Using Traveling Salesman Problem Algorithms for Evolutionary Tree Construction

Chantal Korostensky and Gaston H. Gonnet
Institute for Scientific Computing
ETH Zurich, Switzerland
e-mail: {gonnet,korosten}@inf.ethz.ch

Abstract

We present a new tree construction method that constructs a tree with minimum score for a given set of sequences. 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 of the optimal tree can be used to construct the topology of the tree in $O(n \log(n))$ time where n is the number of input sequences. We can guarantee that we reconstruct a correct evolutionary tree if the error for each distance measurement is smaller than $\frac{x}{2}$, where x is the shortest edge in the tree. For data sets with large errors, a dynamic programming approach is used to reconstruct the tree.

Keywords: tree construction, Traveling Salesman, circular order, evolution

1 Introduction

The construction of optimal evolutionary trees is a very challenging problem, since most versions of the problem are NP complete [1]. Even though the problem has been studied extensively, evolutionary tree construction remains still an open problem.

Definition 1.1 *A* <u>*tree*</u> $T(S) = (V, E, S)$ *is a binary, leaf labeled tree with leafset* $S = \{s_1, .., s_n\}$.

In our context, a <u>tree</u> $T(S)$ associated with a set of sequences $S = \{s_1, .., s_n\}$ is the tree that corresponds to the evolutionary history of the sequences of S. 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: the parsimony and compatibility methods [28, 6, 7, 8], the distance based methods [12, 13, 20, 21, 4], and maximum likelihood methods [9, 11, 30].

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 is needed as input, is also NP complete [22]. In addition, many algorithms for calculating MSAs need an evolutionary tree as input, which makes the problem circular.

Definition 1.2 *Given is a set of sequences $S = \{s_1, .., s_n\}$ with $s_i \in \Sigma^*$ where Σ is a finite alphabet. A* Multiple Sequence Alignment *(MSA) consists of a set of sequences* $\mathrm{A} = \{a_1, a_2, .., a_n\}$ *with $a_i \in \Sigma'^*$ where $\Sigma' = \Sigma \cup \{"\_"\} \notin \Sigma$. $\forall a_i \in \mathrm{A} : |a_i| = k$. The sequence obtained from $a_i \in \mathrm{A}$ by removing all "_" gap characters is equal to s_i.*

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.

Maximum Likelihood The maximum likelihood approach chooses the tree which maximizes the probability that the observed data would have occurred from the constructed tree. We are interested in tree construction methods that only need the raw, unaligned sequences as input and do not require any further information such as an MSA. All our scoring functions are related to probabilities, and our goal is to construct the tree with maximum probability for a given set of sequences.

In this extended abstract, we present an algorithm that runs in $O(n \log(n))$ time and produces a tree with optimal score, if the error of each distance is not greater than $\frac{x}{2}$, where x is the shortest edge in the tree. In the following sections we briefly explain how the probability of an evolutionary configuration can be derived. We then introduce the main ideas and algorithm. Finally, we show how the error influences the results and present experimental results.

1.1 Definitions

Definition 1.3 *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 [5, 14] is used. We refer to a pairwise alignment of two sequences s_1, s_2 with $\langle s_1, s_2 \rangle$.*

Definition 1.4 *A 1-*PAM *unit is the amount of evolution which will change, on average, 1% of the amino acids. In mathematical terms, this is expressed as a matrix M such that $\sum_{i=1}^{20} f_i(1 - M_{ii}) = 0.01$, where f_i is the frequency of the i^{th} amino acid and M_{ij} is the probability that an amino acid i mutates into an amino acid j at one PAM distance [5]. The function* $PAM(s1, s2)$ *is the PAM distance of two sequences s_i, s_2 that maximizes the score.*

Usually the optimal score is determined via standard dynamic programming [29, 16]. An affine gap cost is used according to the formula $a + l \cdot b$, where a is a fixed gap cost, l is the length of the gap and b is the incremental cost [2].

Definition 1.5 *A* Tree scoring function *is a function $F : T \rightarrow I\!R$.*

Definition 1.6 *Let $\mathcal{T}$ be the set of all possible trees that can be generated for a given set of sequences $S = \{s_1, s_2, .., s_n\}$. The* optimal tree *$\bar{T} \in \mathcal{T}$ is a tree such that $F(\bar{T}) = \min_{T \in \mathcal{T}} F(T)$. (For some scoring functions we use the maximum instead of the maximum.)*

2 Methods

Originally we intended to develop a probabilistic scoring function for MSAs that takes into account an associated evolutionary tree. We then realized that this scoring function can be used to evaluate evolutionary trees as well. From this scoring function, which we call CS measure, we then derived an algorithm for reconstructing an optimal evolutionary tree.

2.1 Sum of Pairs versus Circular Sum measure

The sum of pairs (SP) measure is a well known scoring function for MSAs [3, 18, 19, 25]. We introduce the SP measure to show our motivation to find a better scoring function for MSAs. The connection to evolutionary trees will be clear after the following paragraph.

To calculate the score of an MSA with the SP measure [3], all $\binom{n}{2}$ scores of the pairwise alignments are added up. SP 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.

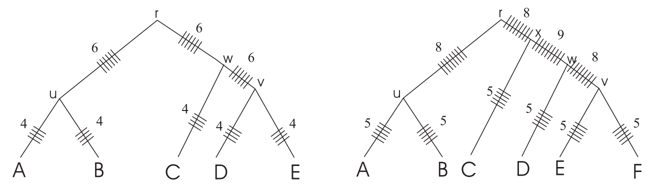

Figure 1: Traversal of a trees using the SP measure. Some edges are traversed more often than others.

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 family are counted a different numbers of times. In the example tree on the left side, edges (r, u), (r, w) and (w, v) are each counted six times by the SP method, while edges (u, A), (u, B), (v, D), (v, E), and (w, C) are each counted four times. It gets worse as the tree grows (see tree on the right).

Thus, SP methods are intrinsically problematic from an evolutionary perspective for scoring MSAs. This was the motivation to developed a scoring method that evaluates each edge equally. In addition, we wanted a scoring function for MSAs that does not depend on the actual tree structure. How this can be achieved is explained in the next section.

2.2 Traveling Salesman

Definition 2.1 *A <u>Circular order</u> $C(T)$ of set of sequences $S = \{s_1, .., s_n\}$ is any tour through a tree $T(S)$ where each edge is traversed exactly twice, and each leaf is visited once.*

When a tree is traversed 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) [15].

2.3 Scoring an Evolutionary Tree

Since traversing a tree $T(S)$ in circular order counts each edge in the tree exactly twice, we can derive a function for determining the score of an evolutionary tree using this order. First, we reorder the sequences in S according to the circular order $C(S)$. If we now add the PAM

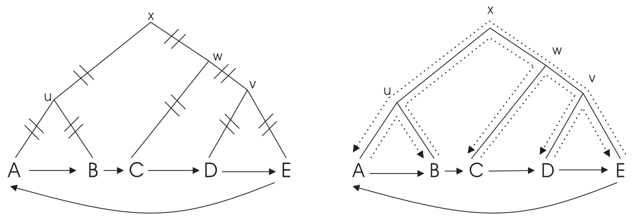

Figure 2: Traversal of a tree in circular order

distances of the pairwise alignments in this order and divide this sum by two, we score each edge of the tree once. So the function for scoring a tree $T(S)$ is simple:

$$F(T) = \frac{1}{2}\sum_{i=1}^{n} PAM(s_i, s_{i+1})$$

where $s_{n+1} = s_1$, n is the number of sequences, and PAM is the PAM distance of the optimal pairwise alignment (see Definition 1.3). We abbreviate this function with

$$F(T) = \sum C(S)$$

2.4 Finding a circular order

Since we want to construct an evolutionary tree and therefore do not have any tree, 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 [15].

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 [24, 23]. 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. There are heuristics for large scale problems that calculate near optimal solutions that are within 1% to 2% of the optimum [27, 17]. For real applications we have seldom more than 100 sequences to compare simultaneously, and 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 scores.

Definition 2.2 *The <u>TSP order</u> $TSP(S)$ of 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 2.3 *A tour C_i is <u>shorter</u> than a tour C_j ($C_i < C_j$) if the number of edges that are traversed by C_i is larger than the number of edges that are traversed by C_j. If $C_i < C_j$ then the score of the tour C_i is smaller than the score of the tour C_j ($\sum C_i > \sum C_j$).*

Problem 2.1 (TSP problem) *Given is a set of sequences $S = \{s_1, .., s_n\}$ and the corresponding scores of the optimal pairwise alignments. The problem is to find the shortest tour where each sequence is visited once.*

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. Since $\sum C$ is the score of an evolutionary tree, $\sum C_{min}$ is the minimum score.

Lemma 2.1 *C_{min} is a circular order of the optimal tree T_{min}.*

Proof 2.1 *Assume T_{min} does not have the circular order $C_{min}(S)$. We know that a circular order is the shortest tour through a tree, and any other ordering leads to a longer tour. Since C_{min} does not belong to the set of circular orders of T_{min}, then the tour that results from C_{min} must be longer. Hence the score derived from C_{min} cannot be optimal, which is a contradiction.*

Lemma 2.2 *There exists no tree with a lower score than T_{min}.*

Proof 2.2 *Assume this statement is wrong and there exists a tree T' with a lower 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 ($\sum C'$) is the score of the tree. If T' really has a lower score, then the score derived from C' would have to be smaller than the score derived from C_{min}, which is a contradiction.*

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

3 Error Bound

Before we present the tree construction algorithm, we want to determine how big the errors of the distance measurement can be in order to get a correct circular tour.

Definition 3.1 *Let L be a set of n leaves of a tree T. The <u>distance</u> between two leaves $x, y \in L$ is $d(x, y)$. It is the unique path length from leaf x to leaf y. We assume that distances are symmetric, hence $d(x, y) = d(y, x)$.*

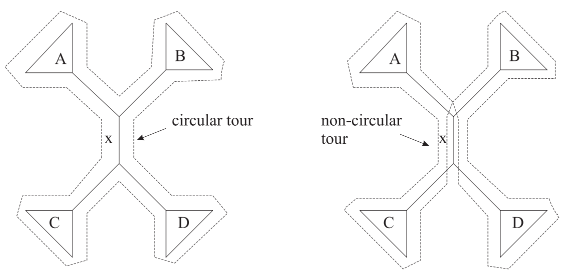

Figure 3: A non-circular order traverses at least one edge (x) at least four times

Given is a tree T (see Figure 3). We want to find the maximum error of the distance measurement, such that the circular order C' derived from an exact TSP algorithm is incorrect, and therefore at least one edge x (the shortest edge) is traversed more than twice. A correct circular order C will pass edge x exactly twice. Now assume that our distances have some error ϵ s.t we get an incorrect order C' (right side of Figure 3). In both 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 we have a wrong circular order C', the following inequality must be satisfied:

$$d(a,b) + d(b,c) + d(c,d) + d(d,a) > d(a,d) + d(d,b) + d(b,c) + d(c,a)$$

We now include the errors ϵ in the inequality and simplify the above inequality:

$$d(a,b) + \epsilon_{ab} + d(c,d) + \epsilon_{cd} - d(b,c) - \epsilon_{bc} - d(c,a) - \epsilon_{ca} > 0$$

where ϵ_{ij} is the error in the distance measurement $d(i,j)$. But when our distance matrix would be additive, without any error, then the following equation holds (see Figure 4):

$$d(a,b) + d(c,d) - d(b,c) - d(c,a) = -2x$$

If we subtract the second from the first equation, we get:

$$\epsilon_{ab} + \epsilon_{cd} - \epsilon_{bc} - \epsilon_{ca} > 2x$$

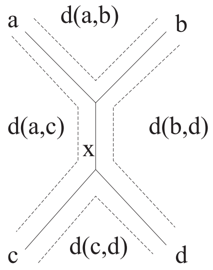

Figure 4: $d(a,b) + d(c,d) - d(b,c) - d(c,a) = -2x$

The conclusion is: we only get a wrong circular order if each of the four distances involved has an error of at least $\frac{x}{2}$ (the length of the shortest edge).

Note that this is the error if we simply take *any* random circular order C. But we do not take a random circular order, our circular order is the solution of a TSP algorithm. Hence to get a wrong tour, *all* other circular tours that involve the same distances to cross edge x must also be larger than the 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.

First we have to determine the number of circular orders we have to consider. We want to find the error with respect to the shortest edge x. So we have to figure out how many different circular orders there are for traversing x. In Figure 5 this situation is depicted: for the subtree A there are $|A|$ possible ways to start and end the cycle in the subtree. The same accounts for subtrees B, C and D. So there are in total $|A| \cdot |B| \cdot |C| \cdot |D| \cdot 2$ circular orders that have to be greater than the wrong order C' (the 2 comes from interchanging the subtrees themselves).

Let p be the probability that a circular order C is wrong in terms of the shortest edge x. Then the probability that the TSP order is wrong is:

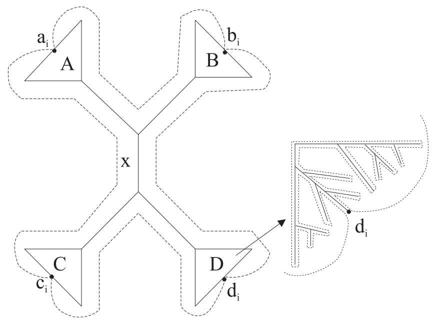

Figure 5: If the tour C' that was obtained from the optimal solution of a TSP, then all $|A| \cdot |B| \cdot |C| \cdot |D| \cdot 2$ other circular orders have to be greater than the wrong TSP order C'

$$P(\text{wrong TSP tour}) = p^{|A| \cdot |B| \cdot |C| \cdot |D| \cdot 2}$$

For illustration we give a small example: let the probability $p = 0.9$, that means that 9 out of 10 circular orders are wrong. Let us take a tree with 16 leaves, s.t. $|A| = |B| = |C| = |D| = 4$, so there are 512 circular orders that must be greater than the wrong order C'. The the probability to get a wrong TSP order is then: $P(\text{wrong TSP tour}) = 0.9^{512} = 10^{-24}$

4 Tree Construction Algorithm

We now present a tree construction algorithm that is based on this circular order C_{min}. Since we will only be using C_{min} from now on, we will refer to that order with C only. The input for the algorithm is the scores of the pairwise alignments plus the circular order C of the optimal tree T_{min}.

4.1 Idea of the algorithm

The basic idea of the tree construction algorithm is the following: write down the leaves in order of C. Starting from the leaves we want to find the internal nodes. 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 a tree with the same topology. If we traverse the tree with the swapped leaves in the same circular order as before, we get the same path length.

But when two leaves are swapped that are not connected (see Figure 3), some edges of the tree are traversed too often, which lowers the score. So all we have to do is to swap each of the neighboring pair of leaves and to calculate the resulting total path length or score. If the score stays the same, we know that the two leaves are connected. If the score increases, the leaves are not connected.

Since we use a TSP algorithm to find the order with the smallest score, the resulting score can never decrease. To save computation time we only recalculate the terms that actually change.

Definition 4.1 *Given is a tree T and a circular order C and leaves $L = \{s_1, .., s_n\}$. Rename the leaves in a way s.t. the order of the leaves is in circular order C. We define $\delta(s_i)$ to be: $\delta(s_i) = d(s_{i-1}, s_i) + d(s_{i+1}, s_{i+2}) - d(s_{i-1}, s_{i+1}) - d(s_i, s_{i+2})$*

Example 4.1: Assume leaves B and C were connected (see figure 6). In the tree the distance $d(A, B)$ plus $d(C, D)$ is the same as the distance $d(A, C)$ plus $d(B, D)$ (when the leaves B and

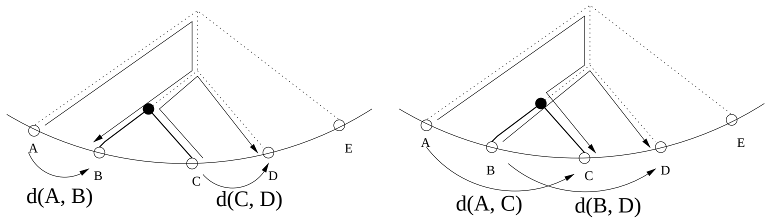

Figure 6: d(A, B) + d(C, D) - d(A, C) - d(B, D) = 0

C are interchanged), because the tree topologies are identical. The difference between those two sums $\delta(B)$ is:

$$\delta(B) = d(A, B) + d(C, D) - d(A, C) - d(B, D)$$

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

Note that $\delta(s_i)$ 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.

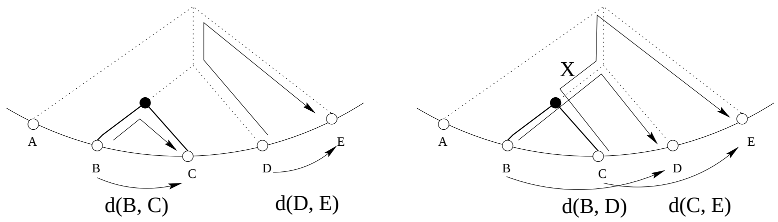

Figure 7: d(B, C) + d(D, E) - d(B, D) - d(C, E) = -2*x

4.1.1 Algorithm

Given is a set of sequences $S = \{s_1, .., s_n\}$. First, the optimal circular order C is calculated with a TSP algorithm. The sequences are renamed with respect to that order. Starting with the optimal circular order C, each of the $n-1$ pairs of neighboring leaves are swapped (see Algorithm 4.1.1) and the path difference $\delta(s_i)$ is calculated for $i = 1..n$. To save computation time, we initially calculate all n $\delta(s_i)$, sort them, and store them in a list D.

The best connection is the one with the smallest $\delta(s_i)$: $\epsilon = \min_{1 \le i \le n} \delta(s_i)$. 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 δ have to be recalculated, since nothing else has changed.

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 [4, 21, 12],

algorithm TreeConstruction$(L) \rightarrow T_{opt}$ **is**
    $C := TSP(L)$
    $L := reorder(S, C)$
    $F_{min} := \sum_{i=1}^{n} OPA(s_i, s_{i+1})$
    $T := L$
    "The initial tree only consists of leaves"
    $D := \{\}$
    for $i = 1..|L|$ **do**
        $D := D \cup \{\delta(i), i\}$
    od
    $D := sort(D)$
    for $k = 1..n-3$ **do**
        $\{\epsilon, best\} := min(D)$
        $T := join(T, L_{best}, L_{best+1})$
        $L := L \backslash L_{best}$
        "represent the two leaves with one leaf"
        recalculate $\delta(best)$ **and put it into the sorted list** D
    od
    return T
end

Figure 8: Algorithm 4.1
Construct a tree T from a set of sequences $S = \{s_1, .., s_n\}$ that has the minimum score

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. If the edge lengths are to be computed as well, then the computation takes $O(n^2)$ 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.

5 Examples

We tested the new tree construction method on two protein families. We also constructed the trees with two other methods, the Fitch algorithm of the PHYLIP package [12] and the ProbModel [13]. In all cases all the three algorithms agreed on the tree topology. We only took 9 and 15 proteins for drawing reasons, but the algorithm basically allows any number of proteins.

The first one is the IA1 and IA2 subunit of Cytochrome P450 (see Figure 10). 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 [26]. The score is the CS measure as described in [15]. This alignment is the optimal alignment, as the maximum possible score is equal to the score.

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

2       _VLSAADKGNVKAAWGKVGGHAAEYGAEALER__MFLSFPTTKTYFPHF_DLSQGSAQVKGHGAKVAAALTKAVEHLDDLPGALSELSDLHAHKLRVDPVNFKLLSHSLLVTLASHLPSDFTPAVHASLDKFLANVSTVLTSKYR
11      _VLSPEDKNNVKAAWSKVGGQAGDYGAEALER__MFLSFPTTKTYFPHF_DLSHGSAQVKGHGKKVGEALTTAVNHMDDLPGALSTLSDLHAYKLRVDPVNFKLLSHCLLVTLACHNPGEFTPAVHASLDKFLASVSTVLTSKYR
3       _VLSPTDKSIVKAAWEKVGAHAGDYGAEALER__MFLSFPTTKTYFPQF_DLSHGSAQVKGHGKKVADALTNAVLHVDDMPSALSALSDLHAHKLTVDPVNFKLLSHCLLVTLACHLPAEFTPAVHASLDKFMASVSTVLTSKYR
5       _VLSPADKNNVKSAWNAIGSHAGEHGAEALER__MFLSFPPTKTYFPHF_DLSHGSAQIKTHGKKVADALTNAVNHIDDMPGALSALSDLHAHKLRVDPVNFKLLSHCLLVTLASHHPAEFTPAVHASLDKFFAAVSTVLTSKYR
```

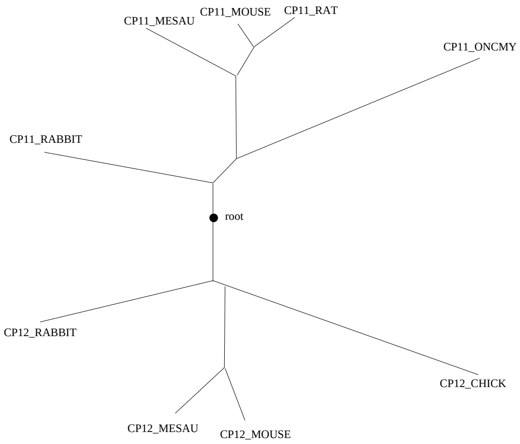

Figure 9: Cytochrome P450 tree constructed with new method

```
1       _VLSPADKTNIKTTWDKLGGHAGEYGGEALER__TFMSFPTTKTYFPHF_DLSPGSAQVKAHGKKVADALTTAVAHMDDLPKALSALSDLHAYKLRVDPVNFKLLSHCLLVTLACHHPAEFTPAVHASLDKFFSTVSTVLTSKYR
10      _VLTDAEKKEVTSLWGKASGHAEEYGAEALER__LFLSFPTTKTYFSHM_DLSKGSAQVKAHGKRVADALTTAAGHFNDMDSALSALSDLHAHKLRVDPVNFKLLAHCFLVVLARHHPAEFTPSAHAAMDKFLSRVATVLTSKYR
4       _VLTEDDKNHIRAIWGHVDNNPEAFGVEALTR__LFLAYPATKTYFAHF_DLNPGSAQIKAHGKKVVDALTQAVNNLDDIPDALAKLADLHAEKLRVDPVNFGLLGHCILVTIAAHNHGPLKADVALSMDKFLTKVAKTLVAHYR
15      _VLTDDDKNHVRAIWGHVSNNPEAFGAEALYR__LFTAHPASKTYFSHF_DLHENSAQIRXXXXKVVDALTQAVNNLDDLSGAISKLSDLHAEK___________________________________________________
14      MKLSADDKHNVKAIWEHVKGHEEAIGAEALCR__MFTSLPTTRTYFPTK_DIKEGSSFLHSHGKKVMGALSNAVAHIDDIDGALSKLSDKHAEELMVDPANFPKLAHNILVVLGIHLKPHLTYSVHSSVDKFLATVGYVLASKYR
8       _KLTAEDKHNVKAIWDHVKGHEEAIGAEALYR__MFCCMPTTRIYFPAK_DLSERSSYLHSHGKKVVGALTNAVAHIDDIDTAFSKLSDKHAEELMVDPANFPKLAHNILVVLGIHLKPHFTYSVHRSVDKFLSTVAYVLASKYR
12      _VLSEGNKKAIKNLLQKIHSQTEVLGAEALAR__LFECHPQTKSYFPKFSGFSANDKRVKHHGALVLKALVDTNKHLDDLPHHLNKLAEKHGKGLLVDPHNFKLFSDCIAVTLAAHLQ_EFSPETHCAVDKFLEEVTYQLSSLYR
7       _SLSDKDKAAVKALWSKIGKSADAIGNDALSR__MIVVYPQTKTYFSHWPSVTPGHPDIKAHGKKVMGGLAIAVSKINDLKAGLSNLSQQHAYKLRVDPANFKILNHCILVVISTMFPKNFTPQAHVSLNKFLSGVALALAQRYR
6       _SLSDKDKAAVRALWSKIGKSADAIGNDALSR__MIVVYPQTKTYFSHWPDVTPGSAHIKAHGKKVMGGIALAVSKIDDLKAGLSDLSEQHAYKLRVDPANFKILNHCILVVISTMFPKDFTPEAHVSLDKFLSGVALALAERYR
13      _SLSDKDKAAVRALWSKIGKSSDAIGNDALSR__MIVVYPQTKIYFSHWPDVTPGSPNIKAHGKKVMGGIALAVSKIDDLKTGLMELSEQHAYKLRVDPSNFKILNHCILVVISTMFPKEFTPEAHVSLDKFLSGVALALAERYR
9       _SLTAKDKSVVKAFWGKISGKADVVGAEALGRDKMLTAYPQTKTYFSHWADLSPGSGPVKKHGGIIMGAIGKAVGLMDDLVGGMSALSDLHAFKLRVDPGNFKILSHNILVTLAIHFPSDFTPEVHIAVDKFLAAVSAALADKYR

  1 -> HBA1_ARCGA      2 -> HBA1_BOSMU      3 -> HBA1_GALCR      4 -> HBA1_IGUIG
  5 -> HBA1_LEMVA      6 -> HBA1_NOTAN      7 -> HBA1_NOTCO      8 -> HBA1_PLEWA
  9 -> HBA1_SALIR     10 -> HBA1_TACAC     11 -> HBA1_TADBR     12 -> HBA1_TORMA
 13 -> HBA1_TRENE     14 -> HBA1_TRICR     15 -> HBA1_UROHA
```

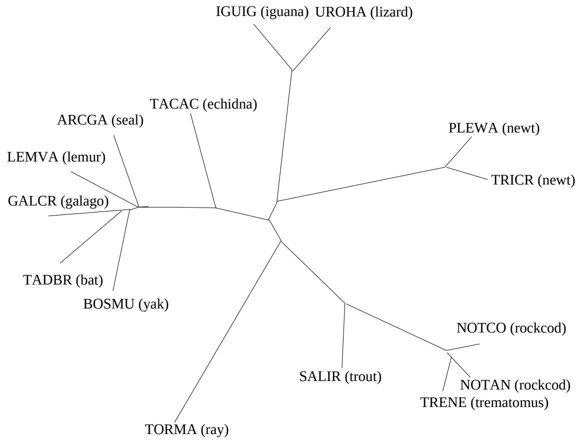

Figure 10: Hemoglobin alpha 1 constructed with new method

6 Discussion

We have introduced a new tree construction method that constructs the tree with the minimum score for a given set of sequences. By using a Traveling Salesman approach to find a circular tour and the sequence data, we can construct the the tree with the minimum score in $O(n \log(n))$ time. We can guarantee that the reconstructed tree is correct, if the error of the distance measurement does not exceed $\frac{x}{2}$, where x is the shortest edge. The probability to get a wrong circular order is very low, as *all* other correct circular orders have to be worse than the wrong order. If the errors of the distance measurements are very large, then better results can be achieved via dynamic programming. But we think it still is important to use different tree construction algorithms and to compare the results and decide in each case which tree is suited best.

References

[1] R. Agarwala, V. Bafna, M. Farach, B. Narangyan, M. Paterson, and M. Thorup. On the approximability of numerical taxonomy: fitting distances with trees. *SIAM J. Comput.*, pages 365 – 72, 1996.

[2] Steven A. Benner, Mark A. Cohen, and Gaston H. Gonnet. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. *J. Molecular Biology*, 229:1065–1082, 1993.

[3] Humberto Carillo and David. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48(5):1073–1082, 1988.

[4] A. Cavalli-Sforza, L. ad Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution*, 32:233 –57, 1967.

[5] Margaret O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model for evolutionary change in proteins. In Margaret O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352. 1978.

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

[7] G. Estabrook, C. Johnson, and F. McMorris. An idealized concept of the true cladistic character. *Math. Biosciences*, 23:263 – 72, 1975.

[8] G. Estabrook, C. Johnson, and F. McMorris. A mathematical foundation for the analysis of cladistic character compatibility. *Math. Biosciences*, 29:181 – 87, 1976.

[9] J. Felsenstein. Maximum-likelihood estimation of evolutionary trees from continuous characters. *Amer. J. Human Genetics*, 25:471–492, 1973.

[10] J. Felsenstein. The number of evolutionary trees. *Systematic Zoology*, 27:401 – 410, 1978.

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

[12] W.M. Fitch and E. Margoliash. The construction of phylogenetic trees. *Science*, 155:279 – 84, 1967.

[13] Gaston H. Gonnet and Steven A. Benner. Probabilistic ancestral sequences and multiple alignments. In *Fifth Scandinavian Workshop on Algorithm Theory, Reykjevik July 1996*, 1996.

[14] Gaston H. Gonnet, Mark A. Cohen, and Steven A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[15] Gaston H. Gonnet and Chantal Korostensky. Evaluation measures of multiple sequence alignments. *J. Comp. Biol.*, 1999. submitted.

[16] O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.

[17] M. Groetschel and O. Holland. Solution of large-scale symmetric traveling salesmand problems. *Math. Programming*, pages 141 – 202, 1991.

[18] S. Gupta, J. Kececioglu, and A. Schaffer. Making the shortest-paths approach to sum-of-pairs multiple sequence alignment more space efficient in practice. *Proc. 6th Symp. on Combinatorial Pattern Matching*, pages 128 – 43, 1995.

[19] Sandeep K. Gupta, John Kececioglu, and Alejandro A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. In *J. Computational Biology*, 1996.

[20] J. Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bull. Math. Biol.*, 51:597 – 603, 1989.

[21] P. Hogeweg and P. Hesper. The alignmen of sets of sequences and the construction of phylogenetic trees: an integrated method. *J. Mol. Evol.*, 20:175 –86, 1988.

[22] T. Jiang and L. Wang. On the complexity of multiple sequence alignment. *J. Comp. Biol.*, 1:337 – 48, 1994.

[23] D.S. Johnson. More approaches to the travelling salesman guide. *Nature*, 330:525, December 1987.

[24] D.S. Johnson. Local optimization and the traveling salesman problem. In *Proc. 17th Colloq. on Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 446 – 461, Berlin, 1990. Springer Verlag.

[25] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. *Proc. 4th Symp. on Combinatorial Pattern Matching*, pages 106 – 19, 1993.

[26] Ch. Korostensky and G. Gonnet. Gap heuristics and tree construction using gaps. *Technical Report, ETH Zuerich*, 321, 1999.

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

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

[29] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[30] Jeffrey Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *J. Molecular Biology*, 34:3–16, 1993.

7 Appendix

7.1 Search Space and Error Analysis

For a tree with n different leaves, there are in the order of $N(n) = 3{*}5{*}..{*}(2n{-}5) = \prod_{k=1}^{n-3} 2{*}k{+}1$ different tree topologies [10]. If a circular order is given, there are $O(2^{n-3})$ different trees that can be built with that order. So a circular order alone reduces the search space. To get an intuition we build a small table (see Table 7.1).

n	no circular order	with circular order
4	3	2
10	2027025	128
20	10^{20}	131000

Figure 11: Number of possible trees that can be built without and with a circular order C

We have seen that the probability to get a wrong TSP order is very unlikely. But given a correct TSP order, there are still 2^{n-3} possible different trees. So far we know that if 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.

As a next step we want to determine the *probability* to get a wrong tree for a given normal distribution of the distance measurement. Our goal is to express the probability to get a wrong edge in terms of the shortest edge x.

At each connection step, we connect leaf s_i and s_{i+1} with the smallest $\delta(s_i)$. Now assume that $\delta(s_i)$ has some error s.t we connect the wrong leaves, and one of the correct choices would have been $\delta(s_j)$. We know that if there was no error, then

$$\delta(s_i) = d(s_{i-1}, s_i) + d(s_{i+1}, s_{i+2}) - d(s_{i-1}, s_{i+1}) - d(s_i, s_{i+2}) = -2 \cdot x$$

We assume that the ϵ_{ij} follow a normal distribution with standard deviation σ. Then the error ϵ_d also follows a normal distribution σ_d, where the standard deviation is the sum of the individual standard deviations: $\sigma_d = 4\sigma$ (see Figure 12).

In order to get a wrong connection, the following condition must be true: for all $j \neq i$, where $\delta(s_j)$ is a correct connection, $\epsilon_{dd} = \delta(s_i) - \delta(s_j) \leq 2 \cdot x$. This again is a normal distribution, with $\sigma_{dd} = 8\sigma$. ϵ_{dd} must be $\leq 2 \cdot x$ for *all* possible valid connections. For a balanced tree with n leaves (tree depth $log(n)$), there are at most $\frac{n}{2}$ valid connections. In the worst case, if the tree is extremely unbalanced, there is only 1 correct connection at each step. Let $P_{dd}(\sigma)$ be the probability that one $\epsilon_{dd} \leq 2 \cdot x$ for a given standard deviation σ :

$$P_{dd}(\sigma) = \frac{2}{\sqrt{2\pi}} \int_{z=\frac{x}{4\sigma}}^{inf} e^{\frac{1}{2}z^2}$$

So the overall probability to get a wrong connection is:

$P_w(\sigma) = P_{dd}(\sigma)^{\frac{n}{2}}$ in the best case, and

$P_w(\sigma) = P_{dd}(\sigma)$ in the worst case.

To illustrate the probability to get the shortest edge x wrong we build a table (see Table 14) for an example tree of 16 leaves. So if the standard deviation σ of the error ϵ of the distance

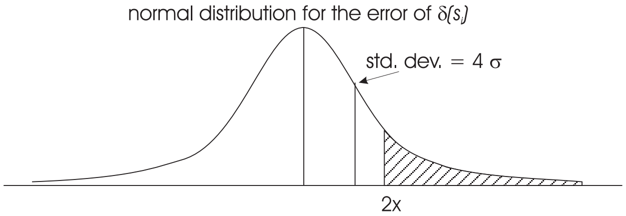

Figure 12: Normal distribution for the error of $\delta(s_i)$

measurement is 50% of the shortest edge x, then the probability to get this edge wrong is 2% in the best, and 60% in the worst case. Hence for very large errors, a dynamic programming approach is more appropriate.

$\frac{\sigma}{x}$	$P_w(\sigma)$ best case	$P_w(\sigma)$ worst case
0.1	$10^{-15}\%$	1%
0.5	2%	60%
1	17%	80%

Figure 13: Probability to get the shortest edge x wrong depending on the standard deviation σ of the error ϵ_{ij} of the distance measurements in terms of x.

7.2 Dynamic Programming Approach

The dynamic programming version of our algorithm is very simple: instead of connecting the two leaves with the smallest error ϵ, the best k connections are chosen (where k is a user specified parameter). At the end, the tree with the overall smallest error ϵ_t is chosen. If the number of trees at the end is t, then the probability get an edge x wrong is much lower: in order to get a wrong connection, *all* k connections have to be wrong:

$P_w(\sigma, k) = P_{dd}(\sigma)^{k\frac{n}{2}}$ in the best case, and

$P_w(\sigma, k) = P_{dd}(\sigma)^k$ in the worst case.

To show the difference we build the same table again, for $k = 3$:

$\frac{\sigma}{x}$	$P_w(\sigma, k)$ best case	$P_w(\sigma, k)$ worst case
0.1	$10^{-45}\%$	$10^{-6}\%$
0.5	$10^{-5}\%$	23%
1	0.5%	50%

Figure 14: Probability to get the shortest edge x wrong depending on the standard deviation σ of the error ϵ_{ij} of the distance measurements in terms of x with dynamic programming ($k = 3$).

If we would simply keep all k trees at each step, the number of resulting would grow exponentially. 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 in the worst case, if the tree is completely unbalanced (if the tree depth is almost n). When the tree is balanced (tree depth is about $log(n)$), then we will get only *one* tree in the best case, and some number between 1 and 2^{n-3} (if most connections are wrong), depending on the error.

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

To identify equal trees during the dynamic programming construction process, a unique number is created for each tree. The number is added to a hash table. Any new tree is compared to the hash table and only kept if it is not already in the list.