8. Reconstructing Optimal Phylogenetic Trees: A Challenge in Experimental Algorithmics

Bernard M. E. Moret[1] and Tandy Warnow[2]

[1] Department of Computer Science, University of New Mexico
Albuquerque, NM 87131, USA
moret@cs.unm.edu

[2] Department of Computer Sciences, University of Texas
Austin, TX 78712, USA
tandy@cs.utexas.edu

Summary.

The benefits of experimental algorithmics and algorithm engineering need to be extended to applications in the computational sciences. In this paper, we present on one such application: the reconstruction of evolutionary histories (phylogenies) from molecular data such as DNA sequences. Our presentation is not a survey of past and current work in the area, but rather a discussion of what we see as some of the important challenges in experimental algorithmics that arise from computational phylogenetics. As motivational examples or examples of possible approaches, we briefly discuss two specific uses of algorithm engineering and of experimental algorithmics from our recent research. The first such use focused on speed: we reimplemented Sankoff and Blanchette's breakpoint analysis and obtained a $200,000$-fold speedup for serial code and 10^8-fold speedup on a 512-processor supercluster. We report here on the techniques used in obtaining such a speedup. The second use focused on experimentation: we conducted an extensive study of quartet-based reconstruction algorithms within a parameter-rich simulation space, using several hundred CPU-years of computation. We report here on the challenges involved in designing, conducting, and assessing such a study.

8.1 Introduction

A *phylogenetic tree* or *phylogeny* is a representation of the evolutionary history of a collection of organisms, in which modern organisms are placed at the leaves of the tree and the (unknown) ancestral organisms occupy internal nodes; the edges of the tree thus denote evolutionary relationships. Due to difficulties in rooting the trees, these phylogenies are usually (but not always) represented by unrooted leaf-labelled trees. Figure 8.1 shows two proposed phylogenies, one for several species of the Campanulaceae (bluebell flower) family (from [8.10]) and the other for herpesviruses that are known to affect humans (from [8.8]). Note that the *Campanulaceae* tree is rooted through the use of a distantly related species (here tobacco), called an *outgroup* in this context (the root is taken to be the internal node to which the outgroup is attached); the herpesvirus tree is unrooted.

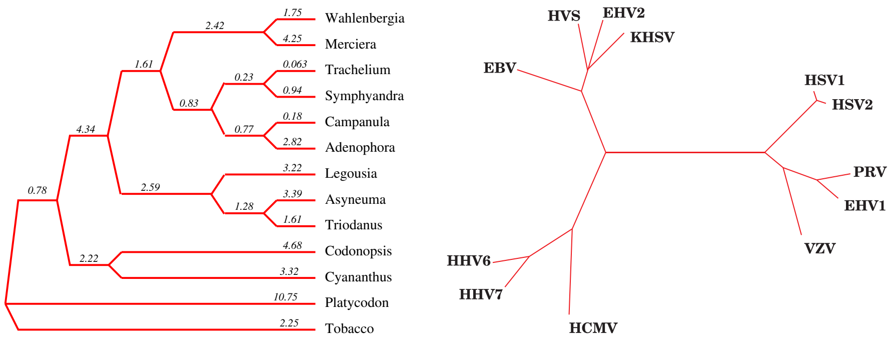

Fig. 8.1. Two phylogenies: some plants of the *Campanulaceae* family (left) and some herpesviruses affecting humans (right)

Reconstructing phylogenies is a major component of modern research programs in many areas of biology and medicine. An understanding of evolutionary mechanisms and relationships is at the heart of modern pharmaceutical research for drug discovery, is helping researchers understand (and defend against) rapidly mutating viruses such as HIV, is the basis for the design of genetically enhanced organisms, etc. In developing such an understanding, the reconstruction of phylogenies is a crucial tool, as it allows one to test new models of evolution. Due to the importance of phylogenetic trees in biological inquiry, there are many methods available for reconstructing phylogenetic trees. Most of these methods are applied to biomolecular sequences, such as DNA, RNA, or amino-acid sequences. More recently, methods have also been developed to reconstruct phylogenies from data on gene content and gene order within a genome.

Evaluating the performance of phylogenetic reconstruction methods is complicated. Since many phylogenetic reconstruction methods are explicit attempts to solve optimization problems, methods may be compared, for example, with respect to the value of these optimization criteria on both real and synthetic data. However, the biological community has also looked at the performance of these methods with respect to an assumed stochastic model of evolution and has evaluated phylogenetic methods with respect to the topological accuracy of the underlying unrooted trees returned by these methods. Thus, phylogenetic methods are evaluated in two different, yet related, ways. Furthermore, the difficulty in establishing true evolutionary histories for many datasets has led the research community to study these questions largely from results on synthetic, rather than real, data, and to use simulations as the main technique. The design of appropriate simulation studies presents interesting and subtle challenges to the researcher in experimental algorithmics.

Fast implementations of phylogenetic methods are also of potentially tremendous impact, since biologists want to apply these methods to large datasets (containing hundreds to thousands of taxa)—and some smaller datasets have required nearly one hundred CPU-years of computation on modern machines for acceptable analyses. Thus, algorithm engineering also has an important role to play in this domain.

In this paper, we review the experimental challenges posed by phylogeny reconstruction, in terms both of algorithm engineering and of data generation, collection, and analysis, and present examples from our own experimental research.

8.2 Data for Phylogeny Reconstruction

Phylogenies are most commonly reconstructed using biomolecular sequences (DNA, RNA, or amino acid) for particular genes or non-coding regions of DNA. More recently, "genomic rearrangement" data have been used to infer deep evolutionary histories (i.e., very ancient evolutionary events), as well as to clarify evolutionary relationships in difficult datasets. The use of genome rearrangement data is part of an increased interest in the development of new sources of phylogenetic information, especially those which can be characterized as "rare genomic changes" (see [8.31] for a survey of these approaches). Sequence data and genomic rearrangement data are highly complementary, with different rates of evolution especially in organelles (chloroplasts and mitochondria), so that using both types of data holds potential for improving accuracy in phylogenetic reconstructions.

DNA, RNA, and amino-acid sequences are used for phylogenetic reconstruction. DNA and RNA sequences can be considered simply as strings over a 4-letter alphabet (A, C, T, and G for DNA), while amino-acid sequences can be considered as strings over a 20-letter alphabet (one for each amino-acid). These sequences evolve through events such as substitutions of one nucleotide by another, insertions and deletions of substrings, etc. Typical sources of biomolecular sequence data are individual genes, so that the sequence coding for a given gene is used in each of the relevant taxa. These sequences are then placed in a multiple alignment through the introduction of spaces; each resulting column of the alignment then corresponds to a place in the sequence and changes can be identified as mutations (two sequences have different entries in that column), insertions (a sequence has an entry, but the other has a space), and deletions (the reverse). Computing a good multiple sequence alignment is itself a hard optimization problem, but outside our scope; we direct the interested reader to [8.38] for an introduction to this problem.

Genome rearrangement data indicate how the genes are ordered within the given genomes. Many organellar genomes are composed of a single chromosome and are relatively small, so that every gene within the genome can be identified and their relative ordering inferred (most accurately through whole genome sequencing, but also through gene mapping). Given an ordering of the genes, we can represent a given genome by an ordering of signed integers. Organellar genomes are thought to evolve via inversions (mechanisms that pick up a segment of a genome and invert it, thus reversing the order of the affected genes), transpositions (mechanisms that pick up a segment of the genome and move it to another position, thus changing the order but not the sign of the affected genes), and inverted transpositions (which are inversions followed by transposition of the inverted segment).

8.2.1 Phylogenetic Reconstruction Methods

For both biomolecular sequences and gene orders, assumptions are made about the mechanism by which these objects evolve. Phylogenetic reconstructions explicitly use assumptions about evolution, but differ in the details of these assumptions. For example, in an analysis of DNA sequences, we may have an explicit model about the evolutionary process; we may know the rates of each type of nucleotide substitution on the true (but unknown) tree. If we assume these rates, then we can seek the tree which is most likely to have generated the given data—the so-called "maximum-likelihood" approach. We can also use these assumptions to infer evolutionary distances between each pair of the given sequences, where the "evolutionary distance" between two sequences is the most likely number of individual changes within the sequence on the path between the two sequences. Both of these approaches have theoretical guarantees, with respect to topological accuracy of the resultant trees, provided that the model is not over-parameterized and that the assumptions about the model are correct. However, maximum-likelihood methods are computationally very intensive, while the second type of methods (called "distance-based" methods) tend to run in polynomial time.

A final class of methods (called "maximum parsimony") does not make any explicit assumption about the model parameters; instead, it seeks a tree with a minimum "number of events". In the context of DNA sequence evolution, these events are nucleotide substitutions, insertions, or deletions, while in the context of gene-order data, they are inversions, transpositions, and inverted transpositions. Maximum parsimony is thus the Steiner Tree Problem for the appropriate space—for instance, the maximum parsimony problem on DNA sequences is the Hamming Distance Steiner Tree problem on strings over a four-letter alphabet. When the input is just the set of taxa (e.g., DNA sequences or gene orders), then the problem is to construct a tree and to label its internal nodes in such a way as to minimize the total number of changes. This problem is NP-hard for both types of data. The point estimation problem, i.e., the scoring of a particular tree topology (in which case the input also includes a specific tree with leaves labelled by the taxa), is solvable in polynomial time for the biomolecular sequence data case, but is NP-hard for gene-order data.

These three types of methods, namely *maximum likelihood*, *distance-based*, and *maximum parsimony*, account for great majority of the methods used by biologists and their relative performance is passionately argued in the biological literature. One of the major limitations of both maximum parsimony and maximum likelihood techniques (even their heuristic versions, which may not have any performance guarantees) is that they take too long. Even some only moderately large datasets can take years of real analysis (hundreds of CPU years), without resolution [8.29]. By comparison, distance-based methods, including the popular Neighbor-Joining (NJ) method [8.32], are often quite accurate (with respect to topological accuracy, as determined using simulation studies) and are very fast (polynomial-time and fast in practice). While the experimental evidence is not yet definitive, the best distance-based methods appear less accurate than the better heuristics for maximum parsimony and maximum likelihood, at least on large trees with high rates of evolution (see, e.g., [8.13]).

8.3 Algorithmic and Experimental Challenges

8.3.1 Designing for Speed

Because both parsimony- and likelihood-based approaches involve NP-hard optimization problems and because poor approximations may lead to biologically incorrect conclusions, developing efficient exact algorithms is a major concern. The range of data used in current analyses is fortunately limited (e.g., the length of available DNA sequences is bounded, as is the number of genes in a mitochondrial or chloroplast genome). This bounded range is tailor-made for applications of algorithm engineering techniques.

8.3.2 Designing for Accuracy

When exact methods fail to terminate, one needs to use approximations. But it is important to keep in mind that the optimization criterion rarely has direct biological significance, so that deviations from optimal, even by small amounts, may yield results that are grossly different from a biological perspective. Thus the development and evaluation of approximation algorithms must be guided by biological considerations. As discussed earlier, the main criterion by which biologists judge the quality of a reconstruction is its topological accuracy, which is only indirectly related to a parsimony or likelihood criterion. Indeed, the great success of the neighbor-joining heuristic, which has no approximation guarantees for the standard optimization criteria, demonstrates that the biological relevance of results matters more than the traditional algorithmic goal of performance guarantees. Simulation experiments can measure topological accuracy, but designing algorithms to produce topologically correct trees is another matter—the methods most closely oriented toward this goal, the family of quartet-based methods, turns out to produce generally much poorer trees than the much simpler neighbor-joining algorithm.

8.3.3 Performance Evaluation

Phylogenetic reconstruction methods are evaluated according to three basic types of criteria: *statistical performance*, which addresses the accuracy of the method under a specified stochastic model of evolution; *computational performance*, which addresses the computational requirements of the method; and *data requirements*, which addresses the requirements, in terms of quality and quantity, placed on the input data. Accuracy in a phylogenetic reconstruction method is determined primarily by comparing the unrooted leaf-labelled tree obtained by the method to the "true" tree. Since the true tree is usually unknown, accuracy is addressed either theoretically, with reference to a fixed but unknown tree in some model of evolution, or through simulation studies. A method is said to be *accurate* if the tree obtained is exactly equal to the unrooted version of the model (or true) tree; degrees of accuracy are quantified typically by the percentage of the edges of the true tree that occur in the estimated tree. A method is said to be *statistically consistent* with respect to a specific model of evolution if it is guaranteed to recover the true tree with probability going to 1 as the amount of data (e.g., sequence length) goes to infinity. The latter property is not as good as it may sound: nature provides us with finite data only—for instance, DNA sequences cannot be of arbitrary length, much less gene orders; thus the rate of convergence is crucial and needs to be evaluated experimentally as well as bounded theoretically (see [8.35] for such an evaluation and [8.37] for a theoretical approach). Data requirements therefore loom large—and indeed may prove more detrimental than computational requirements, since we can always run the program longer.

Because the evolutionary models that biologists favor are parameter-rich, experimental assessment of performance (whether accuracy, convergence rate, or running time) is a daunting task: choosing how to vary the parameters while keeping the total computation down is a difficult tradeoff.

8.4 An Algorithm Engineering Example: Solving the Breakpoint Phylogeny

Blanchette *et al.* [8.6] developed an approach, which they called *breakpoint phylogeny*, for reconstructing phylogenies from gene order data. Their approach is limited to the special case in which the genomes all have the same set of genes and each gene appears once. This special case is of interest to biologists, who hypothesize that inversions (which can only affect gene order, but not gene content) are the main evolutionary mechanism for a range of genomes or chromosomes (chloroplast, mitochondria, human X chromosome, etc.). Simulation studies we conducted suggested that this approach works well for certain datasets (i.e., it obtains trees that are close to the model tree), but that the implementation developed by Sankoff and Blanchette, the

```
For each tree topology do:
  Initially label all internal nodes with gene orders
  Repeat
    For each internal node $v$, with neighbors $A$, $B$, and $C$, do
        Solve the MPB on $A, B, C$ to yield label $m$
        If relabelling $v$ with $m$ improves the score of $T$, then do it
  until no internal node can be relabelled
```

Fig. 8.2. `BPAnalysis`

`BPAnalysis` software [8.33], is too slow to be used on anything other than small datasets with a few genes [8.9, 8.10].

8.4.1 Breakpoint Analysis: Details

When each genome has the same set of genes and each gene appears exactly once, a genome can be described by an ordering (circular or linear) of these genes, each gene given with an orientation that is either positive (g_i) or negative ($-g_i$). 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), such that g_i and g_j appear consecutively in that order in G, but neither (g_i, g_j) nor $(-g_j, -g_i)$ appears consecutively in that order in G'. The breakpoint distance between two genomes is the number of breakpoints between that pair of genomes. The breakpoint score of a tree in which each node is labelled by a signed ordering of genes is then the sum of the breakpoint distances along the edges of the tree.

Given three genomes, we define their *median* to be a fourth genome that minimizes the sum of the breakpoint distances between it and the other three. The *Median Problem for Breakpoints* (MPB) is to construct such a median and is NP-hard [8.27]. Sankoff and Blanchette developed a reduction from MPB to the Travelling Salesman Problem (TSP), perhaps the most studied of all optimization problems [8.15]. Their reduction produces an undirected instance of the TSP from the directed instance of MPB by the standard technique of representing each gene by a pair of cities connected by an edge that must be included in any solution.

`BPAnalysis` (see Figure 8.2) is the method developed by Blanchette and Sankoff to solve the breakpoint phylogeny. Within a framework that enumerates all trees, it uses an iterative heuristic to label the internal nodes with signed gene orders. This procedure is computationally very intensive. The outer loop enumerates all $(2n-5)!!$ leaf-labelled trees on n leaves, an exponentially large value.[1] The inner loop runs an unknown number of iterations (until convergence), with each iteration solving an instance of the TSP (with a number of cities equal to twice the number of genes) at each internal node.

[1] The double factorial is a factorial with a step of 2, so we have $(2n-5)!! = (2n-5) \cdot (2n-7) \cdot \ldots \cdot 3$.

The computational complexity of the entire algorithm is thus exponential in *each* of the number of genomes and the number of genes, with significant coefficients. The procedure nevertheless remains a heuristic: even though all trees are examined and each MPB problem solved exactly, the tree-labeling phase does not ensure optimality unless the tree has only three leaves.

8.4.2 Re-Engineering BPAnalysis for Speed

Profiling. Algorithmic engineering suggests a refinement cycle in which the behavior of the current implementation is studied in order to identify problem areas which can include excessive resource consumption or poor results. We used extensive profiling and testing throughout our development cycle, which allowed us to identify and eliminate a number of such problems. For instance, converting the MPB into a TSP instance dominates the running time whenever the TSP instances are not too hard to solve. Thus we lavished much attention on that routine, down to the level of hand-unrolling loops to avoid modulo computations and allowing reuse of intermediate expressions; we cut the running time of that routine down by a factor of at least six—and thereby nearly tripled the speed of the overall code. We lavished equal attention on distance computations and on the computation of the lower bound, with similar results. Constant profiling is the key to such an approach, because the identity of the principal "culprits" in time consumption changes after each improvement, so that attention must shift to different parts of the code during the process—including revisiting already improved code for further improvements. These steps provided a speed-up by one order of magnitude on the Campanulaceae dataset.

Cache Awareness. The original `BPAnalysis` is written in C++ and uses a space-intensive full distance matrix, as well as many other data structures. It has a significant memory footprint (over 60MB when running on the Campanulaceae dataset) and poor locality (a working set size of about 12MB). Our implementation has a tiny memory footprint (1.8MB on the Campanulaceae dataset) and good locality (all of our storage is in arrays preallocated in the main routine and retained and reused throughout the computation), which enables it to run almost completely in cache (the working set size is 600KB). Cache locality can be improved by returning to a FORTRAN-style of programming, in which storage is static, in which records (structures/classes) are avoided in favor of separate arrays, in which simple iterative loops that traverse an array linearly are preferred over pointer dereferencing, in which code is replicated to process each array separately, etc. (This style of programming is not always easy to reconcile with the currently favored object-oriented style; fortunately, compiler support for this type of code optimization is slowly developing—as of January 2002, for instance, at least one commercial compiler could optimize storage access by breaking an array of record into multiple arrays. We found it easier to code in C, simply because of the much greater transparency of the language.) While we cannot measure exactly how much we gain from this approach, studies of cache-aware algorithms [8.1, 8.11, 8.18, 8.19, 8.20, 8.39] indicate that the gain is likely to be substantial—factors of anywhere from 2 to 40 have been reported. New memory hierarchies show differences in speed between cache and main memory that exceed two orders of magnitude.

Low-Level Algorithmic Changes. Unless the original implementation is poor (which was not the case with `BPAnalysis`), profiling and cache-aware programming will rarely provide more than two orders of magnitude in speed-up. Further gains can often be obtained by low-level improvement in the algorithmic details. In our phylogenetic software, we made two such improvements. The basic algorithm scores every single tree, which is clearly very wasteful; we used a simple lower bound, computable in linear time, to enable us to eliminate a tree without scoring it. On the Campanulaceae dataset, this bounding eliminates over 99.95% of the trees without scoring them, resulting in a 100-fold speed-up. The TSP solver we wrote is at heart the same basic include/exclude search as in `BPAnalysis`, but we took advantage of the nature of the instances created by the reduction to make the solver much more efficient, resulting in a speed-up by a factor of 5–10. These improvements all spring from a careful examination of exactly what information is readily available or easily computable at each stage and from a deliberate effort to make use of all such information.

A High-Performance Implementation. Our implementation, *GRAPPA*[2], incorporates all of the refinements mentioned above, plus others specifically made to enable the code to run efficiently in parallel (see [8.23, 8.24, 8.26] for details). Because the basic algorithm enumerates and independently scores every tree, it presents obvious parallelism: we can have each processor handle a subset of the trees. In order to do so efficiently, we need to impose a linear ordering on the set of all possible trees and devise a generator that can start at an arbitrary point along this ordering. Because the number of trees is so large, an arbitrary tree index would require unbounded-precision integers, considerably slowing down tree generation. Our solution was to design a tree generator that starts with tree index k and generates trees with indices $\{k + cn \mid n \in \mathcal{N}\}$, where k and c are regular integers, all without using unbounded-precision arithmetic. Such a generator allows us to sample tree space (a very useful feature in research) and, more importantly, allows us to use a cluster of c processors, where processor i, $0 \leq i \leq c-1$, generates and scores trees with indices $\{i + cn \mid n \in \mathcal{N}\}$. We ran *GRAPPA* on the 512-processor Alliance cluster *Los Lobos* at the University of New Mexico and obtained a 512-fold speed-up. When combined with the nearly 200,000-fold speedup obtained through algorithm engineering, our run on the Campanulaceae dataset demonstrated a *one hundred million-fold* speed-up over the

[2] Genome Rearrangement Analysis through Parsimony and other Phylogenetic Algorithms.

original implementation [8.26] (a first speedup of one million was reported in [8.2]).

8.4.3 A Partial Assessment

Clearly, generating every single tree is a self-defeating approach: even our huge 10^8-fold speedup allowed us to move from 10 taxa to just 16 taxa—and 20 or more taxa remain forever out of reach of this computational approach. A real search strategy should reduce the cost of computation by an enormous factor. Yet this exercise in algorithm engineering already produced significant results in both biology and computer science: the analysis of the *Campanulaceae* dataset conformed to the expectations of the biologists and thus reinforced the conjecture that gene-order data carries significant information about evolution (and, incidentally, that inversion-driven rearrangements are indeed the main mechanism for such evolution), while the improved understanding of inversion distance computations gained through the implementation enabled us to design the first true linear-time algorithm for this purpose. In turn, the availability of a fast implementation for inversion distance computations and inversion-based phylogenies has spurred renewed interest in the inversion median problem [8.7, 8.34] and other related problems.

8.5 An Experimental Algorithmics Example: Quartet-Based Methods for DNA Data

8.5.1 Quartet-Based Methods

A *quartet tree* is an unrooted binary tree on four taxa. A quartet tree thus induces a unique bipartition of the four taxa and can be denoted by that bipartition. If the taxa are $\{a, b, c, d\}$, we can use $\{ab|cd\}$ to denote the quartet tree that pairs a with b and c with d (see Figure 8.3). A quartet tree $\{ab|cd\}$ *agrees* with a tree T if all four of its taxa are leaves of T and the path from a to b in T does not intersect the path from c to d in T. Equivalently, $\{ab|cd\}$ agrees with a tree if the subtree induced in T by the four taxa is the quartet tree itself. The quartet tree $\{ab|cd\}$ *is an error* with respect to the tree T if it does not agree with T. If $Q(T)$ denotes the set of all quartet trees that agree with T, then T is uniquely characterized by $Q(T)$ and can be reconstructed from $Q(T)$ in polynomial time [8.12].

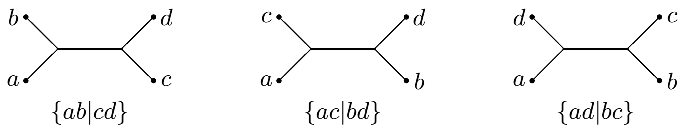

Fig. 8.3. The three possible quartet trees on four taxa $\{a, b, c, d\}$ and their bipartition encodings

Quartet-based methods operate in two phases. In the first phase they construct a set Q of quartet trees on the different sets of four taxa; in the second phase, they combine these quartet trees into a tree on the entire set of taxa. In practice, the input data are not of sufficient quality to ensure that all quartet trees are accurately inferred, so that quartet methods have to find ways of handling incorrect quartet trees. With the exception of Quartet Puzzling, all quartet methods we examine provide guarantees about the edges of the true tree that they reconstruct. These guarantees are expressed in terms of "quartet errors around an edge," a concept we now define.

Consider an edge e in the true tree T; its removal defines the bipartition $A|B$ on the leaves of S. Consider those sets of four leaves $\{a, a', b, b'\}$ with $\{a, a'\} \subseteq A$ and $\{b, b'\} \subseteq B$. A quartet tree t is said to be an "error around e" if we have $t = \{ab|a'b'\}$ or $t = \{ab'|a'b\}$. Similarly, if T' is a proposed tree and Q is a set of quartet trees, then $t \in Q$ is an error around edge $e \in E(T')$ if $t = \{ab|a'b'\}$ or $t = \{ab'|a'b\}$, while e defines the bipartition $A|B$.

Two of the methods we study, the Q^* method (also known as the Buneman method) and the Quartet-Cleaning methods, can be described in terms of an explicit bound on the number of quartet errors around the edges they reconstruct. The Q^* method [8.4] seeks the *maximally resolved* tree T' obeying $Q(T') \subseteq Q$; therefore, there are *no quartet errors* around any edge in the tree T'. *Quartet-Cleaning (QC)* methods [8.3, 8.5, 8.14] have explicit bounds on the number of quartet errors around each reconstructed edge e. These error bounds have the form $m\sqrt{q_e}$, where q_e is the number of quartet trees around edge e and m is a small constant. Thus, the Q^* method is a cleaning method with $m = 0$. The *global cleaning* method sets $m = 1$ and the *local cleaning* method sets $m = \frac{1}{2}$; these methods are guaranteed to recover every edge of the true tree for which Q contains a small enough number of quartet errors. The *hypercleaning* method allows m to be an arbitrary integer and thus has the potential to recover more edges, at the cost of a high running time (proportional to $n^7 \cdot m^{4m+2}$), so that it is impractical for m larger than 5.

The final quartet-based method we examined is the best known and the most frequently used by biologists [8.22, 8.30, 8.17]: the *Quartet-Puzzling (QP)* method [8.36]. This heuristic computes quartet trees using maximum likelihood (ML) and then uses a greedy strategy to construct a tree on which many input quartets are in agreement. QP uses an arbitrary ordering of taxa, constructs the optimal quartet tree on the first four, then inserts each successive taxon in turn, attaching the new leaf to an edge of the current tree so as to optimize a quartet-based score. Because the input ordering of taxa is pertinent, QP uses a large number of random input orderings and computes the *majority consensus* of all trees found. (The majority consensus is the tree that contains all bipartitions that appear in more than half of the trees in the set and is commonly used by biologists.)

8.5.2 Experimental Design

We used Jukes-Cantor model trees with varying numbers of taxa and rates of evolution to generate a large number of synthetic datasets of varying lengths. (The Jukes-Cantor model [8.16] is the simplest of the various evolutionary models, with just one parameter.) For each dataset generated, we computed the neighbor-joining (NJ) and QP trees on the entire dataset and two sets of quartet trees, one based upon ML, Q_{ML}, and one based upon NJ, Q_{NJ}. We then applied various cleaning methods to each of the sets Q_{ML} and Q_{NJ}. We compared quartet trees of Q_{ML}, of Q_{NJ}, and of the reconstructed trees, as well as the reconstructed trees themselves, against the model tree for accuracy.

We randomly generated model tree topologies from the uniform distribution on binary leaf-labelled trees. For each edge of each tree topology, we generated a random number (from the uniform distribution) between 1 and 1000 and used that number as the initial "length" of the edge. We then scaled each such "base" model tree by a multiplicative factor, ranging from 10^{-7} to 10^{-3}. This process produces Jukes-Cantor trees with edge lengths (λ_e for edge e) ranging from a minimum of 10^{-7} to a maximum of 1. The edge length denotes the probability that a particular character in the sequence at the base of the edge will be affected by an evolutionary event along the edge; thus the expected number of changes affecting the sequence at the base of the edge is the product of the edge length by the sequence length. In the following we write $\overline{\lambda_e}$ to denote the average edge length in a collection of trees—which is just 500 times the scaling factor. We generated random DNA sequences for the root and used the program `Seq-Gen` [8.28] to evolve these sequences down the tree under the Jukes-Cantor model of evolution, thus producing sets of sequences at the leaves, our synthetic datasets.

Because the number of distinct unrooted, leaf-labelled trees on n leaves is $(2n-5)!!$ and because our input space is further expanded by the choice of evolutionary rates, it is not possible to take a fair sample of the entire input space. In order to obtain statistically robust results, we followed the advice of McGeoch [8.21] and Moret [8.25] and used a number of *runs*, each composed of a number of *trials* (a trial is a single comparison), computed the mean outcome for each run, and studied the mean and standard deviation over the runs of these events.

A critical parameter of our study, one that has not been explored in most prior studies, is the number of input taxa. Previous experimental studies have often been limited to a small number of taxa due to computational problems. However, to resolve phylogenetic trees of interest to biologists, algorithms must scale reasonably, both in terms of topological accuracy and running time, to problems of the size that biologists typically study (20–200 taxa), as well as those they would like to address (a few hundred to several thousand taxa).

We ran our test suite for 5, 10, 20, 40, and selected sets of 80 taxa. Our tests included a selection of eight expected evolutionary rates, from $5 \times$

10^{-5} to 5×10^{-1} per tree edge. For each evolutionary rate and problem size, we generated a total of 100 topologies, grouped into ten runs of ten trials. All tests were conducted for four sequence lengths: 500, 2,000, 8,000, and 32,000. We note that sequence lengths above 1,000 are considered long and those above 5,000 extremely long; thus our study explores longer sequence lengths than are usually encountered in practice. In all, our study used 16,000 datasets and required many months of computation on two medium-sized clusters.

Our focus was the accuracy of solutions generated by the various tree reconstruction methods. To assess topological accuracy, we measured the number of true positives (edges of the true tree that appear in the reconstructed tree). For cleaning methods, we measured these values before and after cleaning. For each run of ten trials, we retained only the mean values. Our results are composed of the means for each set of ten runs.

8.5.3 Some Experimental Results

We provide only a few illustrative results from our study [8.35]. Because our focus was accuracy, we wanted to find out whether the goal of minimizing quartet errors would correlate closely with the true goal of maximizing topological accuracy. Our results showed convincingly that topological accuracy is a more demanding criterion than quartet accuracy and should therefore be used to assess performance of phylogenetic reconstruction methods; typical results are shown in Figures 8.4 and 8.5. Both NJ and QP can return trees with only 20% of the edges correct from a set of quartet trees that is 80% correct. Worse yet, both methods, except when the percentage of correct quartet trees is close to 100%, can return fewer than 80% of the true tree edges (in

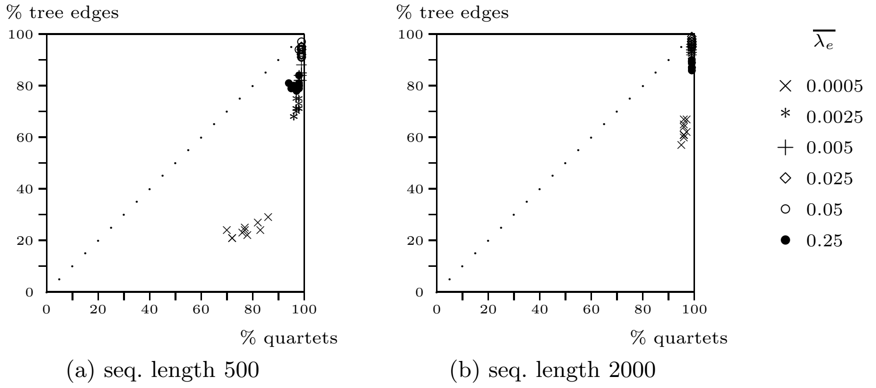

Fig. 8.4. Percent of true tree edges recovered by global NJ for various $\overline{\lambda_e}$ as a function of the percentage of correct induced quartet trees for 40 taxa and two sequence lengths

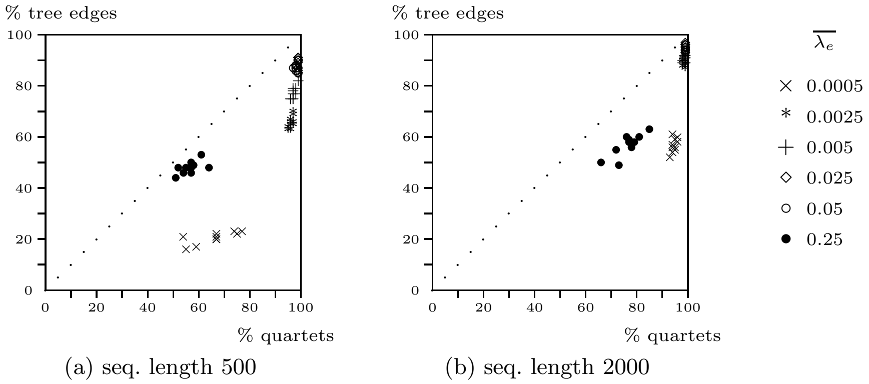

Fig. 8.5. Percent of true tree edges recovered by QP for various $\overline{\lambda_e}$ as a function of the percentage of correct induced quartet trees for 40 taxa and two sequence lengths

the case of QP, some such trees had only 60% of the true tree edges). Because failure to obtain at least 90 or 95% of the edges can be unacceptable to systematists, quartet-based measures of accuracy are not acceptable surrogates for true tree edges.

Theory predicts that the accuracy of methods will degrade as the number of taxa increases while sequence length and average edge length (the expected number of changes for a random site on each edge) are held fixed. Figure 8.6 shows the topological accuracy achieved by all six methods as a function of the number of taxa for a sequence length of 500 and for three different average edge lengths. All methods decrease in accuracy as the number of taxa increases, even though both NJ and QP show an initial increase (particularly for lower evolutionary rates). QC provides a distinct improvement over the Q^* method, whether the quartet trees are computed using ML or local NJ. QCML and QCNJ are very close in performance, although QCNJ slightly outperforms QCML; similarly Q^*NJ slightly outperforms Q^*ML. Of the five quartet methods, QP is the best throughout the range of parameters studied, but NJ completely dominates it.

8.6 Observations and Conclusions

Our two examples illustrate two different facets of computational phylogenetics: the first shows that algorithmic engineering can turn what appears to be strictly a proof of concept into a usable tool, while the second demonstrates the scale required in good experimentation when assessing the behavior of reconstruction algorithms. It is worth stressing that the goal of a biologist is to analyze a specific dataset—so that the biologist will not mind running a cluster on the problem for as long as necessary (weeks or months), since

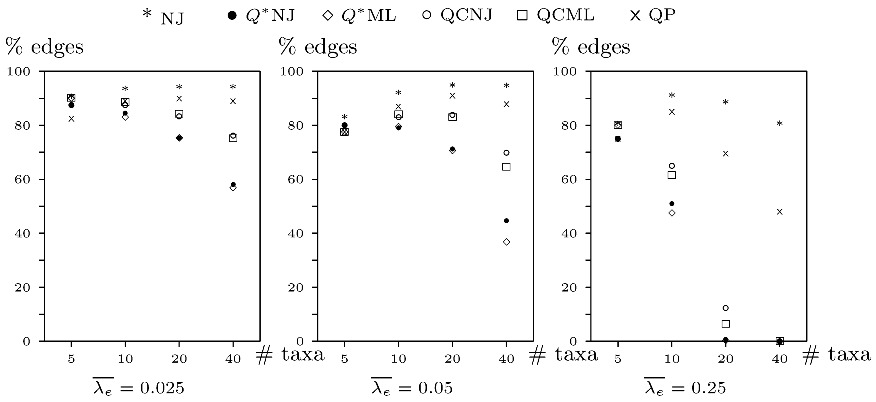

Fig. 8.6. Number of taxa *vs.* percentage of edges correct for sequence length 500 and various $\overline{\lambda_e}$

just one instance must be solved. In contrast, the role of the algorithm designer or engineer is to document the practical performance of an algorithm, which requires many runs on many different sizes and types of data. In consequence, an algorithm that a biologist finds acceptable computationally may be judged completely inadequate by an algorithm engineer—a discrepancy that can only be resolved through a close collaboration between the biologist and the algorithm engineer.

A boon to the experimentalist in the area is the availability of real datasets: academic biologists are generally very free with their data and only too pleased to have an algorithm designer help them answer their questions. We have a national database that stores most known DNA sequences (GenBank) and government laboratories that sequence organellar genomes, providing vast amounts of challenging problems to the algorithms community. On the other hand, the absence of consensus on a model of evolution makes it difficult to obtain definitive results. Current models appear to be brittle, in the sense that small deviations from optimality may cause significant degradation of the quality of the solution as perceived by a biologist; the optimality criteria need to be refined to remedy this problem. Again, working with a research biologist is crucial, as the biologist can sift through the changes in model parameters and the output produced under each model and prepare an analysis and, if necessary, a new model.

On the algorithm engineering side, the area needs a large effort in code production, code that is made freely available to biologists everywhere, in order to replace poor existing codes, implement new ideas, and provide the most efficient tools to the biologists. Perhaps more than in any other area, there is an opportunity in computational biology, in particular in computational phylogenetics, for algorithm designers and engineers to have a profound impact on the course of scientific research.

Acknowledgments

This work was supported in part by NSF grants CCR 94-57800 (Warnow), ACI 00-81404 (Moret), DEB 01-20709 (Moret and Warnow), EIA 01-13095 (Moret), EIA 01-13654 (Warnow), EIA 01-21377 (Moret), and EIA 01-21680 (Warnow), and by the David and Lucile Packard Foundation (Warnow).

References

8.1 L. Arge, J. Chase, J. S. Vitter, and R. Wickremesinghe. Efficient sorting using registers and caches. In *Proceedings of the 4th Workshop on Algorithm Engineering (WAE'00)*. Springer Lecture Notes in Computer Science 1982, 2000.

8.2 D. A. Bader and B. M. E. Moret. GRAPPA runs in record time. *HPC Wire*, 9(47), 2000.

8.3 V. Berry, D. Bryant, T. Jiang, P. Kearney, M. Li, T. Wareham, and H. Zhang. A practical algorithm for recovering the best supported edges of an evolutionary tree. In *Proceedings of the 11th ACM/SIAM Symposium on Discrete Algorithms (SODA'00)*, pages 287–296, 2000.

8.4 V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Theoretical Computer Science*, 240(2):271–298, 2000.

8.5 V. Berry, T. Jiang, P. Kearney, M. Li, and T. Wareham. Quartet cleaning: improved algorithms and simulations. In *Proceedings of the 7th European Symposium on Algorithms (ESA'99)*. Springer Lecture Notes in Computer Science 1643, pages 313–324, 1999.

8.6 M. Blanchette, G. Bourque, and D. Sankoff. Breakpoint phylogenies. In S. Miyano and T. Takagi, editors, *Genome Informatics 1997*, pages 25–34. Univ. Academy Press, Tokyo, 1997.

8.7 A. Caprara. On the practical solution of the reversal median problem. In *Proceedings of the 1st Workshop on Algorithms for Bioinformatics (WABI'01)*. Springer Lecture Notes in Computer Science 2149, pages 238–251, 2001.

8.8 J. I. Cohen. Epstein-barr virus infection. *New England Journal of Medicine*, 343(7):481–492, 2000.

8.9 M. E. Cosner, R. K. Jansen, B. M. E. Moret, L. A. Raubeson, L.-S. Wang, T. Warnow, and S. K. Wyman. An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae. In D. Sankoff and J. Nadeau, editors, *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pages 99–121. Kluwer, 2000.

8.10 M. E. Cosner, R. K. Jansen, B. M. E. Moret, L. A. Raubeson, L. Wang, T. Warnow, and S. K. Wyman. A new fast heuristic for computing the breakpoint phylogeny and experimental phylogenetic analyses of real and synthetic data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'00)*, pages 104–115, 2000.

8.11 N. Eiron, M. Rodeh, and I. Stewarts. Matrix multiplication: a case study of enhanced data cache utilization. *ACM Journal of Experimental Algorithmics*, 4(3), 1999. Online at `www.jea.acm.org/1999/EironMatrix/`.

8.12 P. Erdős, M. A. Steel, L. A. Székely, and T. Warnow. A few logs suffice to build (almost) all trees I. *Random Structures and Algorithms*, 14:153–184, 1997.

8.13 D. Huson, S. Nettles, K. Rice, T. Warnow, and S. Yooseph. Hybrid tree reconstruction methods. *ACM Journal of Experimental Algorithmics*, 4(5), 1999. Online at `www.jea.acm.org/1999/HusonHybrid/`.

8.14 T. Jiang, P. E. Kearney, and M. Li. A polynomial-time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM Journal on Computing.* To appear.

8.15 D. S. Johnson and L. A. McGeoch. The traveling salesman problem: a case study. In E. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley, 1997.

8.16 T. H. Jukes and C. Cantor. *Mammalian Protein Metabolism.* Academic Press, 1969.

8.17 P. J. Keeling, M. A. Luker, and J. D. Palmer. Evidence from beta-tubulin phylogeny that microsporidia evolved from within the Fungi. *Molecular Biology and Evolution*, 17:23–31, 2000.

8.18 R. Ladner, J. D. Fix, and A. LaMarca. The cache performance of traversals and random accesses. In *Proceedings of the 10th ACM/SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 613–622, 1999.

8.19 A. LaMarca and R. Ladner. The influence of caches on the performance of heaps. *ACM Journal of Experimental Algorithmics*, 1(4), 1996. Online at `www.jea.acm.org/1996/LaMarcaInfluence/`.

8.20 A. LaMarca and R. Ladner. The influence of caches on the performance of sorting. In *Proceedings of the 8th ACM/SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 370–379, 1997.

8.21 C. C. McGeoch. Analyzing algorithms by simulation: variance reduction techniques and simulation speedups. *ACM Computing Surveys*, 24:195–212, 1992.

8.22 B. Mishof, C. L. Anderson, and H. Hadrys. A phylogeny of the damselfly genus *Calopteryx* (Odonata) using mitochondrial 16s rDNA markers. *Molecular Phylogeny Evolution*, 15:5–14, 2000.

8.23 B. M. E. Moret, D. A. Bader, and T. Warnow. High-performance algorithm engineering for computational phylogenetics. In *Proceedings of the 2001 International Conference on Computational Science (ICCS'01).* Springer Lecture Notes in Computer Science 2073–2074, 2001.

8.24 B. M. E. Moret, S. K. Wyman, D. A. Bader, T. Warnow, and M. Yan. A new implementation and detailed study of breakpoint analysis. In *Proceedings of the 6th Pacific Symposium Biocomputing (PSB'01).* World Scientific, pages 583–594, 2001.

8.25 B. M. E. Moret and H. D. Shapiro. Algorithms and experiments: the new (and old) methodology. *Journal on Universal Computer Science*, 7(5):434–446, 2001.

8.26 B. M. E. Moret, J. Tang, L.-S. Wang, and T. Warnow. Steps toward accurate reconstruction of phylogenies from gene-order data. *Journal on Computer and System Sciences.* To appear.

8.27 I. Pe'er and R. Shamir. The median problems for breakpoints are NP-complete. *Electronic Colloqium on Computational Complexity*, 71, 1998.

8.28 A. Rambaut and N. C. Grassly. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Computational Applications in Biosciences*, 13:235–238, 1997.

8.29 K. Rice, M. Donoghue, and R. Olmstead. Analyzing large datasets: rbcl500 revisited. *System Biology*, 46:554–562, 1997.

8.30 F. Rodrigues-Trelles, L. Alarcon, and A. Fontdevila. Molecular evolution and phylogeny of the *buzzatii* complex (*D. repleta* group): a maximum likelihood approach. *Molecular Biology Evolution*, 17:1112–1122, 2000.

8.31 A. Rokas and P. W. H. Holland. Rare genomic changes as a tool for phylogenetics. *Trends in Ecology and Evolution*, 15:454–459, 2000.
8.32 N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology Evolotion*, 4:406–425, 1987.
8.33 D. Sankoff and M. Blanchette. Multiple genome rearrangement and breakpoint phylogeny. *Journal on Computational Biology*, 5:555–570, 1998.
8.34 A. C. Siepel and B. M. E. Moret. Finding an optimal inversion median: experimental results. In *Proceedings of the 1st Workshop on Algorithms for Bioinformatics (WABI'01)*. Springer Lecture Notes in Computer Science 2149, pages 189–203, 2001.
8.35 K. St. John, T. Warnow, B. M. E. Moret, and L. Vawter. Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining. In *Proceedings of the 12th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 196–205, 2001.
8.36 K. Strimmer and A. von Haeseler. Quartet puzzling: a maximum likelihood method for reconstructing tree topologies. *Molecular Biology Evolution*, 13:964–969, 1996.
8.37 T. Warnow, B. M. E. Moret, and K. St. John. Absolute phylogeny: true trees from short sequences. In *Proceedings of the 12th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 186–195, 2001.
8.38 M. S. Waterman. *Introduction to Computational Biology: Sequences, Maps and Genomes*. Chapman Hall, 1995.
8.39 L. Xiao, X. Zhang, and S. A. Kubricht. Improving memory performance of sorting algorithms. *ACM Journal of Experimental Algorithmics*, 5(3), 2000. Online at `www.jea.acm.org/2000/XiaoMemory/`.