BIOINFORMATICS

Vol. 16 no. 7 2000
Pages 619–627

Using traveling salesman problem algorithms for evolutionary tree construction

Chantal Korostensky* and Gaston H. Gonnet

Institute for Scientific Computing, 8092 ETH Zurich, Switzerland

Received on September 13, 1999; revised on January 21, 2000; accepted on January 23, 2000

Abstract

Motivation: *The construction of evolutionary trees is one of the major problems in computational biology, mainly due to its complexity.*

Results: *We present a new tree construction method that constructs a tree with minimum score for a given set of sequences, where the score is the amount of evolution measured in PAM distances. To do this, the problem of tree construction is reduced to the Traveling Salesman Problem (TSP). The input for the TSP algorithm are the pairwise distances of the sequences and the output is a circular tour through the optimal, unknown tree plus the minimum score of the tree. The circular order and the score can be used to construct the topology of the optimal tree. Our method can be used for any scoring function that correlates to the amount of changes along the branches of an evolutionary tree, for instance it could also be used for parsimony scores, but it cannot be used for least squares fit of distances. A TSP solution reduces the space of all possible trees to* 2^n*. Using this order, we can guarantee that we reconstruct a correct evolutionary tree if the absolute value of the error for each distance measurement is smaller than* $\frac{x}{2}$*, where x is the length of the shortest edge in the tree. For data sets with large errors, a dynamic programming approach is used to reconstruct the tree. Finally simulations and experiments with real data are shown.*

Availability: *The software may be used via our cbrg server at http://cbrg.inf.ethz.ch/MultAlign.*

Contact: *chantal.roth@nobilitas.com*

Supplementary information: *An html and postscript version of this paper is available at http://chantal.nobilitas.com/ (Papers section).*

Introduction

The construction of optimal evolutionary trees is a very challenging problem, since most versions of the problem are NP complete (Agarwala *et al.*, 1996). Even though the problem has been studied extensively, evolutionary tree construction still remains an open problem.

DEFINITION 1. A *phylogenetic tree* $T = (V, E)$ is a binary connected acyclic graph, where V are the vertices (nodes) and E denotes the edges of the graph. A *leaf* in T has degree 1 and L is used to denote the subset of V which contain the leaves of T. We use $T(S)$ to denote a tree with leafset S.

In our context, a *tree* $T(S)$ associated with a set of sequences $S = \{s_1, \ldots, s_n\}$ is the tree that corresponds to the evolutionary history of the sequences of S. The root of the tree has no relevance in our context. The internal nodes V represent (usually unknown) ancestor sequences. There are three major families of methods for inferring phylogenies that basically use three different classes of scoring functions: parsimony and compatibility methods (Estabrook *et al.*, 1975, 1976; Sankoff, 1975; Dress and Steel, 1993), distance based methods (Cavalli-Sforza and Edwards, 1967; Fitch and Margoliash, 1967; Hogeweg and Hesper, 1988; Hein, 1989; Gonnet and Benner, 1996), and maximum likelihood methods (Felsenstein, 1973, 1981; Thorne *et al.*, 1993).

DEFINITION 2. A *Tree scoring function* is a function $F : \mathcal{T} \to \mathbb{R}$.

DEFINITION 3. Let $\mathcal{T}$ be the set of all possible trees that can be generated for a given set of sequences $S = \{s_1, s_2, \ldots, s_n\}$. The *optimal tree* $\bar{T} \in \mathcal{T}$ is a tree such that $F(\bar{T}) = \min_{T \in \mathcal{T}} F(T)$.[†]

Parsimony. The parsimony methods usually count the number of amino acid or nucleotide substitutions in a weighted or unweighted manner. They take a multiple sequence alignment (MSA) as input and minimize the number of changes to explain the corresponding evolutionary tree. The construction of an optimal MSA, which

*To whom correspondence should be addressed.

[†] For scoring functions where a larger value corresponds to a better alignment multiply the function by −1.

is needed as input, is also NP complete (Jiang and Wang, 1994). In addition, many algorithms for calculating MSAs need an evolutionary tree as input, which makes the problem circular.

Distance matrix methods. Distance matrix methods fit a tree to a matrix of pairwise distances between the sequences. Most distance methods use some form of weighted or unweighted least squares measure. The distances d_{ij} are given. The problem is to find distances $\delta_{i,j}$ such that the score of the tree is minimized. The score of a tree is often defined as:

$$F(T) = \sum_{i<j} \frac{(d_{ij} - \delta_{i,j})^2}{d_{ij}^p} \qquad (1)$$

where where p can be set to 0, 1 or 2. The only rigorous way to get an optimal solution is by trying out all tree topologies. However, finding the optimal topology is generally an intractable problem, since the number of tree topologies grows exponentially with the number of nodes.

Maximum likelihood. The maximum likelihood (ML) approach chooses the tree which maximizes the probability that the observed data would have occurred. One candidate for the best tree is the tree that maximizes the likelihood. The strategy is to search over all trees, and for each topology T to find the lengths of the edges that maximize the likelihood $P(D|T, M)$.

In this paper, we present an algorithm called CircTree that takes unaligned amino acid sequences as input and produces the evolutionary tree with minimum score. It creates the correct evolutionary tree, if the error of each distance is not greater than $\frac{x}{2}$, where x is the length of the shortest edge in the tree. In the following sections we explain our tree scoring function. We then introduce the main ideas and algorithm. Our method can be used for any scoring function that correlates to the amount of changes along the branches of an evolutionary tree, for instance it could also be used for parsimony scores, but it cannot be used for least squares fit of distances. Finally we show the results of a simulation and some real examples.

Definitions

DEFINITION 4. The *optimal pairwise alignment* OPA(s_1, s_2) of two sequences s_1, s_2 is an alignment with the *maximum score* where a probabilistic scoring method (Dayhoff *et al.*, 1978; Gonnet *et al.*, 1992) is used. We refer to a pairwise alignment of two sequences s_1, s_2 with $\langle s_1, s_2 \rangle$.

The optimal score and the PAM distance (see Definition 5) is determined via standard dynamic programming (Smith and Waterman, 1981; Gotoh, 1982), where many Dayhoff matrices are used (for each PAM distance there is a different Dayhoff matrix). An affine gap cost is used according to the formula $a + lb$, where a is a fixed gap cost, l is the length of the gap and b is the incremental cost (Benner *et al.*, 1993).

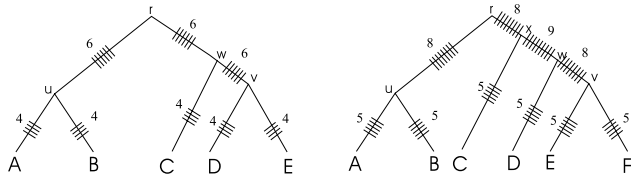

Fig. 1. Traversal of a tree using the SP measure. Some edges are traversed more often than others. The numbers indicate how often an edge was traversed.

DEFINITION 5. An ϵ-PAM unit is the amount of evolution which will change, on average, ϵ% of the amino acids, when ϵ is infinitely small. The function *PAM* (*s1, s2*) is the PAM distance of two sequences s_1, s_2 that maximizes the OPA score.

Methods

Our CircTree method is based on a probabilistic scoring function for MSAs that takes into account an associated evolutionary tree. A variant of this scoring function can be used to evaluate evolutionary trees as well. From this scoring function, which we call CS measure, we derived an algorithm for reconstructing an optimal evolutionary tree. We will first introduce the scoring function and then describe the algorithm.

Sum of pairs versus circular sum measure

The sum of pairs (SP) measure is a well known scoring function for MSAs (Carillo and Lipman, 1988; Kececioglu, 1993; Gupta *et al.*, 1995, 1996). To calculate the score of an MSA with the SP measure (Carillo and Lipman, 1988), all $\binom{n}{2}$ scores of the pairwise alignments are added up. Sum of pairs methods are obviously deficient from an evolutionary perspective. Consider a tree (Figure 1) constructed for a family containing five proteins. The score of a pairwise alignment $\langle A, B \rangle$ evaluates the probability of evolutionary events on edges (u, A) and (u, B) of the tree; that is, the edges that represent the evolutionary distance between sequence A and sequence B. Likewise, the score of a pairwise alignment $\langle C, D \rangle$ evaluates the probability of evolutionary events on edges (C, w), (w, v) and (v, D) of the tree. The edge lengths correspond to the PAM distances.

By adding 'ticks' to the evolutionary tree that are drawn each time an edge is evaluated when calculating the SP score (Figure 1), it is readily seen that with the SP method different edges of the evolutionary tree of the protein

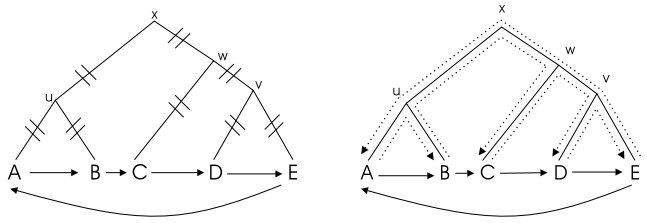

Fig. 2. Traversal of a tree in circular order.

family are counted a different numbers of times. There is no theoretical justification to weight some branches more than others. In addition, it is not simply the root that is weighted more than others, as can be shown with larger examples. Thus, SP methods are intrinsically problematic from an evolutionary perspective for scoring MSAs.

Traveling salesman

DEFINITION 6. A *circular order* $C(T)$ of a set of sequences $S = \{s_1, \ldots, s_n\}$ is any tour through a tree $T(S)$ where each edge is traversed exactly twice, and each leaf is visited once.

This problem can be resolved by traversing the tree in a circular order, that is, from leaf A to B, from B to C, from C to D, from D to E and then back from E to leaf A (Figure 2). All edges are counted exactly twice. A circular order is the shortest possible tour through a tree where each leaf is visited once (shortest means smallest sum of edge lengths) (see Figure 2) (Gonnet *et al.*, 1999).

Scoring an evolutionary tree

Our scoring function is based on this circular order. If we add the PAM distances of all the pairwise alignments in a circular order C with respect to the evolutionary tree T and divide this sum by two, we count each edge of the tree once. Note that we force the tour to go through all leaves in the tree. So the function for calculating the score or total path length of a tree $T(S)$ with a circular order C is:

$$F(T) = \tfrac{1}{2} \sum_{i=1}^{n} \mathrm{PAM}(s_{C_i}, s_{C_{i+1}}) \qquad (2)$$

where $C_{n+1} = C_1$, n is the number of sequences, and PAM is the PAM distance of the optimal pairwise alignment (see Definition 4). We will use $F(C)$ to denote the path length when a circular order C is used as input instead of a tree:

$$F(T) = F(C) = \tfrac{1}{2} \sum_{i=1}^{n} \mathrm{PAM}(s_{C_i}, s_{C_i+1}) \qquad (3)$$

when C is a circular order of T. Note that the scoring function does not depend on the root of the tree or any internal nodes.

Finding a circular order

The problem we consider is to find such an order without having any information about the tree structure. But we know that a circular order is the shortest tour through a tree (Gonnet *et al.*, 1999).

To solve this problem we reduce it to the symmetric Traveling Salesman Problem (TSP): given is a matrix M that contains the $\binom{n}{2}$ distances of n cities (Johnson, 1987, 1990). The problem is to find the shortest tour where each city is visited once. We use a modified version of the problem: in our case, the cities correspond to the sequences and the distances are the PAM distances of the pairwise alignments.

In practice, the TSP is very well studied and optimal solutions can be calculated within a few hours for up to 1000 cities and in a few seconds for up to 100 cities.[‡] For real applications we have seldom more than 100 sequences to compare simultaneously. Furthermore, the calculation of the optimal TSP solution usually takes only a small *fraction* of the time it takes to compute all pairwise alignments to derive the PAM distances.

DEFINITION 7. The *TSP order* TSP(S) of a set of sequences $S = \{s_1, .., s_n\}$ is the order of the sequences that is derived from the optimal solution of a TSP, where the distances between the sequences are the pairwise PAM distances.

DEFINITION 8. A tour C^i is *shorter* than a tour C^j ($C^i < C^j$) if the path length of the tour C^i is smaller than the path length of the tour C^j.

Since we always talk about the same input sequences S, we will use C instead of $C(S)$ and T instead of $T(S)$. Let $C_{\min}$ be the circular tour that is derived with a TSP algorithm, where the input are the pairwise PAM distances of the input sequences S. Since $F(C)$ is the score of an evolutionary tree with circular order C, $F(C_{\min})$ is the minimum score, if $C_{\min}$ is a circular order of $T_{\min}$.

LEMMA 1. *There exists no tree with a lower score than* $T_{\min}$.

PROOF. Assume this statement is wrong and there exists a tree T' with a smaller score than $T_{\min}$. Assume further that we know this tree T'. Hence we can derive a circular order C' easily. The sum of the PAM distances in this order ($F(C')$) is the score of the tree. If T' really has a smaller score, then the score derived from C' would have to be shorter than the score derived from $C_{\min}$, which is a contradiction.

[‡] There are heuristics for large scale problems that calculate near optimal solutions that are within 1–2% of the optimum Groetschel and Holland (1991); Padberg and Rinaldi (1991).

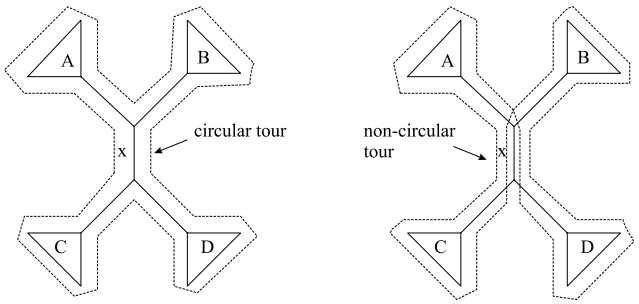

Fig. 3. A non-circular order traverses at least one edge (x) at least four times.

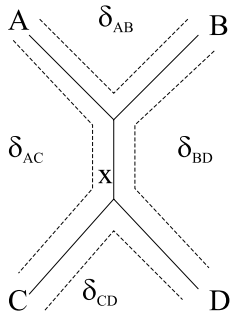

Fig. 4. $\delta_{AB} + \delta_{CD} - \delta_{DB} - \delta_{CA} = -2x$.

In summary, we can compute a circular order $C_{\min}$ without knowing $T_{\min}$. With $C_{\min}$ we can compute the minimum score $F(C_{\min})$ of the unknown tree $T_{\min}$.

Error bounds

In reality evolution is a random process and we are estimating distances from a finite sample. So even if our model of evolution was absolutely correct, there is always some variance, which we perceive as an error $\epsilon \in \mathbb{R}$. In addition, our model of evolution is not perfect, which adds further error. Hence each distance measurement has some error. We would like to determine how small the errors of the distance measurement may be in order to get a correct circular tour. We will first consider the error if an arbitrary circular order is chosen and then discuss the error if a TSP order is chosen, as in our algorithm.

DEFINITION 9. Let L be a set of n leaves of a tree T. The *distance* between two leaves $x, y \in L$ is δ_{xy}. It is the unique path length from leaf x to leaf y. We assume that distances are symmetric, hence $\delta_{xy} = \delta_{yx}$.

Error bound for any circular order

Given is a tree T (see Figure 3). We want to determine how large the distance measurement error may be, such that we still get a correct order. To determine this error, we do the opposite and determine the smallest possible error such that we get a wrong circular order C', which means that at least one edge x is traversed more than twice. Since we are interested in the worst case scenario, we only consider the shortest edge x because the longer the edges are, the larger the errors may be. A circular order C will pass edge x exactly twice. Valid circular orders in Figure 3 are [A, B, C, D] and [A, B, D, C]. Let δ_{ij} be the distance without error and let ϵ_{ij} be the error of the distance measurement (so the *measured* distance would be $\delta_{ij} + \epsilon_{ij}$).

DEFINITION 10. A *subtree* $T_u(V', E')$ of $T(V, E)$ is a tree T_u with $V' \in V$, $E' \in$ E, where u is the root of T_u and $u \in V'$, and all the directed paths from u to the leaves in T are also present in T_u.

In other words: if we remove the parent edge above u in T, we get two graphs, one of which is T_u, a subtree of T.

An incorrect order C' (right side of Figure 3) will pass edge x four times. Two non-circular orders are [A, C, B, D] and [A, D, B, C]. In all cases, subtrees A, B, C and D are all traversed in the same way, so we can represent them with any leaf in the subtree. If the output of a TSP algorithm is a wrong circular order C', the following inequality must be satisfied:

$$\delta_{AB} + \delta_{BC} + \delta_{CD} + \delta_{DA} > \quad (4)$$
$$\delta_{AD} + \delta_{DB} + \delta_{BC} + \delta_{CA} \quad (5)$$

We now include the errors ϵ_{ij} and simplify the above inequality (4):

$$\delta_{AB} + \epsilon_{AB} + \delta_{CD} + \epsilon_{CD}$$
$$-\delta_{DB} - \epsilon_{DB} - \delta_{CA} - \epsilon_{CA} > 0 \quad (6)$$

But when our distance matrix would be additive, without any error, then the following equation (7) holds (see Figure 4):

$$\delta_{AB} + \delta_{CD} - \delta_{DB} - \delta_{CA} = -2x \quad (7)$$

If we subtract equation (7) from inequality (6), we get:

$$\epsilon_{AB} + \epsilon_{CD} - \epsilon_{DB} - \epsilon_{CA} > 2x \quad (8)$$

The conclusion is: as long as the absolute value of each error ϵ_{ij} is smaller than $\frac{x}{2}$, we get a correct circular order.

Error bound for a TSP circular order

The circular order we use is not just any circular order, but it is the best (shortest) circular order,[§] as it is the solution of a TSP algorithm.

[§] Since we assume errors in the distances, we assume that not all circular orders have exactly the same length. A TSP algorithm will yield the shortest circular order.

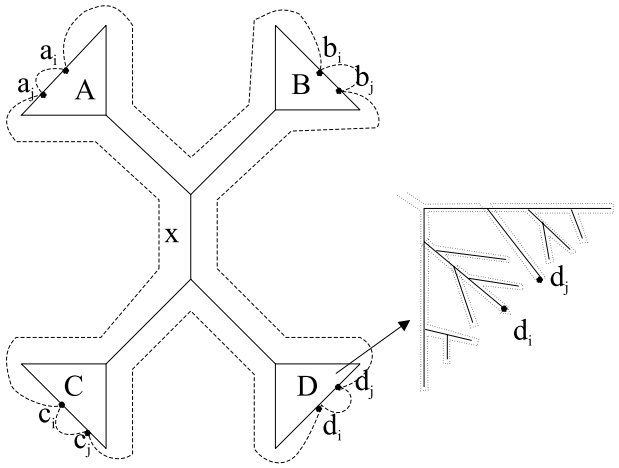

Fig. 5. If the tour C' that was obtained from the optimal solution of a TSP, then all $|A|\cdot(|A|-1)\cdot|B|\cdot)(|B|-1)\cdot|C|\cdot(|C|-1)\cdot|D|\cdot(|D|-1)$ other circular orders have to be greater than the wrong TSP order C'.

In order to get a wrong tour, *all* other circular tours that involve the same distances to cross edge x must be larger than some wrong order C' (otherwise, if just one of the other correct circular orders C_i would yield a shorter tour than C', then the TSP algorithm must return the correct tour C_i).

The number of different circular orders there are for traversing x is depicted in Figure 5: for the subtree A there are $|A|$ possible ways to start and $|A|-1$ end the cycle in the subtree. The same accounts for subtrees B, C and D. There are in total $|A|\cdot(|A|-1)\cdot|B|\cdot(|B|-1)\cdot|C|\cdot(|C|-1)\cdot|D|\cdot(|D|-1)$ circular orders that have to be greater than the wrong order C'. So for instance for a tree with 16 leaves, the edge in the middle has $(4*3)^4$ orders associated with it, which is more than 20 000. And all of them have to be greater than the wrong order C'.

The probability for getting a wrong order cannot be determined in an easy way, as the distances are not all independent in different orders. But we can see that the probability to get a wrong order using a TSP algorithm is extremely low.

Tree construction algorithm

We now present a tree construction algorithm called CircTree that is based on this circular order C. The input for the algorithm is the PAM distances of the pairwise alignments plus the circular order C of the optimal tree $T_{\min}$.

Idea of the algorithm

DEFINITION 11. Two leaves/subtrees a, b are *connected* in a tree T if the two leaves/subtrees have the same parent node in T.

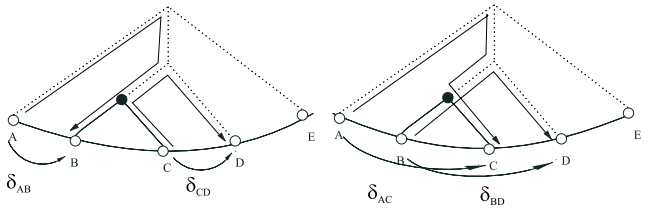

Fig. 6. $\delta_{AB}+\delta_{CD}-\delta_{AC}-\delta_{BD}=0$.

First we reorder (and renumber) the leaves in order of C. Since we consider binary trees only, we know that at least two leaves are *connected*. When two leaves that are connected are swapped, the result is a tree with the same topology. If we traverse the tree with the swapped leaves in the new order, we get the same path length.

But when two leaves are swapped that are not connected (see Figure 3), the new ordering of the leaves is not circular anymore. Hence, some edges of the tree are traversed too often when the score is calculated using this new order. This leads to a larger (worse) score.

Hence the first step is to swap each of the neighboring pair of leaves and to calculate the resulting total score. If the score stays the same, we know that the two leaves are connected. If the score increases, the leaves are not connected.

DEFINITION 12. Given is a tree T and a circular order C and leaves $L=S=\{s_1,\ldots,s_n\}$. Rename the leaves in a way s.t. the order of the leaves is in circular order C. We define $d(s_i)$ to be: $d(s_i)=\delta_{s_{i-1}s_i}+\delta_{s_{i+1}s_{i+2}}-\delta_{s_{i-1}s_{i+1}}-\delta_{s_is_{i+2}}$

EXAMPLE 1. *Assume leaves B and C were connected (see Figure 6). In the tree the distance* δ_{AB} *plus* δ_{CD} *is the same as the distance* δ_{AC} *plus* δ_{BD} *(when the leaves B and C are interchanged), because the tree topologies are identical. The difference between those two sums* $d(B)$ *is:*

$$d(B)=\delta_{AB}+\delta_{CD}-\delta_{AC}-\delta_{BD}=0 \qquad (9)$$

whereas d(C) is $2x$ *[if we try to swap the leaves (C, D)] (see Figure 7), because the two leaves are not connected. So with the function* $d(s_i)$ *we can determine whether leaves* s_i *and* s_{i+1} *are connected or not.*

Note that $d(s_i)$ corresponds exactly to the equation we used in the determination of the bound for the circular order. This means that the same error bound holds for the tree construction as for the circular order.

CircTree algorithm. Given is a set of sequences $S=\{s_1,\ldots,s_n\}$. First, the optimal circular order C is calculated with a TSP algorithm. The sequences are renamed

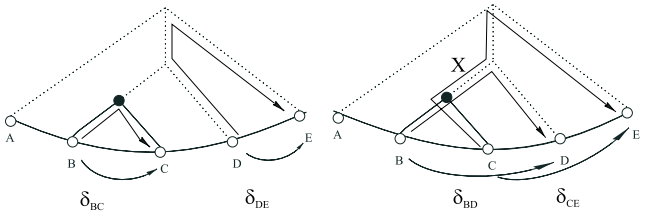

Fig. 7. $\delta_{BC} + \delta_{DE} - \delta_{BD} - \delta_{CE} = -2 \cdot x$.

8 leaves	100 PAM	200 PAM	300 PAM	Time (s)
PROBMODEL	73%	76%	57%	.02
PHYLIP	80%	73%	69%	1.2
TSP	93%	78%	71%	1.4

16 leaves	100 PAM	200 PAM	300 PAM	Time (s)
PROBMODEL	25%	58%	42%	.4
PHYLIP	8%	25%	58%	1.4
TSP	41%	42%	45%	5

Fig. 8. Percentage of correct constructions and time for eight and 16 leaves.

with respect to that order. Starting with the optimal circular order C, each of the n pairs of neighboring leaves are swapped and the path difference $d(s_i)$ is calculated for $i = 1 \ldots n$. To save computation time, we initially calculate all n $d(s_i)$, sort them, and store them in a list D (see Figure 9 in the Appendix).

The best connection is the one with the smallest $d(s_i)$:

$$\epsilon = \min_{1 \le i \le n} d(s_i).$$

The leaves are connected, and one of the connected leaves is chosen as a representative for the next steps. For the next connection step only two path differences have to be recalculated.

Since there are $n - 2$ internal nodes (without the root), the total computation needs $O(n \log(n))$ for the sorting and linear time in n for the actual tree construction. Once the tree structure is known the exact places of the nodes can be obtained with any least squares method (Cavalli-Sforza and Edwards, 1967; Fitch and Margoliash, 1967; Hogeweg and Hesper, 1988), which takes in the order of n^2 time. So given a circular order C, the tree topology can be determined in $O(n \log(n))$ time. When only the input sequences are given, then the overall computation time when only input sequences are given is determined by the TSP algorithm.

Note that the calculation of the TSP takes only a fraction of computation time in real cases (up to about 100 sequences) than the calculation of the $\binom{n}{2}$ pairwise scores.

Search space. For a tree with n different leaves, there are in the order of $N(n) = 3 \times 5 \times \cdots \times (2n - 5) = \prod_{k=1}^{n-3} 2k + 1$ different tree topologies (Felsenstein, 1978). The TSP solution reduces the search space to $O(2^{n-3})$ different trees that could be built with that order. So given a correct TSP order, there are 2^{n-3} possible different trees. So far we know that if the absolute value of the error of each distance measurement is not larger then $\frac{x}{2}$, where x is the shortest edge length, we can guarantee that the tree construction is correct.

Dynamic programming approach

The dynamic programming version of our algorithm is easy: instead of connecting the two leaves with the smallest error, the best k connections are chosen (where k is a user specified parameter). A second parameter, m, limits the number of total trees in each step, because in the worst case the number of trees may grow exponentially. At the end, the tree with the overall smallest error is chosen. This way the probability to get an edge x wrong is much lower: in order to get a wrong connection, *all* k connections have to be wrong in each step.

If we would simply keep all k trees at each step, in the worst case we would get 2^{n-3} trees, as this is the number of possible different trees that can be built given a circular order. Note that this can only happen if the tree depth is close to n, or if k is very large.

In real cases, many trees will end up being the same during the construction process. For instance, whether we first connect leaves 1, 2 and then 3, 4 or 3, 4 and then 1, 2 does not matter. The number of different trees at the end is seldom more than 100 when performing experiments for $n \le 50$.

In order to determine whether two trees are equal, we use a tree signature function $G(T)$. For two trees T and T', $G(T) = G(T')$ if and only if the trees have the same topology.[¶]

Simulation

The simulation is done as follows: based on the amino acid frequencies, a random amino acid sequence is produced, which is the root of a tree. We let the sequence mutate randomly into different directions, according to a stochastic process, where the mutation probabilities are acquired from a Dayhoff matrix according to the estimated PAM distance. No molecular clock is assumed. Insertion and deletion events happen corresponding to the model (Baldi *et al.*, 1994). This process is continued for a certain time. At the end, some number of sequences are collected. Since we keep track of all mutation events, we

[¶] It is theoretically possible that two trees have the same signature even if they are not the same, due to the limited range of integer values in real computer systems, but this is extremely rare.

```
algorithm CircTree(S) → T_opt is
  C := TSP(S) "Circular order"
  L := reorder(S, C) "The leaves, reordered"
  F_min := Σ_{i=1}^{n} PAM(s_i, s_{i+1})
  T := L
  "Initial list of subtrees consists of leaves"
  D := [] "The list of path differences"
  n := |L| "Number of input sequences"
  for i = 1..n do
    D := D ∪ [d(i), i]
  od
  D := sort(D, x → D[1])
  "Sort by the path difference"
  for k = 1..n − 3 do
    i = D[1, 2]
    "Get index i for subtree T[i]
    T[i] := join(T[i], T[i + 1])
    "Join the two subtrees"
    T := T\T[i + 1]
    "Remove T[i + 1] and renumber indices in
D correspondingly"
    D := D\D[1]
    "Remove the corresponding entry in D"
    "recalculate d(i) and put it into the sorted
list D"
    od
    return T[1]
  end
```

Fig. 9. Construct a tree T from a set of sequences $S = \{s_1, .., s_n\}$ that has the minimum score.

know the correct tree for the collected set of sequences. For each simulation, the data (sequences or distances) were fed to different algorithms: ProbModel (Gonnet and Benner, 1996), the FITCH algorithm of the PHYLIP package (Fitch and Margoliash, 1967) and our new CircTree method.

We did the simulation for many combinations of the following parameters: (a) maximum PAM distances of any two sequences; (b) sequence length; (c) number of leaves. The results are summarized in Figure 8. In each experiment we collected the number of exact construction of trees in percentage, i.e. how often the constructed tree was equal to the real tree, time in seconds (CPU) and other data which is not shown here (i.e. tree fitting index, total PAM distance etc). Note that the CircTree method is implemented in Darwin, which is an interpreted language and thus much slower than C.

In the first experiment for a tree with eight leaves and maximum PAM distance of 100 (see Figure 7, top) the CircTree method gave the best result with 93% exact constructions, followed by FITCH (80%) and ProbModel (73%). In general for trees with eight leaves (or less) the

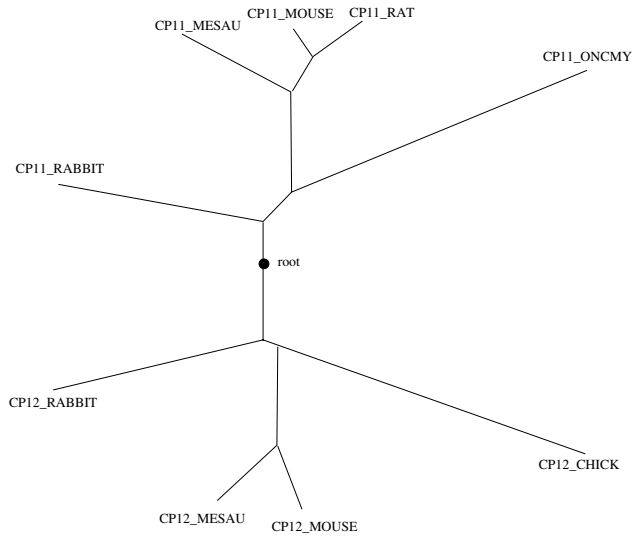

Fig. 10. Cytochrome P450 tree constructed with new method.

CircTree methods gave the best results. For 16 leaves (or more) sometimes the algorithms from the PHYLIP package (see Figure 7, bottom) gave the best results, sometimes the ProbModel and sometimes the Circular method. More simulations need to be done on larger trees to see if one particular method is suited best for a specific tree size and type.

The conclusions we draw from this simulation is that our method is useful and produces good results comparable to other frequently used tree construction algorithms, and in particular for small trees our algorithm produces better results in the simulation. Real examples are shown in the Appendix.

Discussion

In this paper we have introduced a new tree construction method that uses the solution of a TSP problem. The input to the TSP problem are the pairwise PAM distances which are calculated using a dynamic programming algorithm and the output is a circular order of the unknown evolutionary tree. Using this order we can reconstruct the corresponding evolutionary tree. We can guarantee that the correct tree is constructed if the absolute value of the errors of the distance measurement is less than $\frac{x}{2}$, where x is the length of the smallest branch. For larger errors a dynamic programming approach is used. Simulation studies show that the algorithm produces good results, comparable to other algorithms (FITCH, ProbModel), and produces better results for small trees. The algorithms are implemented into the Darwin package (Gonnet, 1994), and the algorithms are available on our server at http://cbrg.inf.ethz.ch.

```
Multiple sequence alignment:
----------------------------
Score of the alignment: 7089.83
Maximum possible score: 7089.83

2   _VLSAADKGHVKAAWGKVGGHAAEYGAEALER__MFLSFPTTKTYFPHF_DLSQGSAQVKGHGAKVAAAL
11  _VLSPEDKNNVKAAWSKVGGQAGDYGAEALER__MFLSFPTTKTYFPHF_DLSHGSAQVKGHGKKVGEAL
3   _VLSPTDKSIVKAAWEKVGAHAGDYGAEALER__MFLSFPTTKTYFPQF_DLSHGSAQVKGHGKKVADAL
5   _VLSPADKNNVKSAWNAIGSHAGEHGAEALER__MFLSFPPTKTYFPHF_DLSHGSAQIKTHGKKVADAL
1   _VLSPADKTNIKTTWDKLGGHAGEYGGEALER__TFMSFPTTKTYFPHF_DLSPGSAQVKAHGKKVADAL
10  _VLTDAEKKEVTSLWGKASGHAEEYGAEALER__LFLSFPTTKTYFSHM_DLSKGSAQVKAHGKRVADAL
4   _VLTEDKKNHIRAIWGHVDWNPEAFGVEALTR__LFLAYPATKTYFAHF_DLNPGSAQIKAHGKKVVDAL
15  _VLTDDDKNHVRAIWGHVSWNPEAFGAEALYR__LFTAHPASKTYFSHF_DLHENSAQIRXXXXKVVDAL
14  MKLSADDKGHVKAIWEHVKGHEEAIGAEALCR__MFTSLPTTKTYFPTK_DIKEGSSFLKSHGKKVMGAL
8   _KLTAEDKKHVKAIWDHVKGHEEAIGAEALYR__MFCCMPTTRIYFPAK_DLSERSSYLKSHGKKVVGAL
12  _VLSEGMKKAIKNLLQKIESQTEVLGAEALAR__LFECHPQTKSYFPKFSGFSANDKRVKHHGALVLKAL
7   _SLSDKDKAAVKALWSKIGKSADAIGNDALSR__MIVVYPQTKTYFSHWPSVTPGHPDIKAHGKKVMGGL
6   _SLSDKDKAAVRALWSKIGKSADAIGNDALSR__MIVVYPQTKTYFSHWPDVTPGSAHIKAHGKKVMGGI
13  _SLSDKDKAAVRALWSKIGKSSDAIGNDALSR__MIVVYPQTKIYFSHWPDVTPGSPMIKAHGKKVMGGI
9   _SLTAKDKSVVKAFWGKISGKADVVGAEALGRDKMLTAYPQTKTYFSHWADLSPGSGPVKKHGCIIMGAI

 1 -> HBA1_ARCGA    2 -> HBA1_BOSMU    3 -> HBA1_GALCR    4 -> HBA1_IGUIG
 5 -> HBA1_LEMVA    6 -> HBA1_MOTAN    7 -> HBA1_NOTCO    8 -> HBA1_PLEWA
 9 -> HBA1_SALIR   10 -> HBA1_TACAC   11 -> HBA1_TADBR   12 -> HBA1_TURMA
13 -> HBA1_TRENE   14 -> HBA1_TRICR   15 -> HBA1_UROHA
```

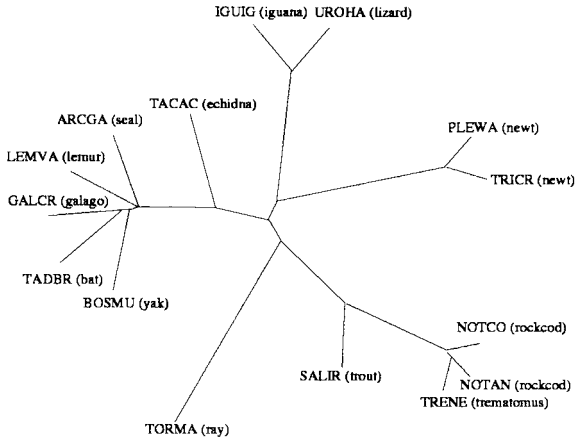

Fig. 11. Hemoglobin alpha-I constructed using the new method.

Appendix

Algorithm

Figure 9 is a rough overview of the algorithm. Only the non-dynamic programming algorithm is shown. T is a list of subtrees that is filled when leaves are connected. At the beginning, T simply consists of the leaves. Some details are left out, for instance, if the algorithm is implemented it is faster to keep a list of pointers that points a leaf to the corresponding $T[i]$ instead of renumbering the leaves. But too keep it simple enough we did not want to go too much into details.

Examples

We tested the new tree construction method on several protein families, two of which are shown here. We constructed the trees with two other methods, the FITCH algorithm (Fitch and Margoliash, 1967) and the ProbModel (Gonnet and Benner, 1996). In all cases all the three algorithms agreed on the tree topology. We only took nine and 15 proteins for drawing reasons, but the algorithm allows any number of sequences.

The first one is the IA1 and IA2 subunit of Cytochrome P450 (see Figure 11). In this example you can clearly see a gene duplication (symmetry). The upper half is the IA1 subunit, the lower half is the IA2 subunit.

The second example is the Hemoglobin alpha-I chain from 15 species. The MSA below was calculated using the ProbModel and gap heuristics (Korostensky and Gonnet, 1999). The score is the CS measure as described in (Gonnet *et al.*, 1999). This alignment is the optimal alignment, as the maximum possible score is equal to the score. The TSP order is: [2, 11, 3, 5, 1, 10, 4, 15, 14, 8, 12, 7, 6, 13, 9, 2].

References

Agarwala,R., Bafna,V., Farach,M., Narangyan,B., Paterson,M. and Thorup,M. (1996) On the approximability of numerical taxonomy: fitting distances with trees. *SIAM J. Comput.*, 365–372.

Baldi,P., Chauvin,Y., Hunkapiller,T. and McClure,M.A. (1994) Hidden markov models of biological primary sequence information. *Proc. Natl. Acad. Sci. USA*, **91**, 1059–1063.

Benner,S.A., Cohen,M.A. and Gonnet,G.H. (1993) Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Mol. Biol.*, **229**, 1065–1082.

Carillo,H. and Lipman,D. (1988) The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, **48**(5), 1073–1082.

Cavalli-Sforza,L. and Edwards,A. (1967) Phylogenetic analysis: models and estimation procedures. *Evolution*, **32**, 233–257.

Dayhoff,M.O., Schwartz,R.M. and Orcutt,B.C. (1978) A model for evolutionary change in proteins. In Dayhoff,M.O. (ed.), *Atlas of Protein Sequence and Structure* vol. 5, pp. 345–352.

Dress,A. and Steel,M. (1993) Convex tree realization of partitions. *Appl. Math. Lett.*, **5**, 3–6.

Estabrook,G., Johnson,C. and McMorris,F. (1975) An idealized concept of the true cladistic character. *Math. Biosci.*, **23**, 263–272.

Estabrook,G., Johnson,C. and McMorris,F. (1976) A mathematical foundation for the analysis of cladistic character compatibility. *Math. Biosci.*, **29**, 181–187.

Felsenstein,J. (1973) Maximum-likelihood estimation of evolutionary trees from continuous characters. *Am. J. Hum. Gen.*, **25**, 471–492.

Felsenstein,J. (1978) The number of evolutionary trees. *Syst. Zool.*, **27**, 401–410.

Felsenstein,J. (1981) Evolutionary trees from gene frequencies and quantitative characters: finding maximum likelihood estimates. *Evolution*, **35**, 1229–1242.

Fitch,W. and Margoliash,E. (1967) The construction of phylogenetic trees. *Science*, **155**, 279–284.

Gonnet,G., Korostensky,C. and Benner,S. (1999) Evaluation measures of multiple sequence alignments. *J. Comp. Biol.*.

Gonnet,G.H. (1994) A tutorial introduction to computational biochemistry using Darwin.

Gonnet,G.H. and Benner,S.A. (1996) Probabilistic ancestral sequences and multiple alignments. *Fifth Scandinavian Workshop on Algorithm Theory, Reykjevik July 1996*.

Gonnet,G.H., Cohen,M.A. and Benner,S.A. (1992) Exhaustive matching of the entire protein sequence database. *Science*, **256**, 1443–1445.

Gotoh,O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Groetschel,M. and Holland,O. (1991) Solution of large-scale symmetric traveling salesman problems. *Math. Program.*, 141–202.

Gupta,S., Kececioglu,J. and Schaffer,A. (1995) Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. *Proceedings of the 6th Symposium on Combinatorial Pattern Matching*, 128–143.

Gupta,S.K., Kececioglu,J. and Schaffer,A.A. (1996) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J. Comput. Biol.*.

Hein,J. (1989) An optimal algorithm to reconstruct trees from additive distance data. *Bull. Math. Biol.*, **51**, 597–603.

Hogeweg,P. and Hesper,P. (1988) The alignmen of sets of sequences and the construction of phylogenetic trees: an integrated method. *J. Mol. Evol.*, **20**, 175–186.

Jiang,T. and Wang,L. (1994) On the complexity of multiple sequence alignment. *J. Comp. Biol.*, **1**, 337–348.

Johnson,D. (1987) More approaches to the travelling salesman guide. *Nature*, **330**, 525.

Johnson,D. (1990) Local optimization and the traveling salesman problem. *Proceedings of the 17th Colloq. on Automata, Languages and Programming* volume 443 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 446–461.

Kececioglu,J. (1993) The maximum weight trace problem in multiple sequence alignment. *Proceedings of the 4th Symposium on Combinatorial Pattern Matching* pp. 106–119.

Korostensky,C. and Gonnet,G. (1999) Gap heuristics and tree construction using gaps. *Technical Report* Institute of Scientific Computing, ETH Zuerich, pp. 321.

Padberg,M. and Rinaldi,G. (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, **33**, 60–100.

Sankoff,D. (1975) Minimal mutation trees of sequences. *SIAM J. Appl. Math.*, **28**, 35–42.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Thorne,J., Kishino,H. and Felsenstein,J. (1993) Inching toward reality: an improved likelihood model of sequence evolution. *J. Mol. Biol.*, **34**, 3–16.