Toward New Software for Computational Phylogenetics

Phylogenetic software must combine the best precepts of high-performance algorithm engineering and a new level of algorithmic sophistication if it is to handle very large data sets.

Bernard M.E. Moret
University of New Mexico

Li-San Wang
Tandy Warnow
University of Texas, Austin

How a group of genes or organisms evolved is a fundamental question in biology. Systematists—biologists who study such evolution—are now setting their sights on the Tree of Life challenge: to reconstruct the evolutionary history of all known living organisms. To meet this challenge, they will need far more powerful algorithms and software than are currently available.

A typical phylogenetic reconstruction starts with biomolecular data, such as DNA sequences, for modern organisms and builds a tree, or *phylogeny*, for these sequences, which represents a hypothesized evolutionary history. Figure 1 shows two sample phylogenies in different graphical formats. The leaves represent the given sequences, the internal nodes represent putative ancestral sequences (from which the given sequences evolved), and the edges represent lines of descent. The edges of the tree can be assigned lengths, indicating the evolutionary distance—the number of mutations that occur on the edge, which can also be viewed as an indication of time whenever evolution is thought to obey a clock-like model.

Finding the best tree for a data set can be a computationally intensive problem. Most of the favored approaches attempt to optimize an NP-hard optimization problem; they typically use hill-climbing heuristics to search through a very large space (the number of possible unrooted trees on just n organisms grows very quickly with n: It is close to 14 billion for $n = 13$ and more than 2×10^{20} for $n = 20$).

Many software packages have been developed for these problems, of which PAUP*[1] is the most popular among biologists. Reconstructions using these packages have typically been limited (mostly because of running time) to sets of some dozens, or at most some hundreds, of organisms. Yet current applications may involve thousands of organisms, while the grand challenge of inferring the Tree of Life—the evolutionary history of all known living organisms (see tolweb.org)—will require scaling algorithms up (in terms of speed and accuracy) to more than a million organisms.

Designing new techniques that can obtain more accurate solutions to these hard optimization problems is thus a major focus of phylogenetics research.[2] Phylogenetic software for the Tree of Life and for the new era of whole-genome bioinformatics will require entirely new approaches in four main areas:

- *Statistical models of evolution.* Optimization criteria for phylogenetic reconstruction must be based on more realistic models of evolution so that the solution of these optimization problems will lead to trees that more closely approximate the true tree.
- *Phylogeny reconstruction algorithms.* New algorithms are needed that can obtain optimal (or nearly optimal) solutions and scale gracefully to large instance sizes.
- *High-performance implementations.* Implementations that take advantage of the entire range of hardware platforms will significantly boost the speed with which we can analyze data sets.

0018-9162/02/$17.00 © 2002 IEEE

- *Data analysis and visualization.* Biologists must be able to explore the space of optimal and near-optimal solutions, to intervene in the optimization process by adding fixed constraints, and to interact with very large databases of previously computed trees.

We address the related problems of designing and implementing better algorithms for phylogeny reconstruction. (The other two problems—developing better statistical models of evolution and data analysis and visualization tools—involve research focused on statistics, databases, data mining, and visualization.)

To illustrate the problems and techniques of algorithm development, implementation, and testing in phylogeny reconstruction, we use three examples from our own research. The first example focuses on new algorithm development, the second on how performance can be improved through more sophisticated use of statistical models, and the third on high-performance implementation. All three include experimental testing, a crucial component in the assessment of an algorithm's performance.

Our research addresses phylogeny reconstruction from biomolecular sequences, focusing on the accuracy of reconstructions and the use of simulations. We have developed some new polynomial-time algorithmic techniques explicitly for this kind of data, which our simulation studies show provide greater accuracy in reconstructing the true tree than standard polynomial-time methods. The studies also show that the improvement in accuracy grows with the number of sequences in the input.

Our research also addresses new data types derived from whole-genome sequencing. Given the growing availability of fully sequenced whole genomes, the order in which genes appear along chromosomes has become a potentially important new source of information in phylogenetic reconstructions.

We have developed two complementary approaches to using such data. Our first approach uses statistically based estimation techniques to improve the accuracy of distance-based phylogeny reconstruction methods. Our second approach uses algorithm-engineering techniques to obtain significantly shorter running times for reconstruction. We were able to speed up earlier heuristics for the NP-hard breakpoint phylogeny problem[3] more than a millionfold.

RECONSTRUCTION DATA AND METHODS

Phylogenies are most commonly reconstructed using biomolecular sequences (DNA, RNA, or protein) for particular genes or noncoding regions of DNA. More recently, gene-order data have been used to infer deep evolutionary histories (very old evolutionary events), as well as to clarify evolutionary relationships in difficult data sets. DNA and RNA sequences can be considered simply as strings over a four-letter alphabet, while protein sequences can be considered as strings over a 20-letter alphabet. These sequences evolve through events such as

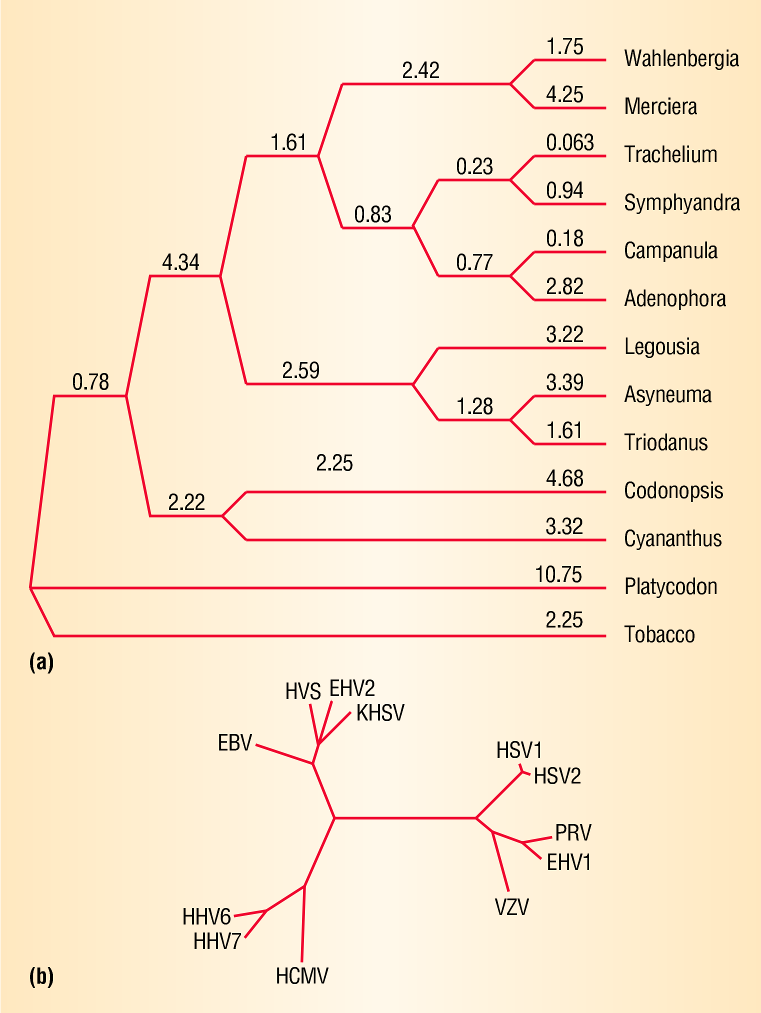

Figure 1. Two phylogenies in different graphical formats: (a) some plants of the Campanulaceae (bluebell) family and (b) herpesviruses known to attack humans. The phylogeny in (a) aligns species for easier reading of the groupings and indicates edge lengths by associated values. The phylogeny in (b) uses proportional edge lengths and visually represents the speciation (branching) events. The Campanulaceae chloroplast gene-order data set consists of 12 taxa from the Campanulaceae family, with tobacco as the outgroup; each taxon has 105 gene segments. (The herpesvirus phylogeny is adapted from J.I. Cohen, "Medical Progress: Epstein-Barr Virus Infection," New England J. Medicine, vol. 343, no. 7, 2000, pp. 481-492.)

substitutions of one nucleotide by another, insertions and deletions of nucleotides or substrings, and so on. Typical sources of biomolecular sequence data are individual genes that are common to all organisms in the groups; the exact sequence defining the gene differs slightly from one organism to another.

Biologists can use gene-order data to answer open questions about the early origins of life.

Gene-order data indicates how the genes are ordered within the given genomes, as well as the strand on which they are located. We can then represent a given genome by an ordering of signed integers, where the sign indicates the strand on which the gene is located. Whole genomes evolve not only through molecular mechanisms that include those described earlier (local changes to DNA), but also through larger scale changes that modify the ordering and strandedness of genes. These mechanisms correspond to transformations of the orderings of signed integers, thus allowing us to model the evolution of whole genomes as a process that operates on signed permutations.

Biologists are interested in using gene-order data in phylogenetics because events that affect gene orderings are relatively rare[4] compared to mechanisms that modify DNA sequences. Consequently, they hope to use gene-order data to answer open questions about the early origins of life.

Gene-order and DNA sequence data are thus complementary: gene-order phylogenetics has the potential to enable discoveries about deep evolution, while DNA sequence phylogenetics allows us to study evolution at smaller time scales.

Simulation studies

The goal of a phylogenetic reconstruction method is the inference of the true tree—the rooted tree whose branching structure is the correct representation of the evolutionary history of the given organisms. Because evolution is historical, it is not possible in most cases to be completely certain about the accuracy of any reconstruction. For this reason, the accuracy of phylogenetic reconstruction methods is studied through simulation studies. In such studies, a model tree is created or chosen and evolution is simulated along the tree, from its root to its leaves.

Model trees are often random tree topologies, but a researcher may also select a topology for specific studies. A DNA sequence (or gene order) is then assigned to the root of the tree (typically that sequence or order is random) and then made to evolve along the tree, using a stochastic process that is based on a specific evolutionary model. In DNA sequence evolution, a typical stochastic process randomly modifies randomly chosen positions within the sequences, substituting one nucleotide with another, all according to some probability model. In gene-order evolution, a typical stochastic process rearranges the gene orders.

This mechanism generates the two children of the root independently and then repeats the process on the newly generated children, until it has assigned DNA sequences (or gene orders) to all leaves. The resulting taxa (DNA sequences or gene orders) at the leaves of the tree then become input for the reconstruction methods under study and the trees produced by these methods are compared to the model tree.

Topological accuracy is measured by counting the number of (non-zero-length) edges in the model tree for which corresponding edges exist in the reconstructed tree (where two edges correspond to each other if they induce the same bipartition on the leaves). The topological error rate is the proportion of (non-zero-length) missing edges relative to the number of (non-zero-length) internal edges in the model tree.

Current methods

The biological community commonly uses three basic phylogenetic reconstruction techniques.[2]

Distance-based methods use a matrix of estimated leaf-to-leaf distances to construct a phylogenetic tree. The most popular of these methods, neighbor-joining (NJ), is an agglomerative clustering technique. It operates by repeatedly joining pairs of leaves (or subtrees) on the basis of a sophisticated numerical optimization, each time updating the distance matrix by replacing the rows and columns corresponding to these two leaves by a single row and column corresponding to the root of the small subtree. It then repeats this process on the new, smaller matrix.

NJ runs in low polynomial time (typically cubic) and is one of the fastest reconstruction methods in common use; moreover, simulation studies on small to moderately large data sets have shown good performance. Yet theoretical research and simulation studies show that NJ and other distance-based methods have poor accuracy when the distance matrix contains large values (indicating that the input contains taxa that appear nearly unrelated). In other words, the evolutionary diameter of the data set negatively affects the topological accuracy of trees inferred using these distance-based methods. This suggests that standard distance-based methods may have poor accuracy on the kinds of data sets needed to infer large portions of the Tree of Life.

Maximum parsimony (MP) methods seek the tree that minimizes the total number of evolutionary events (nucleotide substitutions or gene rearrangements) on the edges of the tree. In the context of DNA sequence evolution, MP seeks a tree whose leaves are labeled by the input set of DNA sequences and whose internal nodes are labeled by additional sequences to minimize the sum of the Hamming distances on the edges. In the context of gene-order phylogeny, MP seeks a tree whose leaves are labeled by the input set of gene orders and whose internal nodes are labeled by additional gene orders to minimize the sum of the edit-distances on the edges.

Maximum likelihood (ML) methods assume that a specific stochastic process was responsible for the evolutionary history and seek the parameter values of that process (such as the number of random events on each edge of the tree). With these parameters, the probability of generating the observed data on a particular tree can be computed; an ML algorithm then returns the tree for which this probability is maximized.

For many stochastic models of biomolecular sequence evolution, both NJ and ML, but not MP, are guaranteed to recover the true tree with high probability, given long enough sequences. While many distance-based methods run in low polynomial time, both MP and ML are known to be computationally hard (MP is provably NP-hard, and ML is conjectured to be so). With MP, scoring a tree built from biomolecular sequences can be done in linear time, but the scoring is an NP-hard problem for gene-order data. ML is far more computationally expensive than even MP, although it appears to be capable of returning better solutions.

Experimental evidence for biomolecular sequence evolution suggests that exact solutions to ML or MP will return trees with good topological accuracy. The main limitation of ML and MP methods is running time: Some data sets of just a few hundred organisms remain unsolved, despite years of analysis.

All three approaches for inferring phylogenies thus have limitations: ML and MP, even in heuristic form, can take too long on large data sets, while NJ and other polynomial-time distance-based methods have poor topological accuracy on data sets with large evolutionary diameter. Unfortunately, the data needed to reconstruct the Tree of Life possesses both these attributes.

IMPROVING RECONSTRUCTION ALGORITHMS FOR BIOMOLECULAR SEQUENCES

Because polynomial-time methods, such as NJ, suffer from poor topological accuracy when the model tree has a large diameter, one possible improvement is to reduce the diameter of the data sets on which these algorithms must run.

Our basic idea is to break the data set into subsets with diameters small enough that NJ will return highly accurate trees, but also with sufficient overlap that, when taken together, the trees on these subsets will uniquely define the tree on the whole data set. A second benefit of such an approach is that it may enable the use of expensive reconstruction methods (such as ML), since they will only be called on smaller data sets (the subsets of limited diameter).

The DCM is a meta-algorithm that works with a base method to improve its performance in terms of accuracy or speed.

Disk-covering methods

We developed a class of methods, disk-covering methods (DCMs),[5,6] that are based on this idea. Some DCMs have theoretical performance guarantees, but we focus here on DCMs that give the best empirical performance.

A DCM works in two phases. In the first phase, it computes a collection of trees one for each maximum subproblem diameter of interest. For a given diameter q, the DCM divides the input data set into overlapping subproblems, so that each subproblem has an evolutionary diameter of at most q. It then reconstructs a tree for each subproblem, using a base method such as NJ. Finally it combines the resulting trees (one for each subset) into a tree t_q for the whole data set. This first phase yields a collection of trees, one for each choice of q.

In the second phase, the DCM selects the best tree from the collection on the basis of some optimization criterion. Our study has examined several criteria for this second phase; some criteria (for example, the SQS, or short quartet support, criterion) have promising theoretical properties; others, such as the MP or ML criterion, may demonstrate an empirical performance advantage.

The DCM is thus a meta-algorithm, one that works with a base method to improve its performance, in terms of accuracy (by limiting the diameter) or speed (by reducing the size of the data sets on which the base methods are called). Because the DCM needs both a base method and a selection criterion, a complete phylogenetic reconstruction method using DCM is denoted by a triple, such as DCM–NJ+MP, which indicates that the base method is NJ and the selection criterion is MP.

We studied methods based on DCM–NJ with two different criteria for the second phase: MP and SQS.[5,6] These two methods (DCM–NJ+MP and DCM–NJ+SQS) have an identical first phase and

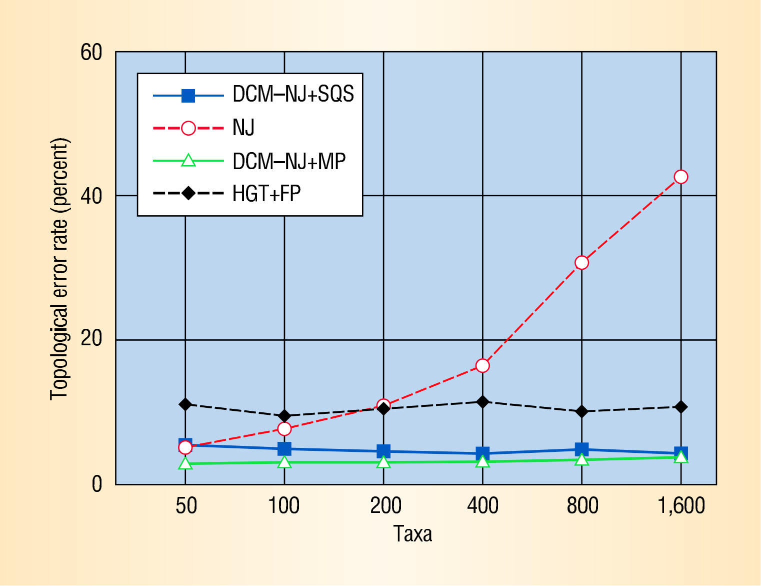

Figure 2. A comparison of two DCM-boosted methods (DCM–NJ+MP and DCM–NJ+SQS), NJ, and HGT+FP on random trees under the K2P+Gamma model, for sequence length of 1,000 and average branch length of 0.05.

thus select a tree from the same collection, but they use different criteria.

Simulation study results

To illustrate the type of improvement seen with DCM boosting, we present selected results from one of our simulation studies.[7] These results address the question, "How will topological errors grow with increasing numbers of organisms if we fix the average edge length in the tree and the total sequence length available?" This question bears directly on the feasibility of inferring the Tree of Life, where both the evolutionary diameter and the number of organisms will be very large, yet the amount of data available on any given species will remain limited.

We examined four methods: NJ, two DCM-boosted versions of NJ (heuristic DCM–NJ+SQS and heuristic DCM–NJ+MP), and the Harmonic Greedy Triplets plus Four Point (HGT+FP) method.[8] We include HGT+FP because it runs in polynomial time and is provably absolute fast-converging[9]—it recovers the true tree with high probability from sequences of polynomial length. This guarantee does not hold for the other three methods, although heuristic DCM–NJ+SQS is based on a provably absolute fast-converging method (which, however, can take exponential time). The inclusion of these methods allowed us to study these two questions, among others:

- Does DCM-boosting improve the topological accuracy of NJ?
- How well does HGT+FP perform, relative to the other methods (does absolute fast convergence confer any performance advantages)?

We generated random tree topologies (from the uniform distribution) of between 50 and 1600 taxa, with random branch lengths selected so that the expected probability of a mutation on each edge for each site is 0.05. For each tree topology, we then generated sequences of length 1,000 under the K2P+Gamma model of evolution[2] with standard settings for its parameters. We generated 100 data sets for each setting of the parameters.

Figure 2 shows the results of just one experiment with high evolutionary rates. In all experiments, we saw a rate-dependent decrease in accuracy as the number of organisms increases for the tree inferred with NJ, but saw no change for HGT+FP or for the two variants of DCM–NJ. The DCM-boosted methods are clearly superior to the others, with DCM–NJ+MP best of all. In all experiments we ran, DCM–NJ+MP was always the best, for all parameter combinations—indicating that using maximum parsimony (MP) for the second phase selection criterion is very helpful.

Our experiments suggest that the relative advantage obtained by using DCM–NJ+MP will increase as the number of organisms increases, so truly large phylogenetic analyses that might not be feasible under NJ may be feasible using methods such as DCM–NJ+MP. (We do not know if this claim holds for HGT+FP or DCM–NJ+SQS, since these methods do not reliably outperform NJ.) Because DCM–NJ+MP, although considerably slower than simple NJ or HGT+FP, runs in polynomial time, it can be used for sizable data sets.

Experimental studies are typical of the development of any new algorithm: Although the algorithm can often be analyzed from a theoretical point of view to predict its running time, predicting its accuracy is not possible except in abstract asymptotic terms. Thus, new phylogenetic algorithms must be assessed through detailed and comprehensive experimental studies.

WHOLE GENOMES: MODELS AND ALGORITHM ENGINEERING

Whole genomes evolve through large-scale events such as inversions, transpositions, and inverted transpositions, which change the order and direction of genes within the genomes, as well as deletions, insertions, and duplications of entire genes, which change the gene content of the genome. Figure 3 illustrates the first three types of events on a circular genome of eight genes, the latter arbitrarily numbered 1 through 8. The same subsequence of three genes (2 3 4) is affected in all three events depicted; when it is transposed, it is placed between genes 6 and 7 in the original order.

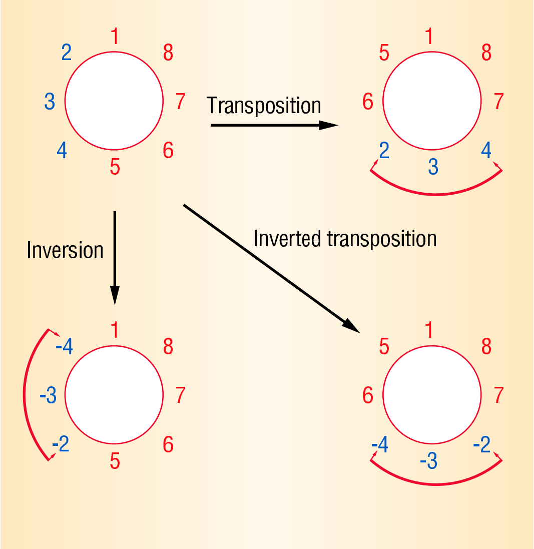

Figure 3: The three types of genome rearrangement. Genes with minus signs are at the opposite DNA strand (with respect to the unrearranged genome)

In our work, we consider only genome rearrangement events (those depicted in the figure). Because these events affect gene order but not gene content (the genome contains the same genes before and after the event), they can be viewed as transforming one signed permutation into another.

We use the Generalized Nadeau-Taylor (GNT) model of evolution, in which events of the same type have equal probability (for example, any two inversions have the same probability of occurrence, regardless of the range of indices affected), but the proportions of the three types of events are dictated by two parameters *a* and *ß*. Here, *a* is the probability that a rearrangement event is a transposition and *ß* that it is an inverted transposition—and thus the probability that an event is an inversion is 1 – (*a* + *ß*). The number of events on each edge *e* obeys a Poisson distribution.

Distance-based methods

As in biomolecular sequence evolution, the first step of a distance-based method is to estimate the number of evolutionary events in the evolutionary history between every pair of taxa (genomes) in the data set.

Our simulation studies have shown that standard distance measures—the inversion distance (the minimum number of inversions needed to transform one genome into another) and the breakpoint distance (the number of positions in one genome in which the ordering changes relative to the other genome)—significantly underestimate the true evolutionary distances.

Our studies also show that NJ used with either measure infers poor trees except for the special case where the data set has a very small evolutionary diameter.[3] One reason for this poor performance is that, given some initial gene order A and some final gene order B, many possible sequences of inversions, transpositions, and inverted transpositions can produce B from A, sequences that may involve significantly different numbers of events. The so-called edit distances are minimum distances—yet natural evolution rarely uses the shortest path from A to B. Thus the true evolutionary distance—the number of events that actually occurred during natural evolution—is typically greater than the edit distance. Reconstruction algorithms based on distances are thus misled when they use edit distances.

We have developed three polynomial-time estimators of true evolutionary distance for genome rearrangements:

- Exact-IEBP (inverting the expected breakpoint distance), which is based on an ML estimate of the breakpoint distance after *k* rearrangements;
- Approx-IEBP, which approximates Exact-IEBP but is faster; and
- EDE (empirically derived estimator), which is based on an empirical estimate of the inversion distance after *k* rearrangements.[3]

We use EDE to describe the idea behind all three estimators. Given a collection of genomes represented as signed permutations, we computed all pairwise inversion distances, and then, for each pair, we estimated the most likely number of inversion events required to produce the computed inversion distance. To obtain this estimate, we collected simulated data and produced a nonlinear regression formula that computes the expected inversion distance given that *k* random inversion events occurred; the EDE distance is the inverse of this regression. Figure 4[3] shows that the estimators of true evolutionary distance considerably improve the accuracy of NJ reconstructions, even when the model assumptions are a poor match for the situation (as in Figure 4b).

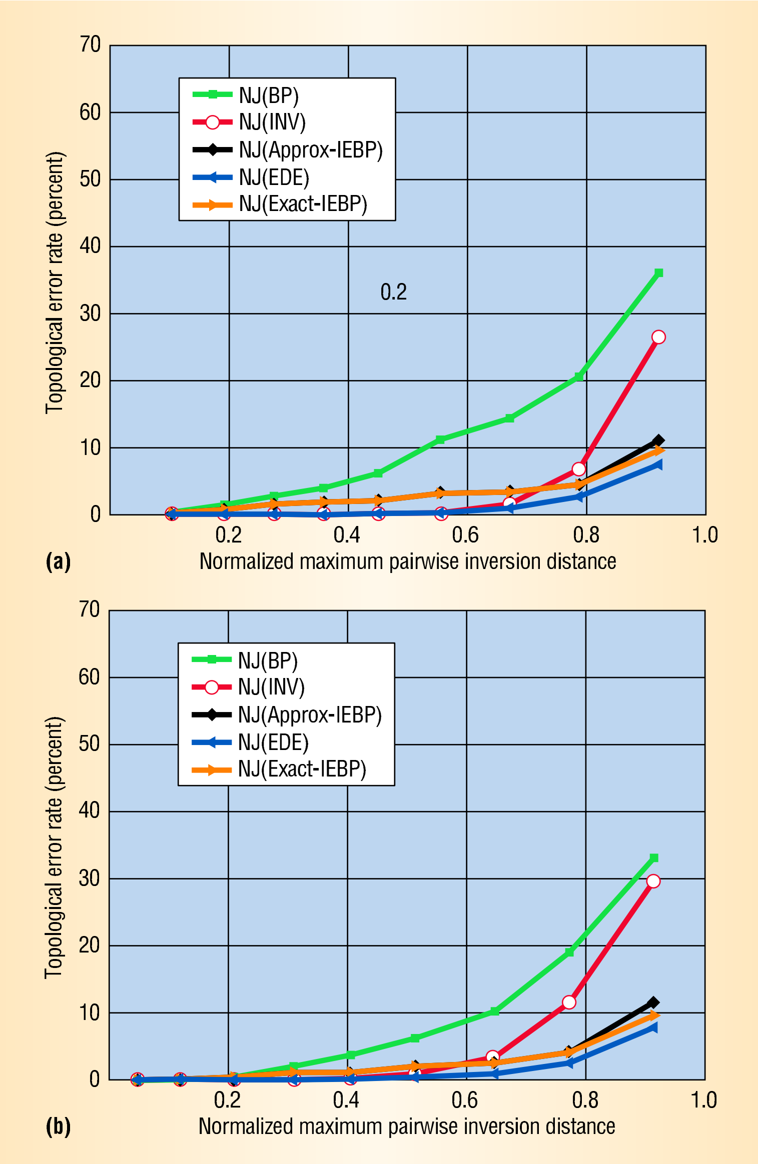

Figure 4: Performance of the neighbor-joining (NJ) algorithm. We used 10, 20, 40, 80, and 160 genomes, all of 120 genes. The evolutionary model is (a) only inversions, and (b) equiprobable inversions, transpositions, and inverted transpositions. The distance estimators used are breakpoint for NJ(BP), inversion for NJ(INV), and EDE (empirically derived estimator) for NJ(EDE).

Maximum parsimony on rearranged genomes

As in biomolecular sequence evolution, we can pose maximum parsimony problems, in which we label internal tree nodes by ancestral gene orders and seek the tree of minimum total length. One such problem, popularized by David Sankoff, is the *breakpoint phylogeny*, which seeks to minimize the total number of breakpoints across all edges.

Breakpoint phylogeny. Reconstructing the most parsimonious tree for this problem involves both identifying the best tree structure and reconstructing ancestral gene orders, the latter required to score each tree. Matthieu Blanchette and David Sankoff developed BPAnalysis[10] to solve this problem when the distance measure is the breakpoint distance. Within a framework that enumerates all trees, this algorithm uses an iterative heuristic to label the internal nodes with signed gene orders. After each internal genome is assigned some initial signed gene order, the tree is traversed repeatedly, and each internal node is assigned a new gene order by calculating the breakpoint median of its three neighbors in the tree until convergence.

Computing the breakpoint median of three genomes is solved by a reduction to the well-studied traveling salesperson problem (TSP). This procedure is computationally very intensive. The outer loop enumerates all $(2n - 5)!!$ leaf-labeled trees on n leaves, while 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. 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 median problem solved exactly, the tree-labeling phase does not ensure optimality unless the tree has only three leaves.

High-performance algorithm engineering. Algorithm engineering refers to the process required to transform a pencil-and-paper algorithm into a robust, efficient, well tested, and easily usable implementation; its main focus is experimentation.[11]

High-performance algorithm engineering targets one particular program attribute: speed. The high-performance aspect does not immediately imply parallelism; in fact, in any highly parallel task, most of the impact of high-performance algorithm engineering tends to come from refining the serial part of the code.

In our GRAPPA (genome rearrangement analysis using parsimony and other phylogenetic algorithms; http://www.cs.unm.edu/~moret/GRAPPA) package, we achieved a speedup of nearly seven orders of magnitude in the serial execution of the code, while practical sizes of parallel machines limit us to a speedup of two to three orders of magnitude from parallelism. Although the asymptotic behavior of the program is identical to that of the original implementation, the enormous speedup makes a huge difference in practice.

On the Campanulaceae data set in Figure 1a, the original code would have taken several centuries to complete. Our latest version runs in less than one hour on a desktop and in a few minutes on a cluster. Thus, high-performance algorithm engineering complements algorithmic developments: The latter enables scaling to truly large problems, while the former ensures that the theoretically pleasing asymptotic behavior is realized at a human scale of time.

In producing the GRAPPA software suite, we began by reimplementing the approach pioneered by Sankoff and Blanchette in BPAnalysis.[6] The original BPAnalysis is written in C++ and has a significant memory footprint (more than 60 Mbytes when running on the Campanulaceae data set) and poor locality (a working set size of about 12 Mbytes). Our C implementation has a tiny memory footprint (1.8 Mbytes on the Campanulaceae data set) and good locality, which enables it to run almost completely in cache (the working set size is 600 Kbytes).

We achieved good cache locality by returning to a Fortran programming style. In this style, storage is static, all our storage is in arrays preallocated in the main routine and retained and reused throughout the computation, records (structures/classes) are avoided in favor of separate arrays, simple iterative loops that traverse an array linearly are preferred over pointer dereferencing, code is replicated to process each array separately, and so on. Indeed, combining efficiency and object-oriented programming is quite difficult because most object-oriented approaches interpose opaque layers of code between the programmer and the computer and because the compiler may not always optimize data layout and access.

Of course, unless the original implementation is poor (not the case with BPAnalysis), profiling and cache-aware programming will rarely provide more than two orders of magnitude in speedup. Low-level improvements in the algorithmic details often yield further gains. In our phylogenetic software, we made three such improvements.

Our TSP solver 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 more efficient, resulting in a speedup by a factor of 5 to 10. The principal change takes advantage of the bounded nature of certain quantities—a general principle of fundamental importance in algorithm engineering. For example, our TSP solver need consider only two possible edge costs, rather than arbitrary integers; taking advantage of this restriction led to a speedup by one to two orders of magnitude.

New software packages must take advantage of both shared-memory and message-passing modes of computation.

The basic algorithm scores every single tree, which is clearly very wasteful. We used a simple lower bound, computable in linear time from the available data, to enable us to eliminate a tree without scoring it. On the Campanulaceae data set, this bounding eliminates 99.9 percent of the trees without scoring them, for a speedup by two orders of magnitude.

Instead of exploring tree space in the order dictated by tree generation (which must be particularly efficient when dealing with trillions of trees), we implemented a two-phase exploration. In the first phase, all trees are generated, their lower bounds calculated, and the results stored in a bucketed hash table, with each bucket storing all trees with the same lower bound. In the second phase, trees are examined by processing each bucket in turn, starting with those associated with the smallest bounds. Because the lower bound of a tree is strongly correlated with its actual score, this approach typically produces the optimal tree much sooner than the search in generation order. On the Campanulaceae data set, our two-phase method examines less than 0.004 percent of the trees, boosting the speedup by another two orders of magnitude.

All these improvements spring from a careful examination of exactly what information is readily available or easily computable at each stage and from a deliberate effort to use all such information. However, the speedup with these improvements is variable—from close to none on particularly hard instances to more than five orders of magnitude on less demanding instances—whereas the coding improvements discussed earlier yield consistent speedups on *all* instances.

Scalability. Scientists today have access to a large variety of computing platforms, from laptops to supercomputers at one of the National Science Foundation's centers. However, few software packages are written to scale gracefully from a laptop to a massively parallel machine, thus few scientists can take advantage of the variety of platforms available to them. New software packages must take advantage of both shared-memory and message-passing modes of computation; the first is well suited to complex combinatorial optimization tasks, and the second is best applied to decomposable problems.

Phylogeny reconstruction involves both kinds of problems. For example, many trees must be scored more or less independently, and processors can carry out these tasks in parallel without taxing the message-passing system. Scoring a single tree for certain criteria—such as parsimony on gene-order data, as in the GRAPPA package—requires the solution of NP-hard discrete optimization problems.

In GRAPPA, the extremely large number of trees is divided evenly among the processors; each processor then begins bounding and scoring its trees and communicates any newly discovered improved upper bounds to the others.

The communication overhead is nearly nonexistent, so we obtained a perfect linear speedup with the number of processors. Because the number of trees is so large compared to the number of processors, there was no need to exploit parallelism for other tasks (such as solving the TSP subproblems). In contrast, our current research version of the software uses branch-and-bound to search tree space and thus needs to focus more on shared-memory parallelism and less on distributed computing approaches.

Our current research version of the software focuses more on shared-memory parallelism and less on distributed computing approaches.

The new generation of phylogenetic software needs to be founded on more sophisticated models of evolution, to incorporate better and faster optimization algorithms, to be implemented according to the best principles of high-performance algorithm engineering, and to provide more powerful modes of user interaction. We have illustrated the potential benefits of algorithm design and high-performance algorithm engineering with three examples drawn from our own research. Combining the large speedups obtained through algorithm engineering with the asymptotic gains derived from new algorithms will enable us to move closer to solving the grand challenge of evolutionary biology, the reconstruction of the Tree of Life.

Because the two main optimization criteria give rise to NP-hard problems, exact algorithms for these criteria must exhibit exponential growth in their running time for at least some instances (unless we have P = NP). Any such behavior forms an absolute barrier to scaling: Even enormous speedups pale in comparison with such growth. (The billionfold speedup on the large cluster has allowed us to increase the instance size by only seven organisms—from 10 to 17.) Thus, algorithmic approaches that reduce the running time to a polynomial are needed; even if these approaches are not exact—or even very good—for all possible instances, they can take advantage of the natural structure present in typical real instances.

Our DCM strategy provides one such algorithmic approach, but it is still in its infancy; tackling these hard optimization problems will require other novel approaches as well. ■

Acknowledgments

Our work on phylogeny reconstruction is supported by the National Science Foundation under grants ACI 00-81404 (Moret), DEB 01-20709 (Moret and Warnow), EIA 01-13095 (Moret), EIA 01-13654 (Warnow), EIA 01-21377 (Moret), EIA 01-21680 (Warnow), and EIA 01-21682 (Warnow), and by the David and Lucile Packard Foundation (Warnow).

References

1. D. Swofford, *PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods) 4.0 Beta for Unix or OpenVMS*, Sinauer Assoc., Sunderland, Mass., 2002.
2. D. Swofford et al., "Phylogenetic Inference," in *Molecular Systematics*, D. Hillis, C. Moritz, and B. Mable, eds., 2nd ed., chapt. 11. Sinauer Assoc., Sunderland, Mass., 1996.
3. B.M.E. Moret et al., "Steps Toward Accurate Reconstruction of Phylogenies from Gene-Order Data," *J. Computation & System Science*, accepted for publication, 2002.
4. A. Rokas and P.W.H. Holland, "Rare Genomic Changes as a Tool for Phylogenetics," *Trends in Ecology and Evolution*, vol. 15, 2000, pp. 454-459.
5. D. Huson, S. Nettles, and T. Warnow, "Disk-Covering: A Fast Converging Method for Phylogenetic Tree Reconstruction. *J. Computational Biology*," vol. 6, no. 3, 1999, pp. 369-386.
6. L. Nakhleh et al., "Designing Fast Converging Phylogenetic Methods," *Proc. 9th Int'l Conf. Intelligent Systems for Molecular Biology*, Bioinformatics, vol. 17, Oxford Univ. Press, Oxford, UK, 2001, pp. S190-S198.
7. L. Nakhleh et al., "The Accuracy of Fast Phylogenetic Methods for Large Data Sets," *Proc. 7th Pacific Symp. Biocomputing*, World Scientific Pub., Singapore, 2002, pp. 211-222.
8. M. Csuros, "Fast Recovery of Evolutionary Trees with Thousands of Nodes," *Proc. 5th Ann. Int'l Conf. Computational Molecular Biology*, ACM Press, New York, 2001, pp. 104-113.
9. T. Warnow, B.M.E. Moret, and K. St. John, "Absolute Phylogeny: True Trees from Short Sequences," *Proc. 12th Ann. ACM/SIAM Symp. Discrete Algorithms*, SIAM Press, Philadelphia, 2001, pp. 186-195.

10. D. Sankoff and M. Blanchette, "Multiple Genome Rearrangement and Breakpoint Phylogeny," *J. Computational Biology*, vol. 5, 1998, pp. 555-570.
11. B.M.E. Moret and H.D. Shapiro, "Algorithms and Experiments: The New (and Old) Methodology," *J. Universal Computer Science*, vol. 7, no. 5, 2001, pp. 434-446.

***Bernard M.E. Moret** is a professor of computer science and electrical and computer engineering at the University of New Mexico. His research interests include algorithm design, algorithm engineering, and computational biology. He received a PhD in electrical engineering from the University of Tennessee and is a member of the ACM, SIAM, and International Society for Computational Biology (ISCB). He is also editor-in-chief of the* ACM Journal of Experimental Algorithmics *and chairs the steering committees of the International Workshop on Algorithms in Bioinformatics and the Workshop on Algorithm Engineering and Experiments. Contact him at moret@cs.unm.edu.*

***Li-San Wang** is a PhD candidate in computer science at the University of Texas at Austin, where his research interests include algorithmic theory and computational biology. He received an MSEE from National Taiwan University and an MS in computer science from the University of Texas at Austin. Wang is a member of the ISCB. Contact him at lisan@cs.utexas.edu.*

***Tandy Warnow** is an associate professor of computer sciences at the University of Texas at Austin, where she is also the codirector of the Center for Computational Biology and Bioinformatics. Her research interests are computational phylogenetics, historical linguistics, graph theory, and graph algorithms. Warnow received a PhD in mathematics from the University of California, Berkeley. She is a Packard fellow and a member of the ISCB board. Contact her at tandy@cs.utexas.edu.*