Pacific Symposium on Biocomputing 5:239-250 (2000)

InVEST: Interactive and Visual Edge Selection Tool for Constructing Evolutionary Trees

Paul Kearney, Adrian Secord and Haoyong Zhang

Department of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1

InVEST is an interactive and visual tool for constructing evolutionary trees from an ordered list of edges. In this paper it is shown that many methods for constructing evolutionary trees reduce to the edge selection problem. Furthermore, through a simulation study, it is shown that noninteractive methods for edge selection often perform poorly and can conceal alternative solutions. InVEST allows the user to interact with and explore an ordered list of edges facilitating the incorporation of user domain knowledge into the evolutionary tree construction process.

1 Introduction

As sequence databases grow in size and diversity, evolutionary studies based on large sequence data sets are becoming commonplace. For example, the Ribosomal Database Project[a] at Michigan State University now contains evolutionary trees describing the evolutionary history of 6200 prokaryote sequences, 2000 eukaryote sequences and 1500 mitochodria sequences [1]. As the scope and magnitude of evolutionary studies increases so does the need to develop more effective computational tools to assist in the evolutionary analysis of sequences.

Current computational tools for constructing evolutionary trees, such as PAUP[2] and PHYLIP[3], are noninteractive. The typical workflow for these tools consists of the following three stages (see Figure 1): First, the sequences are aligned and model parameters are specified. Second, an inference method is applied that constructs an evolutionary tree. Third, the result is either accepted by the user or the process is repeated with modifications to the alignment and model parameters. In this workflow, the user's domain knowledge has the greatest effect in the first stage as it specifies the input for and the scope of the construction stage. The user's domain knowledge is also incorporated in the third stage when the result is evaluated. However, this use of domain knowledge

[a]The RDP's URL is `http://www.cme.msu.edu/RDP/index.html`

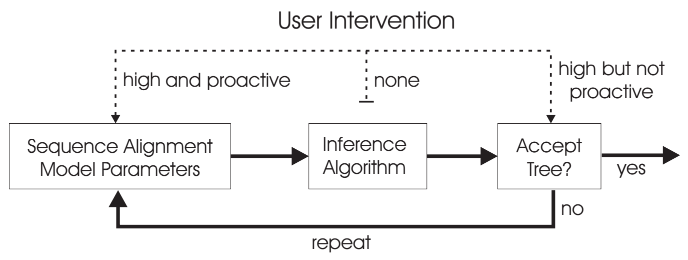

Figure 1: Workflow of Evolutionary Tree Construction

is not proactive in the sense that it is used to reject a hypothesis but not to formulate a hypothesis. The noninvolvement of the user in the second stage is undesirable for two reasons:

- During the construction of an evolutionary tree, inference methods generate information and explore alternatives that may be of interest to the user. In particular, an awareness of alternatives and critical decisions that occur in the construction process could lend insight to the user.
- Most inference methods including maximum parsimony [4] and maximum likelihood [5] are computationally intensive. User domain knowledge could be used both to tailor the inference method to the analysis at hand and also to guide the flow of computation in efficient directions.

This paper explores the utility of interactive and visual tools for constructing evolutionary trees. More specifically, this paper presents an interactive tool, InVEST, that

- allows the user to visualize the details of the construction process;
- allows the user to incorporate domain knowledge at critical points in the construction process and to visualize the consequences of this involvement and
- allows the user to guide the construction process to an efficient solution.

There are many evolutionary tree construction paradigms and methodologies[6]. Our research focusses on the problem of constructing the topology of an evolutionary tree which we define formally in Section 2. We demonstrate through a simulation study that constructing evolutionary tree topology is difficult to do in a noninteractive manner. In Section 3 we overview the design of InVEST. In particular, it is shown how logical steps in the construction of evolutionary tree topology can be visualized and how user domain knowledge can be incorporated in an intuitive manner.

2 Tree Topology and Edge Selection

An evolutionary tree T for a set S of sequences is a rooted and edge weighted tree where the leaves of T are labeled bijectively by S. The topology of T (that is, T without edge weights) describes the speciation events resulting in the evolution of sequences in S from the root. The edge weights of T are proportional to the amount of evolution (sequence substitutions, insertions and deletions) between speciation events.

It is well–known that the topology of an evolutionary tree can be specified by its set of edge-induced bipartitions [7]. An evolutionary tree T labeled by S contains the bipartition (X, Y) of S if there is an edge e in T such that $T - \{e\}$ consists of two trees where one is labeled by X and the other by Y. This is denoted $e = (X, Y)$ and we use the terms 'edge' and 'bipartition' interchangeably. Two bipartitions (A, B) and (C, D) are *compatible* if there is a tree that contains both bipartitions. This is equivalent to one of A or B being a subset of either C or D. A set of bipartitions is compatible if there is a tree that contains these bipartitions.

A bipartition (X, Y) of T is called *trivial* if $|X| = 1$ or $|Y| = 1$. All evolutionary trees labeled by S share the same set of trivial bipartitions. Consequently, when determining the topology of an evolutionary tree *nontrivial* bipartitions are more informative. If $|S| = n$ then T has n trivial bipartitions and $n - 3$ nontrivial bipartitions. Consequently, a set of compatible and nontrivial bipartitions of sequence set S can never be larger than $n - 3$. To illustrate, the evolutionary tree in Figure 2 contains nontrivial bipartitions $(\{1, 2\}, \{3, 4, 5, 6\})$, $(\{1, 2, 3\}, \{4, 5, 6\})$ and $(\{1, 2, 3, 4\}, \{5, 6\})$.

To estimate the topology of T given the set S of sequences, we must determine those bipartitions of S best supported by the sequences S. Let d be a *support function* that measures how well the sequence data S supports a bipartition (X, Y). Let L be the *bipartition list* of all bipartitions of S ordered by d. L is enormously large, and so, cannot be explicitly computed. However, depending

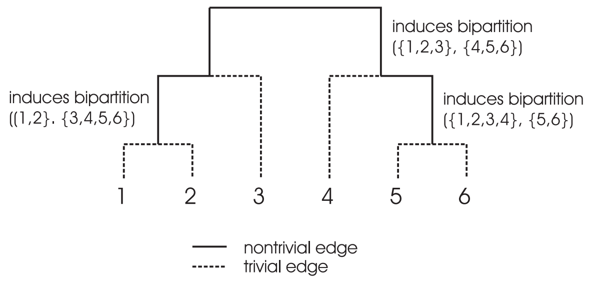

Figure 2: Evolutionary Tree Topology and Bipartitions

on the accuracy of d, it is expected that the edges of T are clustered near the beginning of L, and so, the problem of determining the topology of T is formulated as follows:

EDGE SELECTION: Given a support function d, select a set of $n-3$ compatible and nontrivial bipartitions from a prefix of the bipartition list ordered by d.

Several diverse methods for constructing evolutionary tree topology reduce to the EDGE SELECTION problem including character compatibility[8,9], bootstrapping[10] and ordinal methods[11,12,13]. Here we discuss EDGE SELECTION in the context of the *quartet method* described below.

2.1 Quartet Methods

The quartet method[14,15,13,16,17] constructs the topology of an evolutionary tree T by first estimating the topology for quartets of sequences and then recombining these pieces of T into an estimate of T's topology. More formally, given a quartet of sequences $\{a, b, c, d\}$ and an evolutionary tree T, the *quartet topology* induced in T by $\{a, b, c, d\}$ is the path structure connecting a, b, c and d in T. Given a quartet $\{a, b, c, d\}$, if the path in T connecting labels a and b is disjoint from the path in T connecting c and d, the quartet is said to be *resolved* and is denoted $ab|cd$. Otherwise, the quartet is said to be *unresolved* and is denoted $(abcd)$. The four possible quartet topologies induced by a quartet are depicted in Figure 3.

There are many methods for estimating quartet topology including maximum likelihood[5], maximum parsimony[4], neighbor joining[18] and the ordinal quartet

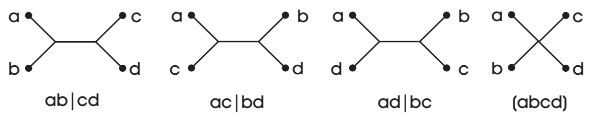

Figure 3: The four quartet topologies for quartet $\{a, b, c, d\}$.

method[13] (please see the last reference for a comparison of these methods). Although many of these methods cannot be feasibly applied to the entire dataset S to infer the topology of T directly, they can be applied feasibly to infer evolutionary tree topologies of size four. Let Q be the set of these $\binom{n}{4}$ inferred quartet topologies. The second step of the quartet method is to recombine the quartet topologies into an evolutionary tree topology T' that is an estimate of T. There are many *noninteractive* methods for recombining quartet topologies including the Q^* method[15], quartet puzzling[17], global edge cleaning[16] and methods based on semi–definite programming[14] and smooth polynomial integer programming[19].

In order to develop an interactive quartet method we define a support measure d and a method for generating prefixes of the bipartition list ordered by d. Define Q_T to be the set of quartet topologies induced in T by sequence quartets from S whereas Q is the set of quartet topologies estimated from S. Under the assumption that Q approximates Q_T, Q can be used to assess the likelihood that a bipartition (X, Y) is an edge of T as follows. Let $Q_{(X,Y)}$ be the set of quartet topologies of the form $xx'|yy'$ where $x, x' \in X$ and $y, y' \in Y$. Define the support function

$$d(Q, (X, Y)) = \frac{4|Q_{(X,Y)} - Q|}{|X|(|X| - 1)|Y|(|Y| - 1)}.$$

$d(Q, (X, Y))$ is the percentage of quartet topologies in $Q_{(X,Y)}$ that differ from Q. Note that the number of quartet topologies in $Q_{(X,Y)}$ is $|X|(|X|-1)|Y|(|Y|-1)/4$. When (X, Y) is trivial ($|X| = 1$ or $|Y| = 1$), $d(Q, (X, Y))$ is defined to be 0.

Prefixes of the bipartition list ordered by d are defined, for parameter m, as follows:

$$Best(Q, m) = \{(X, Y) \mid d(Q, (X, Y)) \leq \frac{2m}{|X||Y|}\}.$$

Larger values of m produce larger prefixes of the bipartition list.

When $m = 0$ the prefix $Best(Q, m)$ corresponds to those bipartitions that have

0 quartet differences with Q. This set of edges is often called the *Buneman tree* [20] and an efficient algorithm, called the Q^* method, exists for recovering the Buneman tree from Q[15]. $Best(Q, 0)$ is a conservative estimate of T's topology that can be improved upon:

Lemma 1 *Let Q be a set of quartet topologies. Then $Best(Q, 1)$ is a set of compatible bipartitions.*

As m is increased $Best(Q, m)$ will eventually contain all edges of T. This value of m is typically a small constant but is almost always greater than 1 (confirmed by the simulation study below). It is known that for some instances of Q and T the bound $2/(|X||Y|)$ implicit in Lemma 1 is tight [16], and so, although $Best(Q, m)$ contains all edges of T when $m > 1$, it may also contain bipartitions that are not edges of T. This poses two algorithmic problems:

- Computing $Best(Q, m)$ efficiently. To solve this problem we developed the first efficient algorithm, called *hypercleaning*, for computing $Best(Q, m)$. This algorithm is utilized in the simulation study presented below. Hypercleaning improves upon previous algorithms for this problem[19,16,15,14,17].
- Selecting from $Best(Q, m)$ the edges of T. We present a noninteractive greedy algorithm, called *Compatible*, for selecting edges of T from $Best(Q, m)$ and assess the performance of *Compatible* in the simulation study below. In Section 3 we present InVEST which is an interactive tool for the selection of edges from $Best(Q, m)$.

2.2 The Inherent Difficulty of EDGE SELECTION*: Theory and Practice*

Let P be a prefix of the bipartition list L ordered by support function d. Compatibility relations between bipartitions in P can be represented by a weighted graph G_P where the vertices of G_P are the bipartitions in P and two bipartitions are adjacent in G_P if and only if they are compatible. Each vertex is assigned the weight $d(S, (X, Y))$ where d is the support function. G_P is called the *compatibility graph* for P and the complement of G_P is called the *incompatibility graph* of P.

EDGE SELECTION is equivalent[b] to solving the well–studied MAXIMUM WEIGHTED CLIQUE problem with input G_P. The goal of MAXIMUM WEIGHTED CLIQUE is to find a complete subgraph of G_P with maximum weight. The problem

[b]This is easy to prove but is omitted here for brevity.

is known to be NP–complete [21] and hard to approximate [22]. Hence, EDGE SELECTION is a computationally challenging problem. As a result, EDGE SELECTION is typically solved using a greedy algorithm which we call *Compatible*: Select the bipartition (X, Y) with the maximum support. Delete (X, Y) and all bipartitions incompatible with (X, Y). Repeat until P is empty.

We conducted a simulation study to evaluate the effectiveness of noninteractive methods for EDGE SELECTION using *Compatible* as an example of such a method. The details of the simulation study follow. DNA sequences were artificially evolved using the Kimura 2 parameter model of evolution [6] with transition/transversion ratio of $5 : 1$ on an evolutionary tree T sampled from the Ribosomal Database Project prokaryotic tree [1]. T represents the evolutionary history of the Methanosarcina and relatives subgroup and contains 21 sequences (see Figure 4). Site–to–site rate variance was simulated using the gamma function with parameter 1. $Best(Q, m)$, for $0 \leq m \leq 5$, was obtained by applying the hypercleaning algorithm to a set Q of quartet topologies obtained from the artificial sequences using the ordinal quartet method[13]. *Compatible* was applied to $Best(Q, m)$ to obtain a set $Compatible(Q, m)$ of compatible bipartitions. To better explore the parameter space, sequence length and edge length were varied. More specifically, the sequence length was varied over values 100, 200 and 300 and T was scaled by factors 0.5, 1.0 and 2.0 so that trees with recently diverged sequences and trees with distantly diverged sequences were examined.

The results appear in Table 1 for scaling factor 2.0. Each cell contains four values. The first value is the percentage of nontrivial edges of T in $Best(Q, m)$. The second value is the number of bipartitions in $Best(Q, m)$ as a percentage of the number of nontrivial edges in T (which is 18). The third value is the percentage of nontrivial edges of T in $Compatible(Q, m)$. Note that this value is bounded by the first value. The fourth value is the number of nontrivial bipartitions in $Compatible(Q, m)$ as a percentage of the number of nontrivial edges in T. All values are computed as an average of 10 trials[c].

As sequence length and m increase most edges of T are included in $Best(Q, m)$. The particular tree used in this simulation is challenging due to several very short nontrivial edges. Focussing on sequence length 300, $Best(Q, 5)$ contains 17 of T's 18 (i.e. 92%) nontrivial edges and a total of 64 bipartitions, on average. From these 64 bipartitions, *Compatible* selects, on average, a maximal set of 18 compatible bipartitions 15 of which are correct (i.e. from T) and 3 of which are erroneous. Similar results hold for the other sequence lengths. Even

[c]These simulations took several days of CPU time.

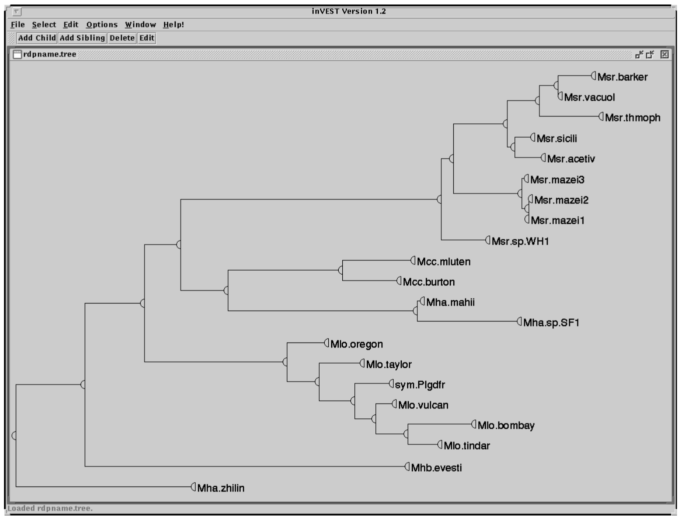

Figure 4: The Methanosarcina and relatives model tree used in the simulation study.

when all edges of T are available in $Best(Q, m)$, the noninteractive *Compatible* does not succeed in selecting these edges.

3 The Visual and Logical Design of InVEST

Designing a visual and interactive tool for the EDGE SELECTION problem poses several logistical and conceptual challenges. Specifically, the tool should present relevant information intuitively and allow the user to implement decisions and view their consequences. In response to these challenges we present InVEST, a tool that allows the user to interactively solve the EDGE SELECTION problem.

InVEST enables the user to interact with several bipartition lists simultaneously. For each bipartition list there is a `bipartition window` and a `tree window` (see Figure 5). The primary function of InVEST is to facilitate visual selection of bipartitions for inclusion in an evolutionary tree. The bipartition list is displayed in the `bipartition window` from which the user manually selects bipartitions. The user can make selections based upon domain knowledge or make selections to explore an hypothesis. An evolutionary tree containing

	Sequence Length					
m	100		200		300	
0	17 (17)	17 (17)	14 (14)	14 (14)	19 (19)	19 (19)
1	27 (33)	27 (33)	41 (45)	41 (45)	51 (53)	51 (53)
2	40 (53)	39 (50)	58 (72)	57 (69)	69 (84)	68 (76)
3	54 (98)	51 (67)	70 (131)	65 (83)	83 (153)	77 (92)
4	60 (146)	54 (78)	72 (201)	65 (88)	87 (236)	81 (97)
5	65 (221)	56 (86)	81 (306)	68 (92)	92 (353)	82 (99)

Table 1: Performance of the noninteractive greedy algorithm for EDGE SELECTION

the current selection of bipartitions is displayed in the `tree window`. The score of the current selection is computed using the score for each bipartition selected. Selections can be reversed. Alternatives can be explored by opening several bipartition/tree windows. The user has the option, at any point, to employ an automated method to complete the current selection of bipartitions. Both the greedy algorithm *Compatible* and an exact method are provided for this purpose.

InVEST displays bipartition compatibility/incompatibility information as arcs joining pairs of bipartitions in the `bipartition window`. Compatibility relations are highly informative and can reveal errors in the ordering of bipartitions by support. For example, in Figure 5 incompatibility arcs are displayed in the top `bipartition window`. Notice that the degree of incompatibility does not increase uniformly from left to right as one might expect. In particular, the 14th bipartition (support value 1.69) is compatible with all bipartitions in the bipartition list whereas most higher ranking bipartitions are not. This indicates that the 14th bipartition should be included in the evolutionary tree; this decision would not be clear using the support information alone. InVEST can also hide those bipartitions incompatible with the selected bipartitions. This has the effect of reducing the density of compatibility information presented to the user as bipartitions are selected.

InVEST incorporates the concept of *local alternatives.* Let C be a set of compatible bipartitions that the user has selected from the bipartition list. Let T_C denote the evolutionary tree containing the bipartitions in C. A bipartition (X, Y) expands a vertex v of T_C if $T_{C \cup \{(X,Y)\}}$ is T_C except with v expanded to the edge (X, Y). Each bipartition can expand at most one vertex of T_C. For each vertex v of T_C let the local alternatives for v be the sublist of bipartitions that expand v.

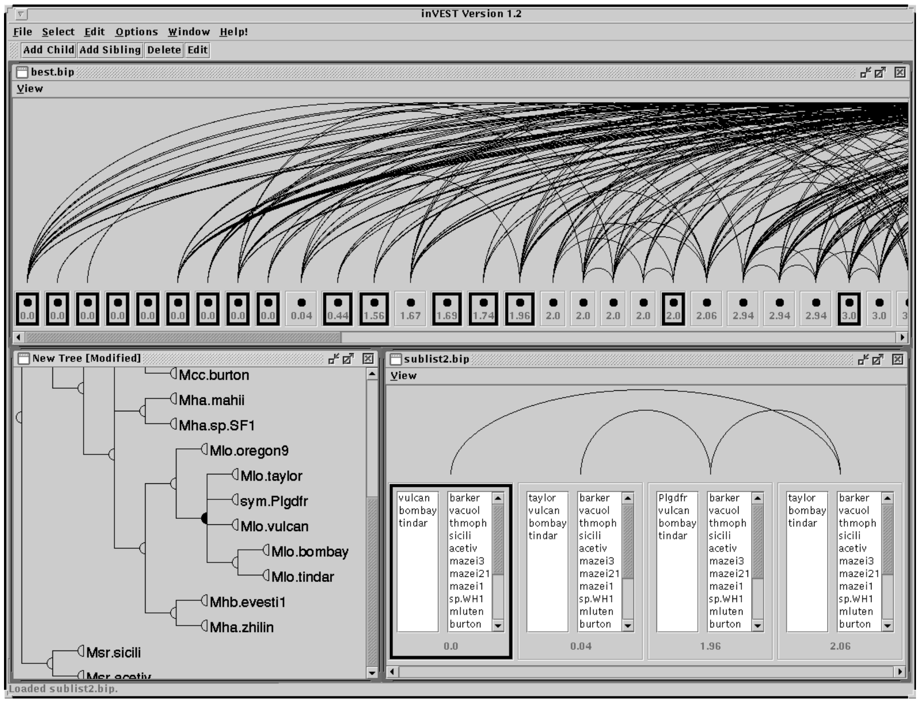
inVEST Version 1.2
File Select Edit Options Window Help!
Add Child Add Sibling Delete Edit
best.bip
View
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.04 0.44 1.56 1.67 1.69 1.74 1.96 2.0 2.0 2.0 2.0 2.0 2.06 2.94 2.94 2.94 3.0 3.0
New Tree [Modified]
Mcc.burton
Mha.mahii
Mha.sp.SF1
Mlo.oregon9
Mlo.taylor
sym.Plgdfr
Mlo.vulcan
Mlo.bombay
Mlo.tindar
Mhb.evesti1
Mha.zhilin
Msr.sicili
Msr.acetiv
sublist2.bip
View
vulcan bombay tindar
barker vacuol thmoph sicili acetiv mazei3 mazei21 mazei1 sp.WH1 mluten burton
0.0
taylor vulcan bombay tindar
barker vacuol thmoph sicili acetiv mazei3 mazei21 mazei1 sp.WH1 mluten burton
0.04
Plgdfr vulcan bombay tindar
barker vacuol thmoph sicili acetiv mazei3 mazei21 mazei1 sp.WH1 mluten burton
1.96
taylor bombay tindar
barker vacuol thmoph sicili acetiv mazei3 mazei21 mazei1 sp.WH1 mluten burton
2.06
Loaded sublist2.bip.

Figure 5: Selecting edges of the Methanosarcina and relatives tree using InVEST.

Every bipartition (X, Y) in the bipartition list is either (1) an edge of T_C; (2) incompatible with an edge of T_C; or (3) expands a vertex v of T_C. In the first case, (X, Y) can be deleted from the bipartition list since it has already been selected for inclusion in T_C. In the second case, (X, Y) can also be deleted from the bipartition list since it cannot be selected for inclusion in the tree. In the third case, (X, Y) is assigned to the sublist of local alternatives for vertex v. The identification of local alternatives has two advantages. First, it allows the user to view only those bipartitions relevant to the expansion of a vertex and to compare these alternatives. Second, the selection of a local alternative at one vertex is independent of the selection of a local alternative at another vertex. This translates into a computational advantage since these vertices can be expanded independently of each other. In Figure 5, the lower right `bipartition window` contains the sublist of bipartitions that expand the vertex selected in the `tree window` on the left.

The main advantage of InVEST over noninteractive methods for edge selection is that it allows the user to explore and interact with the data, and so, the user becomes intimately aware of strengths and weaknesses in the data. Noninteractive algorithms returns an evolutionary tree topology without indication of alternative topologies that might also be significant. This is especially misleading when alternative topologies have similar scores. InVEST presents to the user local alternatives that can be compared and explored.

InVEST was developed using Java's Swing classes, and so, can be used on several platforms.

Acknowledgments

The authors gratefully acknowledge the support of an NSERC undergraduate award, NSERC Research Grant 160321 and CITO.

1. Bonnie L. Maidak, James R. Cole, Charles T. Parker, George M. Garrity Jr, Niels Larsen, Bing Li, Timothy G. Lilburn, Michael J. McCaughey, Gary J. Olsen, Ross Overbeek, Sakti Pramanik, Thomas M. Schmidt, James M. Tiedje, and Carl R. Woese. A new version of the RDP (Ribosomal Database Project). *Nucleic Acids Research*, 27:171–173, 1999.
2. David L. Swofford. *PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods), version 4.0.* Sinauer Associates, Sunderland, Massachusetts, 1996.
3. J. Felsenstein. Phylip (phylogeny inference package), version 3.5c. 1995.
4. W. M. Fitch. Toward defining the course of evolution: Minimal change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.
5. J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
6. D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M Hillis. Phylogenetic inference. In D. M. Hillis, C. Moritz, and B. K. Mable, editors, *Molecular Systematics, 2nd Edition*, pages 407–514. Sinauer Associates, Sunderland, Massachusetts, 1996.
7. J-P. Barthélemy and A. Guénoche. *Trees and Proximity Representations.* Wiley, New York, 1991.
8. C. A. Meacham and G. F. Estabrook. Compatibility methods in systematics. *Ann. Rev. Ecol. Syst.*, 16:431–446, 1985.
9. R. Agarwala and D. Fernández-Baca. A polynomial–time algorithm for the perfect phylogeny problem when the number of character states is

fixed. *SIAM Journal on Computing*, 23(6):1216–1224, 1994.

10. J. Felsenstein and H. Kishino. Is there something wrong with with the bootstrap on phylogenies? *Systematic Biology*, 42:193–200, 1993.
11. S. Kannan and T. Warnow. Tree reconstruction from partial orders. *SIAM Journal on Computing*, 24(3):511–519, 1995.
12. P. E. Kearney. A six–point condition for ordinal matrices. *Journal of Computational Biology*, 4(2):143–156, 1997.
13. P. E. Kearney. The ordinal quartet method. *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 125–134, 1998.
14. A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg. From four–taxon trees to phylogenies: The case of mammalian evolution. *Proceedings of the Second Annual International Conference on Computational Molecular Biology*, pages 9–19, 1998.
15. V. Berry and O. Gascuel. Inferring evolutionary trees with strong combinatorial evidence. *Proceedings of the Third Annual International Computing and Combinatorics Conference*, pages 111–123, 1997.
16. V. Berry, T. Jiang, P. E. Kearney, M. Li, and T. Wareham. Quartet cleaning: Improved algorithms and simulations. to appear, 1999.
17. K. Strimmer and A. von Haeseler. Quartet puzzling: A quartet maximum-likelihood method for reconstructing tree topologies. *Molecular Biology and Evolution*, 13(7):964–969, 1996.
18. N. Saitou and M. Nei. The neighbor–joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.
19. T. Jiang, P. E. Kearney, and M. Li. Orchestrating quartets: Approximation and data correction. *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 416–425, 1998.
20. P. Buneman. The recovery of trees from measures of dissimilarity. In F.R. Hodson, D.G. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.
21. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, 1979.
22. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Proceedings of the Twentieth IEEE Symposium on the Foundations of Computer Science*, pages 14–23, 1992.