf. (WABI'08). Lecture Notes in Comp. Sci., vol. 5251, pp. 1–13. Springer, Berlin (2008)
93. Tesler, G.: Efficient algorithms for multichromosomal genome rearrangements. J. Comput. Syst. Sci. **63**(5), 587–609 (2002)
94. Wang, L.S.: Exact-IEBP: a new technique for estimating evolutionary distances between whole genomes. In: Proc. 1st Workshop Algs. in Bioinf. (WABI'01). Lecture Notes in Comp. Sci., vol. 2149, pp. 175–188. Springer, Berlin (2001)
95. Wang, L.S., Warnow, T.: Estimating true evolutionary distances between genomes. In: Proc. 33rd Ann. ACM Symp. Theory of Comput. (STOC'01), pp. 637–646. ACM Press, New York (2001)
96. Wang, L.S., Warnow, T.: Distance-based genome rearrangement phylogeny. In: Gascuel, O. (ed.) Mathematics of Evolution and Phylogeny, pp. 353–383. Oxford Univ. Press, London (2005)

97. Wang, L.S., Jansen, R., Moret, B., Raubeson, L., Warnow, T.: Fast phylogenetic methods for genome rearrangement evolution: an empirical study. In: Proc. 7th Pacific Symp. on Biocomputing (PSB'02), pp. 524–535. World Scientific, Singapore (2002)
98. Wang, H., Xu, Z., Gao, L., Hao, B.: A fungal phylogeny based on 82 complete genomes using the composition vector method. BMC Evol. Biol. **9**(1), 195 (2009)
99. Wang, L.S., Leebens-Mack, J., Wall, P., Beckmann, K., dePamphilis, C., Warnow, T.: The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**, 1108–1119 (2011)
100. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. J. Theor. Biol. **99**(1), 1–7 (1982)
101. Xu, A., Sankoff, D.: Decompositions of multiple breakpoint graphs and rapid exact solutions to the median problem. In: Proc. 8th Workshop Algs. in Bioinf. (WABI'08). Lecture Notes in Comp. Sci., vol. 5251, pp. 25–37. Springer, Berlin (2008)
102. Xu, A.: A fast and exact algorithm for the median of three problem—a graph decomposition approach. J. Comput. Biol. **16**(10), 1369–1381 (2009)
103. Xu, A., Moret, B.: GASTS: parsimony scoring under rearrangements. In: Proc. 11th Workshop Algs. in Bioinf. (WABI'11). Lecture Notes in Comp. Sci., vol. 6833, pp. 351–363. Springer, Berlin (2011)
104. Yancopoulos, S., Friedberg, R.: Sorting genomes with insertions, deletions and duplications by DCJ. In: Proc. 6th RECOMB Workshop Comp. Genomics (RECOMB-CG'08). Lecture Notes in Comp. Sci., vol. 5267, pp. 170–183. Springer, Berlin (2008)
105. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)
106. Zhang, M., Arndt, W., Tang, J.: A branch-and-bound method for the multichromosomal reversal median problem. In: Proc. 8th Workshop Algs. in Bioinf. (WABI'08), pp. 1–13 (2008)
107. Zheng, C., Zhu, Q., Adam, Z., Sankoff, D.: Guided genome halving: hardness, heuristics and the history of the hemiascomycetes. In: Proc. 16th Int'l Conf. on Intelligent Systems for Mol. Biol. (ISMB'08). Bioinformatics, vol. 24, pp. i96–i104 (2008)

Chapter 8
Status of Research on Insertion and Deletion Variations in the Human Population

Liqing Zhang, Mingming Liu, and Layne T. Watson

Abstract Insertion and deletion (indel) variants comprise a major proportion of human genetic variation. However, little is known about their effect on humans. The void of understanding is largely due to the lack of both biological data and computational resources. Thanks to the progress made by many large-scale genomic projects, a substantial amount of data is now available, enabling the prediction of functional elements in the genome. In this work, we review the impact of indel variants on human biology, evolution, and health, and examine the currently available computational resources for predicting the functional effects of indels and their limitations. We then present a newly developed program for indel effect prediction using a hidden Markov model-based framework and discuss future work for better understanding the effects of indel variants on human biology and health.

8.1 Indel Effects on Human Biology, Health, and Evolution

Indel is the Second Most Common Type of Genetic Variation in Humans The rapid development of sequencing technologies has made possible cataloging the entire set of genetic variants harbored in human populations. The recent pilot study conducted by the 1000 Genome Project Consortium has revealed that there are about 15 million single nucleotide polymorphisms (SNPs), one million short insertions and deletions (indels), and 20,000 structural variants (SVs) harbored by the populations they studied [1]. Thus, indel ranks as the second most common type of genetic variation in humans.

Indel Variants Have Profound Functional Impact on Human-Specific Evolution and Adaptation Comparison of the human genome and several other closely related species' genomes has shown that approximately 0.8 million human-specific

L. Zhang (✉) · M. Liu · L.T. Watson
Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
e-mail: lqzhang@vt.edu

L.T. Watson
Department of Mathmatics, Virginia Tech, Blacksburg, VA, USA

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_8,

indels affect more than 7000 genes, and these genes may have contributed to human traits via changes at the RNA and protein levels [2]. In addition, indels have been found to contribute to about 5 % of the human–chimpanzee divergence, much higher than the 1.5 % nucleotide divergence, suggesting that indels might have played an even bigger role than nucleotide divergence in human-chimpanzee differentiation [20]. Some human-specific indels show evidence of positive selection and might have played important roles in human adaptation both at the species level and the subpopulation level [3]. The importance of indels to the evolution of genomes is further supported by a study that has found an increased rate of mutations (higher levels of SNPs) near the regions with indels [7] at both the species and population levels.

Indels May Hold the Key to Understanding Human Diseases [17] Depending on the locations of the indels, indels can potentially lead to frame shift and thus influence proteins by changing protein sequences, gene expression patterns (by affecting promoter regions, introns, or UTRs), and exon splice patterns. A well-known case of indel effects is cystic fibrosis, a genetic disease frequently caused by a 3-bp deletion within the coding region of CFTR [5]. Although it is in-frame, the deletion leads to abnormal protein folding and protein degradation. Indels in noncoding regions can also cause human diseases. For example, when indels occur within the promoter region of the FMR1 gene, they can change the promoter methylation pattern and thus the gene expression pattern of FMR1, resulting in fragile X syndrome [5]. Mill et al. [17] have shown that about 42 % of the nearly two million indels they identified are mapped to human genes and more than 2000 indels affect coding exons and likely disrupt protein function and cause phenotypic changes in humans. Their analysis of the experimental data in mice shows that 83 % of the coding indels yield abnormal phenotypes. Moreover, the indels tend to have strong linkage disequilibrium (LD) with the SNPs identified in genome wide association studies (GWAS). Diseases with an indel genetic basis might have been mistakenly determined as SNP related, only because of strong LD. In these cases, accurate indel effect prediction is the only way to improve our understanding of these diseases.

8.2 Current Research on Indel Variants

Despite all the evidence suggesting the importance of indels, their research has lagged behind studies of other variant types such as SNPs and SVs. From a biological point of view, it is time consuming and difficult to experimentally characterize the impact of indels on genes or protein function. Computationally, there are two main problems: the lack of specialized database resources for indel curation and function annotation, and the lack of computational methods/programs to predict the effect of indel variants.

The Lack of Specialized Database Resources for Indel Curation and Function Annotation Currently, indel polymorphisms are loosely stored in dbSNP, where simple annotation based mostly on location of indel variants with respect to

genes is provided. The current dbSNP (build 135) contains 6,312,022 nonredundant or reference indels, which are clustered from 7,806,204 indels submitted by various researchers, with a major proportion generated by a few large-scale studies [1, 11, 12, 16, 19]. Indels are roughly annotated to categories including introns, intergenic regions, UTRs, and frameshift indels. Evidently, annotation of indel variants by dbSNP is so coarse-grained and overly simplistic that it does not help researchers prioritize and choose from the sea of indels the strongest candidate indels for traits or diseases of interest. Other large data servers, such as the UCSC Genome Browser and Ensembl, import indel annotation directly from dbSNP. Hence, there is no dedicated computational resource and database for fine-grained annotation of indel effect. It must be noted that indels cannot be simply taken as repeats or mini- or microsatellites as the majority (70 %) of them are nonrepetitive [16].

The need for a database dedicated to indels is further emphasized by several recent studies that demonstrate the far-from-completeness of our current catalog of indel variants in humans. A 2011 study shows that more than 63 % of the nearly two million indels identified in the 79 diverse human genomes are novel [17], compared to the ones in dbSNP. Most recently (August 2012), sequencing and analysis of an Indian female's genome reveals that about 84 % of this person's indels are unique, i.e., not documented in any of the sequenced genome databases, in contrast to less than 3 % of the SNPs being unique [9]. Thus compared to SNPs, the research on cataloging indel variants is still in its infancy and intense effort is needed in order to have a complete inventory. A specialized database devoted to indels would greatly facilitate this task and thus take an important step towards understanding their effect on human traits and diseases.

The Lack of Computational Methods/Programs to Predict the Effect of Indel Variants A survey of the tools for predicting SNP variant effect shows that there are a few dozen computer programs and web servers devoted to such a purpose [13]. In contrast, the computational resources devoted to indel effect prediction is very limited and nearly nonexistent. At the time of this writing, only three studies were found that propose computational methods for predicting the functional effect of indel variants. The first recent study proposed an evolutionary conservation-based approach to score and predict the effect of indel variants for both coding and noncoding regions [21]. Although the results are encouraging, there is no readily available source program. The provided online web server has several major limitations. First, it has limited prediction power, restricted to only one indel on one sequence per analysis. Ideally, the user should have the freedom to upload an input file for batch analyses. Second, although the paper has predictions for both coding and noncoding indels, the web server does not have noncoding indel prediction. Third, the prediction score indicates the deleteriousness of an indel, but does not have any information on what functions are likely affected. Finally, the online server has bugs, returning randomly truncated amino acid sequences in some tests. Another recent study proposed SIFT-Indel that uses a simple decision tree approach to classify the effect of indel variants [10]. For indel effect prediction, four features are extracted for each indel: fraction of affected conserved DNA bases, indel location relative to

the transcript, fraction of affected conserved amino acids, and minimum distance of the indel to the exon boundary of all the affected transcripts. Though easy to interpret due to the nature of a decision tree, the predictive power of SIFT-Indel is rather limited due to two major drawbacks. First, the method only applies to frameshift indels, which account for a tiny proportion (~ 0.05 %) of indel variants [18]. Second, it can only make coarse-grained qualitative predictions, that is, an indel can be either "gene-damaging" or "neutral". However, a computational method or program that can produce quantitative ranking of variant effect is much more useful for indel filtering and prioritization than qualitative assessment [6].

The third latest study introduced an alignment-based score to predict the effect of genetic variants, including single SNPs, indels, and multiple mutations [4]. The corresponding program PROVEAN also uses an evolutionary conservation-based method to evaluate the deleteriousness of variants. Though promising, the program is only applicable to in-frame indels. However, frameshift indels are expected to be more deleterious and thus are also an important type of indels that require function effect prediction. To address the limitations of the current programs, the authors recently developed HMMvar [15], a program using a hidden Markov model (HMM)-based scoring method to predict the effect of indels. The following section gives an overview of the program and some results on its application.

8.3 The Hidden Markov Model-Based Scoring Method for Predicting Indel Effects

The HMM-based method to score the effect of indel variants incorporates hypothesis testing naturally and formally into a probabilistic framework. A profile HMM can be used to describe the probabilities of multiple sequences generated from the HMM model, thus representing a family of proteins. Briefly, a profile HMM, named for the characteristic output "profile" of a particular hidden Markov model, is a finite state machine consisting of a series of nodes, each of which corresponds roughly to a position (column) in the alignment from which it was built. Most of the previous prediction methods are based on the principle that important amino acids will be conserved in the protein family, and so mutations occurring at well-conserved positions tend to be deleterious to the functions of the protein. This principle can be reflected exactly by the profile HMMs. Basically, a HMM profile is a probabilistic description of the consensus of a multiple sequence alignment. Thus it is reasonable to use a profile HMM to gauge how far mutations take the original sequence away from the set of sequences represented by the HMM. The further away from the representation, the more likely the mutation is deleterious.

Figure 8.1 shows the flowchart of profile HMM-based prediction, or the workflow of the HMMvar program. The pipeline consists of five steps: (1) find "seed" proteins that are associated with indels; (2) for each seed protein, find homologous sequences from a database; (3) do multiple sequence alignment (MSA) for each set of homologous sequences; (4) build a profile HMM based on each MSA; (5) predict

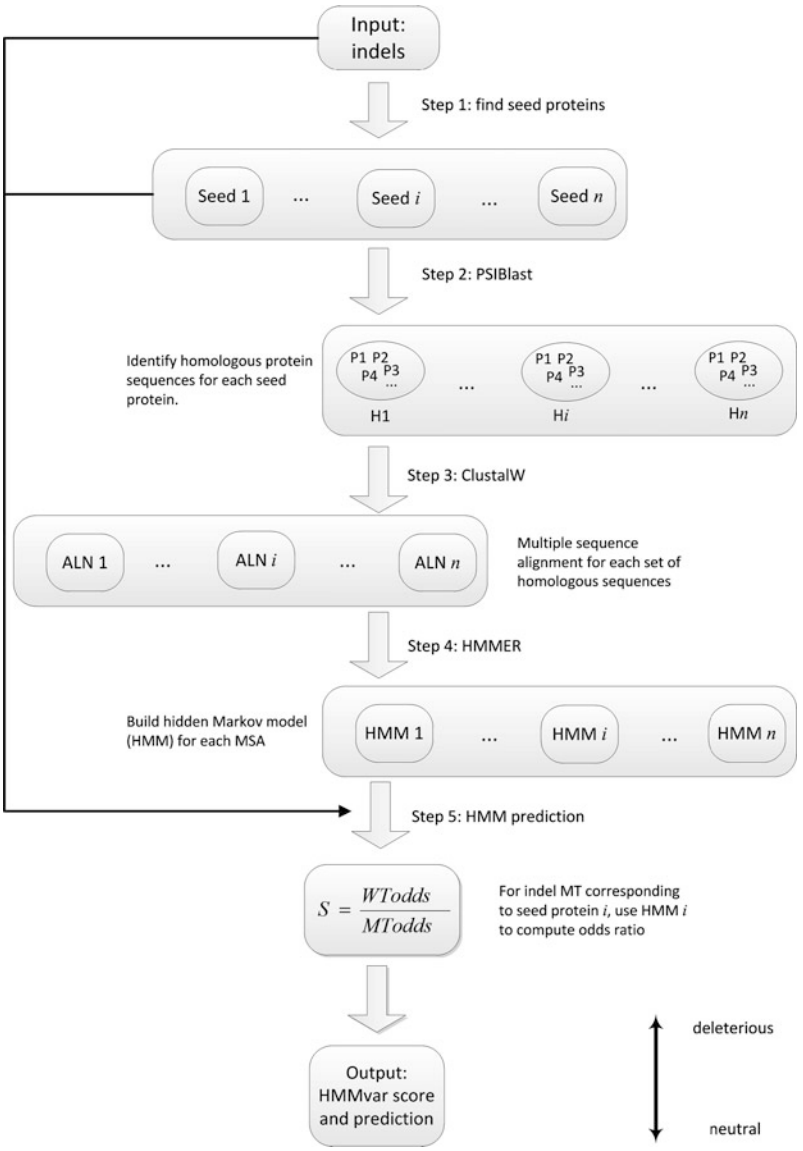


Fig. 8.1 An overview of profile HMM-based variant prediction

the functional effects of indels using the profile HMMs; precisely, for each mutated protein with the indel (MT, mutant type) and corresponding seed protein i (WT, wild type), use the ith HMM to compute the odds ratio (odds that the HMM could have generated the WT sequence)/(odds that the HMM could have generated the MT sequence)—this odds ratio is the HMM indel score.

Table 8.1 Numbers of different indel types with or without the LSDB records

Indel types	LSDB	NonLSDB	Total
Nonsense	112	15	127
Missense	0	56	56
Frameshift	2519	1387	3906
Total	2631	1458	4089

The bit scores calculated from the HMMs are used to quantitatively evaluate the effect of indels. Specifically, the bit score from HMMER3 [8] measures the similarity of a query sequence with the set of homologous sequences used to define the profile HMM. The HMMER3 bit score is a base 2 logarithm of a ratio of probabilities (homology hypothesis over the null hypothesis),

$$B = \log_2 \frac{P(o_1 o_2 \ldots o_n|\text{HMM})}{P(o_1 o_2 \ldots o_n|\text{NULL})}, \tag{8.1}$$

where $o_1 o_2 \ldots o_n$ is the observed protein sequence and "HMM" is the trained profile HMM. "NULL" is the "null model", which is a one-state HMM configured to generate "random" sequences of the same length as the target sequence, with each residue drawn from a background frequency distribution (in HMMER3, for proteins, the frequencies of the 20 amino acids are set to the amino acid composition of SWISS-PROT 34). Since this logarithm score has no direct statistical interpretation, the constituent probabilities are extracted and used to define the odds ratio of the HMM probabilities,

$$S = \frac{P_w/(1 - P_w)}{P_m/(1 - P_m)}, \tag{8.2}$$

where $P_w(P_m)$ is the probability that the wild type (mutated type) protein sequence could have been generated by the profile HMM trained on a seed protein homologous sequence set (i.e., the numerator in B). The greater S is, the more likely that the mutation is deleterious. Usually S is expected to be greater than 1 as most mutations tend to be deleterious. When S is less than 1, it suggests that the mutant sequence better fits the HMM profile and the mutation may lead to amino acids that are more compatible than the wild type proteins. Experiments were done to set a threshold for the odds ratio, S_t, below which, the indel is considered as neutral, otherwise deleterious.

To demonstrate the effectiveness of HMMvar in predicting the effect of indels, indel variant data was obtained from dbSNP, and the indel effects scored using HMMvar. There are three types of indel variants in dbSNP, nonsense, missense, and frameshift indels. Missense indels refer to the indels that add or remove amino acids to or from the original protein sequence. Nonsense indels refer to indels that cause a stop codon where the indel occurs. Frameshift indels refer to indels that are not a multiple of three base pairs, thus change the reading frame of the original protein. Note, these categories of indels are mutually exclusive, that is, an indel can be either

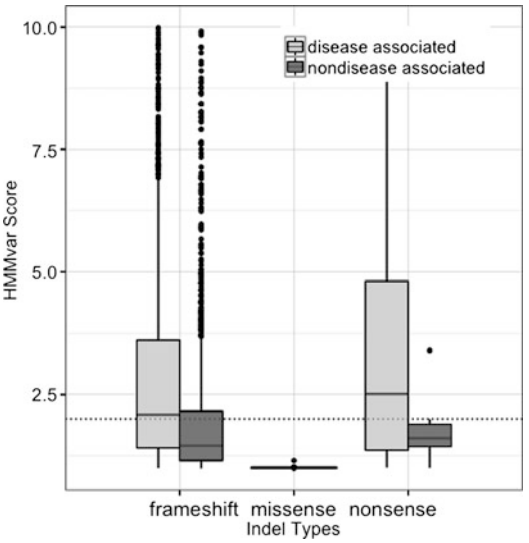


Fig. 8.2 Distributions of HMM scores for different types of indel variants. The *dotted line* shows the HMM score cutoff ($S_t = 2.0$) for determining whether an indel is deleterious or not

frameshift or in-frame, and if in-frame, it can be either missense or nonsense, but not both. The data contains altogether 4089 indels, among which 127, 56, 3906 are nonsense, missense, and frameshift indels, respectively (Table 8.1). These indels are further classified into two groups, indels that have locus-specific mutation database (LSDB) [14] annotation, which are expected to be disease associated and have more harmful effects, and indels that do not have LSDB annotation, which are expected to be nondisease (or unknown) associated and have less harmful effects (Table 8.1). The indels were fed into HMMvar and the odds ratio of the HMM probabilities were computed for each class of indels. Figure 8.2 shows the distributions of the HMM scores for three types of indel variants, frameshift indels, nonsense indels, and missense indels. The most remarkable feature is that the score of missense indels is much lower than the scores of the other two types, consistent with the notion that missense mutations tend to be less deleterious than frameshift indels and nonsense mutations. In each type of indel, the median of the nondisease associated group is lower than the median of the disease associated group, demonstrating that the HMM score is effective in evaluating the deleteriousness of indel mutations. Further comparison shows that the HMM odds ratio score has comparable performance to PROVEAN, with the added advantage of having smaller variance in the predicted scores, a desirable property for a scoring metric.

8.4 Future Directions

As an increasing amount of sequence data shows the prevalence and dominance of indels as the second most common type of mutation in human populations, much effort is required in order to fully understand indel effect on human biology and health.

Future research needs to focus on designing new, or improving existing, algorithms for predicting indel effects, by a combination of methods such as evolutionary-based approaches and sophisticated machine learning algorithms. Integration with diverse data and analysis results promises to provide a complete picture of indel effects on various aspects such as protein function, gene splicing, and gene expression.

References

1. 1000 Genomes Project Consortium: A map of human genome variation from population-scale sequencing. Nature **467**(7319), 1061–1073 (2010)
2. Chen, F.-C., Chen, C.-J., Li, W.-H., Chuang, T.-J.: Human-specific insertions and deletions inferred from mammalian genome sequences. Genome Res. **17**(1), 16–22 (2007)
3. Chen, C.-H., Chuang, T.-J., Liao, B.-Y., Chen, F.-C.: Scanning for the signatures of positive selection for human-specific insertions and deletions. Genome Biol. Evol. **1**, 415–419 (2009)
4. Choi, Y., Sims, G.E., Murphy, S., Miller, J.R., Chan, A.P.: Predicting the functional effect of amino acid substitutions and indels. PLoS ONE **7**(10), e46688 (2012)
5. Collins, F.S., Drumm, M.L., Cole, J.L., Lockwood, W.K., Vande Woude, G.F., Iannuzzi, M.C.: Construction of a general human chromosome jumping library, with application to cystic fibrosis. Science **235**(4792), 1046–1049 (1987)
6. Cooper, G.M., Shendure, J.: Needles in stacks of needles: finding disease-causal variants in a wealth of genomic data. Nat. Rev. Genet. **12**(9), 628–640 (2011)
7. De, S., Madan Babu, M.: A time-invariant principle of genome evolution. Proc. Natl. Acad. Sci. USA **107**(29), 13004–13009 (2010)
8. Finn, R.D., Clements, J., Eddy, S.R.: HMMER web server: interactive sequence similarity searching. Nucleic Acids Res. **39**(Web Server issue), W29–W37 (2011)
9. Gupta, R., Ratan, A., Rajesh, C., Chen, R., Lim Kim, H., Burhans, R., Miller, W., Santhosh, S., Davuluri, R.V., Butte, A.J., Schuster, S.C., Seshagiri, S., Thomas, G.: Sequencing and analysis of a South Asian–Indian personal genome. BMC Genomics **13**, 440 (2012)
10. Hu, J., Ng, P.C.: Predicting the effects of frameshifting indels. Genome Biol. **13**(2), R9 (2012)
11. International HapMap Consortium: The international HapMap project. Nature **426**(6968), 789–796 (2003)
12. International HapMap Consortium: A haplotype map of the human genome. Nature **437**(7063), 1299–1320 (2005)
13. Karchin, R.: Next generation tools for the annotation of human SNPs. Brief. Bioinform. **10**(1), 35–52 (2009)
14. Kato, S., Han, S.-Y., Liu, W., Otsuka, K., Shibata, H., Kanamaru, R., Ishioka, C.: Understanding the function-structure and function-mutation relationships of p53 tumor suppressor protein by high-resolution missense mutation analysis. Proc. Natl. Acad. Sci. USA **100**(14), 8424–8429 (2003)
15. Liu, M., Watson, Layne.T., Zhang, L.: HMMvar: Predicting the functional effects of indels and SNPs based on HMM profiles. BMC Bioinform. (under review)
16. Mills, R.E., Luttig, C.T., Larkins, C.E., Beauchamp, A., Tsui, C., Stephen Pittard, W., Devine, S.E.: An initial map of insertion and deletion (INDEL) variation in the human genome. Genome Res. **16**(9), 1182–1190 (2006)
17. Mills, R.E., Stephen Pittard, W., Mullaney, J.M., Farooq, U., Creasy, T.H., Mahurkar, A.A., Kemeza, D.M., Strassler, D.S., Ponting, C.P., Webber, C., Devine, S.E.: Natural genetic variation caused by small insertions and deletions in the human genome. Genome Res. **21**(6), 830–839 (2011)
18. Mullaney, J.M., Mills, R.E., Pittard, W.S., Devine, S.E.: Small insertions and deletions (INDELs) in human genomes. Hum. Mol. Genet. **19**, R131–R136 (2010)

19. Siva, N.: 1000 Genomes Project. Nat. Biotechnol. **26**(3), 256 (2008)
20. Wetterbom, A., Sevov, M., Cavelier, L., Bergstrom, T.F.: Comparative genomic analysis of human and chimpanzee indicates a key role for indels in primate evolution. J. Mol. Evol. **63**(5), 682–690 (2006)
21. Zia, A., Moses, A.M.: Ranking insertion, deletion and nonsense mutations based on their effect on genetic information. BMC Bioinform. **12**, 299 (2011)

Chapter 9
A Retrospective on Genomic Preprocessing for Comparative Genomics

Binhai Zhu

Abstract In this paper, we present a survey of research on genomic preprocessing for comparative genomics, i.e., handling genomes with gene repetitions, missing or redundant genes, initiated by David Sankoff in 1999. The development of this research ends with several interesting results within and beyond computational biology and bioinformatics, with possible new contributions in the future. We will describe the history of development of this research and review the current status of the corresponding problems. For the problem of handling missing genes (scaffold filling), we also present some technical details which are not given in the previous papers. Some open problems will be listed at the end for further research.

9.1 Introduction

In computational biology, we constantly need to process various biological data to extract meaningful biological relation, like building a phylogenetic tree. Such a process sometimes involves computing the genomic distance between two genomes, which was first investigated as early as in 1926 [61, 62]. The problem was more formally studied in 1990s and is in general polynomially solvable for signed genomes, e.g., under the signed translocation distance [7, 37, 49, 56], under the signed reversal distance [3, 38, 48, 63, 64], and under the DCJ distance [69]. For unsigned genomes, the problems are typically NP-hard, e.g., sorting by reversals [17], sorting by translocations [71], sorting by DCJ operations [19], and sorting by transpositions [16]. But these problems on sort unsigned genomes do admit small-factor ($\leq$1.5) polynomial-time approximations, e.g., sorting by reversals [9, 28], sorting by translocations [31, 46], sorting by DCJ operations [19, 20, 42], and sorting by transpositions [33].

The above results are all under the assumption that each genome is given in a form where there is no loss and duplication of genes and a genome is represented as a permutation of genes. For many genomes, due to the fast evolution/self-reproduction process, duplicated (paralogous) genes are common. So it is useful to

B. Zhu (✉)
Department of Computer Science, Montana State University, Bozeman, MT 59717-3880, USA
e-mail: bhz@cs.montana.edu

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_9,

select the ancestral ortholog of a gene family on an evolutionary basis. In 1999, David Sankoff first formulated this problem as an algorithmic problem, now known as the *Exemplar Breakpoint/Genomic Distance* problem [59]. In Sect. 9.2, we will survey the development of the follow-up research since 1999, mostly with negative complexity results. Some of these methods and results have already been applied in other (biological and non-biological) problems [6, 58, 67].

In some eukaryotic genomes, under many situations, like sequencing error or errors due to an inappropriate design of the biological experiments, we might have noise and redundant genes. Before eliminating these redundancies, using the given genomes for many biological studies might introduce further errors. While this problem was known to the biologists long time ago, in 2007 David Sankoff again first formulated this as an algorithmic problem, now known as the *Maximal Strip Recovery* and the *Complementary Maximal Strip Recovery* problems [27, 70]. This again led to a series of research on fixed-parameter tractable and approximation algorithms, performed by several groups in US, Canada, Europe and China. In Sect. 9.3, we will survey the most recent development of these researches.

Genome sequencing has been a hot research area for the last 20 years. Behind the huge success a commonly ignored fact is that most genomes sequenced are not really 'sequences'; in fact, most of them are made of scaffolds, i.e., composed of incomplete gene markers. David Sankoff and his group initiated this problem of scaffold filling in 2010 [54]. My group and a group led by Prof. Daming Zhu at Shandong University (China) have been following up this research. While initially the work was done on filling scaffolds with no gene repetitions, which is a problem polynomially solvable, recently a lot of effort has been put on filling scaffolds with gene repetitions (which is in general NP-hard). In Sect. 9.4, we will survey the current status of this research.

In the area of bioinformatics and computational biology, for a lot of NP-complete problems one would typically apply three methods to handle them. One is to find an approximation solution, with the requirement being that the approximation factor is small (better close to one). The other is to look for an exact solution (FPT algorithm) when some parameter (say, the solution size) of the problem is small. The vast majority of practical solutions for bioinformatics and computational biology are heuristic ones, which are possibly based on some formal methods like integer linear programming, branch-and-bound, etc.

In this survey, we focus on the approximability and fixed-parameter tractability results for the above three general problems related to computing genomic distance with some preprocessing. In these problems, we are given some genomes or genetic maps and we try to optimize some solution values by deleting some genes or gene markers. So these problem fit naturally for approximation and/or FPT solutions. Unfortunately, as we will review a bit later, some of these problems are very hard in both aspects. In other words, it might be impossible to design good approximation and/or FPT algorithms for them, unless $P = NP$, $NP = ZPP$ or $FPT = W[1]$. On the other hand, many problems are still open along these lines.

The paper is organized as follows. In Sect. 9.2, we first review the approximability and fixed-parameter tractability for the Exemplar Breakpoint Distance (EBD)

problem. We then review the approximability and fixed-parameter tractability for the Exemplar Non-breaking Similarity (ENbS) problem (which is the dual of EBD). In Sect. 9.3, we review the approximability and fixed-parameter tractability for the Maximal Strip Recovery (MSR) problem and its complement, the Complementary Maximal Strip Recover (CMSR) problem. In Sect. 9.4, we review the approximation results for the Scaffold Filling Problem, focusing on the One-sided Scaffold Filling Problem with Gene Repetitions. In Sect. 9.5, we list a set of open problems to conclude this paper.

9.2 The Exemplar Breakpoint Distance and Related Problems

As we have covered in the introduction, in the genome comparison and rearrangement area, a standard problem is to compute the number (i.e., genetic distance) and the actual sequence of genetic operations which converts a source genome to a target genome. This problem is important in evolutionary molecular biology as it gives some useful information on genome evolution. Typical genetic distances include edit [53], signed reversal [4, 38, 52, 57] and breakpoint [66], etc. In fact, the idea of signed reversal and, implicitly, breakpoint, was initiated as early as in 1926 by Sturtevant [61]. In the past years, conserved interval distance was also proposed to measure the similarity of multiple sequences of genes [8]. Interested readers are referred to [35] for a summary of the research performed in this area.

In genome rearrangement research, it is usually assumed that each gene appears in a genome exactly once. Under this assumption, the genome rearrangement problem is in essence the problem of comparing and sorting signed permutations [35, 38]. However, this assumption is very restrictive and is only justified in several small virus genomes. For example, this assumption does not hold on eukaryotic genomes where paralogous genes exist [55, 59]. So we have to handle this gene duplication problem.

David Sankoff first considered the problem of computing the breakpoint distance with duplicated genes. In [59], Sankoff proposed a way to select, from the duplicated copies of genes, the common ancestor gene such that the breakpoint distance between the reduced genomes (*exemplar genomes*) is minimized. The distance is called the *exemplar breakpoint distance* henceforth. A general branch-and-bound algorithm was also implemented in [59]. In [55], Nguyen, Tay and Zhang proposed to use a divide-and-conquer method to compute the exemplar breakpoint distance empirically.

For the theoretical part of research, it was shown that both of the problems of computing the signed reversal and breakpoint distances between exemplar genomes are NP-complete [14]. A few years ago, Blin and Rizzi further proved that computing the conserved interval distance between exemplar genomes is NP-complete [11]; moreover, it is NP-complete to compute the minimum conserved interval matching (i.e., without deleting the duplicated copies of genes). Starting in 2005, we showed much stronger inapproximability results for the exemplar breakpoint and conserved

interval distance problems (even under a weaker model of approximation) [21, 24]. (In fact, a series of workshops were organized at University of Texas—Pan American between 2005 and 2008, focusing on this topic.) While various exemplar genomic distances have been researched before, in this survey we will focus on the exemplar breakpoint distance. In fact, all the inapproximability result for exemplar breakpoint distance holds for any other genomic distance $d(-,-)$ satisfying $d(G,H)=0$ implies $G=H$ or $G=-H$.

9.2.1 Problem Definitions

In the genome comparison and rearrangement problem, we are given a set of genomes, each of which is a signed sequence of genes where the order of the genes corresponds to the position of them on the linear chromosome and the signs correspond to which of the two DNA strands the genes are located. Here we interpret a genome as a set of such sequences (chromosomes), though we focus mostly on *singleton* genomes, i.e., a single sequence, in this paper. When the input genomes contain gene repetitions, Sankoff proposed a method to select an *exemplar genome*, by deleting redundant copies of a gene, such that in an exemplar genome any gene appears exactly once; moreover, the resulting exemplar genomes should have a property that a given genetic distance between them is minimized [59].

The following definitions are very much following those in [11]. Given n *gene families* (alphabet) $\mathcal{F}$, a genome $\mathcal{G}$ is a sequence of elements of $\mathcal{F}$ such that each element is with a sign ($+$ or $-$). In general, we allow the repetition of a gene family in any genome. Each occurrence of a gene family is called a *gene*, though we will not try to distinguish a gene and a gene family if the context is clear. Given a genome with no repetition of any gene $G=g_1g_2\ldots g_m$, we say that gene g_i *immediately precedes* g_j if $j=i+1$. Given genomes G, H (with no gene repetition), if gene a immediately precedes b in G and neither a immediately precedes b nor $-b$ immediately precedes $-a$ in H, then they constitute a *breakpoint* in G. The *breakpoint distance* is the number of breakpoints in G (symmetrically, it is the number of breakpoints in H), denoted as $\mathrm{bd}(G,H)$.

The number of a gene g appearing in a genome $\mathcal{G}$ is called the cardinality of g in $\mathcal{G}$, written as $card(g,\mathcal{G})$. A gene in $\mathcal{G}$ is called *trivial* if g has cardinality exactly 1; otherwise, it is called *non-trivial*. A genome $\mathcal{G}$ is called r*-repetitive*, if all the genes from the same gene family appear at most r times in $\mathcal{G}$. For example, $\mathcal{G}=c-adc-bdeb$ is 2-repetitive.

Given a genome $\mathcal{G}$ over $\mathcal{F}$, an *exemplar genome* of $\mathcal{G}$ is a genome G' obtained from $\mathcal{G}$ by deleting duplicating genes such that each gene family in $\mathcal{G}$ appears exactly once in G'. For example, let $\mathcal{G}=-bcaadag-e$, there are two exemplar genomes: $-bcadg-e$ and $-bcdag-e$.

The Exemplar Breakpoint Distance (EBD) problem is defined as follows:

Instance: Genomes $\mathcal{G}$ and $\mathcal{H}$, each is of length $O(m)$ and each covers n gene families (i.e., at least one gene from each of the n gene families appears in both $\mathcal{G}$ and $\mathcal{H}$); integer K.

Question: Are there two respective exemplar genomes of $\mathcal{G}$ and $\mathcal{H}$, G and H, such that $\text{bd}(G, H) \leq K$?

9.2.2 Algorithmic Foundations

In the next subsection, we present some hardness results on the approximability and fixed-parameter tractability for EBD, namely, the hardness to compute or approximate the minimum value K in the above formulation. Here we give some standard definitions regarding approximation and FPT algorithms. Given a minimization (maximization) problem Π, let the optimal solution value of Π be OPT. We say that an approximation algorithm $\mathcal{A}$ provides a *performance guarantee* of α for Π if for every instance I of Π, the solution value returned by $\mathcal{A}$ is at most $\alpha \times \text{OPT}$ (at least OPT/α). Usually we say that $\mathcal{A}$ is a factor-α approximation for Π. For the obvious reason, we are only interested in polynomial-time approximation algorithms. Readers are referred to [29, 34] for more details regarding the definitions related to approximation algorithms and NP-completeness.

As a well-known subject as well, an FPT algorithm for a decision problem with parameter k is an algorithm which solves the problem in $O(f(k)n^c)$ time, where f is any function only on k and c is some fixed constant not related to k. More details on FPT algorithms can be found in [32].

9.2.3 Hardness Results

In [21], we presented the first set of inapproximability results for the Exemplar Breakpoint Distance problem, given two genomes each containing only one sequence of genes drawn from n gene families. We showed that even if a gene appears at most three times, deciding whether the optimal exemplar breakpoint distance is zero, i.e, whether $G = H$, is NP-complete. It was left as an open problem whether the result holds when each gene appears at most twice in each of the input genomes [2, 21]. Recently, this open question was finally answered, i.e., it remains NP-complete even when each gene appears at most two times [13, 47]. Combining these results, we have the following inapproximability result.

Theorem 1 *If both $\mathcal{G}$ and $\mathcal{H}$ are 2-repetitive genomes, then the Exemplar Breakpoint Distance problem does not admit any polynomial-time approximation (regardless of its approximation factor), unless $P = NP$.*

Proof If we view the Exemplar Breakpoint Distance problem as a minimization problem, then the result in [13], with an example presented at the end of this subsection, implies that deciding whether $\text{OPT} = 0$ is NP-complete (even if the input

genomes are 2-repetitive). Let $\mathcal{A}$ be any approximation algorithm for EBD with factor α. By definition, $\mathcal{A}$ returns an approximation solution value APP, with

$$\text{APP} \leq \alpha \times \text{OPT}.$$

When $\text{OPT} = 0$, clearly APP must also satisfy $\text{APP} = 0$. In other words, $\mathcal{A}$ would be able to solve the instance in [13] in polynomial time. This, however, contradicts with the corresponding NP-completeness result (unless $\text{P} = \text{NP}$). □

Regarding the fixed-parameter intractability for EBD, we have the following theorem.

Theorem 2 *If both $\mathcal{G}$ and $\mathcal{H}$ are 2-repetitive genomes, then the Exemplar Breakpoint Distance problem does not admit any FPT algorithm, unless $P = NP$.*

Proof Again, if we view the Exemplar Breakpoint Distance problem as a minimization problem, then the result in [13, 47] implies that deciding whether $\text{OPT} = 0$ is NP-complete (even if the input genomes are 2-repetitive). Let $\mathcal{B}$ be any FPT algorithm for EBD which runs in $O(f(k)n^c)$ time. When $\text{OPT} = k = 0$, $\mathcal{B}$ solves EBD in $O(f(0)n^c) = O(n^c)$ time. In other words, $\mathcal{B}$ would be able to solve the instance in [13] in polynomial time. This, again, contradicts with the corresponding NP-completeness result, unless $\text{P} = \text{NP}$. □

On the other hand, it is necessary to point out that the reduction in [21, 24] is much simpler than in [13, 47]. As a matter of fact, it has been applied to show the NP-hardness of other problems in computational geometry [6], computational biology [67] and program download [58]. We show a simple example on this reduction.

Given a 3SAT formula $\phi = F_1 \wedge F_2 \wedge F_3 \wedge F_4$, where $F_1 = (x_1 \vee \overline{x_2} \vee x_3)$, $F_2 = (\overline{x_1} \vee x_2 \vee \overline{x_4})$, $F_3 = (\overline{x_2} \vee \overline{x_3} \vee x_4)$, and $F_4 = (x_1 \vee \overline{x_3} \vee \overline{x_4})$, we want to find a truth assignment for ϕ. For each variable x_i, define S_i (resp. S_i') as the list of clauses containing x_i (resp. $\overline{x_i}$) followed by clauses containing $\overline{x_i}$ (resp. x_i). So $S_1 = F_1F_4F_2$ and $S_1' = F_2F_1F_4$, etc.

Then we construct two sequence $\mathcal{G} = S_1g_1S_2g_2S_3g_3S_4$, $\mathcal{H} = S_1'g_1S_2'g_2S_3'g_3S_4'$, where g_j's are peg genes only appearing once. Each gene appears at most three times as each clause contains three literals. The truth assignment can be set as follows: if $x_i = \text{TRUE}$, then keep the clauses in S_i and S_i' which contain x_i; if $x_i = \text{FALSE}$, then keep the clauses in S_i and S_i' which contain $\overline{x_i}$. If there are still duplicated clauses after this, then keep one such clause and delete the remaining ones arbitrarily. Regarding the above example, we can have $x_1 = x_3 = \text{TRUE}$, $x_2 = x_4 = \text{FALSE}$. So the corresponding exemplar genomes obtained are $G = H = F_4g_1F_3g_2F_1g_3F_2$, whose breakpoint distance is zero.

In different applications, F_i's and g_j's can be constructed to fit the corresponding problems, for instance as geometric points [6, 67] or programs to be downloaded [58].

9.2.4 The Complement Problem—ENbS

We comment that the negative results in Sect. 9.2.3 hold for any genomic distance $d(-,-)$ satisfying that $d(G,H)=0$ implies $G=H$ or $G=-H$. This, of course, implies that all the exemplar genomic distance problems (like exemplar reversal, exemplar transposition, and exemplar conserved interval distances) do not admit any polynomial-time approximation algorithms or any FPT algorithm, unless P = NP.

There have been two ways to handle this problem. One is to use a weak model of approximation, which will be covered as related to open problems in Sect. 9.5. The other, on the other hand, is to use a different similarity measure. In this case, one would try to maximize certain similarity measure. The most notable of such measures include non-breaking similarity (or number of adjacencies) [23] and the number of common intervals [12]. (A common interval is a pair of substrings appearing in the two genomes with the same genes, but possibly different orders. Example. $G=abced$, $H=deacb$. (abc, acb) is a length-3 common interval.) We will focus on the non-breaking similarity, which is really the complement of the breakpoint distance.

For two exemplar genomes G and H over the same alphabet of size n, recall that a breakpoint in G is a two-gene substring g_ig_{i+1} such that neither g_ig_{i+1} nor $-g_{i+1}-g_i$ is a substring in H. A *non-breaking point* (or an *adjacency*) is a common two-gene substring g_ig_{i+1} that appears either as g_ig_{i+1} or as $-g_{i+1}-g_i$ in G and H. The number of non-breaking points between G and H is also called the *non-breaking similarity* between G and H, denoted as $\mathrm{nbs}(G,H)$. Clearly, we have $\mathrm{nbs}(G,H)+\mathrm{bd}(G,H)=n-1$. For two genomes $\mathcal{G}$ and $\mathcal{H}$, their *exemplar non-breaking similarity* $\mathrm{enbs}(\mathcal{G},\mathcal{H})$ is the maximum $\mathrm{nbs}(G,H)$, where G and H are exemplar genomes derived from $\mathcal{G}$ and $\mathcal{H}$. Again we have $\mathrm{enbs}(\mathcal{G},\mathcal{H})+\mathrm{ebd}(\mathcal{G},\mathcal{H})=n-1$.

The Exemplar Non-breaking Similarity (ENbS) problem is formally defined as follows:

Instance: Genomes $\mathcal{G}$ and $\mathcal{H}$, each is of length $O(m)$ and each covers n gene families (i.e., at least one gene from each of the n gene families appears in both $\mathcal{G}$ and $\mathcal{H}$); integer K.

Question: Are there two respective exemplar genomes of $\mathcal{G}$ and $\mathcal{H}$, G and H, such that the non-breaking similarity between them is at least K?

We have the following negative results which have been proved in [23, 26].

Theorem 3 *If one of $\mathcal{G}$ and $\mathcal{H}$ is exemplar and the other is 2-repetitive, then the Exemplar Non-breaking Similarity problem does not admit any factor-$n^{0.5-\epsilon}$ polynomial-time approximation unless NP = ZPP.*

Proof We give a sketch of proof from [23, 26]. In [23, 26], it was shown that Independent Set can be linearly reduced to ENbS; i.e., the input graph has an independent set of size k iff the constructed ENbS instance has a non-breaking similarity (or number of adjacencies) equal to k. As Independent Set cannot be approximated

within a factor of $|V|^{1-\epsilon}$ unless NP = ZPP [39] and as in the reduction we use $\Theta(|V|^2)$ genes (where $|V|$ is the number of vertices in the input graph), the theorem follows. □

In [26], a factor-$O(\sqrt{n})$ approximation was presented for ENbS, show that the above inapproximability result is tight.

Theorem 4 *If one of $\mathcal{G}$ and $\mathcal{H}$ is exemplar and the other is 2-repetitive, the Exemplar Non-breaking Similarity problem does not admit an FPT algorithm unless FPT = W[1].*

Proof It is noted that the reduction from Independent Set to ENbS in [23, 26] is in fact an FPT reduction. As Independent Set is W[1]-complete [32], the theorem simply follows. □

In fact, with the lower bound results proved in [18], Independent Set (hence ENbS) cannot be solved in $O(f(k)n^{o(k)})$ time even if k is bounded by an arbitrarily small function of n, unless ETH fails. (ETH—Exponential Time Hypothesis: 3SAT cannot be solved in subexponential time.)

In the next section, we will survey another problem initiated by David Sankoff on computing syntenic blocks from genetic maps.

9.3 Maximal Strip Recovery and Its Complement

In a genome or physical map, the distance between two genes is exact. This is different in a genetic map, where only the relative positions between gene markers along chromosomes are indicated. A genetic map is usually constructed from DAGs (Directed Acyclic Graphs) which represent the partial order of gene markers. We omit the construction of genetic maps and interested readers are referred to [10, 68]. It should be noted that in a genetic map all the gene markers are distinct.

Given two genetic maps G and H represented by a sequence of n gene markers, a *strip* (syntenic block) is a sequence of distinct markers of length at least two which appear as subsequences in both of the input maps, either directly or in reversed and negated form. The problem *Maximal Strip Recovery* (MSR) is to find two subsequences G' and H' of G and H, respectively, such that the total length of disjoint strips in G' and H' is maximized An example is as follows: $G = abcdefgh$, $H = h - g - fcbdae$ and the optimal solution is $G' = cdefg$ and $H' = -g - fcde$, each containing two syntenic blocks cde and fg.

The MSR problem was proposed to handle the elimination of noise and ambiguities in genetic maps. This is related to the well-known problem in comparative genomics—to decompose two given genomes into syntenic blocks, i.e., segments of chromosomes which are deemed to be homologous in the two input genomes. In 2007, a heuristic method was proposed to handle the MSR problem [27, 70]. In [25], a factor-4 polynomial-time approximation algorithm was proposed for the problem. This was done by applying the Maximum Weight Independent Set on 2-interval

graphs, which admit a factor-4 approximation [5]. We also proved that several close variants of MSR, MSR-d (with $d > 2$ input maps), MSR-DU (with marker duplications), and MSR-WT (with markers weighted) are all NP-complete. It was left as an open problem whether the problem can be solved in polynomial time or is NP-complete [25].

Recently, in [65] we showed that MSR is in fact NP-complete, via a polynomial-time reduction from One-in-Three 3SAT (which was shown to be NP-complete in [34, 60]). We summarize the results in [25, 65] as follows.

Theorem 5 *MSR is NP-complete, and it admits a factor-*4 *polynomial-time approximation.*

As an effort to solve the MSR problem practically, we tried to handle MSR by solving its complement (CMSR) with FPT algorithms, i.e., showing that CMSR is fixed-parameter tractable [65]. Note that CMSR is a minimization problem where one deletes some markers such that the remaining ones in the genetic maps all belong to some syntenic blocks. With the previous example $G = abcdefgh$ and $H = h - g - fcbdae$, the optimal CMSR solution is to delete markers a, b, h.

Let k be the minimum number of markers deleted in some optimal solution of CMSR, the running time of known algorithms are $O(3^k n + n^2)$ [43], and $O(2.36^k n + n^2)$ [15]. In [45], we proved a $18k$ parameterized search space for CMSR and subsequently obtained a linear kernel of size (the actual size should be $78k$, slight better than in the conference version). Combining all these results, we have the following theorem.

Theorem 6 *Let k be the optimal number of gene markers deleted from the input genetic maps. CMSR can be solved in $O(2.36^k k + n^2)$ time; i.e., CMSR is fixed-parameter tractable.*

Note that as k is typically greater than 50 in real datasets, our FPT algorithms are not yet practical.

At the same time, approximation algorithms are presented for CMSR in the last couple of years. In [43], a factor-3 approximation was presented. The current best approximation factor is 2.33 [50]. Further improvement of approximation and FPT algorithms for CMSR remains open.

In the next section, we will survey the scaffold filling problem, again initiated by David Sankoff. Due to the technical difficulty of handling breakpoints and adjacencies in sequences (which was not completely given in [44]), this time we focus more on the details.

9.4 Approximation for Scaffold Filling with Gene Duplications

With respect to a target singleton genome, possibly with gene repetitions, a *scaffold* is simply an incomplete sequence. It was found that most of the sequenced

genomes are in fact in the form of scaffolds. Muñoz et al. first formulate the problem of filling an incomplete scaffold H into H', using a reference genome G, such that certain genomic distance between H' and G is minimized [54]. More specifically, they showed for multichromosomal genomes, this (one-sided) scaffold filling problem under the DCJ distance is polynomially solvable. David Sankoff visited Montana State University in early 2010 and gave a talk on this topic. We then started to collaborate by showing that for singleton genomes without gene repetitions, under the breakpoint distance, even the two-sided scaffold filling problem (i.e., both G, H are incomplete scaffolds or permutations) is polynomially solvable [40]. Then this result is generalized to multichromosomal genomes under the DCJ distance [44].

When genomes contain some duplicated genes, the scenario is completely different. There are three general criteria (or distance) to measure the similarity of genomes: the exemplar genomic distance [59], the minimum common string partition (MCSP) distance [30] and the maximum number of common string adjacencies [2, 41, 44]. Unfortunately, as covered in Sect. 9.2, unless P = NP, there does not exist any polynomial-time approximation (regardless of the factor) for computing the exemplar genomic distance even when each gene is allowed to repeat three times [21, 24] or even two times [13, 47]. The MCSP problem is NP-complete even if each gene repeats at most two times [36] and the best known approximation factor for the general problem is $O(\log n \log^* n)$ [30]. Based on the maximum number of common string adjacencies, Jiang et al. proved that the one-sided scaffold filling problem is also NP-complete, and designed a 1.33-approximation algorithm with a greedy strategy [41, 44]. As some of the details on handling breakpoints/adjacencies for sequences are missing in [44], we try to present the complete solution here. We comment that handling breakpoints/adjacencies for permutations is much easier.

9.4.1 Preliminaries

At first, we revise some necessary definitions, which are also defined in [44], but not in a perfect way. (Also, note that the breakpoint and adjacency definitions are more general than in Sect. 9.2 which only handle permutations.) We assume that all genes and genomes are unsigned, and it is straightforward to generalize the result to signed genomes. Given a gene set Σ, a string P is called *permutation* if each element in Σ appears exactly once in P. We use $c(P)$ to denote the set of elements in permutation P. A string A is called *sequence* if some genes appear more than once in A, and $c(A)$ denotes genes of A, which is a multi-set of elements in Σ. For example, $\Sigma = \{a, b, c, d\}$, $A = abcdacd$, $c(A) = \{a, a, b, c, c, d, d\}$. A *scaffold* is an incomplete *sequence*, typically obtained by some sequencing and assembling process. A substring with m genes (in a sequence) is called an *m-substring*, and a 2-substring is also called a *pair*, as the genes are unsigned, the relative order of the two genes of a pair does not matter, i.e., the pair xy is equal to the pair yx. Given a scaffold $A = a_1a_2a_3 \ldots a_n$, let $P_A = \{a_1a_2, a_2a_3, \ldots, a_{n-1}a_n\}$ be the set of pairs in A.

$$
\begin{aligned}
\textit{scaffold}\quad A &= \langle c\ b\ c\ e\ d\ a\ b\ a \rangle \\
\textit{scaffold}\quad B &= \langle a\ b\ a\ b\ d\ c \rangle \\
P_A &= \{cb, bc, ce, ed, da, ab, ba\} \\
P_B &= \{ab, ba, ab, bd, dc\} \\
\textit{matched pairs} &: (ab \leftrightarrow ba), (ba \leftrightarrow ab) \\
a(A, B) &= \{ab, ba\} \\
b_A(A, B) &= \{cb, bc, ce, ed, da\} \\
b_B(A, B) &= \{ab, bd, dc\} \\
\textit{bp-strings in } A &: c\ b\ c\ e\ d\ a \\
\textit{bp-strings in } B &: a\ b, b\ d\ c
\end{aligned}
$$

Fig. 9.1 An example for adjacency, breakpoint and the related definitions

Definition 1 Given two scaffolds $A = a_1a_2 \ldots a_n$ and $B = b_1b_2 \ldots b_m$, if $a_ia_{i+1} = b_jb_{j+1}$ (or $a_ia_{i+1} = b_{j+1}b_j$), where $a_ia_{i+1} \in P_A$ and $b_jb_{j+1} \in P_B$, we say that a_ia_{i+1} and b_jb_{j+1} are matched to each other. In a maximum matching of pairs in P_A and P_B, a matched pair is called an *adjacency*, and an unmatched pair is called a *breakpoint* in A and B, respectively.

It follows from the definition that scaffolds A and B contain the same set of adjacencies but distinct breakpoints. The maximum matched pairs in B (or equally, in A) form the *adjacency set* between A and B, denoted as $a(A, B)$. We use $b_A(A, B)$ and $b_B(A, B)$ to denote the set of breakpoints in A and B, respectively. A gene is called a *bp-gene*, if it appears in a breakpoint. A maximal substring T of A (or B) is call a *bp-string*, if each pair in it is a breakpoint. The leftmost and rightmost genes of a bp-string T are call the *end-genes* of T, the other genes in T are called the *mid-genes* of T. We illustrate the above definitions in Fig. 9.1.

Given two scaffolds $A = a_1a_2 \ldots a_n$ and $B = b_1b_2 \ldots b_m$, as we can see, each gene except the four ending ones is involved in two adjacencies or two breakpoints or one adjacency and one breakpoint. To get rid of this imbalance, we add "#" to both ends of A and B, which fixes a small bug in [41, 44]. From now on, we assume that $A = a_0a_1 \ldots a_na_{n+1}$ and $B = b_0b_1 \ldots b_mb_{m+1}$, where $a_0 = a_{n+1} = b_0 = b_{m+1} = \#$.

For a sequence A and a multi-set of elements X, let $A + X$ be the set of all possible resulting sequences after filling all the elements in X into A. Now, we define the problems we study in this paper formally.

Definition 2 Scaffold Filling to Maximize the Number of (String) Adjacencies (SF-MNSA).

Input: Two scaffolds A and B over a gene set Σ and two multi-sets of elements X and Y, where $X = c(B) - c(A)$ and $Y = c(A) - c(B)$.
Question: Find $A^* \in A + X$ and $B^* \in B + Y$ such that $|a(A^*, B^*)|$ is maximized.

The one-sided SF-MNSA problem is a special instance of the SF-MNSA problem where one of X and Y is empty.

Definition 3 One-sided SF-MNSA.

Input: A complete sequence G and an incomplete scaffold I over a gene set Σ, a multi-set $X = c(G) - c(I) \neq \emptyset$ with $c(I) - c(G) = \emptyset$.
Question: Find $I^* \in I + X$ such that $|a(I^*, G)|$ is maximized.

Note that while the two-sided SF-MNSA problem is more general and more difficult, the One-Sided SF-MNSA problem is more practical as a lot of genome analysis are based on some reference genome [54].

We now list a few basic properties of this problem.

Lemma 1 *Let G and I be the input of an instance of the One-sided SF-MNSA problem, and x be any gene which appears the same times in G and I. If x does not constitute breakpoint in G (resp. I), then it also does not constitute any breakpoint in I (resp. G).*

Proof W.L.O.G, assume that x appears q times in I and G, respectively. Also, assume that there are q_1 adjacencies in the form "xx", and q_2 adjacencies in the form "xy" ($y \neq x$) in G. In G, since each copy of x is involved in two adjacencies: one adjacency on its left and one adjacency on its right, but the two x's share the adjacency "xx", so the total number of adjacencies containing x is $2q - q_1$, then we have $2q - q_1 = q_1 + q_2$, which implies $2q - 2q_1 = q_2$.

In the scaffold I, there must be at least q_1 "xx" adjacencies. As x appears only q times, x has $2q$ neighbors where there are at least $2q_1$ x's. So x has at most $2q - 2q_1$ neighbors which are not x, which means that there are at most $2q - 2q_1$ ($=q_2$) pairs in the form "xy" ($y \neq x$) in I. Since there are q_2 "xy" ($y \neq x$) adjacencies in G, there must be q_2 "xy" ($y \neq x$) adjacencies in I. Therefore, there are exactly q_1 adjacencies in the form "xx", and all the q_2 pairs in the form "xy" ($y \neq x$) are adjacencies in I, and none of them is a breakpoint. □

Lemma 2 *Let G and I be the input of an instance of the One-sided SF-MNSA problem, let $bp(I)$ and $bp(G)$ be the multi-set of bp-genes in I and G, respectively. Then any gene in $bp(G)$ appears in $bp(I) \cup X$, and $bp(I) \subseteq bp(G)$.*

Proof Assume to the contrary that there exists a gene x, $x \in bp(G)$, but $x \notin bp(I) \cup X$. Since $x \notin X$, x appears the same number of times in G and I; moreover, $x \notin bp(I)$, then all the pairs in I containing x are adjacencies. From Lemma 1, all the pairs involving x in G are adjacencies, contradicting the assumption that $x \in bp(G)$. So any gene in $bp(G)$ appears in $bp(I) \cup X$. By a similar argument, we can prove $bp(I) \subseteq bp(G)$. □

Each breakpoint contains two genes, from what we discussed in Lemma 2, every breakpoint in the complete sequence G belongs to one of the three multi-sets according to the affiliation of its two bp-genes.

$$\begin{aligned}
\textit{scaffold}\ \ G &= \langle \#1\ 2\ 3\ 4\ a\ x\ y\ b\ \# \rangle \\
\textit{scaffold}\ \ I &= \langle \#1\ 3\ 2\ 4\ a\ b\ \# \rangle \\
X &= \{x, y\} \\
a(I, G) &= \{\#1, 23, 4a, b\#\} \\
b_G(I, G) &= \{12, 34, ax, xy, yb\} \\
BP_1(G) &= \{ax, yb\} \\
BP_2(G) &= \{xy\} \\
BP_3(G) &= \{12, 34\} \\
b_I(I, G) &= \{13, 24, ab\}
\end{aligned}$$

Fig. 9.2 Classification of the breakpoints

$BP_1(G)$: breakpoints with one bp-gene in X and the other bp-gene not in X.
$BP_2(G)$: breakpoints with both of the bp-genes in X.
$BP_3(G)$: breakpoints with both of the bp-genes not in X.

An example is shown in Fig. 9.2.

9.4.2 Approximation Algorithm for One-Sided SF-MNSA

In this subsection, we present a 1.33-Approximation algorithm for the one-sided SF-MNSA problem. The goal of solving this problem is, while inserting the genes of X into the scaffold I, to obtain as many adjacencies as possible. No matter in what order the genes are inserted, they appears in groups in the final $I' \in I + X$, so we can consider that I' is obtained by inserting strings (composed of genes of X) into I.

Obviously, inserting a string of length one (i.e., a single gene) will generate at most two adjacencies, and inserting a string of length m will generate at most $m + 1$ adjacencies. Therefore, we will have two types of inserted strings.

1. Type-1: a string of k missing genes $x_1, x_2, \ldots, x_k$ are inserted in between $y_i y_{i+1}$ in the scaffold I to obtain $k+1$ adjacencies (i.e., $y_i x_1, x_1 x_2, \ldots, x_{k-1} x_k, x_k y_{i+1}$), where $y_i y_{i+1}$ is a breakpoint.

   In this case, $x_1 x_2 \ldots x_k$ is called a k-Type-1 string, $y_i y_{i+1}$ is called a *dock*, and we also say that $y_i y_{i+1}$ *docks* the corresponding k-Type-1 string $x_1 x_2 \ldots x_k$.
2. Type-2: a sequence of l missing genes $z_1, z_2, \ldots, z_l$ are inserted in between $y_j y_{j+1}$ in the scaffold I to obtain l adjacencies (i.e., $y_j z_1$ or $z_l y_{j+1}$, $z_1 z_2$, $\ldots$, $z_{l-1} z_l$), where $y_j y_{j+1}$ is a breakpoint; or a sequence of l missing genes $z_1, z_2, \ldots, z_l$ are inserted in between $y_j y_{j+1}$ in the scaffold I to obtain $l + 1$ adjacencies (i.e., $y_j z_1$, $z_1 z_2$, $\ldots$, $z_{l-1} z_l$, $z_l y_{j+1}$), where $y_j y_{j+1}$ is an adjacency.

This is the basic observation for devising our algorithm. Most of our work is devoted to searching the Type-1 strings.

Searching the 1-Type-1 Strings To identify the 1-Type-1 strings, we use a greedy method. For each gene x_i of X and each breakpoint $y_j y_{j+1}$ of $b_I(I, G)$, if we can obtain two adjacencies by inserting x_i in between $y_j y_{j+1}$, then insert x_i to $y_j y_{j+1}$.

Algorithm 1: *Greedy*1(G, I)
1 Insert x_i in between $y_j y_{j+1}$ whenever two new adjacencies are generated, where $x_i \in X$ and $y_j y_{j+1} \in b_I(I, G)$.

Searching the 2-Type-1 Strings To identify the 2-Type-1 strings, we again use a greedy method. For each pair of missing genes $x_i x_k$ if we can obtain three adjacencies by inserting $x_i x_k$ in between $y_j y_{j+1}$, where $y_j y_{j+1} \in b_I(I, G)$, then insert $x_i x_k$ in between $y_j y_{j+1}$.

Algorithm 2: *Greedy*2(G, I)
1 Insert $x_i x_k$ in between $y_j y_{j+1}$ whenever three new adjacencies are generated, where $x_i, x_k \in X$ and $y_j y_{j+1} \in b_I(I, G)$.

Inserting the Remaining Genes In this subsection, we present a polynomial-time algorithm guaranteeing that the number of adjacencies increases by the same number of the genes inserted. A general idea of this algorithm was mentioned in [44], with many details missing, and we will present the details here.

Given the complete sequence G and the scaffold I, as we discussed in Sect. 9.4.1, the breakpoints in G can be divided into three sets: $BP_1(G)$, $BP_2(G)$, and $BP_3(G)$. In any case, the breakpoints in $BP_3(G)$ cannot be converted into adjacencies; so we try to convert the breakpoints in $BP_1(G)$ and $BP_2(G)$ into adjacencies.

Lemma 3 *If $BP_1(G) \neq \emptyset$, then there exists a breakpoint in I where after some gene of X is inserted, the number of adjacencies increases by one.*

Proof Let $t_i t_{i+1}$ be a breakpoint in G, satisfying that $t_i t_{i+1} \in BP_1(G)$, $t_i \in X$, and, from Lemma 2, $t_{i+1} \in bp(I)$. Then, there exists a breakpoint $t_{i+1} s_j$ or $s_k t_{i+1}$ in I. Hence, if we insert t_i in between that breakpoint, we will obtain a new adjacency $t_i t_{i+1}$ without affecting any other adjacency. □

Thus, it is trivial to obtain one more adjacency whenever $BP_1(G) \neq \emptyset$.

Lemma 4 *For any $x \in X \cap c(I)$, if there is an "xx" breakpoint in G then after inserting x in between some "xy" pair in I, the number of adjacencies increases by one.*

Proof If "xy" is a breakpoint, then after inserting an 'x' in between it, we obtain a new adjacency "xx". If "xy" is an adjacency, then after inserting an 'x' in between it, we have "xxy". The adjacency "xy" still exists, and we obtain a new adjacency "xx". □

Lemma 5 *If there is a breakpoint "xy" in $BP_2(G)$ and a breakpoint "xz" (resp. "yz") in I, then after inserting y (resp. x) in between "xz" (resp. "yz") in I, the number of adjacencies increases by one.*

Proof From the definition of $BP_2(G)$, we know that $x, y \in X$. Since "xy" is a breakpoint in G and "xz" is a breakpoint in I, we obtain a new adjacency "xy" by inserting y in between "xz", without affecting any other adjacency. A similar argument for inserting x in between "yz" also holds. □

Next, we show that the following case is polynomially solvable. This case satisfies the following conditions.

1. $BP_1(G) = \emptyset$;
2. It does not contain a breakpoint like "xx" in G unless $x \notin X \cap c(I)$;
3. For any breakpoint of the form "xy" in $BP_2(G)$, all the pairs in I involving x or y are adjacencies.

Let $BS_2(G)$ be the set of bp-strings in G with all breakpoints belonging to $BP_2(G)$.

Lemma 6 *In the case satisfying* (1), (2) *and* (3), *the number of times a gene appears as an end-gene of some bp-string of $BS_2(G)$ is even.*

Proof Let gene $x = t_i$ be an end-gene of some bp-string $t_i t_{i+1} \ldots t_j$ of $BS_2(G)$. Since $BP_1(G) = \emptyset$ and x will not be involved in any breakpoint of $BP_3(G)$, $t_{i-1}t_i$ must be an adjacency. Assume that x appears q times in G and q' ($<q$) times in I. As there is no breakpoint in the form "xx" in G and I, we could assume that there are q_1 adjacencies in the form "xx" in G and I. Then, the total number of pairs (adjacencies and breakpoints) involving x in G is $2(q - q_1)$, and of which, $2(q' - q_1)$ are adjacencies. So the number of breakpoints involving x in G is $2(q - q')$, which is even. An end-gene only constitutes one breakpoint and other mid-genes each constitutes two breakpoints. Therefore, any gene should appear at the end of some bp-string of $BS_2(G)$ for an even number of times. □

From Lemma 6, if we denote each bp-string of $BS_2(G)$ by a vertex, and there is an edge between two vertices iff their corresponding bp-strings have a common end-gene, the resulting graph contains a cycle of distinct vertices. Traveling this cycle, concatenating the bp-strings corresponding to the vertices, and deleting one copy of the common end-gene, eventually we can obtain a string composed of genes of X. The following lemma and corollary shows that this string can be inserted into I entirely, generating no breakpoint at all.

Lemma 7 *In the case satisfying* (1), (2) *and* (3), *for a gene x, let q_1 be the number that it appears as an end-gene, let q_2 be the number that it appears in some bp-string of $BS_2(G)$ as a mid-gene, and let r be the number that it appears in X. Then, we have $r = q_1/2 + q_2$.*

Proof Assume that x appears q times in G, q' ($< q$) times in I, and there are p adjacencies in the form "xx" in G and I. Then, the total number of pairs (adjacencies and breakpoints) involving x in G is $2(q - p)$, and of which, $2(q' - p)$ are adjacencies. So the number of breakpoints involving x in G is $2(q - q')$. Each x of q_1 end-genes contributes to one breakpoint, and each x of q_2 mid-genes contributes to two breakpoints, thus, $2(q - q') = q_1 + 2q_2$. Note that $(q - q')$ is exactly r; and following Lemma 6, q_1 is even. Then, $r = q_1/2 + q_2$. □

We summarize the above ideas as the following algorithm, which ensures us to obtain as many adjacencies as the number of missing genes inserted.

For two strings s_1 and s_2, if the right end-gene $r(s_1)$ of s_1 is the same as the left end-gene $\ell(s_2)$ of s_2, we use $s_1 \bowtie s_2$ to represent the string obtained by first concatenating s_1 with s_2 and then delete one copy of $r(s_1)$ and $\ell(s_2)$. For example, $s_1 = acbd$, $s_2 = decb$, then $s_1 \bowtie s_2 = abcdecb$.

Theorem 7 *The algorithm Insert-Whole-Strings*(•) *guarantees that the number of adjacencies increased is not smaller than the number of genes inserted.*

Proof At step 2, 3, 4 of the algorithm, one gene is inserted into I and each time one more adjacency is obtained. At each round of step 6, a string of length l is inserted in between an adjacency in I, then we obtain $l+1$ new adjacencies with one destroyed. So the number of adjacencies increased is not smaller than the number of genes inserted. □

Algorithm 3: *Insert-Whole-Strings*(G, I)
1 Identify the adjacencies and breakpoints in G and I.
2 If $BP_1(G) \neq \emptyset$,
  Insert a gene of X into I according to Lemma 3.
3 If there is an "xx" breakpoint in G, $x \in X$
  Insert x into I according to Lemma 4.
4 If there is an "xy" breakpoint in G and an "xz" breakpoint in I, $x, y \in X$,
  Insert y into Y according to Lemma 5.
5 Compute the set of bp-strings $BS_2(G) = \{s_1, \ldots, s_p$, where $s_j = x_{j,1} \ldots x_{j,u_j}$ and all $x_{j,k} \in X\}$.
6 WHILE ($BS_2(G) \neq \emptyset$)
  \\ Compute a string L composed of some bp-strings of $BS_2(G)$ whose two end-genes are the same.
  {
  (6.1) Choose any bp-string of $BS_2(G)$, say s_j. Let $L = s_j = x_{j,1} \ldots x_{j,u_j}$.
  (6.2) WHILE ($\ell(L) \neq r(L)$)
    Find a bp-string $s_i = x_{i,1} \ldots x_{i,u_i}$ (or its reversal $\overline{s_i} = x_{i,u_i} \ldots x_{i,1}$) of $BS_2(G)$, such that $r(L) = \ell(s_i)$ or $r(L) = \ell(\overline{s_i})$.
    Update $L \leftarrow L \bowtie s_i$ or $L \leftarrow L \bowtie \overline{s_i}$.
  (6.3) Replace some gene identical to $\ell(L)$ in I by the string L.
  (6.4) Update the set $BS_2(G)$.
  }
7 Return the resulting I.

We run the above algorithm on the following example.

$$G = \#daebxceafceb1234\#, \qquad I = \#dafcxb1324\#, \qquad X = \{a, b, c, e, e, e\},$$

$$BP_1(G) = \emptyset, \qquad BP_2(G) = \{ae, eb, ce, ea, ce, eb\}, \qquad BP_3(G) = \{12, 34\},$$

then the set of breakpoint strings $BS_2(G) = \{aeb, cea, ceb\}$. According to the algorithm, we have $L = aebecea$. Gene a in I is replaced with string L to obtain sequence $I^* = \#daebeceafcxb1324\#$. The number of adjacencies is added to by 6 and no new breakpoint is generated.

9.4.3 Analysis of the Approximation Algorithm

In this subsection, we will prove that the approximation factor of our algorithm is $4/3$. Firstly, we present a lower bound of the optimal solution.

A Lower Bound Given an instance of One-sided SF-MNSA, let $I^* \in I + X$ be the final scaffold in the optimal solution after inserting all genes of X into I. Compared to I, all genes belonging to X appear as substrings in I^*. Let $x_1x_2 \ldots x_l$ be a string inserted in between $y_i y_{i+1}$ in I^*, then either $y_i x_1$ or $x_l y_{i+1}$ or both are adjacencies. Since otherwise, we could delete this string from I^* (number of adjacencies decreases by at most $l - 1$), re-insert it following the algorithm *Insert-Whole-Strings*($\bullet$) (number of adjacencies increases by at least l), and obtain one more adjacency. Thus, we have the following corollary of Theorem 7,

Corollary 1 *Each substring in I^* composed of genes of X is either Type-1 or Type-2.*

Now, we present a lower bound for the optimal number of adjacencies.

Lemma 8 *Let OPT be the number of adjacencies between G and I^*, k_0 be the number of adjacencies between G and I, and $k_1 = |X|$. Let b_i be the number of i-Type-1 substrings and q be the maximum length of Type-1 substrings in the optimal solution between G and I^*. Then*

$$OPT - k_0 = k_1 + b_1 + b_2 + \cdots + b_q \leq \frac{4}{3}\left(k_1 + \frac{1}{2}b_1 + \frac{1}{4}b_2\right) \tag{9.1}$$

Proof Define C as the total number of genes in Type-2 substrings in I^*. Since inserting an l-Type-1 string will generate $l + 1$ more adjacencies, and inserting a l-Type-2 string will generate l more adjacencies, we have,

$$OPT = k_0 + \sum_{i=1}^{q}(i + 1) \times b_i + C.$$

By the definition of Type-1 and Type-2 substrings, we have

$$k_1 = \sum_{i=1}^{q} (i \times b_i) + C \geq b_1 + 2b_2 + 3(b_3 + b_4 + \cdots + b_q) + C.$$

Thus,

$$\sum_{i=3}^{q} b_i \leq (k_1 - C - b_1 - 2b_2)/3.$$

Hence, we have

$$\begin{aligned} OPT - k_0 &= C + \sum_{i=1}^{q} i \times b_i + b_1 + b_2 + \cdots + b_q \\ &= k_1 + b_1 + b_2 + \cdots + b_q \\ &\leq k_1 + b_1 + b_2 + (k_1 - C - b_1 - 2b_2)/3 \\ &\leq \frac{4}{3}\left(k_1 + \frac{1}{2}b_1 + \frac{1}{4}b_2\right). \end{aligned}$$

□

Lemma 8 shows that if the number of Type-1 substrings computed in the approximation algorithm is not smaller than $(2b_1 + b_2)/4$, then the approximation factor is $4/3$.

Description of the Main Algorithm There are three main steps in our algorithm. Firstly, we try to search the 1-Type-1 strings; secondly, we try to search the 2-Type-1 strings; finally, we insert the remaining genes in X, guaranteeing that on average we will obtain at least one adjacency for each inserted missing gene.

Main Algorithm	
Input: Complete sequence G and incomplete scaffold I, $X = c(G) - c(I)$.	
Output: $I' \in I + X$	
1	Call *Greedy*1(G, I), let the resulting incomplete scaffold be I_1.
2	Call *Greedy*2(G, I_1), let the resulting incomplete scaffold be I_2.
3	Call *Insert-Whole-Strings*(G, I_2). Let the resulting complete scaffold be I'.
4	Return I'.

9.4.4 Proof of the Approximation Factor

In our algorithm, we make effort to insert Type-1 substrings as much as possible. But a Type-1 substring (say I_s) inserted by our algorithm may make other Type-1 substrings in some optimal solution infeasible, we say I_s *destroys* them. The following lemma shows the number of Type-1 substrings that could be destroyed by a given Type-1 substring.

Lemma 9 *A i-Type-1 substring can destroy at most $i+1$ Type-1 substrings in some optimal solution.*

Proof Assume that an i-Type-1 substring I_s is inserted in between some breakpoint $y_j y_{j+1}$ in I. Then each of the genes in I_s, if not use by I_s, could form a distinct Type-1 substring in some optimal solution. Also, there may exist another Type-1 substring that could be inserted in between the breakpoint $y_j y_{j+1}$ in the optimal solution. Totally, at most $i+1$ Type-1 substrings in the optimal solution could be destroyed by I_s. □

We have the following lemma regarding this greedy algorithm.

Lemma 10 *Let b'_1, b'_2 be the number of type-1 1-substrings and 2-substrings inserted at Step 1 and Step 2 of our greedy algorithm, respectively. Then $b'_1 + b'_2 \geq \frac{b_1}{2} + \frac{b_2}{4}$.*

Proof Let k'_1, k'_2 be the number of missing genes inserted at Step 1 and Step 2, respectively. (So $b'_1 = k'_1$ and $b'_2 = k'_2/2$.) First, by Lemma 9, each of the k'_1 inserted missing genes can destroy at most two type-1 1-substrings in some optimal solution. Moreover, each of the k'_1 inserted missing genes can destroy at most two type-1 2-substrings in some optimal solution, this will be illustrated with an example at the end of this paragraph. Let b'_{10} be the number of missing genes inserted at Step 1 which destroy exactly one type-1 1-substring (and some type-1 m-substring, with $m \geq 3$) in some optimal solution. Let b'_{11} be the number of missing genes inserted at Step 1 which destroy exactly two type-1 1-substrings in some optimal solution. Let b'_{12} be the number of missing genes inserted at Step 1 which destroy one type-1 1-substring and one type-1 2-substring in some optimal solution. Let b'_{13} be the number of missing genes inserted at Step 1 which destroy exactly two type-1 2-substrings in some optimal solution. Obviously,

$$k'_1 = b'_1 = b'_{10} + b'_{11} + b'_{12} + b'_{13}.$$

Then, we show an example for a, one of the b'_{13} inserted missing genes that destroy two type-1 2-substrings in the optimal solution (i.e., counted into b_2). Let $G = \ldots\alpha a\beta\ldots\gamma ab\delta\ldots\alpha uv\beta\ldots$ and let $I = \alpha\ldots\alpha\beta\ldots\gamma\delta\ldots\beta\ldots a\ldots$. We need to insert a, b, u, v into I. Due to the greedy fashion of the algorithm, a is inserted between α, β in I to have $\alpha a\beta$ (destroying the possibility of inserting uv at the same location). On the other hand, due to the insertion of a (instead of ab), ab cannot be inserted in between γ and δ. Therefore, we destroy the optimal adjacencies $\langle\alpha uv\beta\rangle$ and $\langle\gamma ab\delta\rangle$ (with the corresponding two type-1 2-substrings: uv and ab).

Again, by Lemma 9, at Step 2, each of the inserted 2-type-1 substrings can destroy at most three 2-type-1 substrings in some optimal solution.

Now, putting all together,

$$b_1 \leq b'_{10} + 2b'_{11} + b'_{12},$$

and

$$b_2 \leq 3b'_2 + b'_{12} + 2b'_{13}.$$

Then

$$\begin{aligned}\frac{b_1}{2} + \frac{b_2}{4} &\leq \frac{b'_{10} + 2b'_{11} + b'_{12}}{2} + \frac{3b'_2 + b'_{12} + 2b'_{13}}{4} \\ &= \left(\frac{b'_{10}}{2} + b'_{11} + \frac{3b'_{12}}{4} + \frac{b'_{13}}{2}\right) + \frac{3b'_2}{4} \\ &\leq b'_1 + b'_2\end{aligned}$$

□

Theorem 8 *There is a greedy algorithm which approximates One-sided SF-MNSA with a factor of* 1.33.

Proof Following the greedy algorithm, Theorem 7, Lemma 8, and Lemma 10, we have the approximation solution value *APP*, which satisfies the following inequalities:

$$APP - k_0 = k_1 + b'_1 + b'_2 \geq k_1 + \frac{1}{2}b_1 + \frac{1}{4}b_2 \geq \frac{3}{4}(OPT - k_0).$$

So, we have $APP \geq \frac{3}{4}OPT + \frac{1}{4}k_0 \geq \frac{3}{4}OPT$. Hence $\frac{OPT}{APP} \leq 1.33$, and the theorem is proven. □

In [51], a better factor-1.25 approximation was proposed. While the overall framework is similar, the details are quite different. The new approximation is achieved by a combination of maximum matching, local improvement and greedy search.

9.5 Concluding Remarks and Open Problems

The negative results on EBD and ENbS do not mean that we have absolutely no way to tackle these problems. For instance, in [1], with integer linear programming, very nice empirical results are obtained. Here, we try to present a different way to handle these problems formally.

In many biological problems, the optimal solution value OPT could be zero. (Besides EBD, in some minimum recombination haplotype reconstruction problems the optimal solution value could be zero.) As implied by Theorem 1, if computing such an optimal solution with zero solution value is NP-complete then the problem does not admit *any* polynomial-time approximation (unless P = NP). However, in reality one would be satisfied to obtain a solution with value one or two. Due to this reason, we can relax the traditional definition of approximation to a *weak approximation*. Given a minimization problem Π, let the optimal solution of Π be OPT. We say that a weak approximation algorithm $\mathcal{W}$ provides a *performance guarantee* of α

for Π if for every instance I of Π, the solution value returned by $\mathcal{W}$ is at most $\alpha \times (OPT + 1)$.

In [21, 22, 24] we showed that EBD and the exemplar conserved interval distance problems are both hard to approximate even under the weak approximation model. But for the exemplar reversal distance problem, no such result is known yet.

For the exemplar common interval number problem [12], the only negative result is its NP-hardness. It would also be interesting to know whether it admits an efficient polynomial-time approximation. We conclude this paper with a list of open problems.

1. For the One-sided Exemplar Breakpoint Distance problem, does there exist a factor-$o(n)$ approximation? The only known negative result is the APX-hardness of the problem.
2. For the exemplar common interval number problem, does there exist a good approximation?
3. For the CMSR problem, does there exist faster FPT algorithm and/or a smaller linear kernel?
4. For the One-side SF-MNSA problem, does there exist an FPT algorithm?

Acknowledgements I would like to thank my collaborators for this series of research: Zhixiang Chen, Richard Fowler, Bin Fu, Haitao Jiang, Minghui Jiang, Zhong Li, Guohui Lin, Nan Liu, David Sankoff, Weitian Tong, Lusheng Wang, Boting Yang, Zhiyu Zhao, Chunfang Zheng and Daming Zhu.

References

1. 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**, 1093–1115 (2008)
2. Angibaud, S., Fertin, G., Rusu, I., Thevenin, A., Vialette, S.: On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. **13**(1), 19–53 (2009)
3. 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)
4. Bafna, V., Pevzner, P.: Sorting by reversals: genome rearrangements in plant organelles and evolutionary history of X chromosome. Mol. Biol. Evol. **12**, 239–246 (1995)
5. Bar-Yehuda, R., Halldórsson, M.M., Naor, J.(S.), Shachnai, H., Shapira, I.: Scheduling split intervals. SIAM J. Comput. **36**, 1–15 (2006)
6. Bereg, S., Jiang, M., Wang, W., Yang, B., Zhu, B.: Simplifying 3D polygonal chains under the discrete Fréchet distance. In: Proc. 8th Latin American Theoretical Informatics Symposium (LATIN'08), April 7–11, 2008. LNCS, vol. 4957, pp. 630–641 (2008)
7. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. J. Comput. Biol. **13**(2), 567–578 (2006)
8. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. In: Proc. 9th Intl. Ann. Comput. and Combinatorics (COCOON'03). LNCS, vol. 2697, pp. 68–79 (2003)
9. Berman, P., Hannenhalli, S., Karpinski, M.: 1.375-approximation algorithm for sorting by reversals. In: Proceedings of the 10th Annual European Symposium on Algorithms (ESA'02), pp. 200–210 (2002)

10. Bertrand, D., Blanchette, M., El-Mabrouk, N.: Genetic map refinement using a comparative genomic approach. J. Comput. Biol. **16**(10), 1475–1486 (2009)
11. Blin, G., Rizzi, R.: Conserved interval distance computation between non-trivial genomes. In: Proc. 11th Intl. Ann. Comput. and Combinatorics (COCOON'05). LNCS, vol. 3595, pp. 22–31 (2005)
12. Blin, G., Chauve, C., Fertin, G., Rizzi, R., Vialette, S.: Comparing genomes with duplicates: a computational complexity point of view. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**, 523–534 (2007)
13. Blin, G., Fertin, G., Sikora, F., Vialette, S.: The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In: Proc. 3nd Workshop on Algorithm and Computation (WALCOM'09). LNCS, vol. 5431, pp. 357–368 (2009)
14. Bryant, D.: The complexity of calculating exemplar distances. In: Sankoff, D., Nadeau, J. (eds.) Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families, pp. 207–212. Kluwer Academic, Dordrecht (2000)
15. Bulteau, L., Fertin, G., Jiang, M., Rusu, I.: Tractability and approximability of maximal strip recovery. Theor. Comput. Sci. **440–441**, 14–28 (2012)
16. Bulteau, L., Fertin, G., Rusu, I.: Sorting by transpositions is difficult. SIAM J. Discrete Math. **26**(3), 1148–1180 (2012)
17. Caprara, A.: Sorting permutations by reversals and Eulerian cycle decompositions. SIAM J. Discrete Math. **12**, 91–110 (1999)
18. Chen, J., Huang, X., Kanj, I., Xia, G.: Linear FPT reductions and computational lower bounds. In: Proceedings of the 36th ACM Symposium on Theory of Computing (STOC'04), pp. 212–221 (2004)
19. Chen, X.: On sorting permutations by double-cut-and-joins. In: Proc. of the 16th International Conf. on Computing and Combinatorics (COCOON'10), pp. 439–448 (2010)
20. Chen, X., Sun, R., Yu, J.: Approximating the double-cut-and-join distance between unsigned genomes. BMC Bioinform. **12**(Suppl. 9), S17 (2011)
21. Chen, Z., Fu, B., Zhu, B.: The approximability of the exemplar breakpoint distance problem. In: Proc. 2nd Intl. Conf. on Algorithmic Aspects in Information and Management (AAIM'06). LNCS, vol. 4041, pp. 291–302 (2006)
22. Chen, Z., Fu, B., Fowler, R., Zhu, B.: Lower bounds on the approximation of the exemplar conserved interval distance problem of genomes. In: Proc. 12th Intl. Ann. Comput. and Combinatorics (COCOON'06). LNCS, vol. 4112, pp. 245–254 (2006)
23. Chen, Z., Fu, B., Yang, B., Xu, J., Zhao, Z., Zhu, B.: Non-breaking similarity of genomes with gene repetitions. In: Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM'07). LNCS, vol. 4580, pp. 119–130 (2007)
24. Chen, Z., Fu, B., Fowler, R., Zhu, B.: On the inapproximability of the exemplar conserved interval distance problem of genomes. J. Comb. Optim. **15**(2), 201–221 (2008)
25. Chen, Z., Fu, B., Jiang, M., Zhu, B.: On recovering syntenic blocks from comparative maps. J. Comb. Optim. **18**, 307–318 (2009)
26. Chen, Z., Fu, B., Goebel, R., Lin, G., Tong, W., Xu, J., Yang, B., Zhao, Z., Zhu, B.: On the approximability of the exemplar non-breakpoint similarity problem of genomes with gene repetitions. Theor. Comput. Sci. (2013, to appear)
27. Choi, V., Zheng, C., Zhu, Q., Sankoff, D.: Algorithms for the extraction of synteny blocks from comparative maps. In: Proc. of the 7th International Workshop on Algorithms in Bioinformatics (WABI'07), pp. 277–288 (2007)
28. Christie, D.: A 3/2-approximation algorithm for sorting by reversals. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98), pp. 244–252 (1998)
29. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
30. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. In: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms (SODA'02), pp. 667–676 (2002)

31. Cui, Y., Wang, L., Zhu, D., Liu, X.: A $(1.5+\epsilon)$-approximation algorithm for unsigned translocation distance. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**(1), 56–66 (2008)
32. Downey, R., Fellows, M.: Parameterized Complexity. Springer, Berlin (1999)
33. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Trans. Comput. Biol. Bioinform. **3**, 369–379 (2006)
34. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
35. Gascuel, O. (ed.): Mathematics of Evolution and Phylogeny. Oxford University Press, Oxford (2004)
36. Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partitioning problem: hardness and approximations. In: Proc.15th Intl. Symposium on Algorithms and Computation (ISAAC'04). LNCS, vol. 3341, pp. 473–484 (2011). Also in: Electron. J. Comb. **12**, paper R50 (2005)
37. Hannenhalli, S.: Polynomial-time algorithm for computing translocation distance between genomes. Discrete Appl. Math. **71**(1–3), 137–151 (1996)
38. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM **46**(1), 1–27 (1999)
39. Hästad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**, 105–142 (1999)
40. Jiang, H., Zheng, C., Sankoff, D., Zhu, B.: Scaffold filling under the breakpoint distance. In: Proc. of the 2010 International RECOMB-CG Workshop (RECOMB-CG'10). LNBI, vol. 6398, pp. 83–92 (2010)
41. Jiang, H., Zhong, F., Zhu, B.: Filling scaffolds with gene repetitions: maximizing the number of adjacencies. In: Proc. 22nd Annual Symposium on Combinatorial Pattern Matching (CPM'11). LNCS, vol. 6661, pp. 55–64 (2011)
42. Jiang, H., Zhu, B., Zhu, D.: Algorithms for sorting unsigned linear genomes by the DCJ operations. Bioinformatics **27**(3), 311–316 (2011)
43. Jiang, H., Li, Z., Lin, G., Wang, L., Zhu, B.: Exact and approximation algorithms for the complementary maximal strip recovery problem. J. Comb. Optim. **23**(4), 493–506 (2012)
44. Jiang, H., Zheng, C., Sankoff, D., Zhu, B.: Scaffold filling under the breakpoint and related distances. IEEE/ACM Trans. Bioinform. Comput. Biol. **9**(4), 1220–1229 (2012)
45. Jiang, H., Zhu, B.: A linear kernel for the complementary maximal strip recovery problem. In: Proc. 23rd Annual Combinatorial Pattern Matching Symposium (CPM'12). LNCS, vol. 7354, pp. 349–359 (2012)
46. Jiang, H., Wang, L., Zhu, B., Zhu, D.: A $(1.408+\epsilon)$-approximation algorithm for sorting unsigned genomes by reciprocal translocations. In: RECOMB'13, poster (2013)
47. Jiang, M.: The zero exemplar distance problem. In: Proc. of the 2010 International RECOMB-CG Workshop (RECOMB-CG'10). LNBI, vol. 6398, pp. 74–82 (2010)
48. Kaplan, H., Shamir, R., Tarjan, R.: A faster and simpler algorithm for sorting signed permutations by reversals. SIAM J. Comput. **29**, 880–892 (1999)
49. Li, G., Qin, X., Wang, X., Zhu, B.: A linear-time algorithm for computing translocation distance between signed genomes. In: Proc. of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM'04), pp. 323–332 (2004)
50. Lin, G., Goebel, R., Li, Z., Wang, L.: An improved approximation algorithm for the complementary maximal strip recovery problem. J. Comput. Syst. Sci. **78**(3), 720–730 (2012)
51. Liu, N., Jiang, H., Zhu, D., Zhu, B.: An improved approximation algorithm for scaffold filling to maximize the common adjacencies. In: Proc. of the 19th Intl. Conf. on Computing and Combinatorics (COCOON'13). LNCS, vol. 7936, pp. 397–408 (2013)
52. Makaroff, C., Palmer, J.: Mitochondrial DNA rearrangements and transcriptional alternatives in the male sterile cytoplasm of Ogura radish. Mol. Cell. Biol. **8**, 1474–1480 (1988)
53. Marron, M., Swenson, K., Moret, B.: Genomic distances under deletions and insertions. Theor. Comput. Sci. **325**(3), 347–360 (2004)
54. Muñoz, A., Zheng, C., Zhu, Q., Albert, V., Rounsley, S., Sankoff, D.: Scaffold filling, contig fusion and gene order comparison. BMC Bioinform. **11**, 304 (2010)

55. Nguyen, C.T., Tay, Y.C., Zhang, L.: Divide-and-conquer approach for the exemplar breakpoint distance. Bioinformatics **21**(10), 2171–2176 (2005)
56. Ozery-Flato, M., Shamir, R.: An $O(n^{\frac{3}{2}}\sqrt{\log n})$ algorithm for sorting by reciprocal translocations. J. Discrete Algorithms **9**(4), 344–357 (2011)
57. Palmer, J., Herbon, L.: Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. J. Mol. Evol. **27**, 87–97 (1988)
58. Peng, C., Zhou, J., Zhu, B., Zhu, H.: The program download problem: complexity and algorithms. In: Proc. of the 19th Intl. Conf. on Computing and Combinatorics (COCOON'13). LNCS, vol. 7936, pp. 688–695 (2013)
59. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics **16**(11), 909–917 (1999)
60. Schaefer, T.: The complexity of satisfiability problem. In: Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78), pp. 216–226 (1978)
61. Sturtevant, A.: A crossover reducer in Drosophila melanogaster due to inversion of a section of the third chromosome. Biol. Zent.bl. **46**, 697–702 (1926)
62. Sturtevant, A., 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**, 448–450 (1936)
63. Swenson, K., Rajan, V., Lin, Y., Moret, B.: Sorting signed permutations by inversions in $O(n \log n)$ time. J. Comput. Biol. **17**(3), 489–501 (2010)
64. Tannier, E., Sagot, M.-F.: Sorting by reversals in subquadratic time. In: Proc. of 15th Symp. Combinatorial Pattern Matching (CPM'04), pp. 1–13 (2004)
65. Wang, L., Zhu, B.: On the tractability of maximal strip recovery. J. Comput. Biol. **17**(7), 907–914 (2010). (Correction, **18**(1) (Jan. 2011))
66. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. J. Theor. Biol. **99**, 1–7 (1982)
67. Wylie, T., Zhu, B.: Protein chain pair simplification under the discrete Frechet distance. IEEE/ACM Trans. Comput. Biol. Bioinform. 2013). doi:167B699B-E22D-471A-8EE7-01F51E8230D4. Special Issue of ISBRA'12
68. Yap, I., Schneider, D., Kleinberg, J., et al.: A graph-theoretic approach to comparing and integrating genetic, physical and sequence-based maps. Genetics **165**, 2235–2247 (2003)
69. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**, 3340–3346 (2005)
70. Zheng, C., Zhu, Q., Sankoff, D.: Removing noise and ambiguities from comparative maps in rearrangement analysis. IEEE/ACM Trans. Comput. Biol. Bioinform. **4**, 515–522 (2007)
71. Zhu, D., Wang, L.: On the complexity of unsigned translocation distance. Theor. Comput. Sci. **352**(1–3), 322–328 (2006)

Chapter 10
The Emperor Has No Caps! A Comparison of DCJ and Algebraic Distances

Joao Meidanis and Sophia Yancopoulos

Abstract In this chapter we investigate the *DCJ* and *algebraic* distances and how they are found. We introduce a new graphical method to determine the *permutation cycles* which embody the composition permutation for the genome transformation in the algebraic method. This graphical method helps tie the two approaches together. In the usual approaches, the two methods differ only in the distance component due to the *even paths* in the adjacency graph of Bergeron, Mixtacki, and Stoye involving operations changing type and number of chromosomes, such as fission, fusion, altering chromosome type from circular to linear, and vice versa. Discussing each distance individually, we compare their underlying assumptions. Both methods resort to *cycles* to determine the distance, but the basic DCJ uses "caps" to close paths. Without caps the algebraic distance differs from the standard DCJ for *even* paths. However, if caps and null chromosomes are added, the weighting schemes agree. A convention which can be done in multiple ways is the method of *path closure*. We discuss implementation of the *original closure rule* to arrive at the usual weighting scheme for the DCJ. Instead, by a new *alternative closure rule* which we introduce, the distance diverts to the algebraic distance. Finally, we note that although the Bergeron, Mixtacki, and Stoye DCJ approach via the adjacency graph does away with "fictitious" caps and nulls, vestiges of *fictitious operations* may remain, as the resulting weighting scheme is equivalent to that of the basic DCJ.

> So now the Emperor walked under his high canopy in the midst of the procession, through the streets of his capital; and all the people standing by, and those at the windows, cried out, "Oh! How beautiful are our Emperor's new clothes! What a magnificent train there is to the

J. Meidanis
Scylla Bioinformatics, Campinas, SP, Brazil
e-mail: meidanis@scylla.com.br

J. Meidanis
University of Campinas, Campinas, SP, Brazil

S. Yancopoulos (✉)
The Feinstein Institute for Medical Research, Manhasset, NY, USA
e-mail: sopheetsa@aol.com

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_10,

> mantle; and how gracefully the scarf hangs!" in short, no one would allow that he could not see these much-admired clothes; because, in doing so, he would have declared himself either a simpleton or unfit for his office. Certainly, none of the Emperor's various suits had ever made so great an impression, as these invisible ones.
> (Hans Christian Andersen, *The Emperor's New Clothes, Tales*)

10.1 Introduction

The early history of genome rearrangements harks back to a groundbreaking paper analyzing rearrangement scenarios in the fruit fly by Dobzhansky and Sturtevant in 1938 [7]. The onset of whole genome sequencing technologies made such studies truly viable. Hot in pursuit of these developments, the field really took off when David Sankoff and collaborators ushered in a new era of computationally based gene order comparisons [16–18]. In a prescient paper, David Sankoff made a brilliant intuitive leap to go from using edit distance based on the sequence level, to "non-local" large-scale genome rearrangement operations [15].

The language and ideas of permutations have frequently been inextricably linked with genome rearrangement studies [11, 12], however, much of the actual mechanics of permutations has taken a back seat in theoretical developments, subsumed by a graphical formalism that contains permutations implicitly, from the breakpoint graph introduced by Bafna and Pevzner (1993) [3] to the more recent adjacency graph of Bergeron et al. (2006) [4].

The recent reinjection of a more explicit algebraic formalism into the parlance of genome rearrangements has reinvigorated the discourse and challenged some of the basic underlying assumptions. Feijao and Meidanis's *Adjacency Algebraic Method* [9], a "hybrid" approach which combines the algebraic method with that of the adjacency formalism inherent in the use of the adjacency graph, arrives at some refreshingly unexpected results; these include a new weighting scheme for previously considered operations, particularly linear fissions and fusions, and the circularization or linearization of chromosomes.

In this chapter we focus on understanding the assumptions built into the new (adjacency) algebraic formalism, comparing with those underlying the standard DCJ approach. We explore how fundamental differences in the two approaches ultimately lead to differences in the distance and weighting schemes.

We begin with an introduction to permutations and genome rearrangements, followed by a brisk tour of classical models and operations. We introduce the DCJ and the essentials of algebraic rearrangement theory. We proceed to examine transformations involving circular genomes, fundamentally suited to the algebraic approach.

We continue by presenting the DCJ as originally conceived, using caps and nulls to effectively "circularize" all genomes. We transition to the algebraic method via two new approaches, a capping scheme for the algebraic method which results in the same distance as the standard DCJ and a new closure scheme that results in the new distance implied by the algebraic method. These differences based on the closure scenario allows us to realize the deep dependence of the capping and closure schemes with the resulting distance and weights of operations.

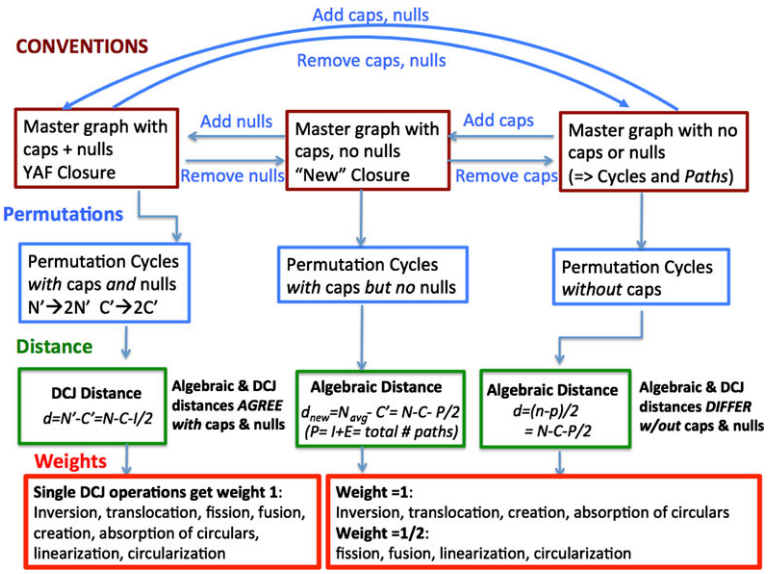


Fig. 10.1 Chapter overview. Fundamental assumptions affect distance and weighting schemes in the standard DCJ and Algebraic methods

We move on to examine the Adjacency Algebraic Theory and how it can be applied to linear chromosomes and transformations. We discuss a decomposition of the *Adjacency Graph* (AG) into components, and how these are independent and contribute independently to the distance. The essential components of the Adjacency Graph are *cycles*, and *even* and *odd paths*. We examine how these contribute to the genomic distance by the Adjacency Algebraic Theory, and contrast these contributions with the corresponding contributions for the DCJ.

Having developed the essential formalism for both methods, we go on to understand the consequences of the formalism on operations for different transformations as well as on the weighting scheme. We examine issues that arise from the introduction of "fictitious elements" (caps and nulls) including the possible artificial operations that may result as a consequence. We compare weighting schemes to see what consequences there are for these operations, and speculate on possible alternative weighting schemes as well as generalizations. We explore the implications including the correspondence to biology and conclude with open questions.

Figure 10.1 shows an overview of the results of this chapter.

10.1.1 Permutations and the Genome Rearrangement Problem

Mathematically, a *permutation* is a bijective function, or a *bijection*, taking a set A to itself. Bijections can be represented as directed graphs where all vertices have

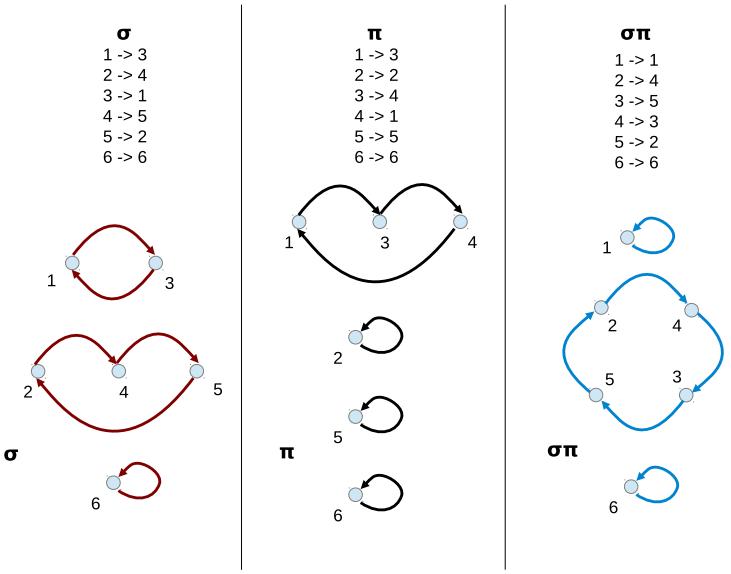


Fig. 10.2 Two permutations and their product. *On top*, we have the textual description of the process. *Below* this, we show the directed graphs representing each permutation, with σ in *red*, π in *black*, and the product $\sigma\pi$ in *blue*

indegree 1 and outdegree 1. Such graphs are collections of directed cycles that we call *permutation cycles* in this text.

Permutations are functions and can be composed as such. The composition of two permutations σ and π, where π is applied before σ, is indicated as a product and denoted by $\sigma\pi$ (Fig. 10.2). In general $\sigma\pi \neq \pi\sigma$. However, when two permutations are *disjoint*, that is they do not have any elements in common, they commute and $\sigma\pi = \pi\sigma$. The *support* of a permutation π is the set of elements that it moves, that is, $\pi(x) \neq x$. *Disjoint permutations* are permutations with disjoint supports.

In comparative genomics, evolutionary mutational processes causing large-scale rearrangements involve a shuffling of *syntenic segments* [14]. Researches have modeled this shuffling using permutations in a variety of ways.

For instance, in the paper by Bafna and Pevzner on the transposition problem [3], a genome is defined as a function $\pi : A \mapsto A$, where $A = \{1, 2, \ldots, n\}$ is used to indicate both *chromosome positions* in the domain of π as well as *genes* in the image of π. Transpositions are also modeled as permutations by Bafna and Pevzner. Several other rearrangement problems have been modeled via permutations, with adaptations for signed genes and multichromosomal settings [11, 12].

Meidanis and Dias [13] also use permutations, but in a different way. For them, a genome is a function $\pi : G \mapsto G$, where G is the set of *genes and their reverse complements*. No chromosomal positions are explicitly involved. The function π indicates which gene follows another gene in the genome (they restrict their study to circular genomes, where every gene in a chromosome has a successor). Feijao

and Meidanis [9] introduced yet another way of coding a genome as a permutation, by using the function π to indicate *adjacencies* between *gene ends*.

10.1.2 Genomes and the Chromosomal Notation

In this chapter, we deal with multichromosomal, signed genomes. We consider genomes that are collections of strictly linear or circular chromosomes, or a combination of both, made up of genes from a fixed set G. No gene repetitions are allowed, but all genes are always used in a pairwise comparison. We will represent linear chromosomes as lists of genes comprised in brackets, e.g., $[1, 2, 3]$, and circular chromosomes as lists of genes in parentheses, e.g., $(1, 2, 3)$. Gene orientation is indicated by a sign. So, for instance, $\pi = \{[1, 2], (-3, -4, 5)\}$ is a two-chromosome genome, with one linear and one circular chromosome. When there is no risk of confusion, we will drop the curly brackets, writing just $\pi = [1, 2], (-3, -4, 5)$.

Notice that this representation is not unique. For linear chromosomes, their reverse complement indicates the same chromosome, e.g., $[1, 2] = [-2, -1]$. For circular chromosomes, apart from the reverse complement, we also arbitrarily choose a gene to start, as there is no preferred starting point, e.g, $(-3, -4, 5) = (-4, 5, -3) = (5, -3, -4)$.

Observe that there is nothing in this notation that requires genes to be identified by integers. We can use letters, as in $[a, -c, -b, d]$, or even the very names used in biology to denote genes, such as *dnaA*, *cox1*, *adh*, or larger, contiguous regions, e.g., *MHC*, and so on. Nevertheless, for compatibility with the majority of theoretical papers on rearrangements, we will represent genes by integers here.

We typically denote the total number of genes in genome π by N_π, or just N depending on context. We will also represent a gene a by its two *gene ends* or *extremities*, with the *tail* denoted a_t, and its *head*, denoted by a_h, where a gene is typically oriented from tail to the head.

10.1.3 Genome Rearrangement Operations and Models

Given two genomes of equal gene content and a set of allowed operations, the most basic question is to find the *distance* between these genomes, defined as the smallest number of allowed operations that will transform one genome into the other. In its full generality such a scheme allows for *weights* assigned to the operation, and the distance then becomes the total weight of a minimum-weight series of allowed operations that transforms one genome into the other. An example of such an operation is a *reversal* where a stretch of contiguous genes in a chromosome gets inverted, and their signs flipped. For instance, genomes $\pi = [1, 2, 3, 4]$ and $\sigma = [1, -3, -2, 4]$ differ by a reversal. Transforming one genome to another by a series of reversals is a problem that has been well studied [3, 11, 12].

Translocations involve multiple linear chromosomes and result in swapping chromosomal ends between two chromosomes. For instance, genomes

$$\pi = \{[1, -2, -3], [-4, 5, 6]\}$$

and

$$\sigma = \{[1, -2, 5, 6], [-4, -3]\}$$

differ by a translocation. Formally, they are similar to reversals if one considers concatenated chromosomes [19].

A *transposition* is defined as an operation that swaps two adjacent substrings in a permutation. Genomes

$$\pi = (1, -2, -3, -4, 5, -6)$$

and

$$\sigma = (1, -4, 5, -2, -3, -6)$$

differ by a transposition. Sorting scenarios with transpositions were introduced by Bafna and Pevzner [2]. This is more difficult than the reversal distance problem, and there were various improvements in approximation methods for this problem.

Christie introduced *block interchanges* [6], which swap any two non-intersecting substrings, a natural generalization of transpositions. Genomes

$$\pi = [1, -2, -3, -4, 5, -6]$$

and

$$\sigma = [5, -6, -3, -4, 1, -2]$$

differ by a block interchange.

It is reasonable for biologically realistic models of genome rearrangements to include more than one kind of operation, however, in trying to consider generalized reversals and transpositions together in the menu of operations, researchers were somewhat baffled how to weight transpositions relative to reversals and translocations, which occur more frequently. In an intriguing paper, Blanchette et al. (1996) [5] allowed the weight of transpositions to vary relative to a weight of 1 for inversions in order to deduce the best weight. The authors noted there was a trade-off between inversions and transpositions, although some transpositions do not seem to be replaceable by inversions even with high values of the weighting function for transpositions. Using a greedy algorithm for genome rearrangements, they concluded that a weight just over 2 for transpositions and inverted transpositions was best able to optimize the rearrangement distance score for bacterial and mitochondrial genomes.

Recently, Bader and Ohlebusch [1], provided an algorithm for sorting by weighted reversals, transpositions and inverted transpositions using realistic weights.

Table 10.1 Examples of rearrangement operations

Operation	Example
Linear reversal	$[1, 2, 3] \mapsto [1, -2, 3]$
Circular reversal	$(1, 2, 3) \mapsto (1, -2, 3)$
Linear translocation	$[1, 2], [3, 4] \mapsto [1, 4], [3, 2]$
Circular fission	$(1, 2) \mapsto (1), (2)$

Other operations are possible, for instance, chromosome fissions and fusions, excisions of linear or circular pieces from linear or circular chromosomes, linearization of a circular chromosome, circularization of a linear chromosome, and so forth. Examples of some of these appear in Table 10.1.

10.1.4 The Basic DCJ

The DCJ, or *double cut and join* [4, 20], is a universal operation capable of modeling a number of genome rearrangement operations, including inversions, translocations, fissions, fusions, and the creation and absorption of circular chromosomes. The corresponding DCJ distance spurred a plethora of theoretical papers, and found its way in the implementation of several genome comparison systems. Perhaps the main reasons for its success are that, on the one hand it is easily computable by both humans (simple theory) and machines (low computational complexity), and, on the other hand, it models most operations observed to occur in real genomes, with reasonable weights.

The DCJ paradigm permits generalizations which allow insertions, deletions and duplications. Here we only entertain DCJ scenarios with equal gene content.

10.1.5 Algebraic Rearrangement Theory

Meidanis and Dias [13] used permutations to represent genomes, assigning a cycle to each chromosomal strand. The permutation formalism is particularly suited for *circular genomes* as these genomes automatically contain cycles by virtue of the circularity of their chromosomes; permutations, when drawn as directed graphs, also result in a collection of cycles. One interesting aspect of this approach is that the usual rearrangement operations become permutations with small support.

Recently, Feijao and Meidanis [9] studied a novel way of representing genomes as permutations, focusing on the *adjacencies between gene ends* rather than trying to model each genome strand as a permutation cycle of chromosomes. This allows both linear as well as circular chromosomes to be modeled. They also discovered a formula relating their *adjacency algebraic* representation to the *chromosomal algebraic* representation of Meidanis and Dias. By going backwards from the adjacency

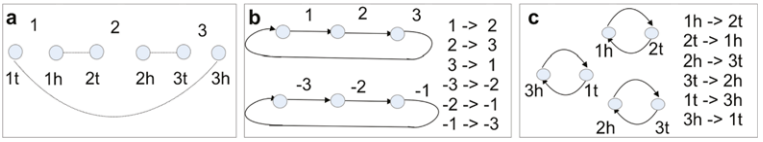


Fig. 10.3 Algebraic representation of a circular genome. (**a**) Genome representation with adjacencies. (**b**) Algebraic chromosomal representation. (**c**) Algebraic adjacency representation

to the chromosomal representation, using this formula, they were able to unify both theories, thus extending the chromosomal theory to encompass linear chromosomes as well. The result was a new rearrangement distance between two signed, multichromosomal genomes. The new distance is easy to compute and its value on random genomes correlates well with the DCJ distance [9].

In Fig. 10.3 we see the chromosomal and adjacency representations of a genome consisting of a single circular chromosome containing three genes.

10.1.6 Linear Chromosomes and "Fictitious" Elements (Caps)

Capping a linear chromosome is a technique that has been fruitful in many situations. For instance, in studying the unichromosomal transposition distance, Bafna and Pevzner extend their linear genomes π with $\pi(0) = 0$ and $\pi(n+1) = n+1$ [3]. The extra elements 0 and $n+1$ connected to the extremities of the linear chromosome are called *caps* and are useful in simplifying notation and arguments, avoiding awkward special cases. Caps have been used extensively in the multichromosomal context as well [12], and in studies with other operations, such as reversals and translocations.

The "basic" DCJ [20] is also heavily rooted in the use of caps. In a way, capping a linear chromosome is a device which transforms it into a circular one. Circular chromosomes seem to be easier to deal with particularly, as we shall see, in the context of permutations, and this is the rationale behind the use of caps.

However, a question lingers on. Does the introduction of caps somehow affect the genomic distance being computed? We show that the addition of caps may affect the resulting distance. As there is some flexibility in the ways of circularizing the genomes via caps and closure scheme, according to the choices made, one may get different distances. We will see that the DCJ and algebraic distances are two facets of this phenomenon.

10.2 Transformations Involving Circular Genomes

We now consider distance scenarios involving circular genomes. These are particularly suited to methods involving permutations. In Fig. 10.3 we saw the chromosomal representation for a circular chromosome contains complete cycles.

The goal of this section is to show that for circular genome transformations, the DCJ distance is equivalent to the algebraic distance. Although both distances have been covered in the literature, we offer here a self-contained treatment for the benefit of newcomers. We begin by introducing the main graphs mentioned in the literature, namely, the *breakpoint graph*, the *adjacency graph*, as well as the *master graph*, from which the previous two can be derived. These are important for visualizing the transformation and computing the genomic distance. We move on to some examples of circular operations, that is, simple operations that take a circular genome and transform it into another circular genome.

We note that even though such transformations may involve temporary states containing linear chromosomes, these are ultimately either circularized or absorbed into circular chromosomes so that the initial and final genomes are all circular. It is interesting to observe that temporary linear chromosomes can result from single-step operations such as single cuts in a circular. Operations such as *single cuts* or *single joins* have been considered by others, including by Feijao and Meidanis [8]. This highlights the importance of the particular model used to effect the transformation.

Sections 10.2.3 and 10.2.5 show how to compute DCJ and algebraic distances, respectively, and Sect. 10.2.6 contains a proof of the main result.

10.2.1 The Master, Breakpoint and Adjacency Graphs

The *master graph*, introduced by Friedberg, Darling, and Yancopoulos in 2008 [10], is a graph that specifies both genomes, and also connects between them. The initial genome, which we call π, is usually represented at the top (but as the diagram is completely symmetric it could also be done the other way, and be on the bottom). The gene extremities in this genome are linked by adjacencies which we color black here. These are *undirected* edges. The target genome which we call σ, is at the bottom and its gene extremities are linked by adjacencies that are colored red and are also undirected. Finally we use "green edges" to link corresponding gene ends in the two genomes. These are the "bridges" between the initial and target genomes. The red and black edges run horizontally, while the green edges connect the two genomes vertically. An example is shown in Fig. 10.4 in the next section. It is interesting to note that the master graph is in some sense the "precursor" graph for other important graphs. To see this:

- contract along the green edges and curve the black edges in the master graph to get the *breakpoint graph* for the *inverse transformation*
- for the *forward transformation breakpoint graph*, contract along the green edges; curve the *red* edges in the master graph, and invert the whole thing vertically
- to get the *adjacency graph* for the transformation, contract both red and black edges in the master graph to become vertices.

In their paper on polynomially sorting by reversals, Hannenhalli and Pevzner define a breakpoint graph slightly differently from the one defined here [11]. The reason

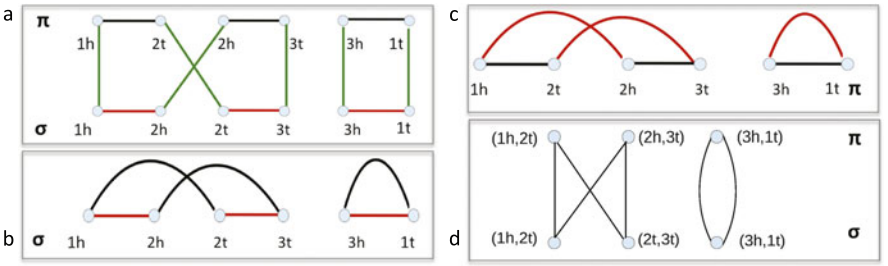


Fig. 10.4 Inversion in circular. (**a**) Master graph (MG), (**b**) & (**c**) Breakpoint Graphs (BPG) for inverse and forward transformation, (**d**) Adjacency Graph (AG)

is that the HP graph is for a linear problem and uses caps, while the master graph described above has no caps. However, we will see in Sect. 10.3, which treats linear chromosomes, there is a capped version of the master graph, for which the previous bulleted items above still apply. The next section shows examples of these graphs.

10.2.2 Examples of Transformations in Circular Genomes

In this section we consider two examples of transformations in circular genomes. First we consider an internal inversion and then a case of a fission.

10.2.2.1 An Inversion in a Circular from $\pi = (1, 2, 3)$ to $\sigma = (1, -2, 3)$

We construct the master graph and the other graphs as described in Sect. 10.2.1.

The master graph MG(π,σ) is shown in Fig. 10.4(a). Below it is the breakpoint graph (BPG) for the inverse transformation. Note it is the *inverse* transformation as the genome at the bottom is the target genome, with adjacencies in red. The genome at top, is the current or initial genome. The BPG for the forward transition is shown in Fig. 10.4(c) arrived at by inverting the MG so the target genome is at the bottom. Its adjacencies are represented by black lines. As previously, the green and red lines are then curved and colored red; these "desire lines" connect gene ends in the target genome. The corresponding adjacency graph (AG), is shown in Fig. 10.4(d). As before, we arrive at the AG by contracting red and black edges in the master graph.

Essentially, forward and inverse BPGs are contained in the MG. With practice, we can see them directly. The forward transformation BPG is up side down! The AG can also be visualized by mentally performing the procedure in the previous section.

10.2.2.2 Circular Fission from $\pi = (1, 2, 3)$ to $\sigma = (1), (2, 3)$

We show the master, breakpoint, genome, and adjacency graphs for the *circular fission* in Fig. 10.5(a)–(d). Comparing these diagrams with those for the inversion,

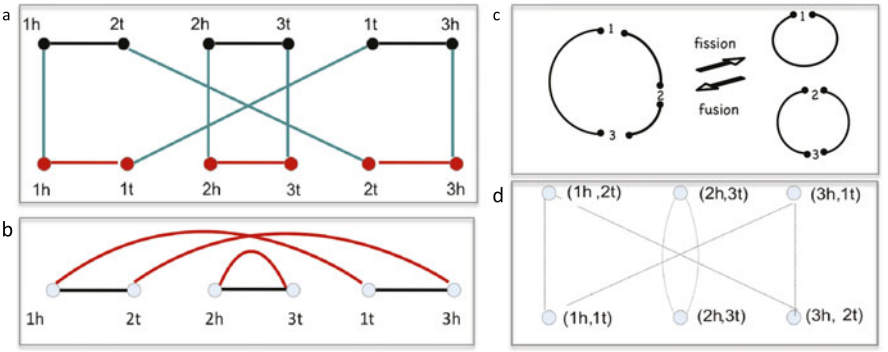


Fig. 10.5 The (**a**) MG, (**b**) BPG, (**c**) Genome Graph, and (**d**) AG for a circular fission

we see they are topologically similar in their adjacency graphs and both contain one 2-cycle and one 1-cycle.

A word about *cycle notation*: in a departure from the usual convention in the genome rearrangement literature, we call a cycle in the adjacency graph containing j adjacencies in only one genome a *j-cycle*. Usually it is called a *2j-cycle*. In the AG, *2-cycles* look like *bow ties*.

10.2.3 How to Compute the DCJ Distance from the Master Graph

For circular genomes, computing the DCJ distance from the master graph is a simple matter. All we have to do is compute

$$d_{\mathrm{DCJ}}(\pi, \sigma) = N - C,$$

where $N = N_\pi = N_\sigma$ is the number of adjacencies (as well as the number of genes) per genome, and C is the number of cycles in the master graph. To see this, we note the master graph (or equivalently, the BPG or AG) for a transformation involving only circular genomes consists completely of cycles and each cycle can be resolved independently.

To resolve a k-cycle such as at the top of Fig. 10.6 by DCJs, we perform one DCJ at a time. A single DCJ can increase or decrease the cycle count by one, or in the case of an "improper" rejoining keep the cycle count unchanged [20]. If we perform two cuts (in "consecutive" adjacencies) we can rejoin so that a target adjacency is achieved. This forms a 1-cycle consisting of the target adjacency, and a remaining cycle with one less adjacency. Proceeding until the remaining cycle is a 2-cycle, the final 2-cycle is resolved by a single DCJ, transforming it into two 1-cycles. As there are k adjacencies, the final graph has k 1-cycles. Each subsequent DCJ produces a new 1-cycle except the last, which produces two. Since a single DCJ can at most augment the total number of cycles by one, we see that it takes a minimum

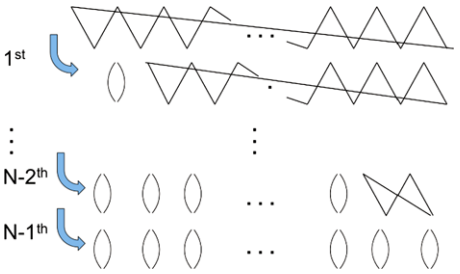


Fig. 10.6 Resolving a k-cycle by DCJs takes $k-1$ DCJs

of $k-1$ DCJs to resolve a k-cycle. A non-minimal path can take more steps by performing DCJs which decrease rather than increase the number of cycles. If the graph is composed of cycles with sizes (counted as the number of adjacencies in each cycle) $k_1, k_2, \ldots$, to find the total distance, we sum on cycles:

$$d_{\mathrm{DCJ}}(\pi, \sigma) = \sum_j (k_j - 1) = N - C \tag{10.1}$$

since the total number of adjacencies N is equal to the adjacencies summed over all cycles, $N = \sum_j k_j$, and similarly, the total number of cycles is just $C = \sum_j 1$.

10.2.4 Finding the Permutation Cycles from the Master Graph

The algebraic (permutation) approach is essentially more *symbolic* whereas genome rearrangement approaches from HP [11, 12], to the DCJ resort to graphical representations. The master graph offers an excellent way to connect the DCJ approach with the Algebraic Adjacency Method. In Fig. 10.7, which revisits the inversion in a circular, we see that we can go directly from the adjacencies in the master graph in (a) to the 2-cycles in the *algebraic adjacency* representation in (b). Hence, the top and bottom of the master graph represent the initial and target genome in terms of their adjacencies, and equivalently their 2-cycle algebraic adjacency representation.

To see how to arrive at the product permutation $\sigma\pi$ from the initial to the target genome, consider the red path in Fig. 10.7(d), where we trace going from a gene end in π (e.g., $1h$, which connects to $2t$ in π) to the connecting gene end in σ (i.e., $2t$ connects to $3t$ in σ). Hence, the product permutation, where π is performed first and then σ, results in a transition from $1h$ to $3t$, shown in the $\sigma\pi$ table in (d) and also outlined in the red *3-step* path in (a), leading to the *blue line*. Superimposing all these blue lines on the master graph ((a) and (c)), will ultimately trace out the product permutation cycles (Fig. 10.7(e)).

To find the product permutation cycles for transformations involving arbitrary circular genomes, we generalize the procedure performed above. We proceed as follows to generate the $\sigma\pi$ *permutation cycles*:

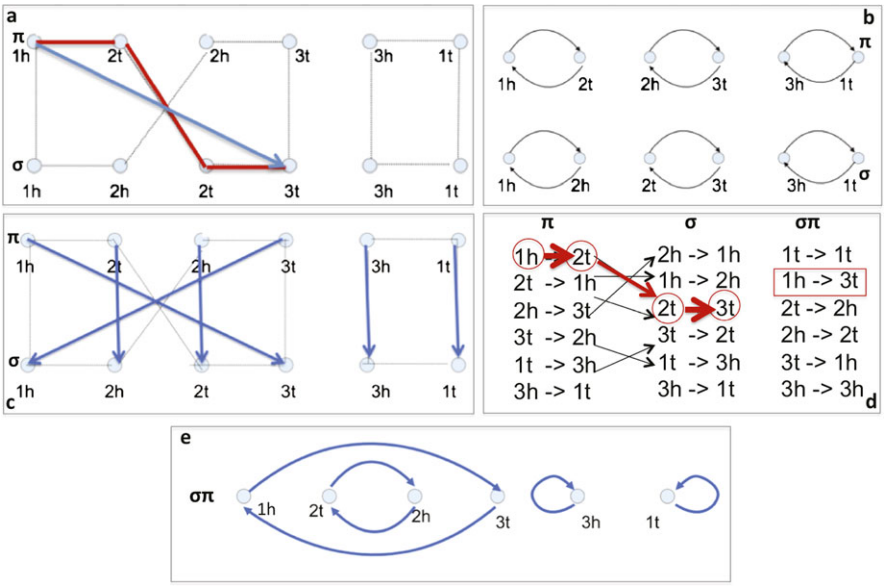


Fig. 10.7 From master graph to permutation cycles

1. construct the master graph (MG) of σ and π
2. for the blue lines, at each extremity in π walk three steps starting on a black line
3. dissolve red and black "adjacency" lines connecting gene ends in adjacencies
4. contract along "green" lines; the blue lines trace the permutation cycles.

10.2.5 Computing the Algebraic Distance for Circular Genomes

An important concept introduced within the algebraic theory is the *norm* of a permutation, which measures the "complexity" of this permutation and equals $n - p$, where n is the number of vertices and p the number of permutation cycles in the corresponding directed graph. We note the DCJ distance is of the same form.

In order for d_{DCJ} and d_{alg} to coincide, the algebraic distance is defined to be the norm of the composition permutation *divided by two* or they would differ by a factor of two. For circular genomes, the master graph for the transformation resolves completely into cycles which double in the permutation representation.

To visualize the cycle doubling from the master graph to the permutation cycles, it is helpful to look at Fig. 10.8(b), where the 2-cycle from the previous example (also shown here in Fig. 10.8(a)) is "opened up". Adding blue lines as in Fig. 10.8(c) one blue line emanates from each gene end in π and bypasses three "BGR" (black-green-red) segments, ending at a gene end in σ. Traversing the next green line returns us to π and after N traversals alternating between blue and green lines, we return to our starting point.

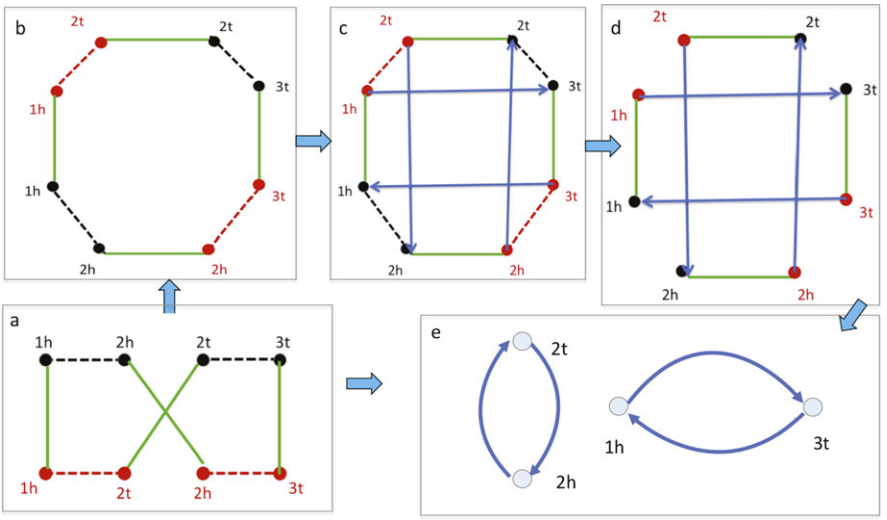


Fig. 10.8 Cycle doubling from master graph to permutation cycles

As the red and black adjacency lines are dissolved, separating gene ends in adjacencies, and the green lines contracted, the number of permutation cycles are double those in the master graph. The number of gene ends ($2N$) is twice the number of genes (N). Hence the norm of the composition permutation is $2N - 2C$ and the corresponding distance is half, or $N - C$ which agrees with the DCJ distance.

10.2.6 Comparing Methods for Circular Chromosomes and Proof

We have seen that for circular genomes the following formula can be used to compute the DCJ distance:

$$d_{\mathrm{DCJ}}(\pi, \sigma) = N - C, \tag{10.2}$$

where $N = N_\pi = N_\sigma$ is the number of adjacencies per genome, and C is the number of cycles in the master graph.

On the other hand, we can compute the algebraic distance directly from the permutation cycles by taking the norm (the difference between the number of vertices and the number of cycles) and dividing by 2, which is equivalent to

$$d_{\mathrm{alg}}(\pi, \sigma) = \frac{n - p}{2} = \frac{2N - p}{2} = N - \frac{p}{2}, \tag{10.3}$$

where n is the number of *gene ends* (double the number of genes, which, in circular genomes agrees with the number of adjacencies per genome, or N), and p is the

number of permutation cycles in the $\sigma\pi$. We saw in the examples in previous sections that $p = 2C$, because each master graph cycle gives rise to two permutation cycles. As a result, the DCJ and algebraic distances agree.

Specifically, for the inversion of a circular (Fig. 10.4) and the circular fusion (Fig. 10.5), both cases consisted of one 2-cycle and one 1-cycle in their adjacency graphs. We recall that our cycle nomenclature labels a j-cycle by the number j of adjacencies in only one of the genomes.

Now, 1-cycles do not contribute to the distance. To see this, note that in the DCJ formulation, the distance contribution for $N = 1$ and $C = 1$ is 0. In the algebraic formulation, we saw (Fig. 10.8) that the 1-cycle *doubles into two cycles* in the permutation $\sigma\pi$, where the notation for the cycles is based on the number of *gene ends* in the adjacency graph, which is $n = 2N = 2$ for a 1-cycle. In the permutation $\sigma\pi$, the two cycles each become $n/2$-cycles, or 1-*cycles*, for which the algebraic distance is $(n - p)/2 = (2 - 2)/2 = 0$. As for the 2-cycle, it contributes $N - C = 2 - 1 = 1$ in the DCJ formulation and $(n - p)/2 = (4 - 2)/2$, which also equals 1, in the algebraic formulation. Hence the two distances agree.

In this section, we go on to prove that the agreement between the two distances (algebraic and DCJ) is a general result, as far as circular genomes are involved. This cannot be considered a new result, because it is implied by formulas for the distances given by Feijao and Meidanis [9], but since our approach here significantly differs from previous treatments, and one of our goals here is to view the algebraic theory in graphical terms, we developed the proof below.

Theorem 1 *If π and σ are circular genomes with the same genes, then*

$$d_{\mathrm{DCJ}}(\pi, \sigma) = d_{\mathrm{alg}}(\pi, \sigma).$$

Proof In view of formulas (10.2) and (10.3), it suffices to prove that the "cycle doubling" occurs in general. To see that this is actually the case, reason as follows. Every cycle in the master graph will have a number of edges that are a multiple of 4. This is because the edges alternate between green and adjacency edges, and, moreover, the adjacency edges alternate between red and black edges. In other words, starting from any gene end in π, for instance, and following the direction of the only black edge incident to it, we necessarily traverse four edges of colors black, green, red, and green again before we have the chance of closing the cycle. So the minimal cycle has four edges. If the cycle does not close after the first four edges, this means we reached a new, free gene end in π, and must traverse a new 4-edge walk (black, green, red, green) before being able to close.

Let's now see what happens with each such cycle as we compute the permutation cycles. Each permutation cycle is formed by blue edges, and each blue edge is the result of traversing a black-green-red walk in the master graph, that is, we are effectively walking over a master graph cycle. Moreover, in the end the green edges get contracted, and the net result of this can be seen to yield a blue line traversing each consecutive black-red segment of the corresponding breakpoint graph cycle.

It follows that each edge in a permutation cycle corresponds to four consecutive edges in the corresponding master cycle. In a master cycle with $4k$ edges, this will

result in a permutation cycle with k edges. However, this permutation cycle involves only half of the π gene ends in the master cycle, because the adjacency partner of each gene end in the permutation cycle is missing from this cycle, since it is reachable in just one step, which is not a multiple of 4. The missing gene ends will be in another k cycle. It follows that each cycle in the master graph gives rise to two permutation cycles, that is,

$$p = 2C,$$

and therefore $d_{\mathrm{DCJ}}(\pi, \sigma) = d_{\mathrm{alg}}(\pi, \sigma)$. □

10.3 General Transformations Using "Fictitious" Elements

Having seen that for circular genome transformations the DCJ and algebraic distances turn out the same, we move on to the general case of multichromosomal genomes with no restrictions on chromosome types: circular, linear, or a mixture.

We review the original construction by Yancopoulos et al. which effectively transforms the general case into the circular case [20] by the addition of caps, and, where needed, null chromosomes to restore the balance in number of chromosomes between genomes. We call this procedure the *original closure rule* and the transformed graph the *augmented* master graph. Once this is done, the DCJ distance can be computed as in the circular case by using the augmented master graph.

Using the approach in the previous section, the augmented graph can also be used to derive permutation cycles, and thereby the algebraic distance. We find, perhaps not surprisingly, this coincides with the DCJ distance. We also investigate a new closure rule that keeps the caps but closes the paths into cycles without resorting to null chromosomes. Perhaps surprisingly, this *alternative closure rule* results in exactly the algebraic distance.

A final subsection summarizes these observations and contains proofs of results.

10.3.1 Examples of Transformations with Linear Chromosomes

In this section we consider examples of transformations with linear chromosomes. In examining their master graphs we observe that in addition to cycles they also contain paths. Paths occur in the event of linear chromosomes, as there are vertices in the adjacency graph corresponding to "telomeric" gene ends, gene ends at the ends of chromosome without partners. We investigate how to close such paths to form cycles, so that we will be able to compute the distance as previously, by subtracting the number of cycles from the number of adjacencies.

10.3.1.1 Linearization

Consider the transformation from a genome with a circular chromosome, $\pi = (1)$, to that of a linear chromosome $\sigma = [1]$. This operation is the linearization of a circular

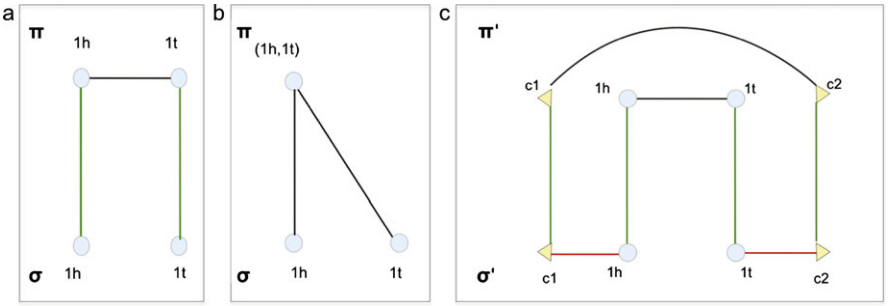


Fig. 10.9 Linearization. (**a**) Master Graph, (**b**) Adjacency Graph, (**c**) Augmented Master Graph with caps and null

chromosome. The master graph for this pair of genomes is shown in Fig. 10.9(a). Contracting red and black edges we arrive at the adjacency graph in Fig. 10.9(b).

As a result of the linear chromosome in the transformation, the master graph and adjacency graph have a path. To circularize this path, we add caps to the telomeres (unpaired extremities) in $\sigma = [1]$. This results in a disequilibrium of adjacencies in the two genomes π and σ. To restore the balance, we add a null chromosome $(c1, c2)$, containing the same caps added in σ, to π, and close the path by adding connecting green lines to the null. The resulting augmented master graph can be seen in Fig. 10.9(c).

10.3.1.2 Fission of a Linear Genome

Consider the transformation from a genome of a linear chromosome $\pi = [1, 2]$ fissioning into two chromosomes $\sigma = \{[1], [2]\}$. The master graph for this pair of genomes is shown in Fig. 10.10(a). Contracting red and black edges we arrive at the adjacency graph, shown in Fig. 10.10(b).

Here again we have paths. To close them, we start by adding caps to each telomere. There are two in π and four in σ. After adding these caps there is a resulting imbalance of adjacencies in the two genomes. We compensate for that by adding a null chromosome, $(c2, c3)$, to π. Connecting corresponding gene ends with green edges we arrive at the augmented master graph in Fig. 10.10(c).

10.3.2 The Original Closure Rule, Completing Paths with Caps and Nulls

To understand the DCJ approach for general transformations including all kinds of chromosomes, we note that, with linear chromosomes present, the master graph contains paths in addition to cycles. The idea behind the method is to convert cases

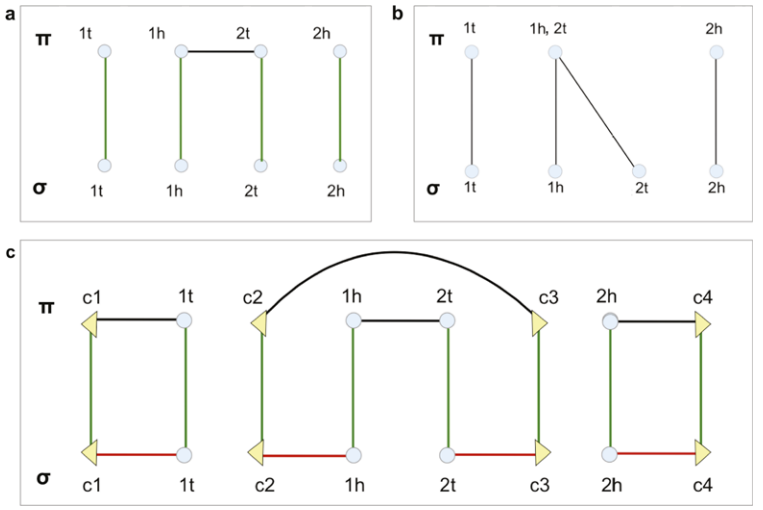


Fig. 10.10 Fission in a linear. (**a**) Master Graph (**b**) Adjacency Graph (**c**) Augmented Master Graph

containing paths to the circular case by augmenting the master graph to circularize paths into cycles.

Linear chromosomes have unpaired gene ends at their ends called *telomeres*. We augment the master graph by attaching new, "dummy" gene ends called *caps* to the telomeres. Effectively we cap the ends of paths with these fictitious elements.

Once we have capped the paths,we note there are two kinds of paths in their master (or equivalently their adjacency) graphs, those with both capped ends in the *same* genome, are called *even paths*, and those with their caps in *opposite* genomes, are called *odd paths*.

To close paths into cycles in the master (or adjacency graph), we add *green lines*, with two different approaches for odd and even paths. The *original closure rules* from Yancopoulos et al. [20] are:

1. For *odd paths* join the caps at the ends of the paths *directly* by a green edge.
2. For *even paths* create a *null chromosome* containing two caps in the genome opposite the genome with the two caps. Draw green edges linking each of the caps in the first genome to the caps of the null chromosome in the other genome.

We notice that, this method generates null chromosomes. The total number of adjacencies is augmented by counting nulls as a single adjacency and single caps as half. The total number of adjacencies is balanced since adjacencies containing actual gene ends are balanced in both genomes in cycles as well as in paths (where telomeres count as half an adjacency), *odd* paths have one cap in each of the genomes, and *even* paths, have two caps in one of the genomes balanced by a null in the other.

Denoting the augmented number of adjacencies in either genome by N', and the total number of cycles including paths closed by the closure rule by C', the DCJ

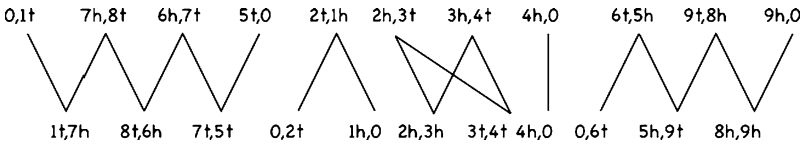


Fig. 10.11 A sample adjacency graph for a general transformation, that is, a transformation involving linear chromosomes. Here the *top* genome is $[1, 2, 3, 4]$, $[5, 6, 7, 8, 9]$, and the *bottom* genome is $[-1, -7, 5, 9, -8, -6]$, $[2, -3, 4]$

distance can be computed as follows:

$$d_{\text{DCJ}} = N' - C'.$$

10.3.3 Understanding the DCJ Distance for General Transformations

Having extended the formalism of closed cycles to the general case including linear chromosomes by our system of capping and the original closure rules, we now wish to make contact with the Bergeron et al. [4] distance formula. Accordingly we will focus on the adjacency graph, keeping in mind it can be derived from the master graph simply by compressing adjacencies. We cap and close paths using the original closure rule. Our goal is to reconsider the path formulation of Bergeron et al. in light of caps, nulls and path closure. For illustrative purposes, consider the adjacency graph in Fig. 10.11 containing paths and cycles.

Using the capping and closure formalism just developed, we know how to complete paths to transform them into cycles. We take note of some general principles:

- In general, the Adjacency Graph (AG) contains cycles and (even and odd) paths.
- A component-wise decomposition of the AG resolves it into these components.
- Each component makes an *independent* contribution to the DCJ distance.

We now consider how each individual component contributes to the total distance arrived at in the previous subsection.

10.3.3.1 Distance Contribution from Cycles

From Sect. 10.2.3 we know the distance contribution of an individual cycle containing k adjacencies is just $k - 1$. Hence, if there are C cycles in the entire graph, involving a total of N_{cycle} genes, there are also N_{cycle} adjacencies, and the total contribution from cycles to the DCJ distance is

$$N_{\text{cycle}} - C.$$

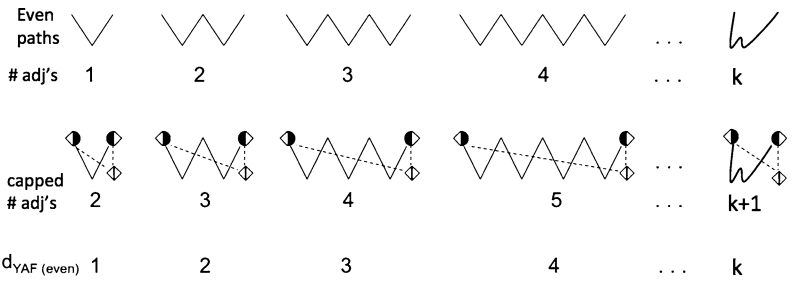


Fig. 10.12 Uncapped and capped even paths and their corresponding DCJ distances (called d_{YAF} in the picture)

10.3.3.2 Distance Contribution from (Closed) Even Paths

At the top of Fig. 10.12 we see a succession of even paths whose telomeric ends start and end in the top genome. Below these are listed the corresponding number of uncapped adjacencies belonging to each path, culminating in a general even path, represented by a "squiggly W", which does not have its adjacencies explicitly enumerated. For such a path involving k genes, there are k adjacencies, as there are $k-1$ "legitimate" adjacencies containing two gene ends, and two telomeric ends counting half an adjacency each.

Below the illustration of the uncapped even paths is a corresponding row with the paths closed by the original closure rule of Sect. 10.3.2 having caps attached in the top genome and a null chromosome in the bottom genome. The caps count as half an adjacency just as the telomeric ends. The two caps in the null chromosome count also as 1, so the amended adjacency total for either genome for each of these closed paths is one more than the previous uncapped version.

The path becomes a $(k+1)$-cycle. Using the DCJ formula for cycles, a closed general even path involving k genes contributes with

$$k+1-1=k$$

to the DCJ distance. The same holds if the even path starts and ends in the bottom genome.

If we consider all even paths, involving N_{even} genes altogether, their contribution to the DCJ distance is, therefore, N_{even}.

10.3.3.3 Distance Contributions from (Closed) Odd Paths

Applying the capping and original closure rules for the odd paths we note that, once capped, an odd path involving k adjacencies becomes a closed $(k+1/2)$-cycle, as shown in Fig. 10.13, since the additional cap adds $1/2$ to the adjacency count. (The final count will still be an integer because odd paths occur in pairs.) Using

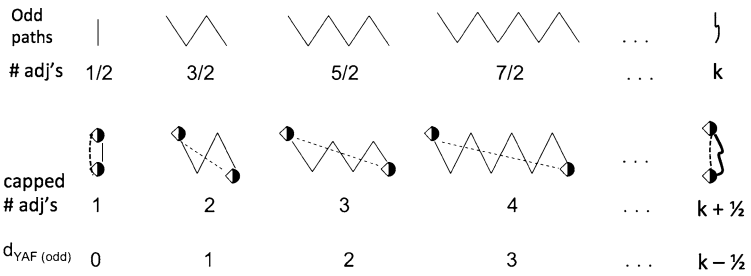


Fig. 10.13 Uncapped and capped odd paths and their corresponding DCJ distances (called d_{YAF} in the picture)

the distance rule for closed cycles which is the number of adjacencies minus 1, this leads to a DCJ distance contribution from this odd path of

$$k - 1/2.$$

Adding over all odd paths, a total contribution of $N_{\mathrm{odd}} - I/2$ follows, where I is the number of odd paths.

10.3.3.4 Adding All Contributions from Cycles, Even and Odd Paths

To make contact with the Bergeron et al. [4] formulation of the DCJ distance, we add all contributions from all components to the distance for a general transformation keeping in mind that each cycle, odd and even path contributes according to what was previously found. Summing on all components we arrive at

$$\begin{aligned} d_{\mathrm{DCJ}}(\pi, \sigma) &= \text{contribution from cycles} + \text{even paths} + \text{odd paths} \\ &= (N_{\mathrm{cycle}} - C) + N_{\mathrm{even}} + (N_{\mathrm{odd}} - I/2) \\ &= N - C - I/2, \end{aligned}$$

because the total number of genes N is equal to $N_{\mathrm{cycle}} + N_{\mathrm{even}} + N_{\mathrm{odd}}$, the number of cycles is C, and the total number of odd paths is I.

10.3.4 Algebraic Distance for Permutation Cycles Using Caps and Nulls

In the previous section we observed that a general transformation contains cycles, and even and odd paths in the Adjacency Graph (AG) as first discussed by Bergeron, Mixtacki and Stoye [4] who introduced the AG. We went on to discuss how each of these components contributes to the DCJ distance, noting that the paths can

be circularized by the addition of caps (for odd paths) and both caps and null chromosomes (for even paths). This circularization procedure using the original closure rules is achieved by the use of "fictitious" elements, but it is still possible to use the formalism for permutations as we did for pure circular genomes. Just as in the strictly circular case, since we have circularized all paths, all cycles will double when we go to the permutation cycles for the composition permutation. In addition, the vertices in the permutations will be twice the number of adjacencies. Hence since the algebraic distance is half of the norm for the permutation cycles, we find that the algebraic distance using capped genomes with nulls is the *same* as the DCJ distance.

10.3.5 Alternative Closure Scheme Bypassing Nulls

Instead of the "original closure rule" we can use an alternative closure scheme which includes caps and closes odd paths in the same way, but, by avoiding nulls, closes even paths by connecting the two cap-ends directly. Unlike the old closure rule, the number of chromosomes is no longer the same in both genomes.

With the new rule we use the number of adjacencies after capping *averaged* over both genomes, remembering that caps contribute $1/2$ an adjacency, as do telomeres. With the new rule, to average the number of adjacencies, we see that when two caps are added to genome π they get "averaged", so there is a resulting contribution of $1/2$ to the total number of adjacencies. The same happens when two caps are added to σ.

With the new closure rule, we introduce a new means of computing a distance, which we will call d_{new}. Since cycles and odd paths do not change their contributions to the total distance, the only difference might possibly come from the even paths. In fact, the new contribution from even paths differs from the DCJ contribution in $-1/2$ for each even path.

Summing over paths, we arrive at the new closure distance:

$$d_{\text{new}}(\pi, \sigma) = N - C - I/2 - E/2 = N - C - P/2,$$

where E is the number of even paths and $P = I + E$ is the total number of paths. But this is a know alternative formula for the algebraic distance [9]. We conclude that the new closure rule yields, in fact, the algebraic distance.

10.3.6 DCJ vs. Algebraic Distances for Capped Genomes

As we saw, there are ways of conceiving and computing the algebraic distance based on a graphical approach, using either the master graph or the adjacency graph. However, there is also a way of computing the DCJ distance using the algebraic approach: using capped genomes and the original closing rule. Indeed, let us recall for

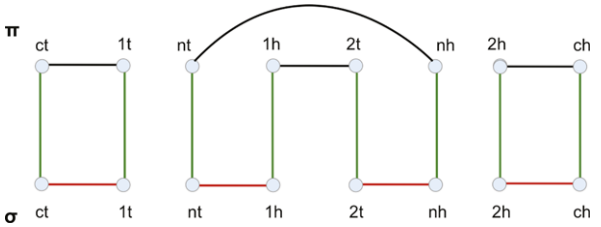


Fig. 10.14 The augmented master graph for genomes $\pi = [1, 2]$ and $\sigma = \{[1], [2]\}$. Caps and nulls were named with subscripts t and h to resemble gene ends. With this, the figure looks exactly like the master graph of $\pi' = \{(1, 2, -c), (n)\}$ and $\sigma' = (1, n, 2, -c)$

a moment the result of the original closure rule on the pair $\pi = [1, 2]$, $\sigma = \{[1], [2]\}$. Figure 10.14 is just the same as Fig. 10.10(c), except that we named the caps with subscripts t and h so that they become more like gene ends.

Doing this, we are able to write the chromosomal expression of the resulting genomes. If we add arrows going from x_t to the corresponding x_h for all entities x, be these genes or dummies, and follow the cycles formed by these arrow plus adjacencies, we end up with two circular genomes, namely, $\pi' = \{(1, 2, -c), (n)\}$ on top and $\sigma' = (1, n, 2, -c)$ in the bottom. We can then compute the algebraic distance between these two genomes, and the result, will be the same as the DCJ distance between these genomes. It turns out that, given a black box that computes algebraic distances, we are able to use it to compute the DCJ distance: just feed it with the capped versions of the genomes, augmented by the original closure rule. This is what the following result states.

Theorem 2 *Let π and σ be multichromosomal genomes over the same gene set, and let π' and σ' be their capped versions, augmented with the original closure rules. Then*

$$d_{\mathrm{DCJ}}(\pi, \sigma) = d_{\mathrm{alg}}\left(\pi', \sigma'\right).$$

Proof By definition, we know that

$$d_{\mathrm{DCJ}}(\pi, \sigma) = d_{\mathrm{DCJ}}\left(\pi', \sigma'\right),$$

since both pairs share the same master graph. Notice that the master graph of π' and σ' does not have to be augmented, because it already consists of cycles only. This means that π' and σ' are circular genomes. But for circular genomes, we know that equality holds between the DCJ and algebraic distances:

$$d_{\mathrm{DCJ}}\left(\pi', \sigma'\right) = d_{\mathrm{alg}}\left(\pi', \sigma'\right),$$

because of Theorem 1. The result then follows from these two equalities. □

The consequence of this result is that properties of the algebraic distance can be, in principle, applied to the DCJ distance via this correspondence. However, one

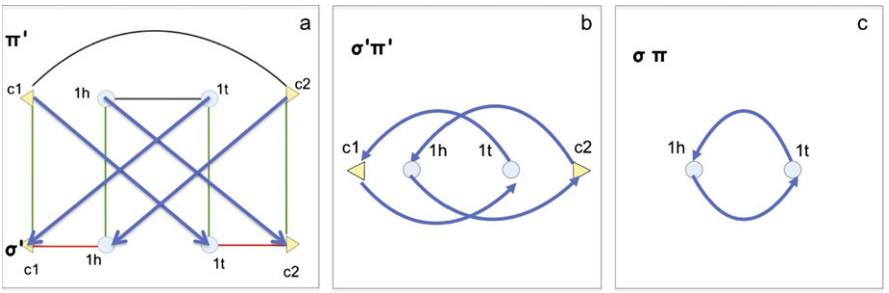


Fig. 10.15 Linearization permutation cycles (PC) . (**a**) *Blue lines* in MG; (**b**) capped PC, (**c**) uncapped PC

possible difficulty here is the fact that π' does not depend solely on π, as the notation seems to imply. It also depends on σ.

10.4 Genome Permutations by the Adjacency Algebraic Theory

In this section we will apply algebraic rearrangement theory in its more recent, adjacency-based formulation to understand the algebraic distance formula and to see how differences with the DCJ distance can arise.

We start by revisiting two examples from the previous section, the linearization example from Fig. 10.9 and the linear fission example from Fig. 10.10. We note they both involve a single cut, which involves the breaking (or cutting) of exactly one adjacency. The master graphs for the two examples are very similar in terms of their both having a 2-cycle except that the master graph (Fig. 10.10(c)) for the linear fission is flanked by two 1-cycles. Since these are identity transformations, they do not contribute to the distance, and so we focus on the 2-cycle as in Fig. 10.9(c). We use the procedure in Sect. 10.2.4 to draw the "blue lines" (Fig. 10.15(a)) and then derive the permutation cycles for the capped (Fig. 10.15(b)) and uncapped (Fig. 10.15(c)) master graphs.

To arrive at the uncapped permutation cycles, we can start from the capped master graph and "identify" the caps, as a device to reconnect the path/cycle without them. We imagine starting at a gene end such as $(1h)$ in the capped master graph and move along on an outgoing blue line; when we arrive at a cap (i.e. $(c1)$) we "jump" immediately to the "identified" cap, (i.e. $(c2)$) and continue along the outgoing blue lines onto gene end $(1h)$, completing the cycle.

Having found the permutation cycles, we note that the 2-cycle in the capped master graph resulted in two 2-cycles in the capped permutation cycles, and only a single 2-cycle in the uncapped case. To derive the permutation cycles for uncapped genomes in a general way, we move on to using the breakpoint graph, although we could do it with other graphs. The virtue of the breakpoint graph here is that its lines end on vertices representing gene ends just like the permutation cycles.

10.4.1 Component-Wise Decomposition of the Adjacency Graph

As mentioned in Sect. 10.1, permutations can be represented by directed graphs that are composed of one or more *cycles*. Every permutation can be written in an essentially unique way as a product of disjoint cycles, apart from the order in which the cycles are multiplied (recall that disjoint cycles commute). This is called the *cycle decomposition* of a permutation. The transformation of a genome π to another genome σ over the same genes is a permutation, and the master graph (MG), adjacency graph (AG), or the breakpoint graph (BPG) of π and σ can graphically represent the components of this transformation. The permutation that transforms π into σ is $\sigma\pi$ in the adjacency algebraic theory—and $\sigma\pi^{-1}$ in the chromosomal algebraic theory, which we will not delve into here.

We discussed the separation of the adjacency graph for a general transformation into its connected components in Sect. 10.3.3, and saw they can be identified visually. As pointed out by Bergeron et al. [4] these components are essentially *cycles* and *paths* in the AG. The same holds true for the breakpoint graph, which will be our main graphical tool in this section. Since separate components have no elements in common, their distance contributions can be resolved separately. We have seen in the section on transformation of circular chromosomes, Sect. 10.2, that every cycle in the MG (and hence in the breakpoint graph) generates two cycles in the permutation $\sigma\pi$. We will show that paths generate a single cycle in $\sigma\pi$ with the same vertices.

10.4.2 The Breakpoint Graph and Permutation Cycles

We are interested in looking at the connected components of the breakpoint graph between genomes π and σ to see how they contribute to the algebraic distance. In terms of components, the master, adjacency, and breakpoint graphs are very much alike: cycles in one graph will correspond to cycles in the other two, perhaps with a different size, but always even. More specifically, a cycle with $4k$ edges in the MG will correspond to a cycle with $2k$ edges in the AG, and to a cycle with $2k$ edges in the BPG. And a path in one graph will always be a path in the other two, again with a difference in size, but not, in general, of the same parity. Specifically, a path with $2k + 1$ edges in the MG will correspond to a path with $k + 1$ edges in the AG, and to a path with k edges in the BPG.

10.4.3 The Algebraic Distance for Paths

To understand more generally how permutation cycles result from paths in the breakpoint graph, we take a look at some generic paths. Instead of going from the construction just described for the previous example in Fig. 10.15, where we found the permutation cycle by starting from the capped master graph and then identified

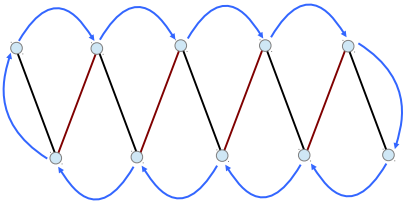


Fig. 10.16 A path in the breakpoint graph that begins and ends with a *black edge*. The breakpoint graph is drawn in zig-zag mode, much like the adjacency graph, to better illustrate the permutation cycle (in *blue*), obtained in general by starting first in π (*black edges*) then σ (*red edges*). A possible pair of genomes that generates such a breakpoint graph is $\pi = \{(1), (2), (3), (4), (5)\}$ and $\sigma = [1, 2, 3, 4, 5]$

caps, we try to find a more direct root by starting with a path in the breakpoint graph and proceeding to add the blue lines to it, as depicted in Fig. 10.16.

To find the permutation cycles associated with a path we compute $\sigma\pi$, that is, apply π first and then σ. We indicate the permutation arrows in blue in Fig. 10.16. They were obtained, from each gene end, following a black edge and then a red edge. If there is no black edge incident to a certain gene end, just follow the red edge to get to the image under $\sigma\pi$. And if, after following a black edge, there is no red edge to continue, just take the result of the first step as the final result. Performing this procedure for every gene end we get the blue arrows indicated in the figure. Notice that they involve all gene ends in the path, "enclosing" the path like a cloud.

Similar results are obtained with any path, not unlike what we have seen already. Paths of any size, starting with either black or red edges, and finishing with either black or red edges, all result in an enclosing blue cycle involving all the gene ends in the path.

We note that this procedure mimics that of Sect. 10.2.4, in tracing out the "blue lines" by walking three steps starting at every vertex, which is a gene end in π. In this method, the "three steps" reduce to two, which are the black and red edges of the breakpoint graph since, in the breakpoint graph, the "green lines" have been contracted out of the picture and are represented by vertex-gene ends.

To see more explicitly how this procedure connects to specific gene ends, we return to the more elaborate approach using the master graph and follow this protocol for all three types of components in the MG, AG, or BPG, that is, for *even* and *odd* paths as well as for *cycles*.

10.4.3.1 Even Paths

Let us consider the telomeres and adjacencies of a general even path in the AG starting and ending in σ:

$$\underbrace{(e1)}_{\sigma}, \underbrace{(e1, e2)}_{\pi}, \underbrace{(e2, e3)}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi}, \underbrace{(e_n)}_{\sigma},$$

where n is an even integer.

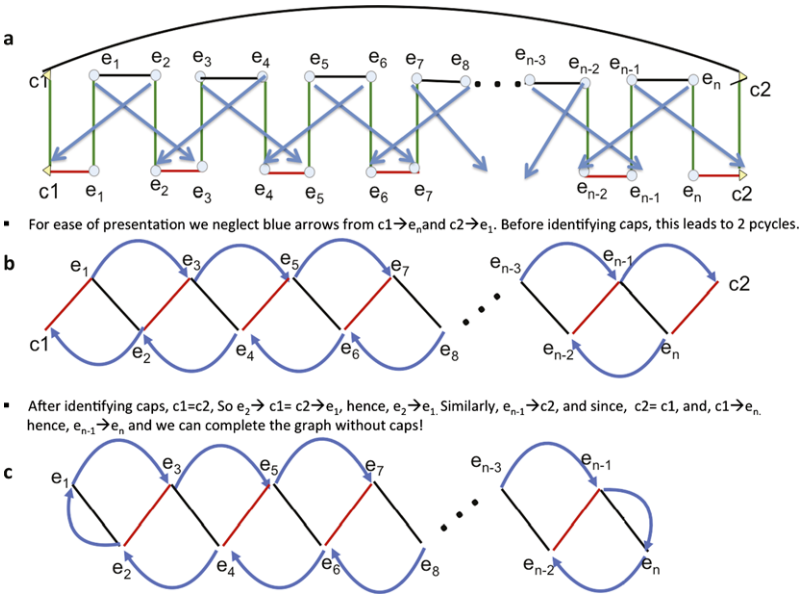


Fig. 10.17 An even path (n is an even integer). (**a**) Master Graph with caps and *blue lines*; π is on *top*. (**b**) Breakpoint Graph and permutation lines with caps. **c** Breakpoint Graph and a *single* permutation n-cycle without caps

Drawing the master graph with caps (Fig. 10.17(a)), we use the procedure in Sect. 10.2.4 for constructing the "blue lines". Contracting the green lines we arrive at the "breakpoint graph" alternating between black and red edges, enveloped by the permutation lines (Fig. 10.17(b)). By identifying the caps at the ends (and eliminating them) we find that the permutation cycles of an even path involving n gene ends (n being therefore an even integer) is a single permutation n-cycle (Fig. 10.17(c)).

The entire construction works just as well if the even path starts and ends in π. Hence, a general "even path" in the MG, AG or BP, (starting and ending in either π or σ), with k adjacencies in one genome in the AG, and $n = 2k$ vertices in the BPG, corresponds to an n-path in the AG and an n-cycle in the permutation $\sigma\pi$. The corresponding contribution to the algebraic distance from the even path is

$$\frac{(n-1)}{2} = \frac{n}{2} - \frac{1}{2}.$$

10.4.3.2 Odds Paths

Similarly, let us consider the telomeres and adjacencies of a general odd path starting in σ and ending in π.

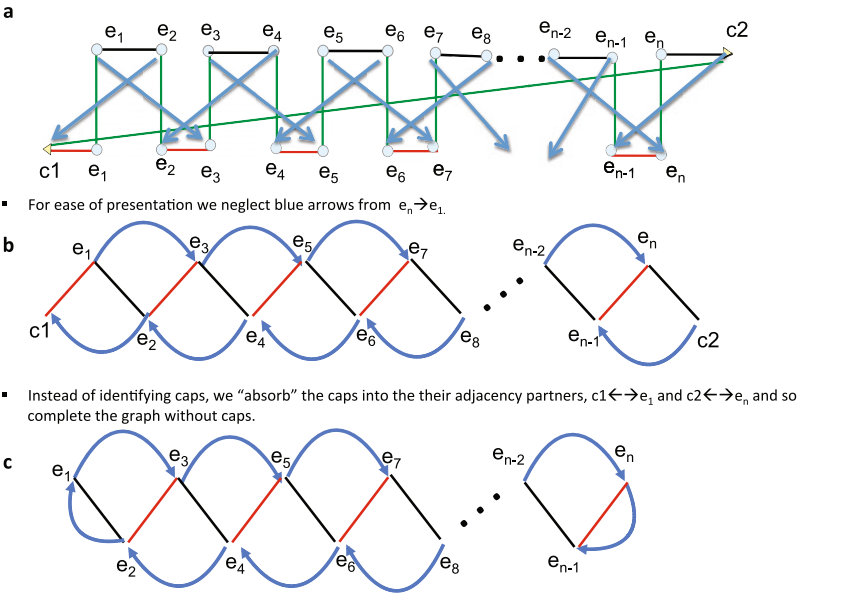


Fig. 10.18 An odd path (n is odd). (**a**) capped MG and *blue lines* (π on *top*). (**b**) capped BPG. (**c**) BPG and n-cycle, no caps

$$\underbrace{(e_1)}_{\sigma}, \underbrace{(e_1, e_2)}_{\pi}, \underbrace{(e_2, e_3)}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\sigma}, \underbrace{(e_n)}_{\pi},$$

where n is an odd integer.

Using a similar construction as for the general even path, we draw the capped master graph (Fig. 10.18(a)) and add blue lines as previously. In Fig. 10.18(b) we show the breakpoint graph containing alternating black and red lines and show the permutation lines with caps. In Fig. 10.18(c) we "absorb" the caps into their adjacency partner. What results is a permutation cycle with n edges, as each of the n vertices has an incoming and outgoing blue permutation line.

Even though this path starts in σ and ends in π, the odd path could "start" in π instead. Hence, for a general "odd path" in the MG, AG or BPG, with n gene ends, and n vertices in the BP, the contribution to the algebraic distance is

$$\frac{(n-1)}{2} = \frac{n}{2} - \frac{1}{2}.$$

10.4.4 The Algebraic Distance for Cycles

We already discussed the algebraic distance for cycles in the section on circular transformations Sect. 10.2. To put our current treatment on par with that for paths, we will also treat cycles using the general framework of the previous sections. Con-

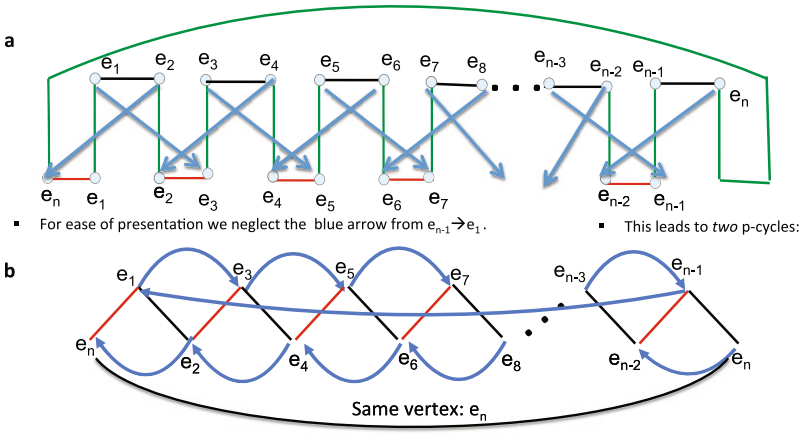


Fig. 10.19 A general cycle (n is even). (**a**) MG with *blue lines* (π on *top*). (**b**) BPG showing an MG cycle becomes *2* permutation cycles

sider a cycle containing n edges (corresponding to n "gene ends") in the adjacency graph of the transformation from π to σ corresponding to a master graph cycle component pictured in Fig. 10.19(a).

$$\underbrace{(e_1, e_2)}_{\pi}, \underbrace{(e_2, e_3)}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi}, \underbrace{(e_n, e_1)}_{\sigma},$$

where n is necessarily even.

As we saw in Sect. 10.2.4, when following the procedure illustrated in Fig. 10.8 to find the composition permutation cycles from the master graph for circular genomes, there is a doubling of cycles which occurs in going from the master graph (MG) to the permutation cycles (PCs). At the same time, the number of vertices in the BPG (equivalent to the number of red and black edges in the AG), is halved in a PC. This is easy to see in Fig. 10.19(b), where it is clear that the splitting of the cycle segregates even and odd numbered adjacencies into two different permutation cycles. We note that $n = 2k$ is even, where k is the number of adjacencies in the cycle in either genome in the AG, and the permutation cycles each contain $n/2$ vertices. To find the contribution to the algebraic distance of a cycle in the AG (BPG, or MG) we find the norm of the corresponding permutation and divide by two. Hence, the contribution is

$$2\frac{(\frac{n}{2} - 1)}{2} = \frac{n}{2} - 1.$$

10.4.5 Total Algebraic Distance in the Adjacency Graph

As we discussed, the adjacencies of the master graph can be segregated separately into *connected components* which contribute independently to the distance. We just

saw how both even and odd paths in the adjacency graph (and also the master graph and breakpoint graph) turn into single cycles in the permutation $\sigma\pi$, whereas cycles in the adjacency graph double in number while halving their vertex counts.

Since we know how to compute the contribution of each individual component, it remains to tally all contributions from components in the adjacency graph. Collecting these, we sum distance contributions over all cycles and even and odd paths. The ith cycle contributes $n_i^{\text{cycle}}/2 - 1$ to the distance, where n_i^{cycle} is the number of gene ends in the ith cycle. As for the paths, as we found, each path (even or odd) contributes $n_j^{\text{path}}/2 - 1/2$, where n_j^{path} is the number of gene ends in the jth path. Summing over all components we get

$$\begin{aligned} d_{\text{alg}}(\pi, \sigma) &= \text{ cycle contribution } + \text{ path contribution} \\ &= \frac{n_{\text{cycles}}}{2} - C + \frac{n_{\text{paths}}}{2} - \frac{P}{2} \\ &= N - C - \frac{P}{2} \end{aligned}$$

where $n_{\text{cycles}} = \sum n_i^{\text{cycle}}$ and $C = \sum_i 1$ are summed over cycles, and $n_{\text{paths}} = \sum n_j^{\text{path}}$, is summed over paths. Notice that $n_{\text{cycles}} + n_{\text{path}} = 2N$, because each gene corresponds to two gene ends. Since both even and odd cycles have the same formal contribution in this approach, we do not have to separate them in the sum over paths.

10.4.6 DCJ Distance vs. Algebraic Distance

The difference between the DCJ distance and the (Adjacency) Algebraic distance boils down to the difference in the distance for even paths. For the DCJ distance the contribution from even and odd paths is asymmetric, whereas, as we just saw, for the algebraic method both contribute in the same way. Since both formulas can be stated simply in terms of the number of adjacencies and the number of odd and even paths, we can easily compute the difference. The distance $d_{\text{DCJ}} = N - C - I/2$, where, as we recall, I is the number of *odd* paths, while $d_{\text{alg}} = N - C - P/2$ where $P = I + E$ is the sum over the number of odd *and* even paths. The difference, is therefore just $E/2$ which is just half the number of even paths!

Since we can say even paths occur whenever the number of chromosomes changes, or the type of chromosome changes, either from linear to circular or vice versa, these transformations may occur relatively rarely in genome transformations, in which case the two distances will appear to be nearly the same. We next re-examine some previous examples to compare similarities and differences between methods.

So, for example, operations considered in Sect. 10.2 which dealt with circular transformations were operations such as (internal) inversions, and the creation and absorption of circular chromosomes (from other circular chromosomes) have only cycles in their adjacency graph, and, as we have seen, cycles are treated in the same way by both methods.

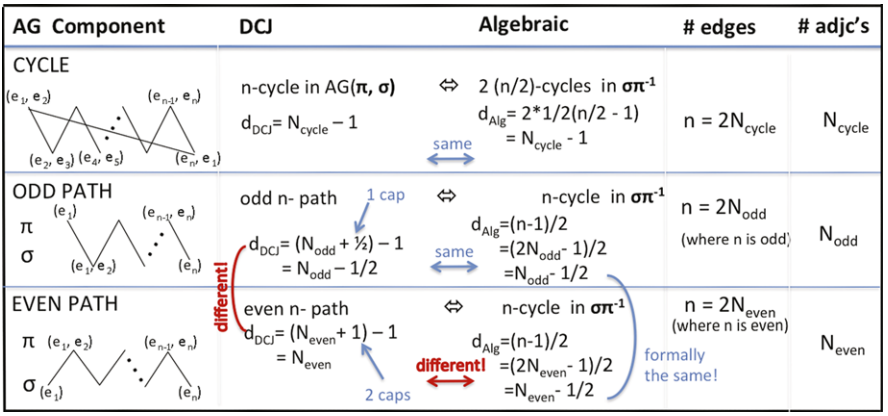


Fig. 10.20 Component-wise summary of DCJ vs. algebraic distances

We also know that differences can arise with the use of caps, and in the section involving such general transformations which include caps (Sect. 10.3), we discussed two transformations in particular where this may be the case, in particular, that of a linearization involving a circular transforming to a linear chromosome, and a linear fission of a single linear chromosome. Not only did these transformations involve caps, with the DCJ approach, but their "bare" uncapped master graphs contained even paths, signaling there is likely to be a discrepancy. In fact, since both transformations contain exactly one even path, we know there is a difference: the DCJ distance is 1 for both the linearization and the linear fission, whereas the algebraic approach yields a distance of $1/2$.

Even though these are rather simple examples, there is more that can be gleaned from them. Not only can we compare the forward transformations, but the inverse transformations as well. And so we can also deduce that for the DCJ, a *circularization* of a linear chromosome, or a *fusion* of two linear chromosomes into one also costs a single DCJ, whereas the algebraic method yields again, a distance of $1/2$ for each. Finally, even if there are more "bystander" genes which are not actually involved in the transformation, these operations will still cost the same.

We conclude by summarizing the similarities and differences of the two methods for cycles, and odd and even paths in Fig. 10.20. This table shows the different contributions of each kind of component, with emphasis on cases where DCJ and algebraic distances differ, namely, the even paths. We examine these differences and the implications in further detail in the next section.

10.5 Weights, Operations, and Biology

In this section we discuss the biological implications of the two approaches particularly by looking at the DCJ and algebraic formulations as methods which minimize the number of weighted operations needed to transform one genome into the other.

Table 10.2 A complete list of DCJ operations

Operation	Example
Linear translocation (incl. fission/fusion)	$[1, 2], [3, 4] \mapsto [1, 4], [3, 2]$
Linear reversal	$[1, 2, 3] \mapsto [1, -2, 3]$
Creation of circular from linear (including circularization)	$[1, 2, 3] \mapsto [1, 3], (2)$
Absorption of circular by linear (incl. linearization)	Inverse of previous
Circular reversal	$(1, 2, 3) \mapsto (1, -2, 3)$
Circular fission/fusion	$(1, 2) \mapsto (1), (2)$

10.5.1 The Relative Weights of Operations

Part of the challenge of modeling genome rearrangements is to find a successful strategy for dealing with the relative weights of operations. One of the original dilemmas for the DCJ was the relative weight of a transposition or a block interchange to an inversion. Our examination of the underpinnings of the DCJ and algebraic methods, has brought to light an interesting challenge: the relative weight of operations such as fissions, fusions, linearizations or circularizations vs. inversion. In comparing these methods we saw that ultimately, the determination of this relative weight comes down to the underlying assumptions in the method. If caps are used, the ratio is 1 and when they are not it depends on the method.

10.5.2 Comparing Weights for DCJ vs. Algebraic Method

Yancopoulos and others [20] describe a complete list of operations characterized as DCJ. Table 10.2 shows the list adapted from their 2005 paper. We use "circular" in places where they use "circular intermediate" because here we study the general case, where the start and target genomes can have both types of chromosomes, linear and circular. Therefore, we do not necessarily think of circular chromosomes as intermediate. Each of these operations has a weight of 1 DCJ.

The same operations are available in the algebraic approach, but the weights are different. Table 10.3 shows the same operations with weights in both the DCJ scheme and the algebraic scheme. Notice that we have to split some operation classes because not all members have the same algebraic weight. In general, operations involving null chromosomes are weighted half a DCJ in the algebraic scheme.

10.5.3 Fact or Artifact: Fictitious Objects and Dummy Elements

We have discussed how it is possible to use caps and nulls and other such "fictitious" objects in order to close paths in the adjacency graph so as to use the formalism developed for cycles. We have shown how it is possible to arrive at the transformation

Table 10.3 A comparison of weights in the DCJ and algebraic schemes. Operations described as 'proper' may not involve null chromosomes

Operation	DCJ weight	Algebraic weight	Example
Linear translocation, proper	1	1	$[1, 2], [3] \mapsto [1], [3, 2]$
Linear fission/fusion	1	1/2	$[1, 2] \mapsto [1], [2]$
Linear reversal	1	1	$[1, 2, 3] \mapsto [1, -2, 3]$
Creation of circular from linear, proper	1	1	$[1, 2, 3] \mapsto [1, 3], (2)$
Circularization of linear	1	1/2	$[1] \mapsto (1)$
Absorption of circular by linear, proper	1	1	$[1, 2], (3) \mapsto [1, 3, 2]$
Linearization of circular	1	1/2	$(1) \mapsto [1]$
Circular Reversal	1	1	$(1, 2, 3) \mapsto (1, -2, 3)$
Circular fission/fusion	1	1	$(1, 2) \mapsto (1), (2)$

distance, using capped versions of genomes. As caps and nulls are not "real" it may be thought that somehow these should not "count" or "weigh in". Hence, some might think that the algebraic method, which does not use them, may seem more legitimate. Others may argue that, though these dummy elements are strictly a device, they aid in simplifying the distance equations.

10.5.4 Fictitious Operations and the "Basic" DCJ

We have made no bones about discussing the artificiality of caps; having given them the "power" of appearances, it remains to discuss the consequences in terms of the kinds of operations that result in the "basic" DCJ.

Ultimately, the double cut and join is a series of four more "elemental" individual *cut* and *join* operations, two of each. When the "single" DCJ operation is performed between two "real" adjacencies, then two "actual cuts" and two "actual joins" happen between "real" gene ends.

With the use of caps in the "basic" DCJ, concomitant *fictitious operations* can arise, which have no ultimate "reality". These fictitious operations involve any type of cut or join involving a cap. There are two possible adjacencies involving caps in the basic DCJ schema: either an adjacency between two caps which is a null chromosome, or an adjacency between a cap and single gene end, also known as a *telomere*. We define a fictitious operation as any operation involving either a cut or a join in an adjacency containing at least one cap.

To understand this in the context of an example, consider the case of the linearization example discussed in Sect. 10.3 using the capped master graph. Just as with any DCJ, there are essentially two cuts and two joins, but now, some of these involve caps. The null chromosome contains an adjacency which is severed as well as the adjacency between the two gene ends in the circular chromosome ($1t$ and $1h$). So one of the cuts performed in this DCJ is fictitious, and the other is real. After these

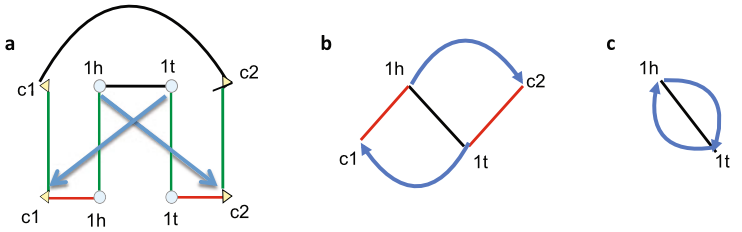


Fig. 10.21 Linearization in BPG representation. (**a**) MG with *blue lines*. (**b**) BPG with capped PC. (**c**) uncapped PC

adjacencies are broken, the gene ends are reconnected with caps, and both of these operations are fictitious. The total operation gets a DCJ "weight" of 1.

One might be tempted to infer that each individual *cut* and *join* is $1/4$, after all, there are four operations and the whole operation has a unit weight. If we do this, then in the linearization example the only "real" operation is the cut between $1t$ and $1h$; the remaining three operations (the cut of the null chromosome and the joining of the two caps) are fictitious, so the "true cost" of this operation should only be $1/4$!

In fact, Feijao and Meidanis [8] considered such a possible scheme with their "SCJ" operation such that any single cut or single join is weighted as little as $1/4$.

10.5.5 Fictitious Operations and the Algebraic Approach

What about fictitious operations for examples like these? First let us start with the algebraic distance for these examples. As we know and saw in Fig. 10.15(c), or here in Fig. 10.21, the permutation cycles for linearization consist of a single 2-cycle, the norm of which is $n - 1 = 2 - 1 = 1$; dividing by two, the distance is $1/2$. We have reached the *irreducible quantum* operation in the algebraic approach in a single fission/fusion/linearization. As we can see from Fig. 10.21(c), the only "step down" from here which is a PC "cloud" surrounding a *single* breakpoint, or adjacency, is a single $n = 1$-cycle with a single gene end. But this last has *no* distance or cost. Hence, in the algebraic schema there is no "step down" below this operation which has a cost. The lowest we can go is to a cost of $1/2$ which straddles between the DCJ and the SCJ. This leaves us with more questions than answers. Had the cost of the operation been $1/4$ of a full DCJ we might not be as perplexed, since it would seem the "fictitious" operations no longer "weigh in".

10.5.6 Does the BMS- DCJ Do Away with Fictitious Operations?

The beautiful formalism developed by Bergeron et al. [4] in their approach to the DCJ, including the adjacency graph and the idea of odd and even paths, seems to

liberate the DCJ from the clumsy use of caps and nulls. The beauty of the new approach was welcomed by many because of its elegant formulation, ease of calculation, and also because at least for some who may have been bothered by them, it seemed to do away with caps, and the issues that come with them; after all, caps are not only a bother, but they are an artificial construction.

Interestingly, even though the DCJ approach as formulated by Bergeron et al. [4] does not *appear* to have caps and nulls explicitly, nevertheless, their version of the distance *agrees in value* with that of the "basic" DCJ distance computed with the use of caps and nulls. The weighting scheme of Bergeron et al. is also identical to that of the DCJ with caps and nulls.

The fact that the algebraic distance *agrees* with the DCJ distance (BMS [4] or YAF [20]) when caps and nulls are used, but *differs* when they are not, begs the question: does the BMS approach somehow retain vestiges of fictitious operations?

10.5.7 Biological Interpretation

The difference in weights between the DCJ and algebraic schemes poses interesting questions. For instance, is it biologically meaningful to give less weight to operations involving null chromosomes? On the one hand, this may make sense because apparently less "modification effort" is needed to effect, say, a linear fission than a linear translocation: in the first case, a single "cut" occurs, while in the second, there are actually two cuts and two rejoins. On the other hand, the creation of a new chromosome is no easy task biologically. A chromosome is not just genes. It needs, to begin with, a duplication apparatus, which includes a centromere or at least an origin of replication. Simplified genome modeling does not take these into account.

We must remember that a mutation has to be accepted by the environment to survive as such. Therefore, a lasting rearrangement must be "easy" with respect to modification effort, but also "stable" in terms of genome structure, and these two components should contribute to its weight.

To further illustrate this discussion, consider the following scenario: $\pi = [1, 2]$ and $\sigma = [2, 1]$. This is a linear transposition, and the DCJ distance between these two genomes is 2. However, the algebraic distance is just 1! How come? Because the algebraic method finds a shorter path of going from π to σ: linear fission of π, giving [1], [2], which is the same as [2], [1], and then linear fusion of these two chromosomes yielding σ. Since a linear fission or fusion is worth $1/2$, the total comes to 1. Notice that this is only possible because the two blocks being interchanged comprise the entire chromosome.

One could argue that the algebraic path is way out of scope, since it involves both the creation and the destruction of a chromosome, which are supposedly expensive biological operations. However, one could also argue that this argument holds only if you assume that these two changes occurred separated by millions of years of evolution. What if they occurred in the *same* cell division?

It is hard to say what is correct or not in modeling biological processes. It may be the case that in some situations one approach is more suitable, while in other

situations the opposite is true. Experimental use of the distances will help us to shed more light into this issue.

10.5.8 Implications and Concluding Remarks

The fact that the algebraic method leads to a *different* weighting scheme than the *standard* DCJ not only lends a new approach to calculating the genomic distance, but offers the possibility of addressing issues having to do with the use of "dummy" elements such as caps and nulls. It is intriguing that we have found that by use of these elements in the algebraic approach, the distance agrees with the *standard* DCJ, but bypassing their use we arrive at a *different* distance and weighting scheme for the operations, primarily those involving even paths in the adjacency graph, and ultimately fissions and fusions in the transformation

By comparing the two formulations we have come to realize the consequences involving the assumptions behind the two methods. Ultimately we feel that neither method is "superior" in that either weighting scheme may be considered valuable under some circumstances and the possible use of either of these two methods increases the number of available options in analyzing genome transformations.

Acknowledgements We would like to thank Richard Friedberg for a long, inspiring conversation over dinner at an Indian restaurant in New York, on February 24th, 2013. We thank Pedro Feijao and Marilia Braga for shorter, but also inspiring, conversations. S.Y. thanks Nick Chiorazzi for his tolerance and support, David Sankoff for introducing Joao to me, and Judith Ficksman for her heartfelt encouragement. Finally, we thank our reviewer for meticulous and insightful comments.

References

1. Bader, M., Ohlebusch, E.: Sorting by weighted reversals, transpositions, and inverted transpositions. J. Comput. Biol. **14**, 615–636 (2007)
2. Bafna, V., Pevzner, P.: Sorting by transpositions. SIAM J. Discrete Math. 224–240 (1998)
3. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. In: Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science, vol. 46, pp. 148–157. IEEE Press, New York (1993)
4. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Moret, B. (ed.) Algorithms in Bioinformatics Proceedings of WABI 2006 (2006)
5. Blanchette, M., Kunisawa, T., Sankoff, D.: Parametric analysis of genome rearrangement. Gene **172**, 11–17 (1996)
6. Christie, D.: Sorting permutations by block interchanges. Inf. Process. Lett. **60**, 165–169 (1996)
7. Dobzhansky, T., Sturtevant, A.H.: Inversions in the chromosomes of *Drosophila pseudoobscura*. Genetics **23**, 28–64 (1984)
8. Feijao, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–1329 (2011). doi:10.1109/TCBB.2011.34.

9. Feijao, P., Meidanis, J.: Extending the algebraic formalism for genome rearrangements to include linear chromosomes. IEEE/ACM Trans. Comput. Biol. Bioinform. **99**(PrePrints), 1 (2012). doi:10.1109/TCBB.2012.161
10. Friedberg, R., Darling, A.E., Yancopoulos, S.: Genome rearrangement by the double cut and join operation. In: Keith, J.M. (ed.) Bioinformatics, Methods in Molecular Biology, vol. 452, pp. 385–416. Humana Press, Clifton (2008)
11. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). J. ACM **46**(1), 1–27 (1999). Previously appeared at Proc. of the 27th Annual Symposium on the Theory of Computing (STOC 95), Las Vegas, Nevada, pp. 178–189 (1995)
12. Hannenhalli, S., Pevzner, P.A.: Transforming mice into men (polynomial algorithm for genomic distance problem). In: Proc. of the 36 Annual Symposium on Foundations of Computer Science (FOCS 95), Milwaukee, Wisconsin, pp. 581–592 (1995)
13. Meidanis, J., Dias, Z.: An alternative algebraic formalism for genome rearrangements. In: Sankoff, D., Nadeau, J. (eds.) Comparative Genomics, pp. 213–223. Kluwer Academic, Dordrecht (2000)
14. Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. USA **81**, 814–818 (1984)
15. Sankoff, D.: Edit distance for genome comparison based on non-local operations. In: Combinatorial Pattern Matching, Third Annual Symposium. Lecture Notes in Computer Science, vol. 644, pp. 121–135. Springer, Berlin (1992)
16. Sankoff, D., Goldstein, M.: Probabilistic models for genome shuffling. Bull. Math. Biol. **51**, 117–124 (1989)
17. Sankoff, D., Cedergren, R., Abel, Y.: Genome divergence through gene rearrangement. Methods Enzymol. **183**, 428–438 (1990)
18. Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B.F., Gene, C.R.: Order comparisons for phylogenetic inference: evolution of the mitochondrial genome. Proc. Natl. Acad. Sci. USA **89**, 6575–6579 (1992)
19. Tesler, G.: Efficient algorithms for multichromosomal genome rearrangements. J. Comput. Syst. Sci. **65**, 587–609 (2002)
20. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, in version and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)

Part III
Promising Directions

Chapter 11
Fractionation, Rearrangement, Consolidation, Reconstruction

David Sankoff and Chunfang Zheng

Abstract The reconstruction of ancestral gene orders based on models of chromosomal rearrangement mechanisms is complicated when some of the input genomes have undergone whole genome duplications followed by fractionation, the massive loss of some or most of the duplicate genes. We describe a reconstruction protocol that uses maximum weight matching in two phases to overcome the fragmented nature of results based on gene adjacency only. We review consolidation methods for recovering synteny patterns from fractionated genomes, and show how to integrate these into the reconstruction protocol. The procedure is applied to reconstruct the common ancestral gene order of grape and poplar. Simulation of the evolution of comparable genomes reveals the narrow ranges within which the rearrangement and fractionation parameters must be set in order to emulate statistical attributes of the extant genomes.

11.1 Introduction

All methods of reconstructing the details of ancestral gene order from a number of extant genomes are based on common gene adjacencies in these genomes, e.g., [1–4], though they all build on these fundamental data in different ways. As evolution progresses *rearrangement* events, notably inversion and reciprocal translocation, successively disrupt gene adjacencies in individual genomes. In addition, more local events such as gene transposition from one site to another, gene deletion and gene duplication also disrupt some adjacencies and establish others. To the extent that many common adjacencies remain unperturbed by these processes in two or more of the extant genomes, they may contain enough evolutionary signal to allow reconstruction of significant portions of the ancestral order. As evolution continues, rearrangement can eventually degrade this signal so that only relatively short fragments—"contigs"—of the ancestral order can be inferred. Reconstruction methods need to transcend their dependence on gene adjacencies if longer range gene orders are required.

D. Sankoff (✉) · C. Zheng
University of Ottawa, Ottawa, ON, Canada
e-mail: sankoff@uottawa.ca

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_11,

A complication arises if one or more of the extant genomes derive from whole genome duplication (WGD) events that occurred since their common ancestor. The duplication itself does not add any new adjacencies or remove any; the pre-existing adjacencies simply continue, but with multiplicity two. What complicates things after WGD is duplicate gene loss on a massive scale, deleting one or the other, but not both, of most duplicate gene pairs, a process called *fractionation* [5]. The loss of an individual gene y from context $\ldots xyz \ldots$ will generally destroy two adjacencies xy and yz and create a new one xz. This is true whether the loss of the gene is a physical deletion of part of the chromosome or by pseudogenization. Even if xy and yz still exist in the homeologous region of the genome, the adjacency xz is an innovation.

WGD and fractionation are particularly prevalent in flowering plants [6], where the slow (tens or hundreds of millions of years) cycle of the two processes also involves the constant excision of excess non-coding DNA, a characteristic of genome dynamics that distinguishes these organisms from other evolutionary domains, such as the mammals.

It is misleading to compare fractionated genomes with non-WGD relatives in terms of rearrangements only, because these yield systematically exaggerated results: the algorithms are forced to account for the missing and the new adjacencies as if they were breakpoints of (non-existent) inversions and translocations. And although there have long been methods for incorporating gene loss into genome rearrangement algorithms [7], these are not designed for the specific scenario of WGD followed by fractionation.

In this paper, we first detail our *scaffolding* approach [4] to overcoming the "short contigs" limitations of adjacency-based reconstruction (Sect. 11.2). After an explanation of the notion of *excess adjacencies* in Sect. 11.3, we briefly review WGD and fractionation (Sect. 11.4). In Sect. 11.5 we then present an improved *consolidation interval* strategy [8, 9] for accounting for fractionation in a descendant of a WGD event. We can then reconstruct an ancestral gene order for the pre-WGD genome from two or more of its direct descendants, where the genes in these descendant genomes are replaced by consolidated intervals. Finally, the genes inside these intervals are sorted.

We illustrate in Sect. 11.6 using the two plant genomes, from poplar (a WGD descendant) and grapevine (no recent WGD history), to reconstruct their common "rosid" ancestor, with and without taking into account fractionation. We find that the consolidation step removes virtually all the artifactual rearrangements inferred when fractionation is ignored.

From this reconstruction, we can calculate the total amount of rearrangement from the ancestor to the two extant genomes, how much this would be inflated by ignoring fractionation history, the distribution of sizes of consolidation intervals, and the fractionation bias—to what extent are genes deleted in an asymmetric manner from the two copies of the genome emerging from WGD.

In Sect. 11.7, we set up a simulation of the gene order evolution of poplar and grapevine from the ancestral rosid, with the same numbers of genes and chromosomes, and the same number of single-copy genes (= number of gene losses) in the

simulated poplar genome. We experimented with parameters reflecting the numbers of rearrangement operations, what proportion of these are short inversions, how many genes are deleted at a time, and how probable a deletion is to affect one or the other of the original two copies of the poplar genome. We determined the unique set of evolutionary parameter values producing the observed values in our analysis of the real plant genomes.

11.2 Reconstruction

Our reconstruction method requires preprocessing annotated genomic data by a syntenic block detection program such as SynMap in the CoGe platform [10, 11] to identify likely orthologous genes in all pairs of the genomes under study, as well as paralogs in self-comparison of each descendant of recent WGD. We then process the combined set of all these orthologies and paralogies with the OMG! procedure [12] to produce homology sets containing at most N paralogous versions of each gene in each $2N$-ploid, including at most one gene in each diploid. We also impose some more or less stringent condition such as at least two genes from different genomes in each set. These homology sets represent candidate genes for the reconstructed ancestral genome. The use of stringent criteria in SynMap and OMG! ensure that each set can be mirrored by only one gene in the eventual reconstruction. Though this lends confidence to the reconstruction of the particular gene and its position in the gene order, it does exclude the possibility of assigning additional genes, even in a tentative way.

Once we have the set of relevant genes and all the homology relations we reconstruct the ancestral order using Maximum Weight Matching (MWM) [13] at two levels. First we identify all the gene adjacencies (considering only the genes within the data set as constructed) in all the genomes and subgenomes, each homology set determining two vertices of a graph G_1, corresponding to the 5' and 3' ends of the genes involved. We weight each adjacency—an edge in G_1—according to how many times homologs of the two genes involved are adjacent, with that particular 5'–3' orientation, in the data, possibly taking into account phylogenetic or data quality considerations, depending on the particular biological problem being analyzed. The MWM then chooses an optimal subset of adjacencies. This gives a set of ancestral "contigs". A small number of these may be circular; we linearize each of these by discarding their lowest weight adjacency—this has a minuscule effect on the total weight of the matching.

For the second application of MWM, we use the contigs as vertices in a graph G_2. Each contig has a mean position (as measured in gene order position) on a chromosome in one or more of the input genomes or subgenomes. These positions order the contigs on chromosomes. The few ambiguous contigs, i.e., containing large proportions of genes originating in two or more chromosomes in the same genome or subgenome are discarded. In addition, to ensure a level of syntenic robustness, if a

contig does not have a minimum number of genes in at least one genome, we discard it. Thresholds are set so that losing these small contigs plus the ambiguous ones satisfies a trade-off between accuracy and gene inclusiveness.

Two successive contigs on a chromosome are considered adjacent, and are joined by an edge in G_2 for the purposes of the second MWM. The orientation of a contig on a chromosome is determined by whether the genes it contains are in largely increasing or decreasing gene order on the subgenome in question. The weights may be the same as in the first MWM, or may be different. The output from this algorithm is a set of "scaffolds", namely a series of contigs alternating with gaps, each corresponding to a chromosome or a fragment of a chromosome in the ancestral genome.

Linearizing circular scaffolds turns out to be a quantitatively more important problem than with contigs, Nevertheless, we have found that shifting the adjacency criterion in mid-analysis from gene adjacency to contig proximity is an effective way of transcending the short contig limitation of ancestral reconstruction [4].

Though the principle of MWM has been used for ancestral genome reconstruction in a variety of theoretical contexts [14–16], it is also well suited to practical problem of scaling up from the gene adjacency-based problem of constructing contigs to the contig-based problem of constructing scaffolds. And as we shall see, since we are using statistics on gene adjacencies in evaluating reconstructions, an MWM approach feeds naturally into this step.

11.3 Excess Adjacencies as a Measure of Rearrangement

Independent rearrangements (inversions and reciprocal translocations) in newly diverging sister species with n genes tends to increase the total number of different adjacencies in the two genomes linearly at an initial rate of $2r$, where r is the total number of rearrangements in the two genomes ($r/2$ in each). At the same time, the number of adjacencies in common *decreases* at the same rate. If one of the genomes undergoes WGD soon after speciation or as part of speciation, the total number of different adjacencies in the two genomes still increases at a rate of $2r$. (The number of common adjacencies only decreases at a rate of r since rearrangement changes in the WGD descendant only affects one of the two identical adjacencies, leaving the other intact, so that only rearrangements in the other genome decreases the number of common adjacencies.) In either case—whether or not one of the genomes is a WGD descendant—the total number of different adjacencies in the two genomes, in excess of n, is an accurate measure of the degree of evolutionary divergence [8].

The advantage of this way of measuring evolutionary divergence over edit distances based on a repertoire of rearrangement operations, and over breakpoint distances, is that it applies equally well to comparing genomes with one or more WGD in their recent history as to those with no such history, and that it requires no special extension, constraint or modification wherever it is applied.

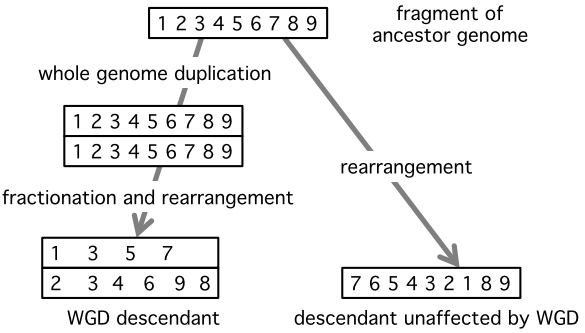


Fig. 11.1 Fractionation leading to different adjacencies in WGD descendant and unaffected genome. The adjacencies between genes 1 and 3, 3 and 5, 5 and 7 as well as 4 and 6 in the WGD descendant are caused by fractionation. The adjacency between 1 and 8 in the unaffected genome is caused by a reversal rearrangement, and the adjacency between genes 6 and 9 in the WGD descendant is caused by deletion of 7 and a rearrangement. Only two of the adjacencies are caused by rearrangement, but ignoring fractionation would lead to the inference of at least three more rearrangements to account for the different sets of adjacencies in the two genomes

11.4 Whole Genome Duplication and Fractionation

During fractionation, gene adjacency disruption follows from the random choice of which of the two copies is deleted, i.e., which copy of a chromosome retains the remaining single copy of the gene. This was first made explicit by Wolfe and Shields [17] in their original demonstration of "reciprocal gene loss" following the ancient WGD of *Saccharomyces cerevisiae*: "...this is the result of random deletion of individual duplicated genes from one or other chromosome subsequent to the initial duplication of the whole region." The pattern was further detailed later by the comparison of the *S. cerevisiae* gene order with that of related diploid yeasts [18, 19], where it was called"interleaving", while Freeling [5] coined the term "fractionation" in the context of plant genomics. Gordon et al. [20] and more recently, Ouangraoua et al. [21], have termed it "double synteny".

The phylogenomic extent of fractionation and the formal treatment of the deletion process have been the subject of numerous papers [22–26].

When a run of adjacent duplicate pairs lose a subset of their redundant genes from one chromosome and another, disjoint, subset from the other copy, as in Fig. 11.1, inference of the rearrangement distances between the WGD descendant and an unduplicated sister genome necessarily suggests that there are rearrangement breakpoints where adjacency no longer exists between the two subsets of single-copy survivors. This exaggerates the inferred number of reciprocal translocations and artificially inflates the overall amount of chromosomal rearrangement inferred between the two sister genomes.

We can correct this through the identification and isolation of "fractionation intervals", regions in both the WGD descendant and its unduplicated sister genome that have become partly or entirely single-copy in the former and may or may not

have been rearranged internally, but have (so far) been unaffected in both genomes by rearrangements exchanging genes from within the interval and genes external to the interval. The statistical properties of the intervals bear on current topics of interest in plant evolutionary genomics, whether duplicated genes are silenced or deleted one by one or through the deletion of longer stretches of DNA [22–24] and whether a fractionation regions tends to lose genes largely from one of the homeologous chromosomal segments or equally from the two [27].

11.5 Consolidation

Ideas about combining the information from the two fractionated regions in reconstructing ancestral genomes may be found in [5] for plants, in [20] for yeast and, more formally, in [21] for ancient vertebrates.

We have been developing a series of *consolidation algorithms* to identify and handle all instances of fractionation in a WGD descendant. The first of these [8] focuses on detecting and accounting for pairs of regions of single-copy genes in the WGD descendant that contain no genes in common (since the genes concerned are single-copy) but whose combined (or *consolidated* [5]) gene content is exactly the same as some contiguous region in a related genome unaffected by the WGD. A recent improvement in collaboration with Katharina Jahn and Jakub Kováč has linear run time, allows duplicate genes to be shared by the two intervals in the WGD descendant, and also extends the analysis to whole genome triplication and higher polyploidies [9]. Current work by Jahn solves the more difficult problem of comparing two fractionated sister genomes while dispensing with any necessity of referencing an unduplicated genome.

In the WGD case, once the pairs of regions or intervals are identified, together with the corresponding interval in the unduplicated genome, all three regions are replaced by a new, labeled, *virtual* gene.

The two genomes, thus altered by the creation of virtual genes replacing fractionated regions, are then examined for excess adjacencies and compared with the corresponding quantity in the untreated genome.

When one of the two intervals in the WGD descendant is empty because of completely biased fractionation, the corresponding virtual gene is still replaced in the appropriate context, deduced by examining the contexts of the other copy of the virtual gene in the WGD descendent and in the unaffected sister genome.

The consolidation algorithm treats the fractionation intervals as identical units, two in the WGD descendant and one in the unaffected genome. In this way it accounts for rearrangements which includes a whole interval in its scope, but also rearrangements which disrupt an interval, in that a fractionation involving such an interval will generally be automatically counted as two intervals, resulting in two virtual units instead of one. What the consolidation algorithm does not account for, however, are rearrangements occurring completely *within* one of the fractionation intervals.

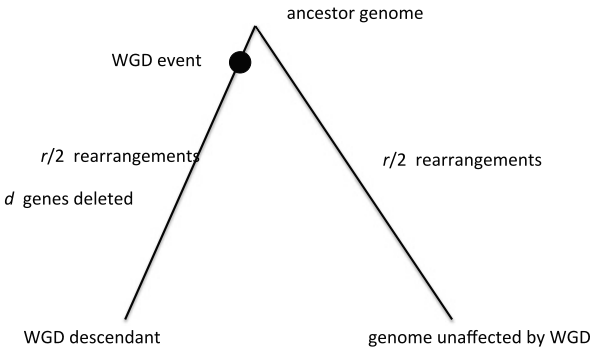


Fig. 11.2 Evolutionary scheme for simulating the production of excess adjacencies in a WGD descendant and an unaffected sister genome, by rearrangement and fractionation in the former, and rearrangement only in the latter. Ancestor contained 24,000 genes, divided among 20 chromosomes. Simulations carried out with number of random rearrangements (10 % reciprocal translocations, 90 % inversions) $r = 0, 600, 1800, 3000, 4200$ and random deletions $d = 0, 3000, \ldots, 21{,}000$

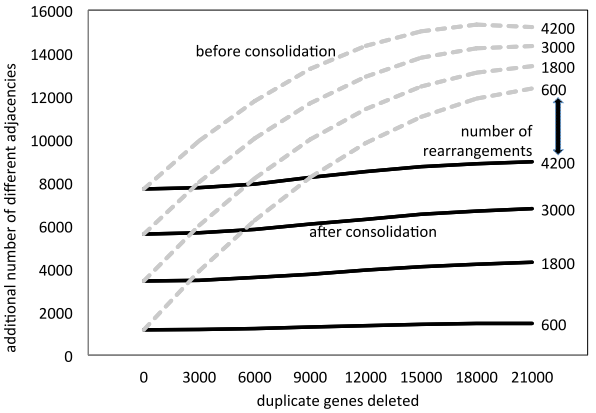


Fig. 11.3 Simulated effect of fractionation on increasing number of excess adjacencies before consolidation (*dashed lines*) and after (*solid lines*) for two genomes, one having undergone a WGD and one unaffected, diverging by various amounts of rearrangement

To correct for this, within each fractionation region, we first consider all the adjacencies in the three component intervals. We find an order for all the genes in the interval in the reconstructed ancestor genome, such that the following condition is satisfied. The induced sub-order determined by the subset of the genes from each of the three intervals in the extant genomes, two in the WGD descendant and one in the unaffected genome, has a minimum number of excess adjacencies when compared with the extant order, summed over all three intervals. We add the number of these interval-internal adjacencies to the set of adjacencies produced by the consolidation algorithm.

Figure 11.2 shows a scheme for simulating fractionation process following a WGD event. Figure 11.3, adapted from [8], shows how the consolidation algorithm wipes out almost all the bias caused by fractionation.

Table 11.1 Statistics on the reconstruction of the common ancestor of grape and poplar, before and after taking into account consolidation of the fractionation intervals. Of note is the decrease in the percentage of excess adjacencies (in boldface), representing artifactual rearrangements when fractionation is not taken into account

Genes in comparison	Grape	Poplar	Ancestor
Single copies	12,494	4,282	8,631
In syntenic pairs	0	$2 \times 8212 = 16{,}424$	0
Total	12,494	20,706	8,631
Adjacency statistics	Before fractionation analysis		
Adjacencies	12,475	20,676	8,588
Distinct (a)	12,475	16,165	8,588 (b)
Distinct overall (c)	19,446		
Excess (c–a)	6,971 (55.9 %)	3,281 (20.3 %)	
With ancestor (d)	9,390	11,094	**total**
Excess (d–b)	802 (9.3 %)	2,506 **(29.2 %)**	3,308 **(38.5 %)**
Virtual genes	After fractionation analysis and consolidation		
Fractionation intervals		2,462	1,888
Single copies	10,674	0	8143[a]
In syntenic pairs	0	$2 \times 10{,}674$[b] $= 21{,}348$	0
Total	10,674	21,348	8143
Adjacency statistics	After consolidation		
Adjacencies	10,655	21,318	8,107
Distinct (a)	10,655	13,309	8,107 (b)
Distinct overall (c)	15,278		
Excess (c–a)	4,623 (43.4 %)	1,969 (14.8 %)	
With ancestor (d)	9,079	9,844	**total**
Excess (d–b)	972 (12.0 %)	1,737 **(21.4 %)**	2,709 **(33.4 %)**

[a]Counting genes within virtual genes: 9502

[b]Includes duplicate genes (not in a fractionation interval) and two copies of virtual genes even if only one gene-containing interval was found in poplar

11.6 Grape and Poplar

We applied our method to the genome of poplar (*Populus trichocarpa*) [28], which descends from a WGD event some 70 million years ago, and grape (*Vitis vinifera*) [29], which has undergone no WGD since the two genomes diverged some 130 million years ago.

As can be seen in Table 11.1, we discovered that a good proportion, over 25 %, of the apparent rearrangement in the poplar lineage, is actually attributable to frac-

tionation. This is remarkable since only about 20 % of the poplar genome is made up of single-copy regions.

Another advantage of consolidation is that it resolves a major part of the "short contigs" problem of the MWM approach. The first stage of the MWM in the reconstruction before consolidation produced 2598 contigs with 12,494 genes. But once we applied the consolidation algorithm only 967 contigs were produced by the MWM, lengthening the average contig size by a factor of 2.6.

The benefits were less striking but still non-negligible in the second stage MWM, itself designed to overcome the problem of short contigs. Here, instead of 43 scaffolds in the reconstruction before consolidation, seven additional "joins" appeared, for a reduction to 36 scaffolds in the consolidated data.

11.7 Simulations

In our complex model of genome divergence through rearrangement, WGD and fractionation can only be validated by seeing how many aspects of the simulated output genomes match those of the real genome, with a minimum of model parameters. The number of genes in the two genomes, and the number of single-copy genes are fixed quantities, determined by the real genomes. Rearrangement can be carried out by a mixture of $(1-\theta)\rho$ short inversions, where the number of genes in the scope of the inversion is geometrically distributed with mean μ, plus $\theta\rho$ unbounded rearrangements whose endpoints are chosen randomly on chromosomes. In each deletion event the number of contiguous genes lost is geometrically distributed with mean λ. Finally, we introduce a parameter π for fractionation bias, the probability that a deletion takes place in a specified "subgenome", one of the two original copies of the duplicated ancestral genome created by the WGD event. There are thus five parameters that must be set for each simulation, ρ, θ, μ, λ, π plus the given structure of the ancestral genome, determined in our case, by the number of homologs in the poplar and grape genomes, and the number of single-copy genes in poplar. To find the appropriate values of the parameters to simulate the data, we can observe the total number of adjacencies between the output genomes, both before consolidation (R_1) and after consolidation (R_2). We can measure the average size L of the fractionation intervals (or, equivalently, the number of intervals N, since the product of the two quantities is fixed). And we can also indirectly observe the fractionation bias P, which is the deviation from an even split of the deleted genes of from the two copes of the fractionation interval in poplar or its simulation. More specifically we can measure $P(1)$, $P(2)$, ... in pairs of poplar fractionation intervals totaling $1, 2, \ldots$ genes, respectively. We term this "indirect" since we do not have access in the real poplar genome to the identity of genes in terms of their origin in one of the other "subgenomes" produced by the WGD event. We simply measure how many more genes there are in the larger fractionation interval compared to its counterpart, a value that is larger, on the average than the "true" bias.

We carried out a cyclical search, one parameter at a time, to find settings (Table 11.2) that gave the same average R_1, R_2 and N over 50 simulations as the values

Table 11.2 Best parameter values

Parameters	ρ	θ	μ	λ	π
Best values	1570	0.05	2.47	1.32	0.70

Table 11.3 Simulation statistics compared to real genomes. In each case the standard deviation over 50 samples was less than 1 % of the mean of the variable

	Real genomes		50 simulations	
	Before Consolidation	After Consolidation	Before Consolidation	After Consolidation
Total adjacencies (R)	19,538	15,363	19,563	15,298
Fractionation intervals (N)		2,462		2,458

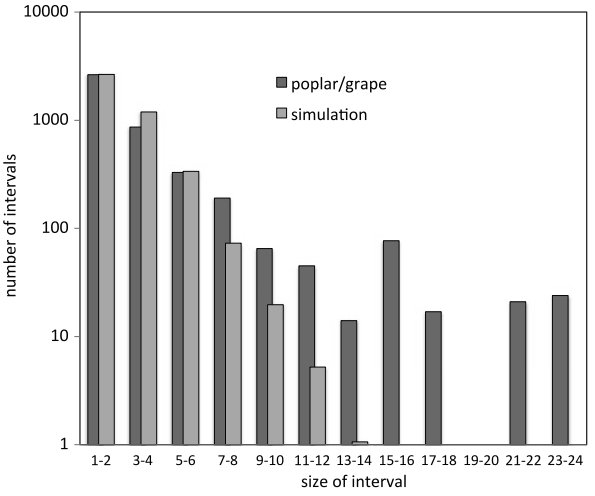


Fig. 11.4 Size of poplar/grape fractionation regions compared to simulations with parameters set so that reconstruction statistics match

calculated from grape and poplar (Table 11.3). We adjusted π so that the plot of the average simulated $P(i)$ resembled that from the real genomes.

Examining the consolidated regions detected by our algorithm, there are a number of regions much longer than those in the simulations (Fig. 11.4), suggesting a non-independence of deletion events affecting neighboring genes, and clear tendency for genes to be deleted in one of the two homeologs, as would be predicted by the recent theory of subgenome dominance [27].

In Fig. 11.5, a high value ($p = 0.85$) for the fractionation bias fits the data from poplar well only for long single-copy intervals, which are relatively rare (see Fig. 11.4), but a lower value ($p = 0.70$) fits the more numerous short intervals better. This is strong evidence that the selection of the various deletion sites does not proceed entirely independently, but rather that there are some regions of the genome that are particularly prone to becoming single-copy.

There are more parameters (five) to set in the simulations than quantities to observe (four) (though P is a vector, we can only observe its trend with any accuracy,

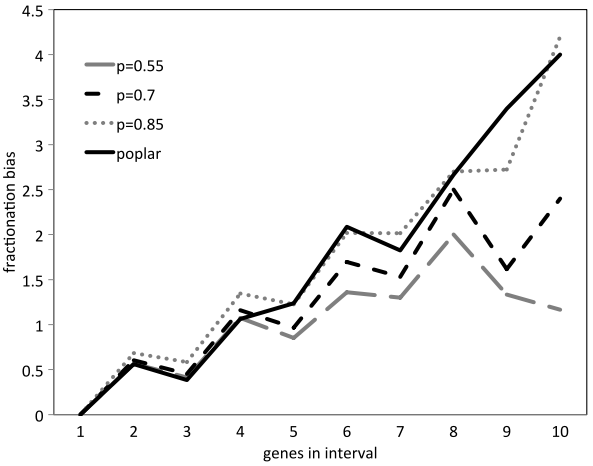


Fig. 11.5 Discrepancy in pairs of poplar intervals in the number of genes, compared with simulations with fractionation bias $p = 0.55, 0.70, 0.85$. Deletion event size geometrically distributed with mean 1.3. Jagged nature of graphs due not to statistical fluctuation but to measurement of discrepancy from an "even" split, which is necessarily calculated slightly differently for even and odd totals of genes in the fractionation intervals

not the individual $P(i)$), so there is inherently some non-uniqueness associated with the best choice of parameters, as suggested in Fig. 11.6. Nevertheless, the parameters can only take on values in a very restricted region. For example, outside a narrow range ρ produces too few or too many adjacencies, no matter what the settings are for the other parameters. And given ρ, the number of intervals N is sensitive to both parameters μ and λ.

11.8 Conclusions

The most important result from this work is that consolidation has the effect of greatly increasing the length of ancestral contigs output from the first MWM stage. The scaffolding approach already compensates for the short contigs problem, but combining the two strategies yields even longer scaffolds.

We have shown that we can closely simulate the gene order evolution of a WGD descendant and an unaffected sister genome, lending some confidence to our reconstruction of their common ancestor. We do detect, however, a significant number of long single-copy intervals, with highly biased fractionation, in the poplar genome, lying well outside the scope of our simulations. Whether there are biological connections among the genes in these intervals, and whether there are genes with no detected homologies in grape that are also present as single copies in these intervals, are questions for further study.

Further work will also involve improvements in parameter estimation as well as the identification of other measurable properties of evolutionary scenarios, restoring the balance between the number of parameters and the number quantities observed, in order to dispel problems of non-uniqueness.

This work has been undertaken as part of a project to formally analyze aspects of WGD fractionation, especially in the context of angiosperm evolution. Other directions include allowing duplicates in pairs of fractionation intervals, treating ploidies

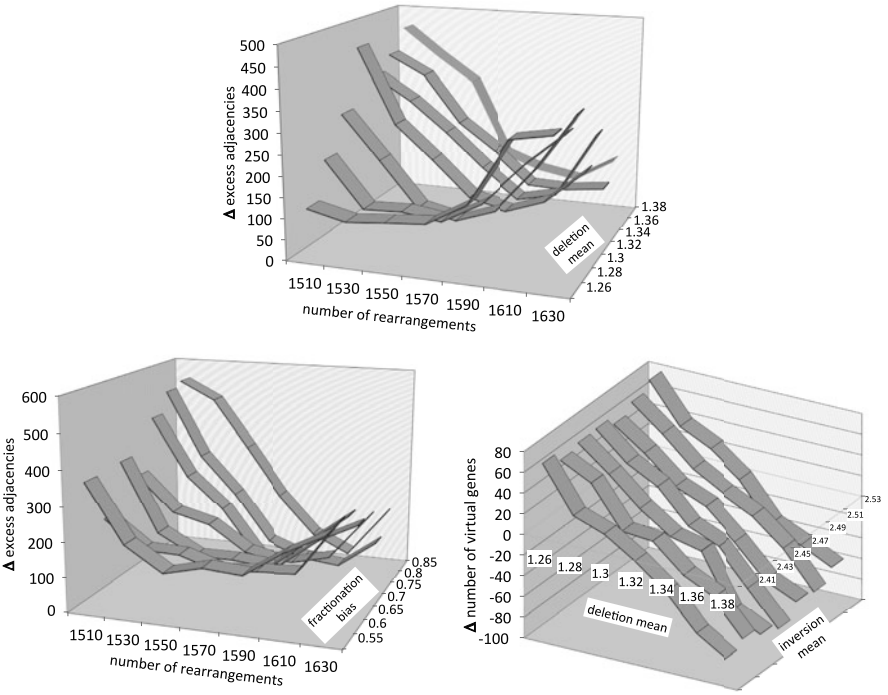


Fig. 11.6 Non-independent effects of simulation parameters on reconstruction characteristics. *Top*: Difference between real and simulated genomes (sum of R_1 and R_2 absolute differences) as a function of ρ and λ, showing the dependence of minimizing values of the parameters. *Left*: Difference between real and simulated genomes as a function of ρ and π, showing the dependence of minimizing values of the parameters. *Right*: Difference between real and simulated genomes (number of virtual genes) as a function of λ and μ, showing the *independence* of minimizing values of these particular parameters

of higher degree than WGD, dispensing with the necessity of an unaffected sister genome, as well as a probabilistic model of the distribution of fractionation interval sizes.

Acknowledgements We thank Katharina Jahn for many valuable comments and suggestions during the work reported here, and Vic Albert and Eric Lyons for guidance to current trends in angiosperm genomics. Research supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC). DS holds the Canada Research Chair in Mathematical Genomics.

References

1. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. PLoS Comput. Biol. **4**, 11 (2008)

2. Alekseyev, M.A., Pevzner, P.A.: Breakpoint graphs and ancestral genome reconstructions. Genome Res. **19**, 943–957 (2009)
3. Gagnon, Y., Blanchette, M., El-Mabrouk, M.: A flexible ancestral genome reconstruction method based on gapped adjacencies. BMC Bioinform. **13**(S19), S4 (2012)
4. Zheng, C., Chen, E., Albert, V.A., Lyons, E., Sankoff, D.: Ancient eudicot hexaploidy meets ancestral eurosid gene order. BMC Genomics **14** (2013, in press)
5. Langham, R.J., Walsh, J., Dunn, M., Ko, C., Goff, S.A., Freeling, M.: Genomic duplication, fractionation and the origin of regulatory novelty. Genetics **166**, 935–945 (2004)
6. Soltis, D.E., Albert, V.A., Leebens-Mack, J., Bell, C.D., Paterson, A.H., Zheng, C., Sankoff, D., dePamphilis, CW, Wall, P.K., Soltis, P.S.: Polyploidy and angiosperm diversification. Am. J. Bot. **96**, 336–348 (2009)
7. El-Mabrouk, N.: Genome rearrangement by reversals and insertions/deletions of contiguous segments. In: Giancarlo, R., Sankoff, D. (eds.) Combinatorial Pattern Matching (CPM 2000), Proceedings of the 11th Annual Symposium. Lecture Notes in Computer Science, vol. 1848, pp. 222–234 (2000)
8. Sankoff, D., Zheng, C.: Fractionation, rearrangement and subgenome dominance. Bioinformatics **28**, 402–408 (2012)
9. Jahn, K., Zheng, C., Kováč, J., Sankoff, D.: A consolidation algorithm for genomes fractionated after higher order polyploidization. BMC Bioinform. **13**(S19), S8 (2012)
10. Lyons, E., Freeling, M.: How to usefully compare homologous plant genes and chromosomes as DNA sequences. Plant J. **53**, 661–673 (2008). http://genomevolution.org/CoGe/
11. Lyons, E., Pedersen, B., Kane, J., Alam, M., Ming, R., Tang, H., Wang, X., Bowers, J., Paterson, A., Lisch, D., 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)
12. Zheng, C., Swenson, K., Lyons, E., Sankoff, D.: OMG! Orthologs in multiple genomes—competing graph-theoretical formulations. In: Przytycka, T.M., Sagot, M.-F. (eds.) Algorithms in Bioinformatics, Proceedings of the 11th International Workshop. Lecture Notes in Computer Science, vol. 6833, pp. 364–375 (2011)
13. Galil, Z.: Efficient algorithms for finding maximum matching in graphs. ACM Comput. Surv. **18**, 23–38 (1986)
14. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinform. **10**, 120 (2009)
15. Warren, R., Sankoff, D.: Genome aliquoting revisited. J. Comput. Biol. **18**, 1065–1075 (2011)
16. Manuch, J., Patterson, M., Wittler, R., Chauve, C., Tannier, E.: Linearization of ancestral multichromosomal genomes. BMC Bioinform. **13**(S19), S11 (2012)
17. Wolfe, K.H., Shields, D.C.: Molecular evidence for an ancient duplication of the entire yeast genome. Nature **387**, 708–713 (1997)
18. Dietrich, F.S., Voegeli, S., Brachat, S., Lerch, A., Gates, K., Steiner, S., Mohr, C., Pöhlmann, R., Luedi, P., Choi, S., Wing, R.A., Flavier, A., Gaffney, T.D., Philippsen, P.: The Ashbya gossypii genome as a tool for mapping the ancient Saccharomyces cerevisiae genome. Science **304**, 304–307 (2004)
19. Kellis, M., Birren, B.W., Lander, E.S.: Proof and evolutionary analysis of ancient genome duplication in the yeast Saccharomyces cerevisiae. Nature **428**, 617–624 (2004)
20. Gordon, J.L., Byrne, K.P., Wolfe, K.H.: Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern Saccharomyces cerevisiae genome. PLoS Genet. **5**, e1000485 (2009)
21. Ouangraoua, A., Tannier, E., Chauve, C.: Reconstructing the architecture of the ancestral amniote genome. Bioinformatics **27**, 2664–2671 (2011)
22. Byrnes, J.K., Morris, G.P., Li, W.-H.: Reorganization of adjacent gene relationships in yeast genomes by whole-genome duplication and gene deletion. Mol. Biol. Evol. **23**, 1136–1143 (2006)
23. van Hoek, M.J., Hogeweg, P.: The role of mutational dynamics in genome shrinkage. Mol. Biol. Evol. **24**, 2485–2494 (2007)

24. Sankoff, D., Zheng, C., Zhu, Q.: The collapse of gene complement following whole genome duplication. BMC Genomics **11**, 313 (2010)
25. Wang, B., Zheng, C., Sankoff, D.: Fractionation statistics. BMC Bioinform. **12**(S9), S5 (2011)
26. Sankoff, D., Zheng, C., Wang, B.: A model for biased fractionation after whole genome duplication. BMC Genomics **13**(S1), S8 (2012)
27. Schnable, J., Springer, N., Freeling, M.: Differentiation of the maize subgenomes by genome dominance and both ancient and ongoing gene loss. Proc. Natl. Acad. Sci. USA **108**, 4069–4074 (2011)
28. Tuskan, G.A., Difazio, S., Jansson, S., Bohlmann, J., Grigoriev, I., Hellsten, U., Putnam, N., Ralph, S., Rombauts, S., Salamov, A. et al.: The genome of Black Cottonwood, Populus trichocarpa (Torr. & Gray). Science **313**, 1596–1604 (2006)
29. Jaillon, O., Aury, J.M., Noel, B., Policriti, A., Clepet, C., Casagrande, A., Choisne, N., Aubourg, S., Vitulo, N., Jubin, C., et al.: The grapevine genome sequence suggests ancestral hexaploidization in major angiosperm phyla. Nature **449**, 463–467 (2007)

Chapter 12
Error Detection and Correction of Gene Trees

Manuel Lafond, Krister M. Swenson, and Nadia El-Mabrouk

Abstract Reconstructing the phylogeny of a gene family and reconciling the obtained gene tree with the species tree reveals the history of duplications, losses, and other events that have shaped the gene family, with important implications towards the functional specificity of genes. However, evolutionary histories inferred by reconciliation are strongly dependent upon the accuracy of the trees, and few misplaced leaves will lead to a completely different history. Furthermore, sequence data alone often lack the information to confidently support a gene tree topology. We outline a number of criteria that can be used to detect erroneous gene trees. Analysing *Ensembl* gene trees of the fish genomes Stickleback, Medaka, Tetraodon, and Zebrafish reveals a significant number of erroneous gene trees. Finally, some potential directions for error correction of gene trees are explored.

12.1 Introduction

Duplication followed by modification is a major mechanism driving evolution. Consequently, genes cannot be seen as independent entities, but rather as entities related through duplication and speciation events. Grouping genes into families of *homologs* (i.e. copies originating from a single ancestral gene) and reconstructing the phylogeny of each gene family is requisite for a variety of annotation, evolutionary, and functional studies. By *reconciling* such a gene tree with a species tree, one can infer the history of duplications, losses and other events that have shaped the gene family. Such a history reveals the orthology (evolution of the ancestral

M. Lafond (✉) · K.M. Swenson · N. El-Mabrouk
Département d'Informatique et de Recherche Opérationnelle (DIRO), Université de Montréal, CP 6128, Succursale Centre-ville, Montréal, QC H3C 3J7, Canada
e-mail: lafonman@iro.umontreal.ca

N. El-Mabrouk
e-mail: mabrouk@iro.umontreal.ca

K.M. Swenson
McGill University, Montreal, QC, Canada
e-mail: swensonk@iro.umontreal.ca

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*, Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_12,

copy by speciation) and paralogy (evolution by duplication) relationship between genes, with important implications towards the functional relationship between gene copies. However, uncertainty on gene trees is a serious limitation to reconciliation, as well as to other applications. In particular, it has been reported that a few misplaced leaves can lead to a completely different history, possibly with significantly more duplications and losses [30]. Thus, a great deal of effort has been put into finding accurate gene trees.

Gene Tree Inference Inferring phylogenies from sequence similarity is a field with a very long history that gave rise to a variety of distance, maximum parsimony, maximum likelihood or Bayesian methods, and a variety of software (PHYLIP [20, 21], NJ [47], PAUP [54], PhyML [28], MrBayes [44], RAxML [51]). However, due to various limitations such as insufficient differentiation, alignment ambiguity, or differing rates of evolution among gene copies, sequences alone do not always support a single gene tree topology with high confidence.

Recently, several approaches have been developed to incorporate other genomic information in the construction of gene trees. For example, the SYNERGY algorithm [60] uses a "synteny similarity score" accounting for the position of genes in the chromosome. Different ways of integrating species tree information have also been considered. For example, the TreeBeST program from TreeFam [32, 45] (used for constructing the *Ensembl Compara* gene trees) uses a likelihood factor reflecting the number of duplications and losses inferred by reconciliation, the goal being to minimize inconsistency with the species tree. Another example is GIGA [57], a simple and fast algorithm using a UPGMA like distance-based approach to construct trees. In addition to the distance criterion, it relies on rules reflecting the species tree constraints (choose topologies in agreement with the species tree), as well as observations on lineage-specific evolution rates. This simple algorithm performs surprisingly well, leading to the conclusion that other constraints are strong enough to compensate for weak or misleading signals in gene sequences.

Other more sophisticated "species tree aware" methods have been developed, such as GSR [1, 2] and Spimap [43] adopting a Bayesian approach, or PhylDog [3] using a probabilistic model for simultaneously coestimating gene trees and the species tree. These models tend to be computationally intensive.

Gene Tree Correction A complementary approach for producing "error-free" gene trees is to develop appropriate evaluation and correction tools, based on various genomic constraints, that can be applied subsequent to gene tree reconstruction. TreeFix [62] offers an additional framework to unify the sequence and genomic approaches, by suggesting a step following gene tree correction that performs statistical evaluation of a corrected tree, choosing it as a viable alternative only if it is statistically equivalent to the original one. The strategies that have been considered for gene tree correction are based on reconciliation, and can be grouped into three different classes:

I. Explore the space of gene trees obtained from the original one by performing some edit operations such as NNI [13, 25], SPR, or TBR [10] and select the tree

having the minimum reconciliation cost. The "soft parsimony" algorithm [8] extends this approach for reconciliation with an uncertain species tree.

II. Collapse weakly supported internal branches [4], which leads to a non-binary gene tree, and then select the resolution minimizing the reconciliation cost [9, 36, 41].

III. Identify potentially misplaced leaves and remove them from the gene tree. In [12], vertices of a gene tree G labeled as Non-Apparent-Duplication (NAD) vertices, were flagged as potentially resulting from the misplacement of leaves in the gene tree. A duplication vertex x of G (according to the reconciliation with a given species tree) is a NAD if genes from the same species do not appear as a descendant of each of x's children. The reason for doubting NADs is that each one of these vertices reflects a phylogenetic incongruence with the species tree that is not due to the presence of duplicated genes in a single genome. Avoidance of NADs is one of the principles behind the GIGA algorithm [57]. We presented algorithmic results for removing, from a given gene tree, the minimum number of leaves or leaf-labels (species) leading to a tree without a NAD vertex, under conditions of a known or an unknown species tree [16, 52]. All known formulations of this version of the problem are NP-hard [14, 15].

Error Detection Known methods for correcting gene trees all rely on errors detected through reconciliation with the species tree. Similarly, in the field of gene tree reconstruction, most integrated methods rely on the species tree information, although other criteria have been suggested such as gene order [60] and variability of evolutionary rates [57]. In this paper, we follow up on this effort by exploring these two directions.

In Sect. 12.3, we show how gene order may be inconsistent with a gene tree, and state two error detection criteria based on gene order. To show the utility of these criteria, we consider the *Ensembl* [23] gene trees for four fish genomes (Stickleback, Medaka, Tetraodon, Zebrafish) with human and mouse as outgroups. We observe that more than 31 % of all trees exhibit at least one gene order contradiction. In Sect. 12.4, we show how the presence of negative and positive selection may be misleading for gene tree reconstruction, and suggest methodology for detecting natural selection bias in a gene tree. Using the non-synonymous (dN) versus synonymous (dS) substitution ratio dN/dS as a criterion for detecting natural selection, a clear selective pressure is observed on Ensembl gene trees as compared to random trees. Finally, in Sect. 12.5 we give some avenues for developing a coherent tool for correcting gene trees, taking advantage of all available sequence and genomic information.

12.2 Genomes, Trees, and Gene Family Histories

We begin by introducing the necessary notations and background concepts. Although some of our experimental results could be explained without such formalities, we find it important to be precise. Indeed, many of the terms introduced in Sect. 12.2.5 have been used in multiple ways under diverse circumstances, some-

times leading to confusion. Many concepts are also presented in a general way, in the hopes of illuminating the potential for related work.

12.2.1 Genomes

Although our methods may be extended to arbitrary genomes, for simplicity of presentation we only consider single chromosomal genomes, represented as strings of, possibly signed, genes. Let $A = a_1 a_2 \dots a_n$ be a string representing a genome. For any i, j such that $1 \leq i \leq j \leq n$, $A[i, j] = a_i a_{i+1} \dots a_j$ is a substring of A. A string obtained from a substring of A by removing a subset of genes (possibly empty), is called a *subsequence* of A. For $1 \leq i_1 < i_2 < \dots < i_p \leq n$, we denote by $A[i_1, i_2, \dots, i_p]$ the subsequence $A[i_1]A[i_2] \dots A[i_p]$ of A.

12.2.2 Trees

A *phylogeny* is a rooted binary tree, uniquely leaf-labeled by some set. A *species tree* S is a phylogeny over a set of species Σ, which represents the evolutionary relationships between these species. Similarly, we can consider the evolutionary relationships amongst a family of homologous genes Γ that appear in the genomes of Σ. A *gene tree* G for Γ is a phylogeny accompanied by a function $s : \Gamma \to \Sigma$ indicating the species where each gene is found. We will make no difference between a node and its associated gene. The tree G from Fig. 12.1 is a gene tree for $\Gamma = \{Z_1, M_1, M_2, S_2\}$ on species set $\Sigma = \{Z, M, S\}$. In this case, $s(M_1) = s(M_2) = M$.

Given a tree T and a node x of T, we denote by T_x the subtree of T rooted at x (i.e. the tree comprises x and all its descendants), and by $\mathcal{L}(T_x)$ the set of leaves of T_x. The species set of x, denoted $\mathcal{S}(T_x)$, is the subset of Σ defined by the labels of the leaves of T_x (if T is a gene tree then $\mathcal{S}(T_x) = \{s(\ell) : \ell \in \mathcal{L}(T_x)\}$). If there is no ambiguity about the tree in question, we write $\mathcal{S}(T_x)$ as $\mathcal{S}(x)$. The *lowest common ancestor* (LCA) of leaves x and y in a tree T, written $lca_T(x, y)$, is the common ancestor of x and y that is farthest from the root. Finally, for any internal node x of a rooted binary tree T, we denote by x_ℓ and x_r the two children (left and right) of x in T.

12.2.3 Histories

As a set of modern species evolves from a single ancestral species, some of the gene content of those species is modified through duplication within the genome, and then loss. Traditionally, reconciliation between gene trees and species trees has been used to reconstruct such histories. The basis for such methodology has been a formal definition of what a reconciliation is, without a definition of the actual history that is the ultimate objective. Indeed, for a family of genes related through

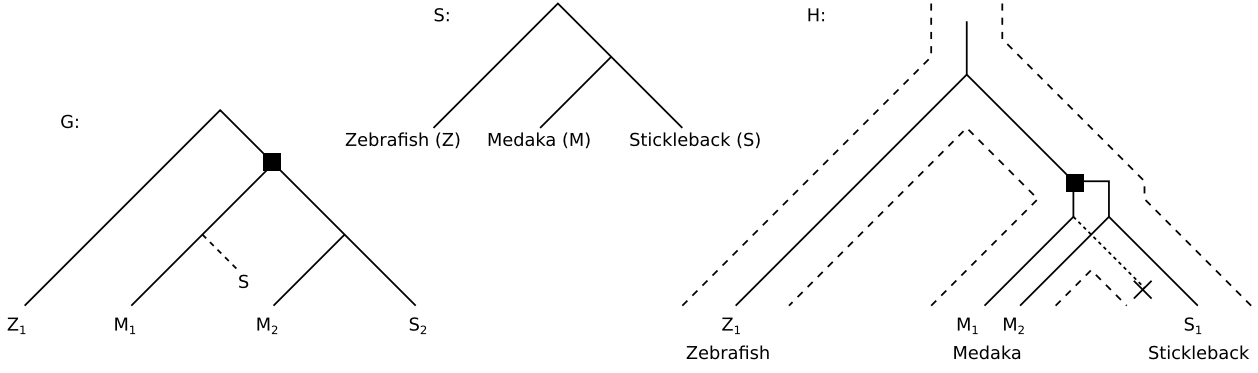


Fig. 12.1 *G* is the gene tree for the gene family "OLA.11555" from Ensembl (ENSORLG00000013558), extended with a loss leaf according to a reconciliation with species tree *S*. *H* is the duplication-loss history corresponding to *G*. Reconciliation of *G* with the species tree *S* gives one duplication and one loss (as marked in *G* and *H*, duplication by a *square*, and losses by *two dotted lines*)

duplications and speciations, there exists some true history—the actual duplications and speciations that occurred in the past. So as not to put the cart in front of the horse, we now define what we mean by a *duplication/loss/speciation-history* (*dls*-history). We then define a *reconciliation* in terms of *dls*-histories. This perspective facilitates the reasoning used in Sect. 12.3; knowing that there is one true history of speciation/loss/duplication for a family of genes, we establish conditions that true gene trees must possess.

A *duplication* of size $k+1$ on genome A is an operation that copies a substrings $A[i, i+k]$ to a location j of A outside the interval $[i, i+k]$ (i.e. preceding i or following $i+k$). A *Loss* of size k is an operation that removes a substring of size k from A. Given a set of genes Γ from a set of genomes Σ, a *duplication/loss/speciation-history* H for Γ is a rooted tree "embedded" in the species tree S of Σ, which reflects the evolution of the set from a single ancestral copy through duplication, loss and speciation events. In other words, each internal node x of H represents the evolution of the set $\mathcal{L}(H_x)$ from an ancestral gene copy x_A, and corresponds to either a speciation or gene duplication event. The leaves correspond to either the genes in question, or to losses, where each of the latter *loss leaves* map to a single node of S. If a loss leaf ℓ maps to a node x of S, we say that S_x is the *label* of ℓ.

Definition 1 (*dls*-history) Let Γ be a set of genes from a set of genomes Σ, and let S be the true phylogeny for Σ. A *duplication/loss/speciation-history* H for Γ consistent with S (or simply a *dls*-history if unambiguous) is a rooted binary tree such that:

- each leaf is uniquely labeled by an element of Γ, or it is a loss leaf labeled by a subtree of S;
- each internal node is labeled as a duplication or speciation; and
- H is *consistent* with S: Consider the tree $\bar{H}$ obtained from H by replacing each loss leaf by the subtree that labels it, and by replacing all other leaves by the species to which the attached gene belongs. Then, for every internal node x of $\bar{H}$ such that $|\mathcal{S}(x)| \geq 2$, there exists a vertex u of S such that $\mathcal{S}(x) = \mathcal{S}(u)$ and: $\mathcal{S}(x_r) = \mathcal{S}(x_\ell)$ if x is a duplication, or $\mathcal{S}(x_r) = \mathcal{S}(u_r)$ and $\mathcal{S}(x_\ell) = \mathcal{S}(u_\ell)$ if x is a speciation node.

The gene tree *in agreement* with H is the tree obtained from H by removing loss leaves and the resulting internal nodes having one child. Consider the trees from Fig. 12.1. The solid lines of G denote the gene tree corresponding to the history H.

As true histories are unknown, gene trees are usually inferred from sequence data, and histories subsequently inferred from *reconciliation* with the species tree (see the next section). In this paper, we will distinguish between the *true gene tree*, which is the tree in agreement with the true *dls*-history of the gene family, and the *gene tree*, which is a tree obtained from the observed gene sequences (e.g. a multiple alignment of the sequences, the observed gene positions, or any other footprint of evolution observed in the extant species).

12.2.4 Reconciliation

Given an inferred gene tree G for a set Γ of genes from genomes Σ, and given a species tree S for Σ, the problem is to recover a *dls*-history for Γ consistent with S, such that G is in agreement with the history. Such a history is called a *reconciliation*. Informally, a *reconciliation* R of G and S is a *dls*-history of Γ obtained by inserting loss leaves in G. Let an *extension* of G be a tree obtained from G by a sequence of loss insertions, where a *loss insertion* denotes the insertion of a new loss leaf labeled by a subtree of S, by means of bisecting an existing edge of G with a new edge. A rigorous definition of reconciliation follows.

Definition 2 (Reconciliation) A *reconciliation* R of gene tree G and species tree S is an extension of G that is a *dls*-history consistent with S.

The parsimony criteria used to choose among the large set of possible reconciliations are usually the number of duplications (*duplication cost*), the number of losses (*loss cost*) or the sum of the two (*mutation cost*). Many algorithms have been developed for computing the most parsimonious reconciliation, the most efficient ones with running time proportional to the size of the gene tree [12, 19, 27, 67].

12.2.5 Perspectives on Homology

There have been many uses of the word homology and the related concepts, the confusion due to the many possible measures of similarity between genes. Indeed, evolutionary, sequence, functional, or positional constraints give rise to definitions that are unfortunately not equivalent [35]. In this paper we adopt the original definitions recommended by Fitch [22], corresponding to the evolutionary concepts.

Definition 3 (Homology) Two genes are *homologous* if and only if they are the leaves of a *dls*-history H. A *gene family* is a set of homologous genes.

Although many genes share a common origin [56], and thus share the same *dls*-history, the definition of homology given by Fitch does not include a necessary limit on the evolutionary closeness between two homologous genes. To our knowledge, this is an unfortunate and unstated ambiguity that we must live with for the time being.

The remainder of the definitions describe a hierarchy of homologous genes, implied by the *dls*-history H.

Definition 4 (Orthology) Genes a and b are *orthologous* if $lca_H(a, b)$ is a speciation node.

As duplications may arise following a speciation event, the orthology relationship is not transitive. This property is inherent to the evolutionary definition of or-

thology, which is not a definition about the functional relationship between genes, nor the positional or direct descendant relationship. In this perspective, Fitch [22] introduced the following notion of *functional orthologs* or *isorthologs*, for a given function (in case of hemoglobin sequences for example, the function is the ability of being the adult transporter of oxygen).

Definition 5 (Isorthology) Two orthologous genes that have retained the same function $\mathcal{F}$ of their LCA in H are called *isorthologous* for function $\mathcal{F}$.

Isorthology relation is transitive. Therefore it makes sense to speak of sets of isorthologs, or isorthogroups. Two genes are in the same *isorthogroup* if and only if they are isorthologous. Finally, we introduce the notion of paralogy.

Definition 6 (Paralogy) Genes a and b are *paralogous* if $lca_H(a, b)$ is a duplication node.

Consider the histories from Fig. 12.2(a). Any two genes denoted by the same letter are homologous. The history for homologous gene family c serves as a good example. The gene from C_1 is orthologous with all occurrences of c in C_3 and C_4, while it is paralogous to the gene in C_2. Further, the last occurrences of c in C_4 is paralogous to the second occurrence of the gene in C_3.

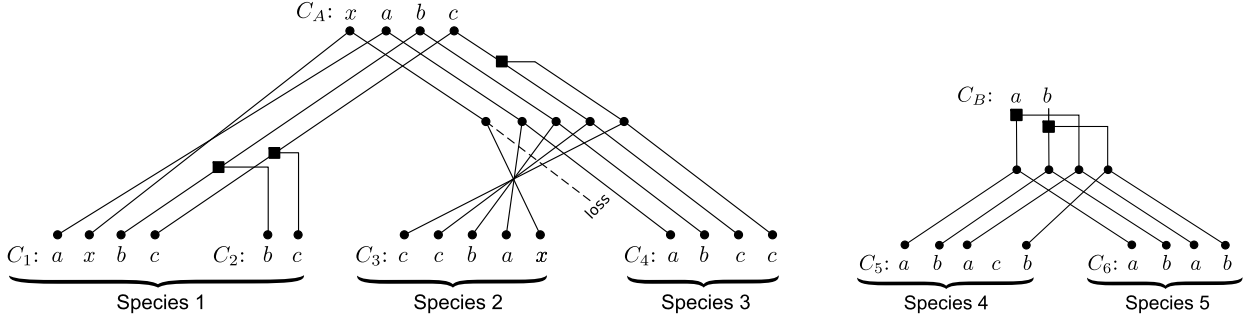
12.3 Gene Order Inconsistency

In this section we explore how information on gene order can be used to discover erroneous gene trees. The general idea is the following: look at the *regions* (formally defined below) surrounding the genes of interest. If they are similar (in terms of gene order), assuming that this cannot happen by chance, we can deduce that they are *homologous*, i.e. they descend, through a duplication or speciation event, from a common ancestral region. Such property on homology for regions leads to properties on underlying genes: homologous genes in the two regions are either all pairwise orthologous or all pairwise paralogous. These properties can then be checked against gene trees, and used as criteria for correcting them.

In Sect. 12.3.1 we formally define homology on regions. This perspective allows us to establish in Sects. 12.3.2 and 12.3.3 properties that sets of true gene trees must possess when genes belong to similar regions, given that the following hypothesis about convergent evolution is assumed:

Hypothesis NoConvergentEvol: Similar regions are homologous.

In the last 15 years many methods have been developed for the classification of similar syntenic regions that have undergone gene order mutation [5–7, 31]. Hoberman and Durand [33] give a nice treatment of the competing interests surrounding a good definition of gene order similarity. David Sankoff has been ever present in the



(a) evolution following a speciation event affecting an ancestral region $C_A = xabc$.

(b) evolution following the duplication of an ancestral region $C_B = ab$.

Fig. 12.2 Gene trees for two different ancestral regions. Duplications are denoted by square nodes, speciations by circles, and losses by *dashed edges*. Next to each C_i is a description of a substring of a genome. Each region C_i is defined as the genes labeling the leaves of a gene tree. For example $C_5 = abab$ (does not include gene c)

discussion [18, 34, 48, 63, 65, 66]. Whatever the definition, the underlying idea is to maximize the probability that similar regions are indeed homologous.

Our study in Sect. 12.3.4 limits regions to the immediate left and right neighbors of the genes in question; the regions of two homologous genes are similar if they are directly surrounded by homologous genes. Under this definition, the substrings aba of region C_5 and aba of region C_6 from Fig. 12.2 are similar, as do abc of C_4 and cba of C_3.

12.3.1 Region Homology

Homology on a set of genes is a property of the true history for that set, independent of any similarity measure amongst them. Homology of a set of regions should also be defined in a manner that is independent of any particular similarity measure on those regions. To accomplish this we leverage the duplication/loss/speciation histories for the genes contained in the regions of interest.

A *region* of a genome A is simply a subsequence of A. An *ancestral region* is a region occurring in some ancestral genome, while a *modern region* is a region occurring in some modern genome.

Definition 7 (Region homology) Let C_k and C_ℓ be two modern regions defined on a gene set Γ, subdivided into the gene families $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_m\}$. Let $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$ be the *dls*-histories corresponding to Γ_is, and let a_i be the root of H_i. Then C_k and C_ℓ are *homologous* if and only if the a_is all belong to a region $C_A = a_1 a_2 \ldots a_m$ of an ancestral genome A, and they are either all speciation nodes or all duplication nodes. We call C_A the LCA region for C_k and C_ℓ.

The case where the roots of the *dls*-histories are speciations corresponds to the divergence of C_A through a speciation event, while the latter case corresponds to the divergence through a duplication event that has duplicated the entire ancestral region C_A.

Notice that the definition of region homology supports the possibility of rearrangements occurring during the evolution of regions; in Fig. 12.2(a) genes a and x have been inverted in the branch from the ancestral genome to Species 1, yet regions C_1 and C_3 are homologous. Local duplications of sub-regions (in tandem or not) are also supported. In Fig. 12.2(a) for example, a duplication of gene c occurs in the branch leading to Species 2 and 3, yet regions C_1 and C_3 are homologous. Insertion and deletion of genes are supported as well. For example, gene c in Species 4, which is not present in Species 5, does not prevent regions C_5 and C_6 from being homologous. Moreover, the ancestral region C_A may contain genes that have lost in the *dls*-histories leading to modern regions.

Notice, however, that, in contrast to the homology relationship on genes, the homology relationship on regions is not transitive. Consequently, we are unable to generalize the notion of gene families to the notion of homologous region families.

12.3.2 Homology Contradiction

Our definition of homologous regions, along with Hypothesis NoConvergentEvol, provides us with a tool for testing the validity of gene trees. Remember that for a pair of homologous regions, the roots of the genes trees that comprise the genes contained in the two regions must all be the same type of node; they must all be speciation nodes, or they must all be duplication nodes. Thus, for a pair of similar regions—assumed from Hypothesis NoConvergentEvol to be a pair of homologous regions—and an inferred set of gene trees—implying a set of homology relationships between genes of the regions—we can confirm that indeed the gene trees have such roots. If they do not, we say that the forest of gene trees exhibits a *homology contradiction*.

12.3.3 Region Overlapping

In this subsection, we define the notion of a *region surrounding a gene* in a strict way ensuring a single region assignment for each gene, and a fixed length for all regions. Formally, for a given set of parameters $0 < l_1 < \cdots < l_p$ and $0 < r_1 < \cdots < r_q$, the region C_x surrounding the gene at position x in genome A is the subsequence $A[x - l_p, \ldots, x - l_1, x, x + r_1, \ldots, x + r_q]$. In Sect. 12.3.4, the underlying parameters are $p = q = 1$, and $l_1 = r_1 = 1$. Now two regions C_k and C_ℓ are *similar* if an only if, for any i, $C_k[i]$ and $C_\ell[i]$ belong to the same gene family. This definition of similarity ensures transitivity, which allows to define a *similarity family* as a family of pairwise similar regions.

A stronger statement on no convergent evolution is also required:

Hypothesis StrongNoConvergentEvol: Two similar regions are homologous. In addition their similarity is inherited from their LCA region and preserved during the course of evolution.

Stated formally, let C_k and C_ℓ be two similar regions surrounding two homologous genes x_k and x_ℓ belonging to a gene family Γ, and let G be the true gene tree for Γ. Then the regions surrounding ancestral genes corresponding to the nodes on the path between x_k and x_ℓ in G are similar to C_k and C_ℓ.

Take a gene tree G such that each gene (leaf of G) is assigned to a region, and that regions are grouped into similarity families $\mathcal{E} = \{F_1, F_2, \ldots, F_p\}$.

Let $V(G)$ be the set of internal nodes of G. Consider the *region labeling function* $\ell_G : V(G) \to 2^{\mathcal{E}}$ (where $2^{\mathcal{E}}$ is the power set of $\mathcal{E}$) that labels the nodes of G with homologous families as follows:

1. for all $x \in V(G)$, initialize $\ell_G(x)$ to $\emptyset$;
2. for each family F_i, include F_i in the label of any node on a path from a pair of leaves with label F_i.

The following lemma provides a second criterion for error detection in gene trees.

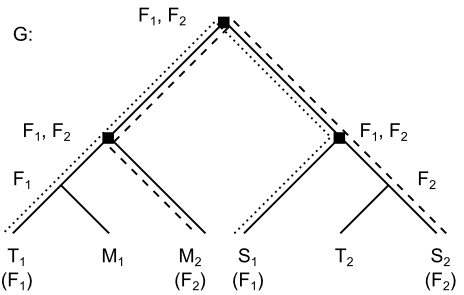


Fig. 12.3 The gene tree of the "RAB27" gene family (ENSGACG00000003336) for the Stickleback (S), Medaka (M) and Tetraodon (T) species, exhibiting a region overlapping. The T_1, S_1 genes are in similarity family F_1, while M_2, S_2 are in another similarity family F_2. The internal nodes are annotated by their $\mathfrak{l}_G$ labeling; all nodes on the dotted path are labeled by F_1, and those on the dashed path by F_2

Lemma 1 *If G is the true gene tree for some set of genes and* Hypothesis StrongNoConvergentEvol *holds, then for each node x of G, $|\ell_G(x)| \leq 1$.*

Proof Let x be an internal node of G with surrounding region C_x, and suppose $\ell_G(x)$ contains at least two elements F_i, F_j of $\mathcal{E}$. From the definition of ℓ_G, it follows that x is on the path between some genes ℓ_i and r_i with regions C_i^ℓ and C_i^r, both belonging to F_i. In the same manner, x is on the path between genes ℓ_j and r_j with region C_j^ℓ and C_j^r belonging to F_j. We see that x has at least one descendant that is ℓ_i or r_i, and at least another descendant that is ℓ_j or r_j. Suppose without loss of generality that ℓ_i and ℓ_j are descendants of x. By Hypothesis StrongNoConvergentEvol, C_i^ℓ and C_j^ℓ are both similar to C_x, and since similarity is transitive, C_i^ℓ is similar to C_j^ℓ. It follows that $F_i = F_j$. □

A gene tree with an internal node possessing multiple labels is said to exhibit a *region overlap*. Notice that for such a node, Lemma 1 holds whether it is a speciation or a duplication. Figure 12.3 shows a gene tree with multiple region overlaps, which are all duplications. Consider the overlapping occurring at the root of G, which we denote by r. It might be tempting to explain this scenario by stating that since r is a duplication, one copy of the ancestral gene belonged to the ancestral region similar to F_1, and the other to the ancestral region similar to F_2, and thus both regions could have propagated to their respective descendants. However, r refers to a single ancestral gene, which may have belonged to one of the two ancestral regions, but not to both, as we assume each gene is assigned a single region.

12.3.4 Results

We wanted to see the impact of using homology contradiction and Lemma 1 to reveal errors in gene trees. To this end, we considered the four fish genomes *Gasteros-*

teus aculeatus (Stickleback), *Oryzias latipes* (Medaka), *Tetraodon nigroviridis*, and *Danio rerio* (Zebrafish) with human and mouse as outgroups. We used the *Ensembl Genome Browser* to collect all available gene trees, and filtered each tree to preserve only genes from the taxa of interest. We then reconciled the trees with the known species trees, and identified duplication and speciation nodes. Genes appearing in the same gene tree in the database are considered to be part of the same homologous gene family.

In this section, a region surrounding a gene is defined as the substring containing the gene and both its left and right adjacencies. Two regions are similar if they contain homologous genes in the same order or inverted order. More precisely, regions $C_k = x_1 a_1 y_1$ and $C_\ell = x_2 a_2 y_2$ (or $C_\ell = y_2 a_2 x_2$) are similar if x_1 and x_2 appear in the same *Ensembl* gene tree, a_1 and a_2 appear in the same gene tree, and y_1 and y_2 appear in the same gene tree. We avoid tandem duplications by requiring the three trees to be different.

In Sect. 12.3.2 we defined the homology contradiction property for a forest of gene trees. Here, we identify problematic forests of gene trees using that property. Let $C_k = x_1 a_1 y_1$ and $C_\ell = x_2 a_2 y_2$ be two similar regions and G_x, G_a, and G_y be the gene trees containing the pairs of homologs (x_1, x_2), (a_1, a_2) and (y_1, y_2), respectively. Then, according to our definition, the forest $\{G_x, G_a, G_y\}$ exhibits a homology contradiction iff the set $\{lca_{G_x}(x_1, x_2), lca_{G_a}(a_1, a_2), lca_{G_y}(y_1, y_2)\}$ contains at least one duplication node and at least one speciation node.

In this section we will focus on the gene tree of the central gene. We say that G_a exhibits a *paralogy contradiction* iff $lca_{G_a}(a_1, a_2)$ is a duplication node, and both $lca_{G_x}(x_1, x_2)$ and $lca_{G_y}(y_1, y_2)$ are speciation nodes. Conversely, we say that G_a exhibits an *orthology contradiction* iff $lca_{G_a}(a_1, a_2)$ is a speciation node, and both $lca_{G_x}(x_1, x_2)$ and $lca_{G_y}(y_1, y_2)$ are duplication nodes. Note that this notion of contradiction is extremely conservative; if only a single neighbor disagrees with the central gene, then we do not report it.

Results are summarized in Table 12.1. Among the 6241 trees in Ensembl, 6118 of them have at least one pair of genes in the same context. More than 31 % of the 6241 trees exhibited at least one contradiction, the most frequent contradiction type being paralogy contradiction. These numbers show that a very conservative application of our methods uncovers a significant number of inconsistencies between gene order and gene tree topology.

It is conceivable that a significant number of missing genes in the gene trees could lead to a false homology contradiction. Also, poor detection of homology relationships in Ensembl could yield false region overlaps. For example, two overlapping regions could have the form $C_k = a_1 b_1 c_1$ and $C_\ell = x_2 b_2 c_2$. But if x_2 should in fact be in the same homologous gene family as a_1, the overlap would no longer exist. This is what happens in the example of Fig. 12.3. The F_1 region consists of "ASH1L" "RPS27" "KCNN3" genes, while the F_2 region is made of "RAB13" "RPS27" "KCNN3" genes. In fact, every single overlapping regions we found had this form. Thus region overlaps in Ensembl gene trees might not occur because of wrong topologies, but rather because of missing homologies. In any case, detection of overlaps can identify possible improvements on the known relationship between some pairs of genes.

Table 12.1 Results obtained for Ensembl gene trees. Reported numbers are not mutually exclusive, in the sense that a given tree may exhibit more than one type of contradiction, and thus be included in more than one list. In brackets are the actual numbers of trees

Number of trees	6241
Region overlap	3.4 % (210)
Paralogy contradiction	22.5 % (1407)
Orthology contradiction	10.8 % (677)
At least one contradiction	31.3 % (1959)

To get an idea of how the numbers can change, we reran the test suite for a more general notion of similarity: C_k and C_ℓ are similar if b_1 and b_2 are homologous, and if there exists a pair of neighbors c_1 and c_2 that are homologous. Note that under this definition, there are fewer region sets so region overlaps are harder to find. The new definition finds 71 (2.38 %) gene trees with overlaps.

Yet our region overlaps and homology contradictions tend to agree with mechanisms already in place for error detection in Ensembl gene trees. Based on the structure of the tree, some duplication nodes, corresponding to NAD nodes [12], are labeled as "dubious" in the Ensembl trees. As paralogy and orthology contradictions are inferred according to duplication nodes (one duplication node involved in a paralogy contradiction and two in an orthology contradiction), we were interested to see to which extent our results correlated with Ensembl observations about dubious duplications. We found that 77.4 % of duplications involved in observed paralogy contradictions are labeled as dubious, while 90.2 % of duplications involved in orthology contradictions are dubious. These number are significantly high considering that the fraction of dubious duplications among the total number of duplications in our trees is only 36 %. These observations validate the fact that gene order inconsistencies are likely to reveal errors in gene trees.

12.4 Positive and Negative Selection Bias

Classical phylogenetic methods, such as those using parsimony, distance or maximum likelihood models, are typically based upon the assumption of stochastic, neutral, and site-independent processes. However, as few mutations may cause structural modification to protein coding genes with deleterious functional consequences, isorthologous gene copies in multiple species are commonly subject to negative (purifying) selection pressure, leading to sequence stability inside isorthogroups. On the other hand, positive selection, responsible for the creation of new function, is also known to play a major role in the evolution of gene families. Under natural (positive and negative) selection, a gene tree best reflecting the sequence similarity of gene copies is more likely to reflect functional constraints rather than evolutionary and ancestral relationships between gene copies. In particular, negative selection may result in isorthologous genes being grouped into a subtree of the gene tree, leading to erroneous ancestral inference for the isorthogroup.

This grouping driven by function has been reported for different gene families, such as GLP-1 [49] and opsin proteins [55]. An interesting study based on simulations is also reported in [38]. In this study, DNA sequences encoding a protein folding, with a predefined active site for the binding of a ligand, have been generated. An A ligand initially bound stably at the beginning of the simulation, while a B ligand did not. The proteins were evolved under constant population size and mutation rate. In every generation the individuals were picked randomly, provided they folded stably and binded to a peptide. Moreover, to simulate positive selection, a selective advantage of 5 % was given to individuals binding the new ligand B. Phylogenetic trees for simulated sequences were then inferred using distance, parsimony and likelihood methods. Every generated tree exhibited a clustering by function rather than by ancestry (two monophyletic groups, one for proteins binding to the ligand A and the other for proteins binding to the ligand B). In the same paper, other results obtained on multiple sequence alignments of Chordate genes also confirmed previous studies on the loss of the evolutionary signal due to negative and positive selection [29, 46, 58].

12.4.1 Detecting Functional Bias

In the presence of negative and positive selection (i.e. confusion of the neutral phylogenetic signal), some studies have recommended different criteria for gene (site) selection when reconstructing phylogenies. In particular, the filtering of fast evolving genes has been suggested to reduce the effect of positive selection [29]. On the other hand, filtering slow evolving sites has been suggested to reduce the effect of negative selection. However, as noticed in [38], these models for data filtering have limitations as evolution speed does not always correlate with selection type.

Instead of an *a priori* selection of appropriate sites, we can alternatively *a posteriori* detect gene trees reflecting a bias due to negative or positive selection. Classical methods for evaluating selective pressures acting on homologous amino acid sequences are based on computing the ratio dN/dS of the number of non-synonymous (dN) versus synonymous (dS) nucleotide substitutions per site of a pairwise alignment [39]. Synonymous substitutions are those that do not result in change of amino acid (for instance most changes at the third codon position), while non-synonymous substitutions are those altering the amino acid (for instance changes at the second codon position). Under negative (purifying) selection, most non-synonymous changes are eliminated, leading to an excess of synonymous changes. On the other hand, positive selection leads to an excess of non-synonymous substitutions. In general, negative selection is inferred if $dN/dS < 1$ and positive selection is inferred if $dN/dS > 1$. We suggest the use of the synonymous/non-synonymous substitution rate measure for detecting gene trees reflecting a selection bias, formalized as trees reflecting the *isolocalization property* which is defined below.

12.4.2 Formalizing the Functional Bias

Under the hypothesis that after a duplication, exactly one of the two gene copies preserve the parental function, the *isolocalization property* was introduced in [53], to characterize gene trees biased towards a grouping of isorthologous genes. Here, we define a less constraining version of this property by asking for at least one isorthogroup to appear as a monophyletic group (an isolated subtree). Notice that results obtained in [53] (stated below and summarized in Sect. 12.5) about the effect on reconciliation remain valid for this new definition.

Definition 8 (Isolocalization) Let G be a gene tree for a gene family Γ. Let $I = \{a_1, a_2, \ldots, a_n\} \subseteq \Gamma$ be a maximal isorthogroup of Γ, meaning that no other gene of Γ is isorthologous to an a_i. A gene tree G respects the *isolocalization* property for I if and only if there exists an x such that $\mathcal{L}(G_x) = I$.

We say that G *respects the isolocalization property* if G respects the isolocalization property for at least one maximal isorthogroup of Γ.

We showed in [53] that isolocalization confounds reconciliation, in the sense that some histories (those with a duplication node descending from a speciation node) can never be recovered through the reconciliation of a gene tree respecting the isolocalization property. Following this observation, we proposed general ideas for inferring true histories. Although presented as tools for correcting reconciliation, they can alternatively be seen as tools for correcting gene trees, i.e. removing the functional constraints exhibited by isorthogroups. An overview of the related open problems is given in Sect. 12.5.

In the following, an *isorthologous subtree* of G is a speciation subtree of G with a set of leaves corresponding a maximal isorthogroup.

12.4.3 Results

By definition, a subtree G_x rooted at node x of a gene tree G is an *isorthologous subtree* if $\mathcal{L}(G_x)$ is a maximal isorthogroup, i.e. elements of $\mathcal{L}(G_x)$ are pairwise isorthologous, and there is no gene outside $\mathcal{L}(G_x)$ which is isorthologous to a gene of $\mathcal{L}(G_x)$. As suggested by the discussion above, this can be tested by comparing the dN/dS ratios of pairs (I_i, I_j) of genes inside $\mathcal{L}(G_x)$, versus pairs (I_k, O_l), with I_k being a gene inside $\mathcal{L}(G_x)$ and O_l being a gene outside $\mathcal{L}(G_x)$. Here, we consider the average dN/dS ratios over all possible pairs. Namely, we define M_x^I to be the average over all (I_i, I_j) *inside pairs* and M_x^O to be the average over all (I_k, O_l) *inside-outside pairs*. For an isorthologous subtree, we expect $\frac{M_x^I}{M_x^O}$ to be lower than one. For any internal node x, if $\frac{M_x^I}{M_x^O} < 1$ we say x is a *winner*; otherwise we say that x is a *loser*. Note that the root of a tree cannot be a winner, since there are no genes outside of its leafset.

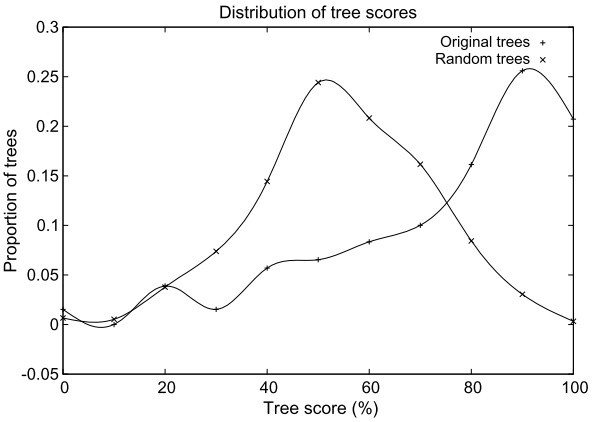


Fig. 12.4 Distribution of original trees scores versus random trees scores. The score of a tree is its number of winner nodes over its number of internal nodes

We wanted to see to what extent the Ensembl gene trees reflect a natural selection bias. We considered the same six species as in Sect. 12.3.4, namely four fish species (Stickleback, Medaka, Tetraodon, Zebrafish) with human and mouse as outgroups. We collected all available gene trees, restricted each of them to the taxa of interest, reconciled the trees with the known species trees, and retained the "interesting" ones according to [53], namely those reflecting a history with a "surviving" duplication followed by a "surviving" speciation event. More precisely, a gene tree G was retained if it contained at least one duplication node x such that G_{x_ℓ} and G_{x_r} were both speciation subtrees, each containing at least two leaves and at least five leaves together. This yielded 815 gene trees. We refer to this set as the *original set*. For each tree G in the original set, we obtained the canonical nucleotide sequences of its genes from Ensembl, and computed every pairwise dN/dS ratio using the PAML package [64], which implements the Nei and Gojobori method [40]. The sequences were aligned and prepared using ClustalW2 [37] in conjunction with the BioPerl library [50].

We expect the topology of each tree G in the original set to contain more winner nodes than most other topologies that share the same leaf set. We tested the null hypothesis, which states that there is no relationship between the gene trees constructed by Ensembl and the proportion of winner nodes they contain. Thus for each tree G, we considered a set of *random trees*, obtained from G by all possible permutations on its leaves. We refer to the set of random trees for all the Ensembl trees as the *random set*.

Figure 12.4 depicts, for both the original and the random tree sets, the proportion of trees by *score*, defined for each tree as the number of winners over the number of internal nodes. The original trees clearly tend to contain a higher ratio of winners than random trees. In fact, the random trees' percentages follow a distribution that is not far from normal, whereas the original trees favor higher scores, hinting at the invalidity of the null hypothesis.

Our analysis also showed some interesting numbers. Among all nodes (excluding roots and leaves), 71 % of them are winners in the original set, as compared to 50 % in the random set. Moreover, 81 % of the original trees have a majority of winner

nodes (more than half), compared to 49 % for random trees. Say that a gene tree G is *optimal* if the number of winner nodes in G is no less than the number in all random trees for G. We find that the proportion of optimal gene trees over the original set is 45 %. Moreover, if we also count as winner nodes those having a winner ancestor (i.e. not only those pointing to isorthogroup but also to subsets of isorthogroups), then the proportion of optimal trees raises to 64 % of all original trees. Finally, 80 % of the original trees have more winner nodes than at least half of their random trees.

More detailed statistical analysis are required to establish criteria for detecting functional bias in a gene tree according to dN/dS ratios. However, this preliminary study already reveals a possible negative selection bias in these Ensembl trees.

12.5 Gene Tree Correction

A significant obstacle to our understanding of evolution is the difficulty of inferring accurate gene trees. It is now clear that methodology based solely on sequence similarity are unable to produce a single well supported gene tree [42, 43, 59, 61]. Opposite to such a "sequence only" paradigm is the "sequence free" paradigm that does not directly use the sequence information. An example is the polynomial-time algorithm developed by Durand et al. [17] for inferring a gene tree minimizing the reconciliation cost with a given species tree. Such an extreme strategy is of theoretical interest only, as an accurate reconstruction model should be "hybrid", e.g. account for both sequence and genomic information, the challenge being to find the right balance between the two. Later in the same paper, a hybrid approach is in fact presented.

Each one of the genomic constraints we have introduced in this paper can be used to define, in the space of gene trees, points that best reflect the desired properties. As exploring the space of all topologies is time and space prohibitive, gene tree correction methods explore the neighborhood of an input gene tree G, according to a tree-distance measure, such as the Robinson–Foulds [11, 26], Nearest Neighbor Interchange (NNI) [13, 24, 25], Subtree Prune and Regraft (SPR), or Tree Bisection and Reconnection (TBR) [10] distances. In order to reduce the space of explored gene trees, tree moves may be restricted to edges deemed suspect by the user, typically those with low bootstrap values [13, 17].

As in Durand et al., almost all hybrid methods that have been developed so far are "species tree-aware" and consist in selecting, from a given neighborhood, a tree minimizing a reconciliation distance with a species tree. Beside reconciliation, other criteria such as the number of NAD nodes [12, 16, 52] may be considered for a "species tree-aware" hybrid method. On the other hand, a "gene order aware" method would select, in a given neighborhood of G, the trees avoiding or minimizing gene order inconsistencies (Sect. 12.3). A "negative selection aware" method would select appropriate alternative trees, as we explain in Sect. 12.5.

A wide range of theoretical and analytical open problems are implicit in the last paragraph. In addition to developing the right data structures and algorithms for efficient exploration of the neighborhood of a gene tree, the challenge is to explore

ways of combining multiple criteria in a unified framework. Do repairs to a gene tree suggested by the diversity of constraints coincide, or do they conflict? If they conflict, how should relative importance be distributed over the various constraints?

Another concern is the development of a unified approach that accounts for both sequence and genomic constraints simultaneously. Indeed, a significant drawback of the hybrid methods developed so far is the sequential manner in which the sequence and genomic information are considered; the corrected gene tree is not subsequently evaluated according to the sequence information, and thus may over fit the species tree. From this perspective, an interesting framework is the one used in TreeFix [62], as well as PhylDog [3] and Spimap [43]. Taking advantage of the fact that phylogenetic methods usually lead to a set of statistically equivalent gene trees, TreeFix is based on a heuristic that searches, among all topologies that are statistically equivalent to the input tree, one that minimizes a user-defined reconciliation cost. The implicit hypothesis used in TreeFix is that regions of tree space with high sequence likelihood and low reconciliation cost overlap, which they show to be true in practice. Such a general framework can easily be adapted to account for various types of constraints. However, the more constraints simultaneously considered, the more challenging the problem of attributing relative weights to each of them and managing conflicting requirements become (see also chapter Chauve et al. in this volume).

We conclude this section by highlighting important results obtained in [53] that show how the selection bias, formalized as the isolocalization property, can be used for gene tree correction.

Isorthology Respecting Histories As recalled in Sect. 12.4.2, we showed that gene trees respecting the isolocalization property can lead to erroneous histories through reconciliation. This observation is not surprising as a gene tree reflecting functional constraints rather than evolutionary constraints can hardly be confidently used to infer evolutionary scenarios. Yet there must be some information in the gene tree and species tree relationship. For instance, we expect subtrees corresponding to isorthogroups in a well-supported gene tree to agree with the species tree. Define a *speciation subtree* of G to be a subtree such that all internal nodes (if any) are labeled as speciations by the reconciliation. The following result comes from Corollary 3 of [53], and is adapted to our new definition of the isolocalization property.

Theorem 1 *Let G be a gene tree satisfying the isolocalization property for an isorthogroup I and reflecting the true phylogeny for I (see a precise definition in [53]). Then I appears in G as the leaf-set of a speciation subtree.*

Based on Theorem 1, the following definition can be used for gene tree correction.

Definition 9 (Isorthology respecting history (IRH)) Given a gene tree G and a species tree S, a *dls*-history H is an *isorthology respecting history* for (G, S) if and only if each isorthogroup inferred from H is the leaf-set of a speciation subtree of G.

Following a duplication, we assume that one of the two gene copies preserves the ancestral function (Hypothesis 1 in [53]). Suppose that gene related by speciation preserve the ancestral function. Then two isorthogroups $\{M1, S1, T1, Z1\}$ and $\{M2, S2\}$ are inferred from the history H in Fig. 12.5, and H is an isorthology respecting history for (G, S). Notice that H leads to the gene tree G', which can be seen as a correction of G.

As many IRHs are possible for a given pair (G, S), an appropriate criterion for choosing most likely histories is required. For example the history R resulting from the reconciliation of G with S in Fig. 12.5 is also an isorthology respecting history for (G, S). However, while R has a mutation cost of 3 (one duplication and two losses), the history H has a mutation cost of one (no loss). In [53] we considered the Minimum Isorthology Respecting History Reconstruction (MIRH) Problem, which asks for the IRH of minimum cost, and developed a linear-time algorithm for the duplication cost. An algorithm for the mutation cost remains open.

The MIRH optimization problem as stated, is very conservative, in the sense that nothing is trusted in the gene tree except the isorthology information. In particular, it ignores all the information on duplication and speciation nodes of G that are above the considered speciation subtrees. An alternative would be to account for the hierarchy of deeper nodes in G. The notion of a *Triplet Respecting History* (TRH) [53] is intended to account for such hierarchy. Efficient algorithms for inferring parsimonious TRHs remain undiscovered.

Notice that Theorem 1 does not *a priori* give us the isorthogroups for a pair (G, S), as the true isorthologous subtree could be part of a larger speciation subtree. A restricted version of the MIRH problem considers the maximal speciation subtrees of G as the definition of the isorthogroups. We showed in [53] that this isorthology respecting partition of G is the one that would minimize the duplication cost, but not necessarily the mutation cost.

An alternative approach would use some isorthogroup detection criteria, such as the one given in Sect. 12.4.1, and correct according to the corresponding isorthologous subtrees. Such targeted reconstruction algorithms remain completely unexplored.

12.6 Conclusion

While gene trees have traditionally been constructed and validated using nucleotide sequence or amino acid sequence information alone, more recently information from the species tree has been used to both correct and validate gene trees. We have introduced new methodology to further validate and correct gene trees through the use of other data. Our novel use of syntenic information (homologous regions) points to a significant number of flawed gene trees in the Ensembl database due to homology contradiction or region overlapping. Our use of the dN/dS ratio on gene trees points to a bias towards clustering of isorthologous genes in gene trees. Although some potential avenues for improving gene trees are explored, our results seem to pose more questions than they answer.

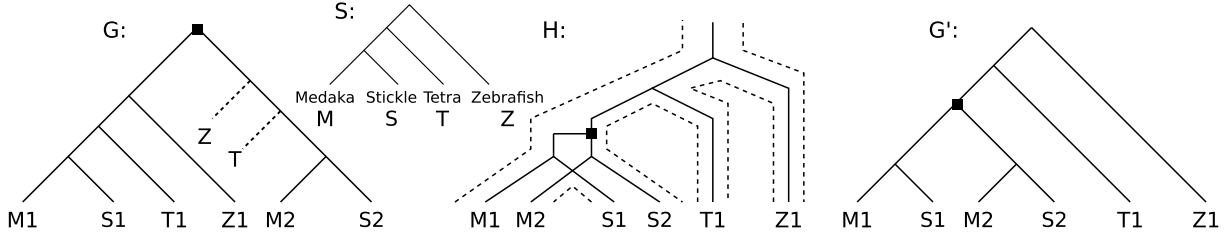


Fig. 12.5 G is the gene tree for the gene family "C2orf47" from Ensembl (ENSGT00390000004145), extended with loss leaves according to a reconciliation with species tree S. M is Medaka, S is Stickleback, T is Tetraodon, and Z is Zebrafish. Reconciliation of G with the species tree S gives one duplication and two losses (as marked in G, duplication by a *square*, and losses by *dotted lines*). Considering the largest speciation subtrees of G as pointing to the isorthogroups ($\{M1, S1, T1, Z1\}$ and $\{M2, S2\}$), H is an isorthology respecting history for (G, S) leading to the gene tree G'

References

1. Akerborg, O., Sennblad, B., Arvestad, L., Lagergren, J.: Simultaneous Bayesian gene tree recons. and reconciliation analysis. Proc. Natl. Acad. Sci. **106**(14), 5714–5719 (2009)
2. Arvestad, L., Berglund, A.C., 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)
3. Boussau, B., Szllosi, G.J., Duret, L., Gouy, M., Tannier, E., Daubin, V.: Genome-scale coestimation of species and gene trees. Genome Res. **23**, 323–330 (2013)
4. Beiko, R.G., Hamilton, N.: Phylogenetic identification of lateral genetic transfer events. BMC Evol. Biol. **6**(15) (2006)
5. Bergeron, A., Chauve, C., Gingras, Y.: Formal models of gene clusters. In: Mandoiu, I., Zelikovsky, A. (eds.) Bioinformatics Algorithms: Techniques and Applications. Wiley, New York (2008). Chap. 8
6. Bergeron, A., Corteel, S., Raffinot, M.: The algorithmic of gene teams. In: Algorithms in Bioinformatics, pp. 464–476 (2002)
7. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. J. Comput. Biol. **13**, 1340–1354 (2003)
8. Berglund-Sonnhammer, A.C., Steffansson, P., Betts, M.J., Liberles, D.A.: Optimal gene trees from sequences and species trees using a soft interpretation of parsimony. J. Mol. Evol. **63**, 240–250 (2006)
9. Chang, W.C., Eulenstein, O.: Reconciling gene trees with apparent polytomies. In: Chen, D.Z., 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)
10. Chaudhary, R., Burleigh, J.G., Eulenstein, O.: Efficient error correction algorithms for gene tree reconciliation based on duplication, duplication and loss, and deep coalescence. BMC Bioinform. **13**(Suppl. 10), S11 (2011)
11. Chaudhary, R., Burleigh, J.G., Fernandez-Baca, D.: Fast local search for unrooted Robinson–Foulds supertrees. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**(4), 1004–1012 (2012)
12. Chauve, C., El-Mabrouk, N.: New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In: RECOMB 2009. LNCS, vol. 5541, pp. 46–58. Springer, Berlin (2009)
13. Chen, K., Durand, D., Farach-Colton, M.: Notung: dating gene duplications using gene family trees. J. Comput. Biol. **7**, 429–447 (2000)
14. Dondi, R., El-Mabrouk, N.: Minimum leaf removal for reconciliation: complexity and algorithms. In: CPM. Lecture Notes in Computer Science, vol. 7354, pp. 399–412. Springer, Berlin (2012)
15. Dondi, R., El-Mabrouk, N., Swenson, K.M.: Gene tree correction for reconciliation and species tree inference: complexity and algorithms. J. Discrete Algorithms (2013). doi:10.1016/j.jda.2013.06.001
16. Doroftei, A., El-Mabrouk, N.: Removing noise from gene trees. In: WABI. LNBI/LNBI, vol. 6833, pp. 76–91 (2011)
17. Durand, D., Haldórsson, B.V., Vernot, B.: A hybrid micro-macroevolutionary approach to gene tree reconstruction. J. Comput. Biol. **13**, 320–335 (2006)
18. Durand, D., Sankoff, D.: Tests for gene clustering. J. Comput. Biol. **10**(3–4), 453–482 (2003)
19. Eulenstein, O., Mirkin, B., Vingron, M.: Duplication-based measures of difference between gene and species trees. J. Comput. Biol. **5**, 135–148 (1998)
20. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**, 368–376 (1981)
21. Felsenstein, J.: PHYLIP(phylogeny inference package). Version 3.6 distributed by the author, Seattle (WA): Department of Genome Sciences, University of Washington (2005)
22. Fitch, W.M.: Homology: a personal view on some of the problems. Trends Genet. **16**(5), 227–231 (2000)

23. Flicek, P., Amode, M.R., Barrell, D., Beal, K., Brent, S., Carvalho-Silva, D., Clapham, P., Coates, G., Fairley, S., Fitzgerald, S., Gil, L., Gordon, L., Hendrix, M., Hourlier, T., Johnson, N., Khri, A.K., Keefe, D., Keenan, S., Kinsella, R., Komorowska, M., Koscielny, G., Kulesha, E., Larsson, P., Longden, I., McLaren, W., Muffato, M., Overduin, B., Pignatelli, M., Pritchard, B., Riat, H.S., Ritchie, G.R., Ruffier, M., Schuster, M., Sobral, D., Tang, Y.A., Taylor, K., Trevanion, S., Vandrovcova, J., White, S., Wilson, M., Wilder, S.P., Aken, B.L., Birney, E., Cunningham, F., Dunham, I., Durbin, R., Fernndez-Suarez, X.M., Harrow, J., Herrero, J., Hubbard, T.J., Parker, A., Proctor, G., Spudich, G., Vogel, J., Yates, A., Zadissa, A., Searle, S.M.: Ensembl 2012. Nucleic Acids Res. **40**(Database Issue), D84–D90 (2012)
24. Gorecki, P., Eulenstein, O.: Algorithms: simultaneous error-correction and rooting for gene tree reconciliation and the gene duplication problem. BMC Bioinform. **13**(Suppl. 10), S14 (2011)
25. Gorecki, P., Eulenstein, O.: A linear-time algorithm for error-corrected reconciliation of unrooted gene trees. In: ISBRA. LNBI, vol. 6674, pp. 148–159. Springer, Berlin (2011)
26. Gorecki, P., Eulenstein, O.: A Robinson–Foulds measure to compare unrooted trees with rooted trees. In: Bleris, L. et al. (eds.) ISBRA. LNBI, vol. 7292, pp. 115–126 (2012)
27. Gorecki, P., Tiuryn, J.: DLS-trees: a model of evolutionary scenarios. Theor. Comput. Sci. **359**, 378–399 (2006)
28. Guidon, S., Gascuel, O.: A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. **52**, 696–704 (2003)
29. Philippe, H., Lopez, P., Brinkmann, H., Budin, K., Germot, A., Laurent, J., Moreira, D., Muller, M., Le Guyader, H.: Early-branching or fast-evolving eukaryotes? An answer based on slowly evolving positions. Proc. R. Soc. Lond. B, Biol. Sci. **267**, 1213–1221 (2000)
30. Hahn, M.W.: Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. Genome Biol. 8(R141) (2007)
31. Heber, S., Stoye, J.: Algorithms for finding gene clusters. In: Algorithms in Bioinformatics, pp. 252–263 (2001)
32. Li, H., Coghlan, A., Ruan, J., Coin, L.J., Hrich, J.K., Osmotherly, L., Li, R., Liu, T., Zhang, Z., Bolund, L., Wong, G.K., Zheng, W., Dehal, P., Wang, J., Durbin, R.: TreeFam: a curated database of phylogenetic trees of animal gene families. Nucleic Acids Res. **34**(D572), 580 (2006)
33. Hoberman, R., Durand, D.: The incompatible desiderata of gene cluster properties. In: Comparative Genomics, pp. 73–87 (2005)
34. Hoberman, R., Sankoff, D., Durand, D.: The statistical analysis of spatially clustered genes under the maximum gap criterion. J. Comput. Biol. **12**(8), 1083–1102 (2005)
35. Koonin, E.V.: Orthologs, paralogs and evolutionary genomics. Annu. Rev. Genet. **39**, 309–338 (2005)
36. Lafond, M., Swenson, K.M., El-Mabrouk, N.: An optimal reconciliation algorithm for gene trees with polytomies. In: WABI. LNCS, vol. 7534, pp. 106–122 (2012)
37. Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., McGettigan, P.A., McWilliam, H., Valentin, F., Wallace, I.M., Wilm, A., Lopez, R., Thompson, J.D., Gibson, T.J., Higgins, D.G.: Clustalw and clustalx version 2. Bioinformatics **23**, 2947–2948 (2007)
38. Massey, S.E., Churbanov, A., Rastogi, S., Liberles, D.A.: Characterizing positive and negative selection and their phylogenetic effects. Gene **418**, 22–26 (2008)
39. Miyata, T., Yasunaga, T.: Molecular evolution of MRNA: a method for estimating evolutionary rates of synonymous and amino acid substitutions from homologous nucleotide sequences and its application. J. Mol. Evol. **16**(1), 23–36 (1980)
40. Nei, M., Gojobori, T.: Simple methods for estimating the number of synonymous and nonsynonymous nucleotide substitutions. Mol. Biol. Evol. **3**, 418–426 (1986)
41. 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**(12) (2013)

42. Rasmussen, M.D., Kellis, M.: Accurate gene-tree reconstruction by learning gene and species-specific substitution rates across multiple complete geneomes. Genome Res. **17**, 1932–1942 (2007)
43. Rasmussen, M.D., Kellis, M.: A Bayesian approach for fast and accurate gene tree reconstruction. Mol. Biol. Evol. **28**(1), 273–290 (2011)
44. Ronquist, F., Huelsenbeck, J.P.: MrBayes3: Bayesian phylogenetic inference under mixed models. Bioinformatics **19**, 1572–1574 (2003)
45. Ruan, J., Li, H., Chen, Z., Coghlan, A., Coin, L.J., Guo, Y., Hrich, J.K., Hu, Y., Kristiansen, K., Li, R., Liu, T., Moses, A., Qin, J., Vang, S., Vilella, A.J., Ureta-Vidal, A., Bolund, L., Wang, J., Durbin, R.: TreeFam: 2008 update. Nucleic Acids Res. **36**(Suppl. 1), D735–D740 (2008)
46. Ruano-Rubio, V., Fares, V.: Artifactual phylogenies caused by correlated distribution of substitution rates among sites and lineages: the good, the bad and the ugly. Syst. Biol. **56**, 68–82 (2007)
47. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
48. Sankoff, D., Ferretti, V., Nadeau, J.H.: Conserved segment identification. J. Comput. Biol. **4**(4), 559–565 (1997)
49. Skovgaard, M., Kodra, J.T., Gram, D.X., Knudsen, S.M., Madsen, D., Liberles, D.A.: Using evolutionary information and ancestral sequences to understand the sequence-function relationship in GLP-1 agonists. J. Mol. Biol. **363**, 977–988 (2006)
50. Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G., Korf, I., Lapp, H., Lehvslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D., Birney, E.: The bioperl toolkit: Perl modules for the life sciences. Genome Res. **12**, 1611–1619 (2002)
51. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analysis with thousands of taxa and mixed models. Bioinformatics **22**, 2688–2690 (2006)
52. Swenson, K.M., Doroftei, A., El-Mabrouk, N.: Gene tree correction for reconciliation and species tree inference. Algorithms Mol. Biol. **7**(31) (2012)
53. Swenson, K.M., El-Mabrouk, N.: Gene trees and species trees: irreconcilable differences. BMC Bioinform. **13**(Suppl. 19), S15 (2012)
54. Swofford, D.L.: PAUP: Phylogenetic Analysis Using Parsimony, 4th edn. Sinauer Associates, Sunderland (2002)
55. Taylor, S.D., de la Cruz, K.D., Porter, M.L., Whiting, M.F.: Characterization of the long-wavelength opsin from Mecoptera and Siphonaptera: does a flea see? Mol. Biol. Evol. **22**, 1165–1174 (2005)
56. Theobald, D.L.: A formal test of the theory of universal common ancestry. Nature **465**(7295), 219–222 (2010)
57. Thomas, P.D.: GIGA: a simple, efficient algorithm for gene tree inference in the genomic age. BMC Bioinform. **11**, 312 (2010)
58. Townsend, J.P.: Profiling phylogenetic informativeness. Syst. Biol. **56**, 222–231 (2007)
59. Vilella, A.J., Severin, J., Ureta-Vidal, A., Heng, L., Durbin, R., Birney, E.: EnsemblCompara gene trees: complete, duplication-aware phylogenetic trees in vertebrates. Genome Res. **19**, 327–335 (2009)
60. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Automatic genome-wide reconstruction of phylogenetic gene trees. Bioinformatics **23**(13), i549–i558 (2007)
61. Wong, K.M., Suchard, M.A., Huelsenbeck, J.P.: Alignment uncertainty and genomic analysis. Science **319**, 473–476 (2008)
62. Wu, Y.C., Rasmussen, M.D., Bansal, M.S., Kellis, M.: TreeFix: statistically informed gene tree error correction using species trees. Syst. Biol. **62**(1), 110–120 (2013)
63. Xu, X., Sankoff, D.: Tests for gene clusters satisfying the generalized adjacency criterion. In: Advances in Bioinformatics and Computational Biology, pp. 152–160 (2008)
64. Yang, Z.: Paml 4: phylogenetic analysis by maximum likelihood. Mol. Biol. Evol. **24**, 1586–1591 (2007)

65. Yang, Z., Sankoff, D.: Natural parameter values for generalized gene adjacency. J. Comput. Biol. **17**(9), 1113–1128 (2010)
66. Zhu, Q., Adam, Z., Choi, V., Sankoff, D.: Generalized gene adjacencies, graph bandwidth, and clusters in yeast evolution. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(2), 213–220 (2009)
67. Zmasek, C.M., Eddy, S.R.: A simple algorithm to infer gene duplication and speciation events on a gene tree. Bioinformatics **17**, 821–828 (2001)

Chapter 13
The Potential of Family-Free Genome Comparison

Marília D.V. Braga, Cedric Chauve, Daniel Doerr, Katharina Jahn, Jens Stoye, Annelyse Thévenin, and Roland Wittler

Abstract Many methods in computational comparative genomics require gene family assignments as a prerequisite. While the biological concept of gene families is well established, their computational prediction remains unreliable. This paper continues a new line of research in which family assignments are not presumed. We study the potential of several family-free approaches in detecting conserved structures, genome rearrangements and in reconstructing ancestral gene orders.

13.1 Introduction

In more than 20 years of research in computational comparative genomics [44, 49] a large variety of questions have been addressed. By now, strong methods are available to study the structural organization of genomes as well as to unravel their shared and individual evolutionary histories. The structural organization of genomes does not only give insights into species' phylogeny, but also hints at interactions within and between sets of genes by means of their involvement in metabolic and regulatory networks. As such, one aims to understand cell functions. Whereas point mutations generally affect one or a few nucleotides, large-scale mutations such as rearrangements, deletions, substitutions, or insertions affect one or more genes. These modifications alter the structural organization of the genome which can cause profound

M.D.V. Braga
Inmetro, Duque de Caxias, Brazil

C. Chauve
Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

C. Chauve
LaBRI, Université Bordeaux I, Talence, France

D. Doerr · K. Jahn · J. Stoye (✉) · A. Thévenin · R. Wittler
Genome Informatics, Faculty of Technology, Bielefeld University, Bielefeld, Germany
e-mail: jens.stoye@uni-bielefeld.de

D. Doerr · K. Jahn · J. Stoye · A. Thévenin · R. Wittler
Institute for Bioinformatics, CeBiTec, Bielefeld University, Bielefeld, Germany

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_13,

changes in the cellular machinery. Identifying and quantifying such structural modifications is crucial in understanding the highly complex functions of organisms and their interactions with the natural environment.

Initial approaches to study genome rearrangement considered pairwise comparisons with well identified one-to-one orthologous markers [44], for many of which polynomial time algorithms for computing distances and evolutionary scenarios could be designed [4, 6, 30, 47, 63]. Extensions considering more than two genomes lead to hard problems [8, 13, 15, 41, 47, 61], with few exceptions [26, 54]. David Sankoff initiated formulations and algorithms for genome rearrangement problems with duplicated markers originating from gene families [45], quickly followed by the outline of a general approach that would consider both gene orders and gene family information as input to genome rearrangement problems [48]. Since then, genome rearrangement with unequal gene content and gene families, where genomes are represented by signed sequences, has been intensively explored; for reviews see [17, 27].

Another line of research in computational genomics aims at the detection of genomic segments that are conserved across different species. The presence of such structures often hints at functional coupling of the contained genes, or indicates remnant ancestral gene order which is valuable information for phylogenetic reconstruction. Initial approaches in this field—like early rearrangement studies—required the identification of one-to-one orthologous markers [5, 32, 33], but in the following most of them were adapted to a more general genome model that allows genomes to differ in their marker set and to have homologous markers on the same genome [21, 31, 50].

All of the above methods, which we call *family-based*, require prior gene family assignments. However, biological gene families are difficult to assess; commonly, they are predicted computationally. In doing so, they can be either obtained from databases [42, 55, 59] or directly computed based on the particular dataset under consideration [36, 40, 51]. In either case, the obtained assignments are predicted by some computational method which typically involves a clustering phase in which genes are partitioned into groups representing the predicted families. Generally, the results of such efforts depend on arbitrary parameters of sequence comparison, similarity quantification and clustering. These parameters are user-controlled and influence the size and granularity of the computed gene families. In particular, when genes within biological gene families are largely diverged, computational means may not be able to resolve gene family assignments accurately [28]. Consequently, errors are introduced into the primary dataset which deteriorate subsequent analyses, a phenomenon that can be amplified when phylogenetic trees for the gene families are considered [17, 39]. The quest to reduce misassignments in gene family construction also led to the use of positional homology [10, 57, 58, 65].

Recently, in an attempt to avoid these problems, a *family-free* method, which does not assume prior gene family assignment, has been proposed for computing the adjacency score between two genomes [22]. In this approach, given the gene similarities, the aim is to find pairwise gene assignments while maximizing the conserved adjacency measure. In other words, next to finding the maximal number of

adjacent genes along different genomes, the method also infers homologies between genes. It should be noted that these homologies are not equivalent to gene families in the classical sense, as by design only one-to-one relationships are detected, while a gene family in general may consist of a potentially large set of orthologous and paralogous genes. Given the nature of the detected one-to-one relationships, they are not unlikely to form sub-families of biological gene families. Therefore they can be further utilized in gene family construction.

Here we go beyond this one application and explore how various problems in computational comparative genomics could be approached in a family-free setting. We do not necessarily provide full solutions to the proposed problems.

This paper is organized as follows. After basic definitions in Sect. 13.2, we extend earlier results on the adjacency measure to more than two genomes and to larger conserved structures (gene clusters) in Sect. 13.3. A more dynamic view is taken in Sect. 13.4, where we apply the ideas to rearrangement distances, most notably the Double Cut and Join distance. In Sect. 13.5, finally, we indicate how the family-free approach could be further extended to the reconstruction of ancestral genomes. The paper concludes with a discussion in Sect. 13.6.

13.2 Basic Definitions

A chromosome is a DNA molecule composed of antiparallel strands and can be read in either of the two possible directions. Since each gene, representing an interval along the DNA, lies in one of the two strands of the chromosome, the orientation of the gene depends on the adopted reading direction. The representation of a gene g in a chromosome can then be the symbol g, if it is read in direct orientation, or the symbol $\overline{g}$, if it is read in reverse orientation. Without loss of generality, we will assume in this paper that each chromosome has a canonical reading direction, giving a natural left to right order of its genes.

A genome consists of one or more chromosomes that can be either linear or circular. For ease of presentation, throughout this paper we will consider only unichromosomal linear genomes. The general case can be easily inferred with minor modifications.

A unichromosomal linear genome is represented as a sequence of distinct symbols, flanked by telomeric ends indicated by the $\circ$ sign: $G = (\circ\ g_1\ g_2\ \ldots\ g_n\ \circ)$. The size of G with n genes and two telomeric ends is $|G| = n + 2$. When we consider a set of genomes, we will assume that all genes can be distinguished from each other, i.e., every two genomes $G \neq H$ share only the telomeric ends.

Let $\mathcal{A}$ be the universe of all genes and let $\sigma : \mathcal{A} \times \mathcal{A} \rightarrow [0, 1]$ be a *normalized similarity measure* between all pairs of genes.

Definition 1 (Gene similarity graph) For a set of k genomes $\{G_1, \ldots, G_k\}$, the *gene similarity graph* is defined as an ordered weighted undirected k-partite graph $B = (G_1, \ldots, G_k, E)$, where each gene and each telomere represents a node, and

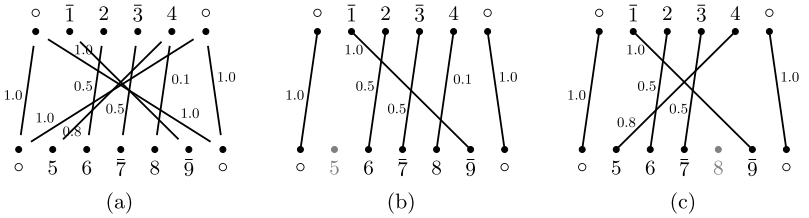


Fig. 13.1 (**a**) Example of a gene similarity graph for $k = 2$. Part (**b**) shows a matching in which the weak edge with weight 0.1 between genes 4 and 8 is selected, creating a conserved adjacency between $(\bar{3}, 4)$ and $(\bar{7}, 8)$. In the matching of (**c**) the stronger edge with edge weight 0.8 between genes 4 and 5 is selected

the nodes are ordered following the chromosomal order. Any two genes g and h, belonging to two distinct genomes, are connected by an edge $e_{g,h} \equiv \{g, h\} \in E$ with weight $w(e_{g,h}) := \sigma(g, h)$, if and only if $\sigma(g, h) > 0$. Telomeres in distinct genomes are always connected with edges of weight 1.

We call a gene $g \in G$ *unconnected* if there exists no other gene h in any of the other genomes $H \neq G$ such that $\sigma(g, h) > 0$. An example of a gene similarity graph for the case $k = 2$ is shown in Fig. 13.1(a). The k-partite gene similarity graph features similarity relationships between genes of different genomes whereas similarities between genes within the same genome are ignored. For now, if information about paralogous relationships between genes within the same genome is desired, it must be gained through a post-processing step incorporating the results obtained by the methods presented herein.

13.3 Detecting Conserved Structures

Many gene order studies quantify conserved structures based on well-defined proximity relations between the chromosomal locations of pairs or groups of genes. Typical proximity relations between pairs of genes are conserved adjacencies [44, 46, 60] and generalized conserved adjacencies [62], whereas proximity relations between groups of genes include common intervals [21, 33, 50, 56], max gap clusters (gene teams) [5, 31], approximate common intervals [12, 34, 43], generalized adjacency clusters [64, 67], and conserved intervals [4]. We discuss conserved adjacencies in Sect. 13.3.1 and common intervals and some of its derivatives in Sect. 13.3.2.

Whenever one-to-one relationships between genetic markers, genes or genome segments (identified through some proximity relation) between genomes must be established, comparative genomics applications commonly incorporate matchings. For example, in aligning whole genomes, one aims to find a matching between genome segments that maximizes the similarity of the respective sequences, but also minimizes the number of breakpoints (or other measures of structural dissimilarity) in the final ordering of segments [19]. Similarly, recent methods in predicting co-orthologs and gene families not only assess the sequence similarity between genes,

but also their position within the genome [20]. In the following we describe approaches that incorporate matchings to identify conserved adjacencies and common intervals without the use of gene family assignments.

13.3.1 Conserved Adjacencies

Previous Work Two genes that are located next to each other in a genome are said to be adjacent, their adjoining extremities form an *adjacency*. An early measure for family-based genome similarity was to count the number of *conserved adjacencies*, i.e. those adjacencies that are common to two genomes, with the restriction that the gene content of both genomes is identical [44, 60]. Thereby, the number of conserved adjacencies constitutes the dual measure of the number of breakpoints between both sequences [46].

With the adoption of gene families, gene duplicates are introduced, i.e., the occurrence of several members of the same family in one genome [45, 48]. Gene duplicates allow for multiple scenarios of ancestral gene order. One possibility to resolve the consequential ambiguities consists in computing a matching between orthologous subsets of given family members, with some predefined constraints on the structure of the matching. This general principle, which relates also to ortholog identification [20], was introduced by David Sankoff with the notion of *exemplar distance* [45], where the main ortholog (the exemplar) of each family is kept. This initial model was later generalized to less constrained classes of matchings where one or more genes per family is kept, always leading to NP-hard computational problems [2, 11, 66], although practically efficient solutions were designed, using heuristics [29] or integer linear programming [1].

Family-Free Adjacencies Recently, a gene family-free model was introduced to compute the number of conserved adjacencies in pairwise comparison [22]. The computational problem being NP-hard, exact and heuristic algorithms were presented with feasible running times in practice. In this section, we advance towards a more general model applicable for the simultaneous study of several genomes. Conserved adjacencies obtained in this approach can further benefit ancestral genome reconstruction, as it will be explained in Sect. 13.5.

The genome model described in Sect. 13.2 is neither restricted to one-to-one relations between genes, nor to closed sets of gene family members. In the subsequent analysis, unconnected genes are omitted from the chromosomal sequences. The remaining genes form connected components of size two or larger. Their size is typically greater than their gene family counterparts. Further, opposing the gene family concept, these connected components are not required to equal their transitive closure.

Given $k \geq 2$ genomes, we aim to find a matching between genes, analogous to previous family-based approaches [1, 9, 45]. One way is to find all completely connected subgraphs of size k in the gene similarity graph and then perform a k-dimensional matching (also known as k-matching). Yet, this approach eliminates

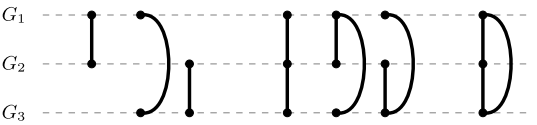


Fig. 13.2 The 7 valid types of components of a partial 3-matching

many connected components that do not form complete cliques or spread over only a smaller subset of genomes. Consequently, with increasing number of genomes in the dataset, the matching size will decrease until only few fully connected genes remain. In this work we use a *partial k-matching* which allows for missing genes and edges:

Definition 2 (Partial k-matching) Given a gene similarity graph $B = (G_1, \ldots, G_k, E)$, a *partial k-matching* $\mathcal{M} \subseteq E$ is a selection of edges such that for each connected component $C \subseteq B_{\mathcal{M}} := (G_1, \ldots, G_k, \mathcal{M})$ no two genes in C belong to the same genome.

Figure 13.2 depicts all valid types of components in a partial k-matching for $k = 3$. The partial k-matching is closely related to the *intermediate matching* [1] for $k = 2$. Just as in the latter, a partial k-matching can saturate an arbitrary number of edges of the initial k-partite graph B but differs in that it is not required to saturate at least one edge per connected component. Our motivation to reject this constraint is discussed further below.

Biological Interpretation Relating to the underlying mechanism of gene family evolution, a connected component in the partial k-matching represents a tentative sub-family assignment in which two intrinsic aspects of gene family prediction are addressed; first, the similarity measure between genes is generally not transitive; second, genes and gene families may arise or vanish along the evolutionary process whereas some genes that are intermittently indispensable for the organism emerge as main orthologs. The biological interpretation of the matching is limited by the restriction to one-to-one assignments between genes and by the fact that the matching does not consider the underlying phylogeny of species and thus is unable to differentiate between orthologs and paralogs. As such, our method is susceptible to non-ortholog assignments in entangled events of gene deletions. Thus it is deceptive to relate a connected component in the partial k-matching to an ortholog assignment. Rather, under the optimization problem stated further below, it represents a tentative sub-family determined by the most parsimonious homology assignment with respect to gene similarity and gene order.

Constructing a Partial *k*-Matching We assume for now that a partial k-matching $\mathcal{M}$ is given. For any two genomes G and H in the gene similarity graph we define $\mathcal{M}_{GH} \subseteq \mathcal{M}$ as the set of matched edges between G and H. We call a gene GH*-saturated* if it is incident to an edge in $\mathcal{M}_{GH}$. Two GH-saturated genes are *consecutive* with respect to G and H if no GH-saturated gene lies between them. Further, two pairs of consecutive GH-saturated genes (g, g') in genome G, with g to the left of g', and (h, h') in genome H, form a *conserved adjacency* if

(a) for h left of h' in H, $\text{sgn}(g) = \text{sgn}(h)$ and $\text{sgn}(g') = \text{sgn}(h')$ or
(b) for h right of h' in H, $\text{sgn}(g) \neq \text{sgn}(h)$ and $\text{sgn}(g') \neq \text{sgn}(h')$,

where the orientation of a gene (or telomere) g is determined by the following function:

$$\text{sgn}(g) = \begin{cases} 1 & \text{if } g \text{ is in forward direction} \\ -1 & \text{if } g \text{ is in backward direction} \\ 0 & \text{if } g \text{ is a telomere} \end{cases}$$

For example, the consecutive pair of genes $(2, \overline{3})$ and $(6, \overline{7})$ in Fig. 13.1(b) represent a conserved adjacency. Following [22], we define a scoring scheme for adjacencies:

$$s(g, g', h, h') = \begin{cases} \sqrt{w(e_{g,h}) \cdot w(e_{g',h'})} & \text{if } (g, g'), (h, h') \text{ form a cons. adjacency} \\ 0 & \text{otherwise} \end{cases}$$

The convex nature of the scoring scheme rewards conserved adjacencies between high weighted edges the most, whereas combinations of high and low weighted, or low weighted edges are decreasingly scored. While a matching that creates many conserved adjacencies is often more appreciated than a matching with few conserved adjacencies, maximizing the number of conserved adjacencies is not desirable at any price. For example, the matching depicted in Fig. 13.1(b) contains an adjacency between genes $(\bar{3}, 4)$ and $(7, 8)$ at the expense of dismissing the stronger edge between genes $(4, 8)$, which is selected in the matching displayed in Fig. 13.1(c). Hence we view a matching as a trade-off between two competing properties, namely similarity and synteny. We quantify both in a matching $\mathcal{M}$ between genomes $\mathcal{G} = \{G_1, \ldots, G_k\}$ by means of the following measures:

$$\text{adj}(\mathcal{M}) = \sum_{G,H \in \mathcal{G}} \sum_{\substack{g \text{ left of } g' \text{ in } G \\ h,h' \text{ in } H}} s(g, g', h, h'), \tag{13.1}$$

$$\text{edg}(\mathcal{M}) = \sum_{e \in \mathcal{M}} w(e). \tag{13.2}$$

Extending [22], we propose to find a partial k-matching that maximizes a linear combination of both quantities:

Problem 1 (FF-Adjacencies) Given a gene similarity graph $B = (G_1, \ldots, G_k, E)$ and some $\alpha \in [0, 1]$, find a partial k-matching $\mathcal{M}$ such that the following formula is maximized:

$$\mathcal{F}_\alpha(\mathcal{M}) = \alpha \cdot \text{adj}(\mathcal{M}) + (1 - \alpha) \cdot \text{edg}(\mathcal{M}). \tag{13.3}$$

Thereby α is a user-controlled parameter that can be adjusted in favor of similarity or synteny.

Rejection of Intermediate Matching Constraints Recall that a partial k-matching for $k = 2$ differs from the intermediate matching only by omitting the constraint that for each connected component at least one edge must be matched. While such restriction is reasonable in gene family studies, where family assignments act as filter in reducing false positive associations between genes, the gene similarity graph can include also small weakly connected components (depending on the particular similarity function) that most likely represent false positives. Substituting the intermediate matching which was used in the initial gene family-free approach [22] for the partial k-matching may have a crucial effect on α in solving Problem FF-Adjacencies. While in pairwise comparison where $\alpha = 0$, both matchings coincide, the choice of edges in the intermediate matching is increasingly limited, when $\alpha > 0$. Discarding the constraint of keeping at least one edge per connected component allows more freedom in the choice of edges included in the matching and thus may lower the number of false positive assignments. However, it does so at the cost of increasing the combinatorial solution space that must be explored in solving Problem FF-Adjacencies. That is because the constraints of the intermediate matching enable the reduction of the solution space by identifying anchors in the gene similarity graph. Using a partial k-matching, we lack sensible constraints of the matching that can be exploited to identify anchors beforehand. Nevertheless, heuristic methods can be applied to establish anchors based on highly conserved structures in the gene similarity graph that are likely preserved in optimal solutions of Problem FF-Adjacencies. These methods will not be discussed here.

13.3.2 Common Intervals

The concept of *common intervals* is used to represent two or more genomic segments (usually from different genomes) that are composed of the same set of genes. The presence of such segments in the genomes of different species suggests either functional coupling of the involved genes, as observed in operons in prokaryotes, or remnant ancestral gene order, often referred to as *syntenic blocks*, which are used to study large-scale genome evolution. Over the past years, the common intervals model has been generalized to increase its applicability: Starting from a model that requires genomes to be permutations of each other [32, 33, 56], it extended to a sequence-based model that allows multiple occurrences of the same gene and differences in the gene composition of genomes [21, 50]. Finally it was redefined in different ways to account for small differences in the gene content of otherwise well-conserved segments. The most notable of the latter extensions are *r-windows* [23], *max-gap clusters* [5, 31] and *approximate common intervals* [12, 34, 43].

Currently, all approaches to common interval detection require as a prerequisite that the genes of the studied genomes are partitioned into gene families. It is evident that errors in this assignment can have a negative impact on common intervals detection. In the classical common intervals model a single unrecognized homology can prematurely end a conserved segment, or even cause the whole segment

to remain unrecognized. Approximate common intervals are to some extent robust against errors in gene family assignment. An unrecognized homology between two genes may be interpreted as a combined gene insertion/gene deletion. However, in presence of a large number of erroneous gene family assignments this workaround quickly reaches its limits. Another drawback of the current approach is that all information on alignment scores is discarded once gene families are assigned, such that later on, it makes no difference if two genes that are each others' counterpart in a pair of common intervals are strong bidirectional best hits or barely made it into the same gene family and may not even be true homologs after all.

To make better use of positional information and pairwise gene similarity scores, we can use a partial k-matching, as introduced earlier in this section, and simply translate each connected component into one gene family. (Strictly speaking, these are rather sub-families, as discussed previously.) However, conserved adjacencies, the only type of positional information currently used to obtain partial k-matchings, are not optimal in the context of common intervals detection. Typically their definition allows for unrestricted internal rearrangements and disregards gene orientation. The rationale behind this approach is not that conservation of gene order and orientation are supposed to be meaningless, but merely that it is difficult to decide *ad hoc* how much internal rearrangement in a conserved segment is plausible. In practice, a post-processing step can be applied to screen the predicted conserved segments for these qualities. A more integrative approach are *generalized adjacency clusters* which employ a user-defined parameter to restrict internal rearrangements [67].

The above considerations suggest that for common intervals more suitable positional information for gene family assignment could be obtained if the partial k-matching was not only based on conserved adjacencies, but the conserved neighborhood of up to $\theta > 0$ genes to the left and right of each gene. To obtain such a matching, we introduce the notion of θ-*neighbors*: Two genes g and g' in genome G are θ-neighbors with respect to G and H if at most $\theta - 1$ GH-saturated genes lie between them. Two pairs of θ-neighbors (g, g') in genome G and (h, h') in genome H form a θ-*adjacency* if the corresponding edges $e_{g,h}$ and $e_{g',h'}$ are part of $\mathcal{M}_{GH}$. An initial scoring scheme for θ-adjacencies could look as follows:

$$s^{\theta}(g, g', h, h') = \begin{cases} \sqrt{w(e_{g,h}) \cdot w(e_{g',h'})} & \text{if } (g, g') \text{ and } (h, h') \text{ form a } \theta\text{-adjacency} \\ 0 & \text{otherwise} \end{cases}$$

It can be extended by a weighting scheme that values pairs of θ-neighbors the higher the closer they are.

While the use of positional information is most likely an advantage for gene family assignment, the restriction of gene families to at most one gene per genome, a consequence of the partial k-matching, is clearly not. In fact, it is not only unnecessary but even unwanted in common intervals detection. It prevents the detection of duplicate occurrences of genes within a common interval, as well as multiple occurrences of common intervals in a genome. Both findings are certainly interesting as they hint at segmental or whole genome duplications.

In the remainder of this section, we broach a gene family-free approach for common intervals detection that avoids the above mentioned restrictions. We first study

the case of two genomes G and H. Any pair of intervals (I, J) on G and H can be common intervals. Therefore we build for each (I, J) a maximum weighted bipartite matching $\mathcal{M}_{I,J}$ between the gene sets of I and J. This is equivalent to solving Problem FF-Adjacencies with $\alpha = 0$ for $G_1 = I$ and $G_2 = J$.

An unmatched gene in I and J is either a duplicate occurrence if it is incident to an unchosen edge within the interval pair, or an inserted gene, if there are no incident edges or all of them point to a gene outside the interval pair. We obtain a matching score $\text{score}(\mathcal{M}_{I,J}) = \mathcal{F}_0(\mathcal{M}_{I,J})$ that needs to be corrected for the number of genes occurring in the intervals. Otherwise, the biggest score is obtained for (G, H), the interval pair defined by the complete genomes. Simply normalizing $\text{score}(\mathcal{M}_{I,J})$ by the length of I and J is also not advisable, as it causes the best-scoring common intervals to be of length one, the best scoring pair of genes. Instead a trade-off between matching score and interval compactness needs to be defined. The corrected score can then be used to decide whether an interval pair should pass for a conserved segment or not. For $k > 2$ genomes, the matching score can be defined as the sum over all pairwise matching scores which equals the score of a partial k-matching over all genomes.

The computation of a single matching $\mathcal{M}_{I,J}$ can be done in $O(\max\{|I|, |J|\}^3)$ time using the Hungarian Method [35]. However, already for two genomes there are $O(|G|^2|H|^2)$ interval combinations that need to be tested. One order of magnitude is saved if the initial definition of common intervals is used that neither allows duplicate genes nor gene insertions/deletions. In this case, only intervals of the same size need to be paired. For larger k, the complexity increases further, as all $O(k^2)$ pairwise genome combinations need to be considered. With polynomials of such high degrees in the asymptotic time complexity, it remains to be seen to what extent matching-based approaches are feasible in practice.

13.4 Genome Rearrangements

The study of genome rearrangements leads to a better understanding of the dynamics of genome structure over time. Typical rearrangement operations are the *inversion* of a piece of a chromosome, the *translocation* of material between two chromosomes, or the *fusion* and *fission* of chromosomes. These operations are explicitly modeling the modification of the genome over time and the methods therefore are called *rearrangement model-based* [30, 44, 63], in contrast to the *rearrangement model-free* methods that we discussed in the previous section, which only study and compare static properties of the genomes.

In rearrangement model-based methods, given two genomes and a set of rearrangement operations, two problem variants are typically considered: (1) calculate the minimum number of steps that are necessary to transform one genome into another, the so-called *genomic distance problem*, and (2) find a series of operations that perform such a transformation, the *genomic sorting problem*. Traditional approaches to analyze these problems are family-based, and the vast majority of methods also

adopt the simplifying assumption that exactly one occurrence of each family appears in each genome, which allows the existence of several polynomially-time computable methods, including for the popular Double Cut and Join (DCJ) rearrangement model [7, 63].

While the sorting problem, especially for the case of multiple genomes and their relation along the branches of a phylogenetic tree, will be addressed briefly in the following Sect. 13.5, here we concentrate on distance calculations in a family-free setting. In general, similarly to the rearrangement model-free measure of conserved adjacencies described in Sect. 13.3, the challenge is finding pairwise gene assignments based on similarities while minimizing the distance. In the following we will sketch a natural modification of existing approaches for the DCJ model. Whether this will lead to meaningful distances and allows for efficient algorithms has yet to be shown.

13.4.1 The Weighted Adjacency Graph

Recall that a gene is an oriented interval of a chromosome. We now represent a gene by the two extremities of its interval, called *tail* and *head*. The tail of gene g is denoted by g^t and the head by g^h. In a family-based setting composed of n gene families, consider that each one of two genomes G and H has exactly n genes, one occurrence of each family. A data structure that has proven to be useful in the study of the DCJ rearrangement model in this context is the adjacency graph $AG(G, H)$. This graph has a vertex for each adjacency of either of the two given genomes, and for each one of the two extremities of each gene there is an edge connecting the two vertices, one in G and the other in H, that contains this extremity. The graph is bipartite and a collection of paths and cycles, because each vertex has either degree one or degree two. The DCJ rearrangement distance can easily be calculated from this graph using the formula $d_{\text{DCJ}} = n - c - i/2$, where c is the number of cycles and i is the number of paths with an odd number of edges in $AG(G, H)$ [7]. Since, in the linear unichromosomal case that we consider in this paper, the adjacency graph has exactly two paths and otherwise only cycles, $i/2$ is either 0 or 1. Therefore, the similarity of two genomes G and H is closely related to the number of cycles in the adjacency graph $AG(G, H)$.

While the original adjacency graph clearly depends on the assignment of gene families, we observe that based on the information in the gene similarity graph from Sect. 13.2 we can obtain a data structure that resembles some of the properties of the adjacency graph. This new data structure might thus be a good basis for DCJ-like rearrangement distance calculations in a family-free setting:

Definition 3 (Weighted adjacency graph) The *weighted adjacency graph* $WAG(G, H)$ of two genomes G and H has a vertex for each adjacency in G and a vertex for each adjacency in H. For a gene g in G and a gene h in H with similarity $\sigma(g, h) > 0$ there is one edge connecting the vertices containing the two heads g^h

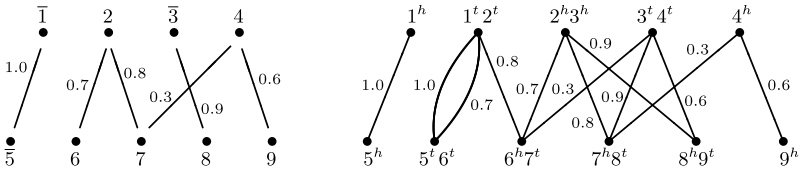


Fig. 13.3 Gene similarity graph (*left*) and the resulting weighted adjacency graph $WAG(G, H)$ (*right*) for two genomes $G = (\circ\ \overline{1}\ 2\ \overline{3}\ 4\ \circ)$ and $H = (\circ\ \overline{5}\ 6\ 7\ 8\ 9\ \circ)$

and h^h and one edge connecting the vertices containing the two tails g^t and h^t. The weight of each of these edges is $w(e_{g,h}) := \sigma(g, h)$.

As an example, the gene similarity graph for the two genomes $G = (\circ\ \overline{1}\ 2\ \overline{3}\ 4\ \circ)$ and $H = (\circ\ \overline{5}\ 6\ 7\ 8\ 9\ \circ)$ and six edges with non-zero weight, and the corresponding weighted adjacency graph are given in Fig. 13.3.

Note that if G and H have the same number of genes and the similarity measure σ forms a perfect matching with weight 1 for all edges of the matching and weight 0 otherwise, then the weighted adjacency graph reduces to the ordinary adjacency graph.

13.4.2 The Weighted Double-Cut-and-Join Distance

As for the case of conserved adjacencies, where instead of the breakpoint distance we calculate a matching maximizing an adjacency score in Eq. (13.3), here we first define a similarity measure that, if needed, can easily be converted into a distance.

Again, the similarity measure is based on a matching $\mathcal{M}$ of the genes in G and the genes in H. Let $\mathcal{I}(G, H; \mathcal{M})$ be a graph derived from the weighted adjacency graph $WAG(G, H)$ and the matching $\mathcal{M}$ by first removing from $WAG(G, H)$ each unmatched gene, consequently merging the two vertices containing its extremities, and second keeping only the edges representing extremities of gene pairs from $\mathcal{M}$. This graph has the shape of a standard adjacency graph and thus is a collection of cycles and paths. We denote by $\mathcal{C}(\mathcal{M}) \equiv \mathcal{C}(G, H; \mathcal{M})$ the set of connected components of $\mathcal{I}(G, H; \mathcal{M})$.

The graph derived from the weighted adjacency graph of Fig. 13.3 and the matching $\mathcal{M} = \{(1, 5), (2, 6), (3, 8), (4, 9)\}$ is given in Fig. 13.4.

Since we know that the number of DCJ operations is closely related to the number of cycles in the adjacency graph, we define a score function whose domain is defined by gene similarities and cycles in the matching. Therefore, in analogy to the corresponding formula for conserved adjacencies in Eq. (13.3), we propose the following objective function:

$$\mathcal{F}_{\alpha}^{\mathrm{DCJ}}(\mathcal{M}) = \alpha \cdot \mathrm{cyc}(\mathcal{M}) + (1 - \alpha) \cdot \mathrm{edg}(\mathcal{M})$$

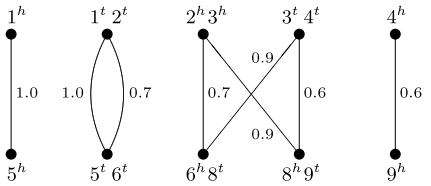


Fig. 13.4 The graph derived from the weighted adjacency graph of Fig. 13.3 and the matching $\mathcal{M} = \{(1, 5), (2, 6), (3, 8), (4, 9)\}$

where

$$\mathrm{cyc}(\mathcal{M}) = \sum_{C \in \mathcal{C}(\mathcal{M})} \left(\frac{1}{|C|} \sum_{e \in C} w(e) \right)$$

and $\mathrm{edg}(\mathcal{M})$ is the same as in Eq. (13.2). Again, $\alpha \in [0, 1]$ is a parameter that allows to balance between the two extremes, here between rearrangements ($\alpha = 1$) and gene similarities ($\alpha = 0$). Nevertheless, even for $\alpha = 1$ gene similarities are not ignored since the weights $w(e)$ also form an essential part of the cycle score $\mathrm{cyc}(\mathcal{M})$. Note that the normalization $1/|C|$ in $\mathrm{cyc}(\mathcal{M})$ is designed such that many short cycles are preferred over fewer long ones. For example, if all edges have the same weight w, two cycles of length 2 receive the score $2w$, which is twice the score of one cycle of length 4. The cycle score of the graph shown in Fig. 13.4 is $\mathrm{cyc}(\mathcal{M}) = \frac{1}{1} \cdot 1.0 + \frac{1}{2} \cdot (1.0 + 0.7) + \frac{1}{4} \cdot (0.7 + 0.9 + 0.6 + 0.9) + \frac{1}{1} \cdot 0.6 = 3.225$.

It is unlikely to find an efficient algorithm to compute a matching $\mathcal{M}$ that maximizes $\mathcal{F}_\alpha^{\mathrm{DCJ}}(\mathcal{M})$, but the solution of this optimization problem through integer linear programming seems possible and will be the subject of further research.

It is also an open question how to treat genes that are not covered by $\mathcal{M}$. They can be explained as being inserted or deleted during the course of evolution. Thus, a more general score function might consider these genes and prefer sorting scenarios with a low number of insertion/deletion events, similar to existing family-based approaches [14, 25].

Even further reaching might be approaches that do not rely on any matching, and instead optimize an objective directly defined on the weighted adjacency graph, for example a weighted version of maximum cycle decomposition.

13.5 Ancestral Genome Reconstruction

Studying conservation of gene order or rearrangement processes in the light of a phylogeny—given or unknown—can provide deeper insight into evolutionary mechanisms, gene functions, or the phylogeny itself. In this section, we will discuss how a partial k-matching can be used for ancestral genome reconstruction.

Phylogeny Aware Optimization A natural first step when reconstructing ancestral gene orders is to take phylogenetic information into account. Apart from ancestral reconstruction, this can actually be done in general to improve the construction of the partial k-matching. Given an edge-weighted phylogenetic tree, say $\mathcal{T}$,

for the species under consideration where the edge weights reflect the phylogenetic/evolutionary distance, the lengths of the paths between all pairs of species define an additive distance matrix $D^{\mathcal{T}}$. As additivity gives a one-to-one correspondence of $D^{\mathcal{T}}$ and $\mathcal{T}$, including the pairwise distances into the optimization implicitly also includes the topology of $\mathcal{T}$. These distances can be used to scale the pairwise scores in the objective function—close relatives receive a higher score than more distant pairs:

$$\begin{aligned}\mathcal{F}_{\alpha,\mathcal{T}}(\mathcal{M}) &= \alpha \cdot \sum_{G,H} \left(D^{\mathcal{T}}_{\max} - D^{\mathcal{T}}_{GH}\right) \operatorname{adj}(\mathcal{M}_{GH}) \\ &\quad + (1-\alpha) \cdot \sum_{G,H} \left(D^{\mathcal{T}}_{\max} - D^{\mathcal{T}}_{GH}\right) \operatorname{edg}(\mathcal{M}_{GH}) \\ &= \sum_{G,H} \left(D^{\mathcal{T}}_{\max} - D^{\mathcal{T}}_{GH}\right)\left(\alpha \cdot \operatorname{adj}(\mathcal{M}_{GH}) + (1-\alpha) \cdot \operatorname{edg}(\mathcal{M}_{GH})\right)\end{aligned}$$

where

$$D^{\mathcal{T}}_{\max} = \max_{G,H}\left\{D^{\mathcal{T}}_{GH}\right\} + \epsilon$$

($\epsilon > 0$, a constant to avoid nullity in the case of the two most distant genomes).

Ancestral Genes To be able to reconstruct ancestral gene orders, we first need to define ancestral genes and the ancestral gene content of ancestral genomes. To this end, we leave the family-free approach and rely on the assignments given by the partial k-matching. From such assignments, gene families can be derived by simply assigning all genes from a connected component in a partial k-matching to one family. As mentioned in Sect. 13.3.2, strictly speaking, these are rather gene sub-families. Recall further that the partial k-matching is defined such that within each connected component formed by saturated edges no two genes belong to the same genome. If all components are k-cliques, then genomes can be modeled as signed permutations. But in general, components might cover less than k genomes, i.e., not all genomes have the same gene content, although genomes do not have duplicated genes, thus leading to easier problems.

Based on the gene sub-families, we can infer the ancestral gene content from standard methods [18] or methods tailored for genome rearrangement problems [24, 53].

Ancestral Gene Orders Similarly to the computation of genomic distances (Sect. 13.4), the reconstruction of ancestral gene orders can be seen from two points of view—incorporating a rearrangement model-based approach or not. Once gene families have been defined from the partial k-matching, we have the gene orders of the extant genomes. Thus, we can apply rearrangement model-based methods allowing for unequal gene content such as [24, 48, 53]. Usually, such methods, following a parsimony approach, would aim at minimizing the total number of operations along the tree edges, which in most cases will lead to computationally hard optimization problems.

In the rearrangement model-free approach, ancestral syntenic characters are determined which induce a (partial) gene order. In our case, adjacencies qualify as ancestral syntenic characters. The remaining questions are then (1) how to infer the ancestral adjacencies, and (2) whether a set of adjacencies assigned to an ancestral node is concordant with some valid gene order, i.e., a collection of linear (and circular) chromosomes where each gene has at most two neighbors.

For a median-of-three, the above questions can easily be answered. Following a parsimony approach, the 0/1-assignment of an adjacency to the median boils down to a majority vote. Further, in almost all rearrangement median models, any adjacency present in at least two genomes is contained in any optimal median. In the case of signed gene orders, this selection will always ensure compatibility with a collection of linear and circular gene orders, and the inferred partial k-matching defines implicitly a set of linear or circular genome segments. Note, however, that this median genome might not be optimal for a given rearrangement model; however, it is a valid set of ancestral genome segments that has been inferred in a joint process, together with putative gene sub-families.

For general trees, one could follow rearrangement model-free approaches that try to find a most parsimonious labeling of the whole tree that is at the same time consistent with some linear or circular gene order [52], or one could concentrate on a single ancestral node as, e.g., done in several recent works [16, 37]. The method by Chauve and Tannier [16] relies on the Dollo principle, where only adjacencies conserved in pairs of genomes whose path in the species tree contain that ancestor are deemed ancestral; other approaches can select or score adjacencies using a Fitch principle [37].

The Dollo principle can easily be included into the optimization of the partial k-matching by introducing a factor π^A_{GH} that equals one if the path between G and H contains the ancestor A and zero otherwise:

$$\mathcal{F}_{\alpha,\mathcal{T},A}(\mathcal{M}) = \sum_{G,H} \pi^A_{GH}\big(D^{\mathcal{T}}_{\max} - D^{\mathcal{T}}_{GH}\big)\big(\alpha\,\mathrm{adj}(\mathcal{M}_{GH}) + (1-\alpha)\,\mathrm{edg}(\mathcal{M}_{GH})\big).$$

Thus, adding this feature to the objective function allows to select a set of putative ancestral adjacencies that can also receive a phylogenetic score as we described it earlier. Then existing methods that select a subset of adjacencies that form a valid genome can be used (see [38] for an example).

In this section we outlined how the family-free principle can fit quite naturally in existing approaches to reconstruct ancestral gene orders. This preliminary study opens several interesting research avenues. For example, it is worth to mention that rearrangement model-free reconstruction methods can utilize larger conserved structures than just adjacencies. Thus, e.g., common intervals could be included by integrating the scoring for θ-adjacencies as proposed in Sect. 13.3.2. Also, progressing toward a fully integrated inference process, it would be natural to incorporate the constraints posed by the structure of an ancestral genome; with adjacencies, this reduces to ensuring that every ancestral gene has at most two adjacent neighboring genes. However, integrating such constraints—even if only for a single internal node of a species tree (ancestral genome)—seems to be very challenging. Finally,

it would also be interesting to move on from reconstructing the states of the internal nodes of a given phylogeny (the small phylogeny problem) to reconstructing the tree itself. It is known that using gene order data for phylogenetic reconstructions can be more accurate and robust than sequence-based methods since they are not affected by gene-tree species-tree issues and less affected by small sequence or alignment errors. Not relying on purely sequence-based homology assignments could be a benefit for such reconstructions.

13.6 Discussion

In this paper we have outlined the potential of family-free methods in various aspects of genome comparison. Gene families are generally computationally predicted and serve as basis for a large variety of current comparative genomics studies. Since the predicted families may not be concordant with the underlying true biological gene families, erroneous gene family assignments can deteriorate subsequent analyses. Most importantly, comparative genomics methods require prior gene family assignments, yet the attained information about the structural organization of the genome may in turn actually help to improve the initially required gene family assignments. Consequently we propose the use of a *gene similarity graph* as underlying data structure in genome comparison. Therein genes are associated with each other by weighted edges according to a normalized similarity measure. In practice, sequence similarity scores can be employed in constructing the graph.

The underlying strategy of almost all presented methods is tantalizingly simple and boils down to obtain a one-to-one matching between orthologous genes of the gene similarity graph by solving an optimization problem. More specifically, a linear combination of a synteny (or rearrangement) score and a similarity score, parameterized by α, between saturated genes is optimized. Here, we give users the choice in favoring one of the two quantities over the other by adjusting α in each particular analysis. At this point, we like to acknowledge an inherent disadvantage of a one-to-one matching, namely its inability to account for inparalogous genes. Thus, the detection of inparalogs remains part of post-processing steps which identify unsaturated genes with high similarities to other genes of the same genome.

In Sect. 13.3 we studied two forms of conserved structures: adjacencies and common intervals. In the former, we generalized the problem of family-free computation of adjacencies of [22], called FF-Adjacencies, towards the simultaneous study of more than two genomes. Thereby we introduced the notion of a *partial k-matching*, which allows to incorporate in solutions of Problem FF-Adjacencies sparsely interconnected genes as well as connected components that are only contained in subsets of the genomes. We also discussed two possible approaches towards family-free common intervals by introducing a scoring scheme for *θ-adjacencies*, which is a co-localization measure for genes similar to adjacencies. We further outlined a more dynamic, but also computationally more expensive approach based on performing local maximum matchings. Complementing the study of conserved structures, we turned in Sect. 13.4 to model-based genome comparison by introducing

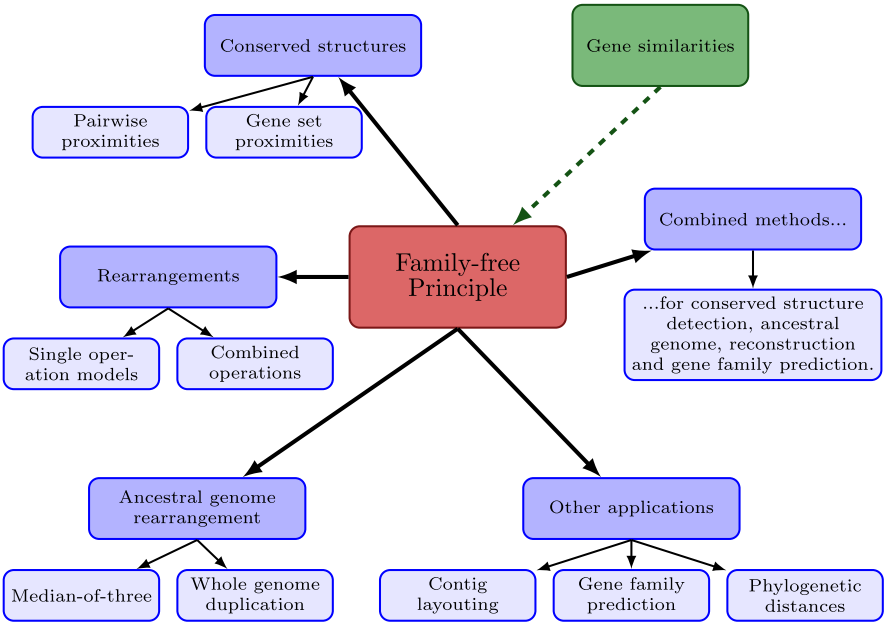


Fig. 13.5 Various fields of comparative genomics can be explored under the family-free model such as conserved structure detection or reconstruction of ancestral genomes, employing different gene similarity measures (based on alignment scores, functional similarity, etc.)

the *weighted adjacency graph*. On this basis we proposed a *weighted DCJ* distance following a similar strategy as in the previous section. We further showed in Sect. 13.5 how the reconstruction of ancestral genomes can be performed using the family-free principle. Thereby we studied the concept of family-free adjacencies in a phylogeny-aware setting using existing approaches of reconstructing ancestral gene orders.

This work presents a number of initial studies in a new field of genome comparison which aims at developing methods where prior gene family assignments are no longer required. It consequently offers many directions in which these studies can be extended (see Fig. 13.5). Most evidently, the principle of family-free genome comparison can be applied to the numerous existing family-based studies. More interestingly, the family-free principle could even be integrated into a methodology for joint inference of gene families, conserved structures and ancestral gene orders at the same time, extending presented work in reconstructing ancestral gene orders. Even though such venture most likely involves a more complex data structure and a potentially increased solution space, the question remains unanswered if the stronger signal gained from harvesting more information from the genomic datasets may reduce the computational cost in finding optimal solutions. Finally, it is worth to mention that the family-free principle may be particularly beneficial in studying partially sequenced (or assembled) genomes, as methods in gene family prediction tend to be susceptible for missing genes. Here, the family-free approach can offer

improvements for inferring phylogenetic distances of incomplete genomes, but also in detecting conserved structures, which may lead to improved methods in contig layouting.

While sequence similarity between genes is an obvious and reasonable measure in constructing the gene similarity graph, similarity scores can also integrate additional information such as functional similarity. Such information can be obtained from various databases, most notably, from the Gene Ontology database [3]. Family-free genome comparisons of this kind may give further insights into the functional organization of the genome.

Acknowledgements MDVB is funded by the Brazilian research agency CNPq grant PROMETRO 563087/10-2. DD receives a scholarship from the CLIB Graduate Cluster Industrial Biotechnology. KJ is funded by DFG grant ST 431/5-1. AT is a research fellow of the Alexander von Humboldt Foundation.

References

1. 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), 1093–1115 (2008)
2. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. **13**(1), 19–53 (2009)
3. 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. the gene ontology consortium. Nat. Genet. **25**(1), 25–29 (2000)
4. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. J. Comput. Biol. **13**(7), 1340–1354 (2006)
5. Bergeron, A., Corteel, S., Raffinot, M.: The algorithmic of gene teams. In: Proceedings of WABI 2002. LNCS, vol. 2452, pp. 464–476 (2002)
6. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. J. Comput. Biol. **13**(2), 567–578 (2006)
7. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Proceedings of WABI 2006. LNBI, vol. 4175, pp. 163–173 (2006)
8. Bernt, M., Merkle, D., Middendorf, M.: Solving the preserving reversal median problem. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**, 332–347 (2008)
9. Blin, G., Chauve, C., Fertin, G.: The breakpoint distance for signed sequences. In: Proceedings of CompBioNets 2004. Texts in Algorithmics, vol. 3, pp. 3–16 (2004)
10. Blin, G., Chateau, A., Chauve, C., Gingras, Y.: Inferring positional homologs with common intervals of sequences. In: Proceedings of RECOMB-CG 2006, pp. 24–38. Springer, Berlin (2006)
11. 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)
12. Böcker, S., Jahn, K., Mixtacki, J., Stoye, J.: Computation of median gene clusters. J. Comput. Biol. **16**(8), 1085–1099 (2009)
13. Bourque, G., Pevzner, P.A.: Genome-scale evolution: reconstructing gene orders in the ancestral species. Genome Res. **12**(1), 26–36 (2002)

14. Braga, M.D.V., Willing, E., Stoye, J.: Double cut and join with insertions and deletions. J. Comput. Biol. **18**(9), 1167–1184 (2011)
15. Caprara, A.: The reversal median problem. INFORMS J. Comput. **15**(1), 93–113 (2003)
16. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. PLoS Comput. Biol. **4**(11), e1000234 (2008)
17. Chauve, C., El-Mabrouk, N., Guéguen, L., Semeria, M., Tannier, E.: Duplication, rearrangement and reconciliation: a follow-up 13 years later. In: Chauve, C. et al. (eds.) Models and Algorithms for Genome Evolution. Computational Biology, vol. 19. Springer, Berlin (2013). In this volume
18. Csurös, M.: Count: evolutionary analysis of phylogenetic profiles with parsimony and likelihood. Bioinformatics **26**(15), 1910–1912 (2010)
19. Darling, A.E., Mau, B., Perna, N.T.: ProgressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. PLoS ONE **5**(6), e11147 (2010)
20. Dewey, C.N.: Positional orthology: putting genomic evolutionary relationships into context. Brief. Bioinform. **12**(5), 401–412 (2011)
21. Didier, G., Schmidt, T., Stoye, J., Tsur, D.: Character sets of strings. J. Discrete Algorithms **5**(2), 330–340 (2007)
22. Doerr, D., Thévenin, A., Stoye, J.: Gene family assignment-free comparative genomics. BMC Bioinform. **13**(Suppl 19), S3 (2012)
23. Durand, D., Sankoff, D.: Tests for gene clustering. J. Comput. Biol. **10**, 453–482 (2003)
24. Earnest-DeYoung, J.V., Lerat, E., Moret, B.M.E.: Reversing gene erosion—reconstructing ancestral bacterial genomes from gene-content and order data. In: Proceedings of WABI 2004. LNCS, vol. 3240, pp. 1–13 (2004)
25. El-Mabrouk, N.: Sorting signed permutations by reversals and insertions/deletions of contiguous segments. J. Discrete Algorithms **1**(1), 105–122 (2001)
26. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(5), 1318–1329 (2011)
27. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press, Cambridge (2009)
28. Frech, C., Chen, N.: Genome-wide comparative gene family classification. PLoS ONE **5**(10), e13409 (2010)
29. 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)
30. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J. ACM **46**(1), 1–27 (1999)
31. He, X., Goldwasser, M.H.: Identifying conserved gene clusters in the presence of homology families. J. Comput. Biol. **12**(6), 638–656 (2005)
32. Heber, S., Stoye, J.: Algorithms for finding gene clusters. In: Proceedings of WABI 2001. LNCS, vol. 2149, pp. 252–263 (2001)
33. Heber, S., Mayr, R., Stoye, J.: Common intervals of multiple permutations. Algorithmica **60**(2), 175–206 (2011)
34. Jahn, K.: Efficient computation of approximate gene clusters based on reference occurrences. J. Comput. Biol. **18**(9), 1255–1274 (2011)
35. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (2006)
36. Li, L., Stoeckert, C.J., Roos, D.S.: OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res. **13**(9), 2178–2189 (2003)
37. Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Zhang, L., Miller, W., Haussler, D.: DUPCAR: reconstructing contiguous ancestral regions with duplications. J. Comput. Biol. **15**(8), 1007–1027 (2008)
38. Manuch, J., Patterson, M., Wittler, R., Chauve, C., Tannier, E.: Linearization of ancestral multichromosomal genomes. BMC Bioinform. **13**(Suppl 19), S11 (2012)

39. Milinkovitch, M.C., Helaers, R., Depiereux, E., Tzika, A.C., Gabaldon, T.: 2× genomes—depth does matter. Genome Biol. **11**, R6 (2010)
40. Ostlund, G., Schmitt, T., Forslund, K., Köstler, T., Messina, D.N., Roopra, S., Frings, O., Sonnhammer, E.L.L.: InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. Nucleic Acids Res. **38**(Database issue), D196–D203 (2010)
41. Pe'er, I., Shamir, R.: The median problems for breakpoints are NP-complete. Electron. Colloq. Comput. Complex. **71**, 5 (1998)
42. Powell, S., Szklarczyk, D., Trachana, K., Roth, A., Kuhn, M., Muller, J., Arnold, R., Rattei, T., Letunic, I., Doerks, T., Jensen, L.J., von Mering, C., Bork, P.: eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. Nucleic Acids Res. **40**(Database issue), D284–D289 (2012)
43. Rahmann, S., Klau, G.W.: Integer linear programs for discovering approximate gene clusters. In: Proceedings of WABI 2006. LNBI, vol. 4175, pp. 298–309 (2006)
44. Sankoff, D.: Edit distances for genome comparisons based on non-local operations. In: Proceedings of CPM 1992. LNCS, vol. 644, pp. 121–135 (1992)
45. Sankoff, D.: Genome rearrangement with gene families. Bioinformatics **15**(11), 909–917 (1999)
46. Sankoff, D., Blanchette, M.: The median problem for breakpoints in comparative genomics. In: Proceedings of COCOON 1997. LNCS, vol. 1276, pp. 251–263 (1997)
47. Sankoff, D., Blanchette, M.: Multiple genome rearrangement and breakpoint phylogeny. J. Comput. Biol. **5**, 555–570 (1998)
48. Sankoff, D., El-Mabrouk, N.: Duplication, rearrangement and reconciliation. 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. Computational Biology Series, vol. 1, pp. 537–550. Kluwer Academic, Dordrecht (2000)
49. Sankoff, D., Cedergren, R., Abel, Y.: Genomic divergence through gene rearrangement. In: Doolittle, R.F. (ed.) Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences. Meth. Enzymol., vol. 183, Chap. 26, pp. 428–438. Academic Press, San Diego (1990)
50. Schmidt, T., Stoye, J.: Quadratic time algorithms for finding common intervals in two and more sequences. In: Proceedings of CPM 2004. LNCS, vol. 3109, pp. 347–358 (2004)
51. Shi, G., Peng, M.C., Jiang, T.: MultiMSOAR 2.0: an accurate tool to identify ortholog groups among multiple genomes. PLoS ONE **6**(6), e20892 (2011)
52. Stoye, J., Wittler, R.: A unified approach for reconstructing ancient gene clusters. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(3), 387–400 (2009)
53. Tang, J., Moret, B.M., Cui, L., Depamphilis, C.W.: Phylogenetic reconstruction from arbitrary gene-order data. In: Proceedings of BIBE 2004, pp. 592–599. IEEE, New York (2004)
54. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinform. **10**, 120 (2009)
55. Tatusov, R.L., Fedorova, N.D., Jackson, J.D., Jacobs, A.R., Kiryutin, B., Koonin, E.V., Krylov, D.M., Mazumder, R., Mekhedov, S.L., Nikolskaya, A.N., Rao, B.S., Smirnov, S., Sverdlov, A.V., Vasudevan, S., Wolf, Y.I., Yin, J.J., Natale, D.A.: The COG database: an updated version includes eukaryotes. BMC Bioinform. **4**, 41 (2003)
56. Uno, T., Yagiura, M.: Fast algorithms to enumerate all common intervals of two permutations. Algorithmica **26**(2), 290–309 (2000)
57. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Automatic genome-wide reconstruction of phylogenetic gene trees. Bioinformatics **23**(13), i549–i558 (2007)
58. Wapinski, I., Pfeffer, A., Friedman, N., Regev, A.: Natural history and evolutionary principles of gene duplication in fungi. Nature **449**(7158), 54–61 (2007)
59. Waterhouse, R.M., Zdobnov, E.M., Tegenfeldt, F., Li, J., Kriventseva, E.V.: OrthoDB: the hierarchical catalog of eukaryotic orthologs in 2011. Nucleic Acids Res. **39**(Database issue), D283–D288 (2011)
60. Watterson, G., Ewens, W.J., Hall, T., Morgan, A.: The chromosome inversion problem. J. Theor. Biol. **99**(1), 1–7 (1982)

61. Xu, A.W., Moret, B.M.E.: GASTS: parsimony scoring under rearrangements. In: Proceedings of WABI 2011. LNBI, vol. 6833, pp. 351–363 (2011)
62. Xu, X., Sankoff, D.: Tests for gene clusters satisfying the generalized adjacency criterion. In: Proceedings of BSB 2008. LNBI, vol. 5167, pp. 152–160 (2008)
63. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16), 3340–3346 (2005)
64. Yang, Z., Sankoff, D.: Natural parameter values for generalized gene adjacency. In: Proceedings of RECOMB-CG 2009. LNBI, vol. 5817, pp. 13–23 (2009)
65. Zhang, M., Leong, H.W.: Identifying positional homologs as bidirectional best hits of sequence and gene context similarity. In: Proceedings of ISB 2011, pp. 117–122. IEEE, New York (2011)
66. Zhu, B.: Approximability and fixed-parameter tractability for the exemplar genomic distance problems. In: Proc. of Theory and Applications of Models of Computation. LNCS, vol. 5532, pp. 71–80 (2009)
67. Zhu, Q., Adam, Z., Choi, V., Sankoff, D.: Generalized gene adjacencies, graph bandwidth, and clusters in yeast evolution. IEEE/ACM Trans. Comput. Biol. Bioinform. **6**(2), 213–220 (2009)

Chapter 14
Genetic History of Populations: Limits to Inference

Daniel E. Platt, Filippo Utro, Marc Pybus, and Laxmi Parida

Abstract The dispersal of the human population to all the continents of the globe is a compelling story that can possibly be unravelled from the genetic landscape of the current populations. Indeed, a grasp on this strengthens the understanding of relationship between populations for anthropological as well as medical applications. While the collective genomes is believed to have captured its own "evolution", undoubtedly, a lot is irrecoverably lost. It is important to try to estimate what fraction of past events become unreconstructable. We used a published population simulation algorithm which includes parameter sets that simulate modern regional human demographics: it reflects the out-of-Africa expansion events, isolation during the Last Glacial Period, Neolithic expansion of agriculture, and the industrial revolution for African, African–American, European, and Asian populations. This simulation tool was used to provide complete genetic histories unavailable in real human population data. Next we used the minimal descriptor device which is an algorithm-independent means to explore the (potential) recoverable genetic events from the final extant population data. We found that, on average, around 65 % of the total number of genetic events are recoverable, with substantial variations among histories. We also found that increases of sequence length tended to yield diminishing returns in new information yield. Lastly, even with a substantial fraction of events unrecoverable, and even for different population history simulations, the recoverable events do yield similar resolution of the whole record of demographic, climatic, and other population events.

D.E. Platt · F. Utro · L. Parida (✉)
IBM T.J. Watson Research, Yorktown Heights, New York, USA
e-mail: parida@us.ibm.com

D.E. Platt
e-mail: watplatt@us.ibm.com

F. Utro
e-mail: futro@us.ibm.com

M. Pybus
Institute of Evolutionary Biology (CSIC-UPF), Barcelona, Spain
e-mail: marc.pybus@upf.edu

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*,
Computational Biology 19, DOI 10.1007/978-1-4471-5298-9_14,

14.1 Background

Every genetic event that is consequential to the genetic landscape of a population is captured in a topological structure called the Ancestral Recombinations Graph (ARG) [1]. The converse of this may not hold, that is some genetic events captured in the ARG may not contribute to resolvable events with definable times in the history of the extant population, but nevertheless are legitimate components of the relevant and common genetic history of the population. In a sense, the ARG is the phylogeny of the individuals of the population. It should be noted that just as in a phylogeny tree topology, an ARG also does not have any extraneous nodes. Further, it is not unreasonable to assume that there exists a "true" ARG for a collection of samples or a population that contains all the events in the sequence that they emerged, and is the underlying target of efforts to reconstruct the history of the population. Recall that the nodes of the ARG represent genetic events. Such a "true" ARG would contain a description of all SNPs, coalescent events, and recombinations that contributed to that collection of samples or population. The topology of the ARG is not necessarily a tree due to genetic exchange events such as recombinations, gene duplications, and so on. However, segments within a sequence that do not contain any recombination crossovers will show well-defined phylogenetic trees. The relationship among these "marginal trees" as their segments were involved in recombination events constitute the information in the complete ARG. These events are represented as nodes with multiple incoming edges in the ARG. The edges are usually annotated with mutations, and the lengths are representative of the ages measured in generations. Thus the topology, together with its annotation and the edge lengths, determines the genetic landscape of the extant samples. In non-recombinant phylogenetic trees, drift-driven coalescence events represent a loss of information, ultimately destroying all information prior to some most recent common ancestor. Equivalent SNPs are multiple single nucleotide polymorphisms which are found together along a single edge connecting two nodes of a phylogenetic tree [2]. Even though these SNPs likely accumulated along the lineage within the population, other lineages differentiating the order that these mutations occurred have been lost in subsequent drift and coalescence events. This has also resulted in the loss of information bounding when each mutation emerged along an edge between two coalescence nodes in the phylogenetic tree. In ARGs, the lost information involving recombination events results in coalescence-like branch nodes, but where none of the segment trees that are involved in the node coalesce at that node. The order and timing of those coalescence events are lost. This may result in ambiguities in determining not only timing and order, but possibly also of topology. It is possible to completely loose the information that two segments accumulated mutations and evolved along a single chromosome's lineage. While the population may be subject to fairly well-defined environmental impacts (desertification, glaciation, moistening, etc), and respond with migrations, contractions, expansions, impacts from new technologies, etc, the genetics that emerge in these processes have a distinctly stochastic character. It is of interest to understand how much can be deduced about the genetic response to these climatic, technological, and cultural changes.

To that extent, this study considers the impact of stochastic variation among graphs representing possible genetic histories of these populations. The reader is directed to [7] for an exposition on random graph representation of the ARG.

In this paper, we define a measure of relevant genetic "history" as the number of nodes in the ARG[1] [14]. These nodes include coalescence events, recombination events, and instances where multiple recombination events along extant lineages lost information due to intervening drift. These genetic events are also responsible for the genetic diversity in a population. Since drift can rapidly diminish diversity if the population is suddenly reduced in size, and the re-accumulation of mutations and diversity is rate-limited by mutation rates, bottlenecks can also significantly impact the extant number of unresolvable nodes. This feature will be visible in the numbers of reconstructable nodes and non-reconstructable nodes, and therefore is reflected in our history measure. glacial expansions. Then well-defined questions to ask are:

1. If the genetic history measure reflects diversity, do the demographic parameters reflect the greater genetic diversity of Africans vs. Asian, European, and other out-of-Africa populations?
2. Do we observe less information due to drift and coalescence in more ancient epochs?
3. Do we obtain higher resolution with (a) increasing sample sizes? (b) increasing sequence lengths? (c) higher or lower recombination rates?
4. What is the largest fraction, f, of the history that is estimable from a given sample? In other words, no matter what analysis is employed, is there always some fraction $(1 - f)$ of the common history that is impenetrable?
5. Is the information in a specific history sufficient to identify population, ecological, and demographic transitions?

Let N be the number of nodes in an underlying, or true, ARG topology. Given the extant samples, let a method estimate $N' \leq N$ nodes. We further assume that the lengths of the edges, as well as the interconnectivity with the labels, are estimated correctly, so the estimated fraction of this ARG, defined as $0.0 \leq N'/N \leq 1.0$, is a natural "overestimate". Let $N'_{\max}$ be the maximum of N'_i from *all* possible methods i. Then f, the penetrable fraction, is $N'_{\max}/N$. However, it is impossible to enumerate all possible estimation methods. So, we resort to the mathematical structure called the minimal descriptor [9] of an ARG: it is an essential substructure of an ARG that preserves the genetic landscape of the extant samples, including the topology and edge lengths of the marginal trees.

The reader is directed to [8] for an exposition on this nonredundant information content of an ARG. The minimal descriptor is also an ARG. Let the number of nodes in the minimal descriptor be $\tilde{N}$, then $N'_{\max} \leq \tilde{N}$. Thus, this methodology-independent scheme gives an upper bound on the penetrable fraction f as $\tilde{N}/N$, and a lower bound on the impenetrable fraction as $1 - \tilde{N}/N$.

In this paper, we seek to characterize the value of f in human populations. However, real genetic histories are not available for extant human populations. We there-

[1] A preliminary version of this work was presented at APBC 2013 [14].

fore sought to estimate bounds through simulation. From the literature, we selected a population simulator that has the capability of providing not only individuals from different demographies, but also the underlying ARG. This is very suitable for our experimental set-up. Next, we designed an algorithm to extract the minimal descriptor from a given ARG. Thus we compute upper bounds on f, as discussed. Recall that each node of the ARG has a specific *age* or *depth* associated with it. It may be noted that the length attribute of an edge can be viewed simply as the non-negative difference between the depths of the two incident nodes. The terminal leaf nodes are the extant individuals. The depth of the extant individual is defined to be zero and the value progressively increases as one traverses the ARG away from the terminal leaf nodes. The nodes of the minimal descriptor are also the nodes of the underlying ARG and the same age is associated with them. Let *epoch* d be defined as a range of depths say $[d_1, d_2]$ with $d_2 \geq d_1$. Then the *history density* at d, N_d, is measured by the number of nodes in the ARG with depth in the epoch d.

Extending this notion, the *estimable density* at d is measured as $f_d = \tilde{N}_d / N_d$, where $\tilde{N}_d$ is the number of nodes in the minimal descriptor with depth in the epoch d. We study the demography characteristics in terms of the history density and the estimable density. Continental populations have distinctive history density profiles, that are invariant under different parameter settings of the simulator. Let tARG denote the true ARG for a given data set with N nodes. Then, $\tilde{N}$ is the number of nodes of tARG that can be reconstructed based on extant genetic information. Thus $\tilde{N}$ is an underestimate of actual nodes N for any specific history, and $(N - \tilde{N})$ measures the deficit of $\tilde{N}$'s estimate of tARG's true historical values.

We used COSI [13], which is the only population simulator, to the best of our knowledge, that provides the ARG as well as produces populations that match the genetic landscape of the observed human populations. We used the *bestfit model* in COSI to simulate the samples with a calibrated human demography for different populations, proposed by Schaffner et al. [13]. This simulator generates data matching three structured continental populations: Africans, Europeans, and Asians. An admixed population, the Afro-Americans, can also be generated. This simulator has also been used in literature as a gold standard for generating the demographies [3, 5, 10, 11]. In order to explore f and the impenetrable fraction $(1 - f)$ of a given demography, all combinations of four different simulation parameters have been explored: mutation rate, sequence length, sample size and recombination rate. These are briefly described below:

Mutation rate: According to different studies *Homo sapiens*, as a species, has a mutation rate around 1.5×10^{-8} per base pair per generation (bp/gen for short) [12]. However, this value could change along the genome.

Sequence length: When simulating genetic population data, sequence length is one of the most important factors. While it may not be computationally feasible to simulate a whole chromosome, enough polymorphisms are required in order to get meaningful results.

Sample size: The sample size needs to be large enough to capture important population features.

Table 14.1 Parameters defining COSI execution environment

Parameters	Values
Mutation Rate (bp/gen $\times 10^{-8}$)	0.7, 1.5, 3.0
Sequence length (Kb)	5, 10, 30, 50, 75, 100, 150, 200
Sample size	5, 10, 30, 60, 120
Recombination rate (cM/Mb)	0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 15, 1.7, 1.9, 2.1, 2.3, 2.5, 2.8, 3.1, 3.5, 3.9, 4.5, 5.1

Recombination rate: The mean recombination rate along the genome in *Homo sapiens* is around 1.3 cM/Mb [4]. However, it has been seen that it can vary widely in a fine-scale manner when focusing on specific regions of the genome [6]. Different simulations are run using recombination rates matching the major portion of the range observed in human data [13].

Based on the above we used different parameters' values, and all possible combinations of them, in order to assess their effects on f (see Table 14.1). Each population of the COSI demography has been tested independently as well as the whole human demography (i.e. all populations together). In total, more than 22800 simulation replicates were generated for each population, each representing distinct histories, and exploring combinations of different values for the four simulation parameters described above. This includes ten replicates for each experiment. For the highest value of sequence length used (i.e. 200 Kb), some experiments were terminated after thirty minutes, since no substantial progress was being made towards its completion. Therefore, the results for 200 Kb sequence length are not reported in the summary plots.

Plots were prepared showing Summary statistics (means, medians, whisker plots showing medians, extremes, and quartiles) across replicated simulations. Further, a computation of the standard deviation over the ensemble of histories was prepared to explore whether ecological and demographic events (post-glacial expansions, Neolithic agricultural expansion, etc) would be expected to leave a genetic signal that could be resolved across the demographic groups' histories.

14.2 Method

Recall that an ARG is a phylogenetic structure that encodes mutation events and duplication events, as well as genetic exchange events such as recombinations: this captures the (genetic) dynamics of a population evolving over generations. From a topological point of view, an ARG is always a directed acyclic graph where the direction of the edges is toward the more recent generation. An edge is annotated with genetic mutation events, such as single nucleotype polymorphisms. Some simulators may give edges with empty labels. Recall that the *length* of the edge, not to be confused with the edge label, represents the epoch defined by the age (or depth) of

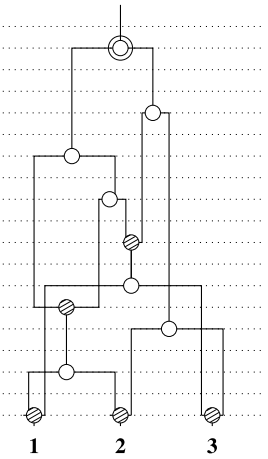


Fig. 14.1a The topology of an ARG G with three extant samples marked *1*, *2*, and *3*. The *dashed horizontal lines* mark the age or depth of the nodes which are the same in all the four figures

the two incident nodes. A chain node has a single incoming edge and a single outgoing edge. In the ARG we define the leaf nodes as nodes with no outgoing edges, they represent the extant unit.

Finally, given two nodes v and w, if there is an outgoing edge from v to w, then v is referred to as *parent* of w and w is referred as a *child* of v.

In [9], a structure-preserving and samples-preserving core of an ARG G, called the minimal descriptor ARG (mdARG) of G, was identified. Its structure-preserving character ensures that the topology and the all the branch lengths of the marginal trees of the minimal descriptor ARG are identical to that of G, and the samples-preserving property asserts that the patterns of genetic variation in the samples of the minimal descriptor ARG are exactly the same as that of G. It was also shown that an unbounded G has a finite minimal descriptor, that continues to preserve critical graph-theoretic properties of G. Thus this lossless and bounded structure is well defined for all ARGs (including unbounded ARGs) and we use the same here. However, a minimal descriptor of an ARG may not be unique. This does not affect the estimation of f, since $N_{\max} \leq \tilde{N}$ are the same, for all possible $\tilde{N}$ corresponding the different minimal descriptors (see Sect. 14.1 for the definitions).

Identification of the reconstructable fraction is started by first computing a minimal descriptor from the ARG. The input to this process is the ARG G derived from the log files of the population simulator COSI. The sequence length is normalized to the interval $[0, 1]$. This ARG is preprocessed as follows.

Firstly, the number of marginal trees, M, is extracted from G, corresponding to the M intervals $[0, l_1], [l_1, l_2], \ldots, [l_{M-1}, l_M = 1.0]$ derived from the segments file of COSI, where $0 < l_1 < l_2 < \cdots < l_M = 1.0$. Next, each node in G is annotated with one or more of these M intervals through their traversal of G. See Figs. 14.1a and 14.1b.

A coalescent node is *t-coalescent* if it is a coalescent node in one of the M marginal trees. Each coalescent node that is not t-coalescent is removed, following the node-removal procedure defined in [9]. To remove node v, an edge is introduced

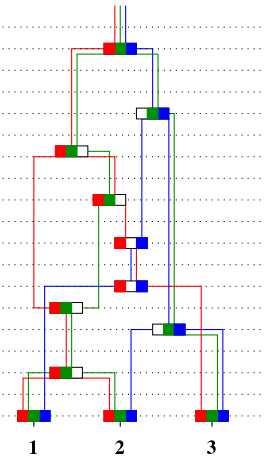


Fig. 14.1b The three nonmixing segments are *red*, *green*, and *blue*, in that order. Each node displays the nonmixing segments. A *white rectangle* indicates the absence of that segment in that node. The three embedded trees, corresponding to each segment, are shown in the same color as that of the segment. The edge labels (mutation events) are not shown to avoid clutter

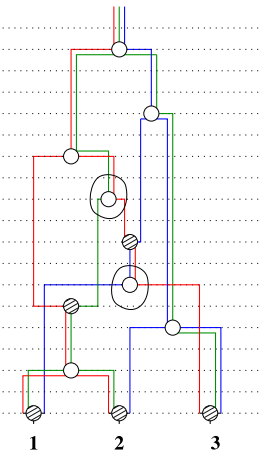


Fig. 14.1c The same as Fig. 14.1b with the two marked nodes that are not t-coalescent

from the parent, u, of v to each child w of v, while maintaining the same age of u as well as the child w. Also the annotation of the edges is adjusted to reflect the same flow of genetic material from u to each of w.

Based on this, we applied the following algorithm to construct mdARGs from the COSI results in the following steps. (1) Remove the coalescent nodes that are not t-coalescent, using the node-removal procedure. See Fig. 14.1d for an example. Since a coalescent node could also be a recombination node, it is possible that such a node is additionally not t-coalescent. In that case, the node continues to belong to the minimal descriptor. (2) The last step is applied till it is no longer applicable. (3) The chain nodes are removed.

Figure 14.2 provides an example, on a G given by COSI, of the above procedure. In particular, given the ARG in Fig. 14.2(a), the node D is not t-coalescent node and then it is removed producing Fig. 14.2(b). Finally, in Figs. 14.2(c)–(d) step 3 is performed removing the nodes E and B.

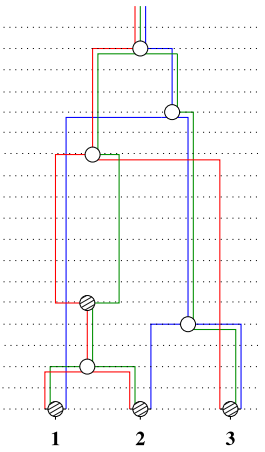


Fig. 14.1d Removing the marked nodes to obtain a minimal descriptor G'. At each node the nonmixing segment corresponding to the embedded tree is are shown separately in Figs. 14.1b–14.1c

Let the resulting graph be G', which is an mdARG. Let N be the number of nodes in G and $\tilde{N}$ the number of nodes in G'. Then, $f \leq \tilde{N}/N$.

14.3 Results and Discussion

Given the genetic landscape of some extant samples, its underlying ARG provides a plausible explanation of the observation, since it is the annotated topological structure that captures the total genetic history that is relevant to the extant samples. Since the specific ARGs reflect specific histories, we explored multiple replicates to characterize the ensemble of histories consistent with the population, ecological, and demographic parameters.

COSI includes population specific parameters in its simulation of African, African–American, European, and Asian populations. Figures 14.3a, 14.3b and 14.3c show that "history", N, measured across specific epochs, identifies distinct

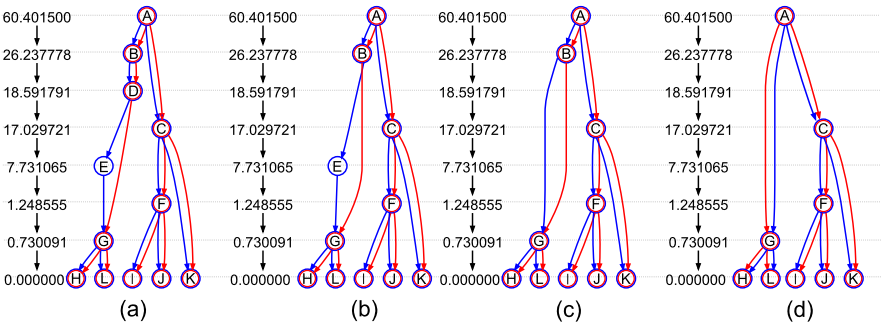


Fig. 14.2 Steps reducing ARG to resolvable nodes in a typical COSI output

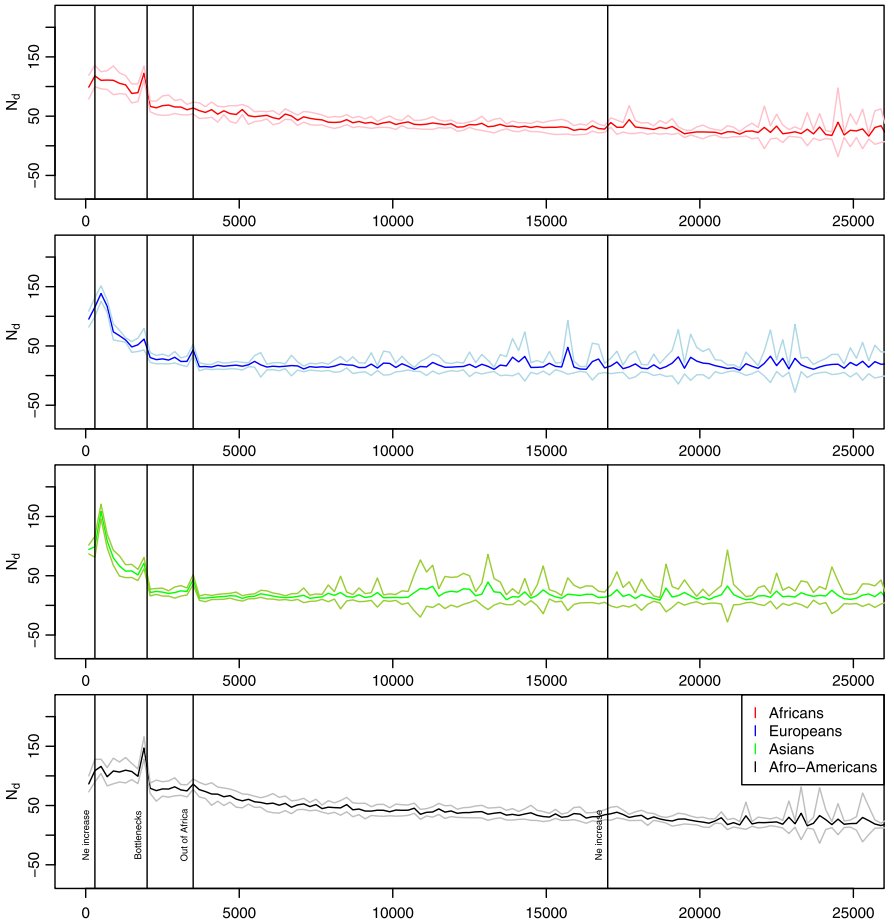


Fig. 14.3a N_d for sequence lengths of 150,000, with time depth back to 25,000 generations, and epoch lengths of 200 generations

differences between the COSI simulated Africans and Afro-Americans in comparison with Europeans and Asians. Specifically, density N_d shows significant impacts consistent with the out-of-Africa event for Europeans and Asians reflecting the migration bottleneck, compared with Africans. While the Africans do not show a founder-effect spike, the admixed Afro-American populations do reflect the spike. Subsequent event counts also seem to reflect post-Glacial expansions and the Neolithic Revolution for out-of-Africa populations. Climatic impacts appear also to be reflected in African populations as well given post-glacial moistening and expansions. In this respect, N_d mirrors diversity and genetic depth, as expected. The recent increase in effective population has had almost no impact on diversity since little time to accumulate new mutations has occurred in the modern era.

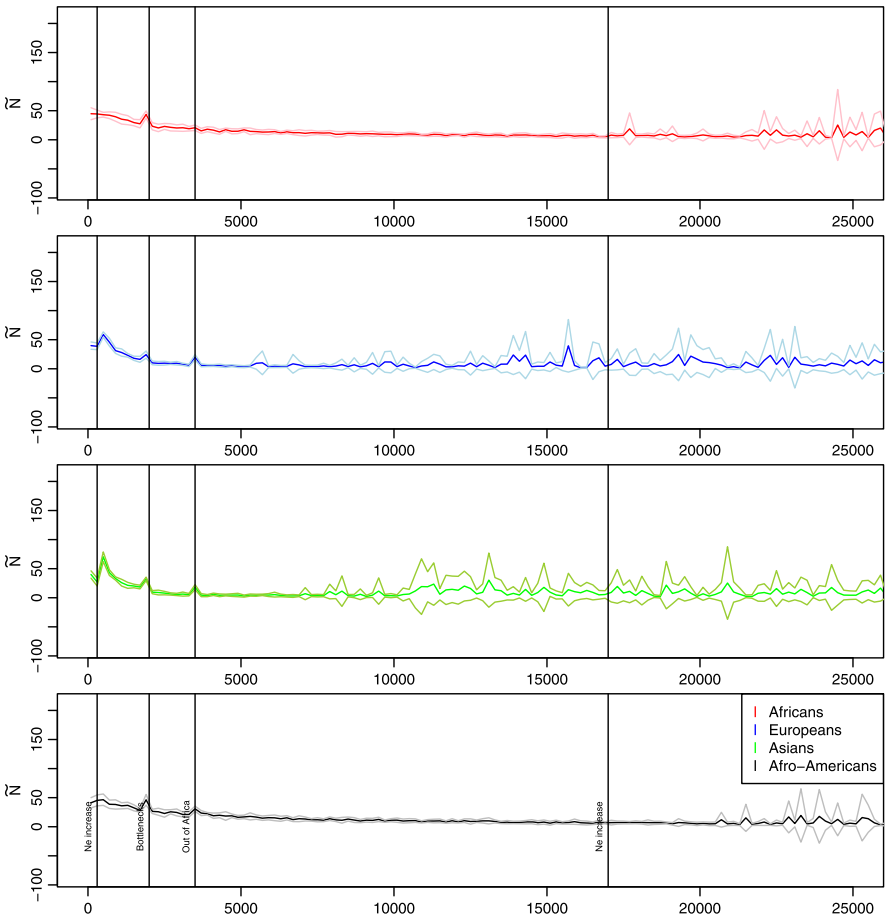


Fig. 14.3b $\tilde{N}$ for sequence lengths of 150,000, with time depth back to 25,000 generations, and epoch lengths of 200 generations

The observable density $f_d = \tilde{N}_d / N_d$ shows very different histories for African genetics compared with out-of-Africa populations. For African populations, the recoverable density f_d appears to decrease with increasing depth, suggesting intervening coalescence events gradually destroy recoverable recombination events, as expected. Interestingly, for out-of-Africa populations, the impact of the out-of-Africa expansion genetic bottleneck appears to mark a transition where prior information shows a highly stochastic regime such that the chances that a recombination survived without being impacted by coalescence is essentially random. A similar signal appears for African populations prior to the initiation of exponential growth among African populations, where the effective population was also quite small. Subsequent to the out-of-Africa expansion, the small effective population sizes in isolated glacial-period populations appears to have

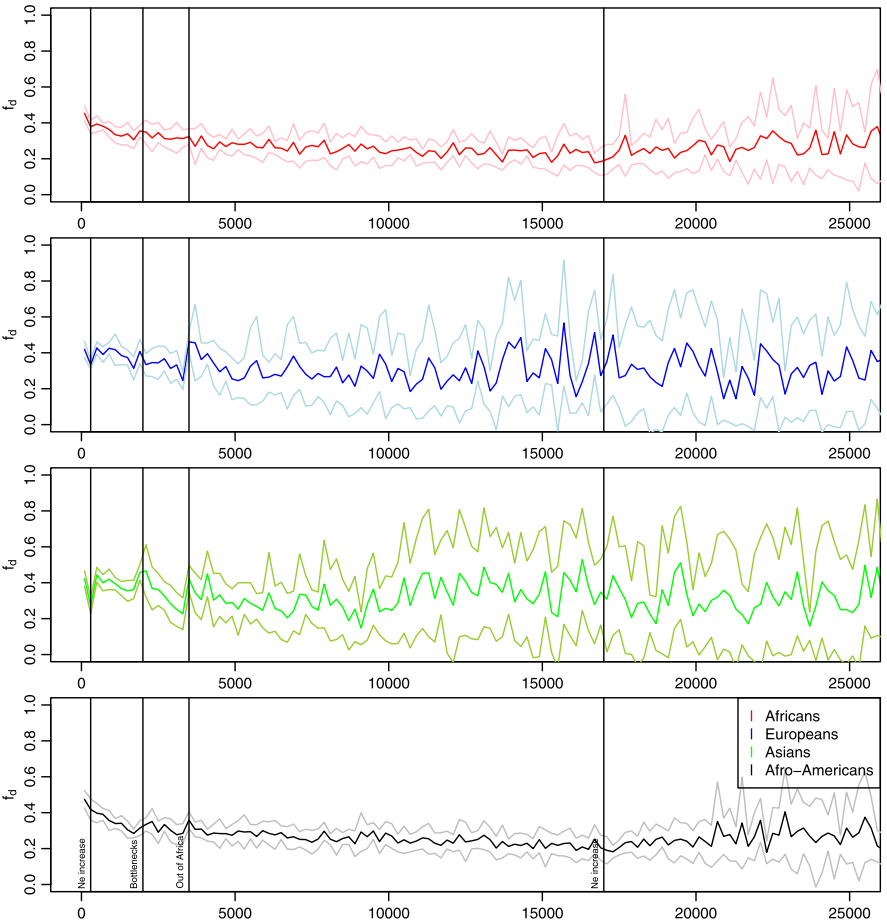


Fig. 14.3c f_d for sequence lengths of 150,000, with time depth back to 25,000 generations, and epoch lengths of 200 generations

been marked by increasing fractions of drift-driven loss of recombination event information. Post-glacial expansions continued the process and isolation between expanding population groups. Overall, lower effective population sizes increases the probability of drift, which appears to be associated with driving higher f_d's.

Figure 14.4 shows that increasing sequence length actually seems to decrease the fraction f of determinable recombinations. This might be expected since, as the number of segments increases, the possibility of more coalescence events destroying information between segments that mark the recombinations also increases. Further, it indicates that increasing mutation rates increases f, which suggests that more SNPs marking edges between recombination events and coalescence events

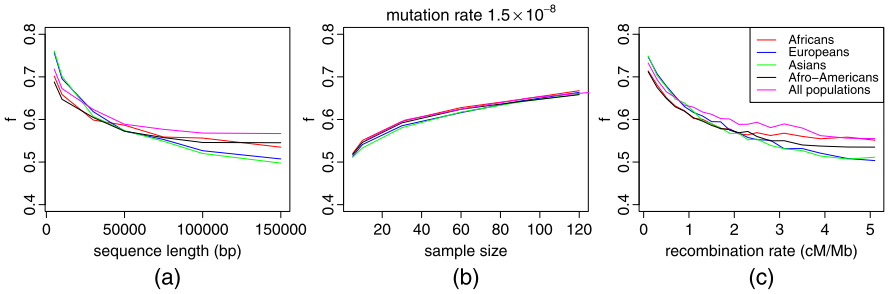


Fig. 14.4 Fraction f of resolvable genetic history averaged over multiple COSI history instances as a function of (**a**) sequence length, (**b**) sample size, and (**c**) recombination rate

provides distinguishing features reducing losses of recombination event information through drift. Lastly, increases in recombination rates relative to SNP mutation rates provides more opportunities for drift related losses of reconstructable events.

Figures 14.5 show, for each demography, the impacts of numbers of base pairs (a–c), sample size (d–f), and recombination rate (g–i), given increasing mutation rates from left to right. Variations over multiple runs have been captured, with 95 % confidence intervals (bars), 50 % confidence intervals (boxes), and medians. As such, these represent the range of variations among different simulated histories given population parameters modeling demographic and environmental parameters that impacted the different demographies.

The means plotted with their standard deviations (Figs. 14.3a–14.3c) do clearly show distinct events revealed across histories, allowing for the possibility of drawing conclusions about population parameters, and for comparative analysis of impacts of ecological changes by comparing extant trends in modern populations.

We observe that the mutation rate does not influence f significantly. Hence, we use a fixed mutation rate of 1.5×10^{-8} bp/gen in the figures. Even though the ARG is not a tree, the density (i.e, number of nodes per epoch) of the relevant genetic events decreases approximately exponentially with depth following Kingman coalescence. Also, the shape of the profiles for the different demographies is independent of the four classes of parameters.

With an increase in sample size averaged over an ensemble of histories, f gradually increases but stabilizes around $f = 0.65$, suggesting that, on average for these demographies, at least about 35 % of the relevant genetic history is impenetrable. However, the most surprising observation comes from the experiments with different sample lengths. It turns out that f decreases with increasing sequence length. We observe that, on average, the reconstructable history of a segment s is actually smaller than the sum of the reconstructable histories of the subsegments of s. This is observed in each of the demographies in isolation, as well as when the demographies are combined into a single universal population.

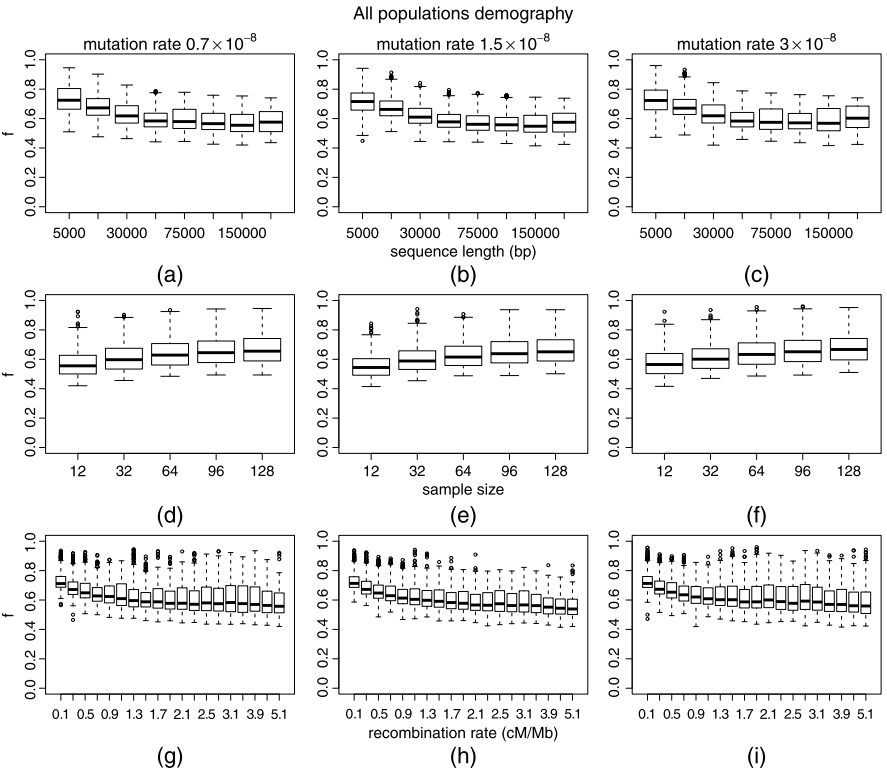


Fig. 14.5 For columns identified by mutation rates 0.7×10^{-8}, 1.5×10^{-8}, and 3×10^{-8} per generation, plots of f vs. sequence length: (**a**), (**b**), and (**c**), plots of f vs. sample size: (**d**), (**e**), and (**f**), and plots of f vs. recombination rates: (**g**), (**h**), and (**i**). f is displayed in whisker plots, showing median, 50 % confidence interval, and extreme range = 1.5 the interquartile range. Outliers beyond the range limit are marked individually

14.4 Conclusions and Future Directions

Reconstructability of common genetic history is a fundamental problem of significant curiosity in the study of populations. While the population evolution models mature and the algorithms get more sophisticated, it becomes important to identify what fraction of the common and relevant genetic history of populations continues to be undeterminable events depends on edges lost to drift, there are no surviving samples against which even variance or other measures would show the existence of those events. They are entirely invisible in the extant record. However, without that information, estimation of how many of these events have left a legacy of genetic diversity is impossible to construct.

We present a framework that enables such an exploration. This is based on the random topological structure, the ARG and a characteristic derived from the generating ARG, for the continental populations. This process does not directly take into account migratory events, bottlenecks or effective population sizes, suggesting

that these events affect the ARG in a manner that is reflected in the history and estimability density.

We built the measure upon a method-independent structure called the minimal descriptor. This was applied to different demographics in a simulation setting. The most surprising observation was that the sum of the reconstructible history of each of the chromosomal segments $s_1, s_2, \ldots, s_m$, is indeed larger than the reconstructible history of the single segment composed of these segments. This appears to be a universal property, holding in all the demographies tested. Also, irrespective of the sample size, we observe that at least one-third of the population genetic history is impenetrable, across all demographies.

We also note that the reconstructable history is generally sufficient to resolve the genetic signatures of demographic, ecological, and climatic events that dominated the history of the demographics, and they are adequate for comparison between specific histories of different populations and demographies.

The framework also opens up possible new directions of investigation. Assume that the characteristics of a population can be derived, say from the linkage disequilibrium landscape and other characteristics of observed extant individuals. Then, can such a generator be used to construct power analysis for study design identifying minimum numbers of samples and sequence lengths required to definitively resolve hypotheses.

14.4.1 Author's Contributions

DP, FU, and MP carried out the experiments and the analysis. LP designed the study. LP, DP, and FU wrote the paper.

References

1. Griffiths, R.C., Marjoram, P.: An ancestral recombinations graph. In: Donnelly, P., Tavare, S. (eds.) Progress in Population Genetics and Human. IMA Vols in Mathematics and Its Applications, vol. 87, pp. 257–270 (1997)
2. Hammer, M.F., Zegura, S.L.: The human y chromosome haplogroup tree: nomenclature and phylogeography of its major divisions. Annu. Rev. Anthropol. **31**, 303–321 (2002)
3. Javed, A., Pybus, M., Melè, M., Utro, F., Bertranpetit, J., Calafell, F., Parida, L.: Iris: construction of arg network at genomic scales. Bioinformatics **27**, 2448–2450 (2011)
4. Kong, A., Gudbjartsson, D., Sainz, J., Jonsdottir, G., Gudjonsson, S., Richardsson, B., Sigurdardottir, S., Barnard, J., Hallbeck, B., Masson, G., Shlien, A., Palsson, S., Frigge, M., Thorgeirsson, T., Gulcher, J., Stefansson, K.: A high-resolution recombination map of the human genome. Nat. Genet. **31**, 241–247 (2002)
5. Li, H., Durbin, R.: Inference of human population history from individual whole-genome sequences. Nature **475**, 493–496 (2011)
6. McVean, G., Myers, S., Hunt, S., Deloukas, P., Bentley, D., Donnelly, P.: The fine-scale structure of recombination rate variation in the human genome. Science **304**, 581–584 (2004)

7. Parida, L.: Graph model of coalescence with recombinations. In: Heath, L., Ramakrishnan, N. (eds.) Problem Solving Handbook in Computational Biology and Bioinformatics, pp. 85–100 (2010)
8. Parida, L.: Nonredundant representation of ancestral recombinations graphs. Methods Mol. Biol. **856** (2012)
9. Parida, L., Palamara, P., Javed, A.: A minimal descriptor of an ancestral recombinations graph. BMC Bioinform. **12**, S6 (2011)
10. Pickrell, J., Coop, G., Novembre, J., Kudaravalli, S., Li, J., Absher, D., Srinivasan, B., Barsh, G., Myers, R., Feldman, M., Pritchard, J.: Signals of recent positive selection in a worldwide sample of human populations. Genome Res. **19**, 826–837 (2009)
11. Sabeti, P., Varilly, P., Fry, B., Lohmueller, J., Hostetter, E., Cotsapas, C., Xie, X., Byrne, E., Mccarroll, S., Gaudet, R., Schaffner, S., Lander, E., Consortium, T.I.H.: Genome-wide detection and characterization of positive selection in human populations. Nature **449**, 913–918 (2007)
12. Sachidanandam, R., Weissman, D., Schmidt, S., Kakol, J., Stein, L., Marth, G., Sherry, S., Mullikin, J., Mortimore, B., Willey, D., Hunt, S., Cole, C., Coggill, P., Rice, C., Ning, Z., Rogers, J., Bentley, D., Kwok, P., Mardis, E., Yeh, R., Schultz, B., Cook, L., Davenport, R., Dante, M., Fulton, L., Hillier, L., Waterston, R., McPherson, J., Gilman, B., Schaffner, S., Van Etten, W., Reich, D., Higgins, J., Daly, M., Blumenstiel, B., Baldwin, J., Stange-Thomann, N., Zody, M., Linton, L., Lander, E., Altshuler, D., International SNP Map Working Group: A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. Nature **409**, 928–933 (2001)
13. Schaffner, S., Foo, C., Gabriel, S., Reich, D., Daly, M., Altshuler, D.: Calibrating a coalescent simulation of human genome sequence variation. Genome Res. **15**, 1576–1583 (2005)
14. Utro, F., Pybus, M., Parida, L.: Sum of parts is greater than the whole: inference of common genetic history of populations. BMC Genomics **14**, S10 (2013)

Index

C. Chauve et al. (eds.), *Models and Algorithms for Genome Evolution*, Computational Biology 19, DOI 10.1007/978-1-4471-5298-9,