# A fast algorithm for optimal alignment between similar ordered trees

Jansson, Jesper; Lingas, Andrzej

*Published in:*
Combinatorial Pattern Matching / Lecture notes in computer science

*DOI:*
10.1007/3-540-48194-X_22

2001

# A Fast Algorithm for Optimal Alignment between Similar Ordered Trees

Jesper Jansson and Andrzej Lingas

Dept. of Computer Science, Lund University
Box 118, 221 00 Lund, Sweden
{Jesper.Jansson,Andrzej.Lingas}@cs.lth.se

**Abstract.** We present a fast algorithm for optimal alignment between two similar ordered trees with node labels. Let $S$ and $T$ be two such trees with $|S|$ and $|T|$ nodes, respectively. An optimal alignment between $S$ and $T$ which uses at most $d$ blank symbols can be constructed in $O(n \log n \cdot (maxdeg)^4 \cdot d^2)$ time, where $n = \max\{|S|, |T|\}$ and $maxdeg$ is the maximum degree of a node in $S$ or $T$. In particular, if the input trees are of bounded degree, the running time is $O(n \log n \cdot d^2)$.

## 1 Introduction

Let $R$ be a rooted tree. $R$ is called a *labeled tree* if each node of $R$ is labeled by a symbol from a fixed finite set $\Sigma$. $R$ is an *ordered tree* if the left-to-right order among siblings in $R$ is given.

The problem of determining the similarity between two labeled trees occurs in several different areas of computer science. For example, in computational biology, methods for measuring the similarity between ordered labeled trees of bounded degree can be used in the comparison of RNA secondary structures [2,4,9]. The problem also occurs in evolutionary trees comparison, organic chemistry, pattern recognition, and image clustering [2,4,8,12].

The similarity between two labeled trees can be defined in various ways analogous to the ways of defining the similarity between two sequences [5,7,8]. For example, one can look for the largest maximum agreement subtree, the largest common subgraph, the smallest common supertree, the minimum tree edit distance *etc.* [2,3,4,7,10,12].

In [2], Jiang *et al.* generalized the concept of an alignment between sequences to include labeled trees as follows. An *insert operation* on a labeled tree adds a new node $u$ which is labeled by a blank symbol $\lambda$ (space) not belonging to $\Sigma$. The operation either (1) turns the current root of the tree into a child of $u$ and lets $u$ become the new root, or (2) makes $u$ the parent of a subset of (if the tree is unordered) or consecutive subsequence of (if the tree is ordered) children of an existing node $v$, and $u$ a child of $v$. An *alignment between two labeled trees* is obtained by performing insert operations on the two trees so they become isomorphic when labels are ignored, and then overlaying the first augmented tree on the other one. The *score* of the alignment is the sum of the scores of

A. Amir and G.M. Landau (Eds.): CPM 2001, LNCS 2089, pp. 232–240, 2001.

all matched pairs of labels, where the score of a pair of labels is defined by a given function $\mu : (\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\}) \to \mathbb{Z}$. An *optimal alignment* between a pair of labeled trees is an alignment between them achieving the highest possible score[1]. See Fig. 1 for an example.

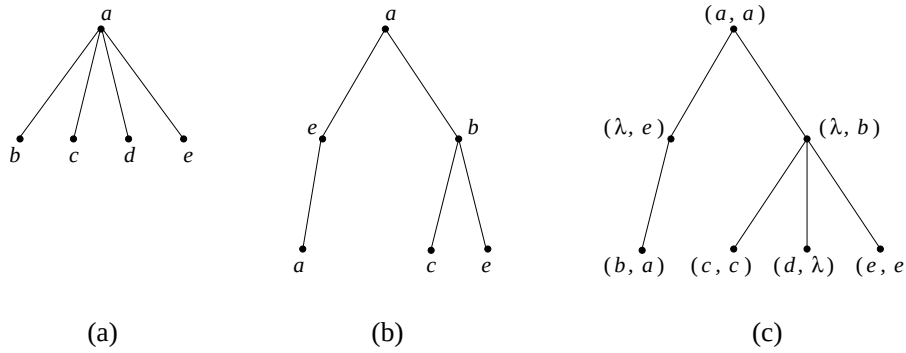


**Fig. 1.** Example: Let $\Sigma = \{a, b, c, d, e\}$ and define the scoring function $\mu$ as $\mu(x, x) = 3$, $\mu(x, y) = -1$, $\mu(x, \lambda) = \mu(\lambda, x) = -2$, $\mu(\lambda, \lambda) = -2$ for all $x, y \in \Sigma$ with $x \neq y$. Then the score of the alignment in (c) of the two ordered trees shown in (a) and (b) is equal to 2.

Jiang *et al.* presented an $O(n^2(maxdeg)^2)$-time algorithm for computing an optimal alignment between two ordered trees with node labels, where $n$ stands for the maximum number of nodes in one of the input trees, and $maxdeg$ for the maximum degree of a node in the input trees. They also provided a polynomial time algorithm for finding an optimal alignment of two unordered trees in case $maxdeg = O(1)$, and showed the latter problem to be MAX SNP-hard in general.

Inspired by the known fast method for an optimal alignment between similar sequences (see Section 3.3.4 in [8]), we give a fast algorithm for optimal alignment between two similar ordered trees with node labels. If there is an optimal alignment between the two input ordered trees which uses at most $d$ blank symbols then our algorithms runs in $O(n \log n \cdot (maxdeg)^4 \cdot d^2)$ time. Hence, under a natural assumption on the scoring, if the maximum possible score of an alignment between the two trees is $O(d)$ apart from the score of a perfect alignment of the first tree with itself then the algorithm runs in $O(n \log n \cdot (maxdeg)^4 \cdot d^2)$ time. In particular, if both trees are of bounded degree the running time reduces to $O(n \log n \cdot d^2)$.

[1] In fact, Jiang *et al.* consider the arithmetically complementary distance measure of an alignment which is the subject of minimization [2].

## 2 *d*-Relevant Pairs

The general idea of our algorithm is to modify the dynamic programming algorithm of Jiang *et al.* to only consider what we call $d$-relevant pairs of subtrees or subforests. In order to introduce our slightly technical concept of $d$-relevance we need the following definition.

**Definition 1.** *For an ordered tree $T$ and a node $u$ of $T$, $T[u]$ denotes the ordered subtree of $T$ rooted at $u$. When $u$ is not the root of $T$, $\overline{T[u]}$ stands for the ordered subtree of $T$ resulting from removing $T[u]$ and the edge between $u$ and the parent of $u$ from $T$. Next, $L(T, u)$ denotes the set of leaves in $T$ that are to the left of the leaves of $T[u]$. The number of nodes in $T$ is denoted by $|T|$ and the cardinality of $L(T, u)$ by $|L(T, u)|$.*

Now, we are ready to introduce the concept of $d$-relevant pairs of subtrees as well as those of $d$-descendant and $d$-ancestor.

**Definition 2.** *Let $d$ be a positive integer. For two ordered trees $S, T$ containing nodes $u$ and $v$ respectively, the pair of subtrees $(S[u], T[v])$ is called* $d$-relevant *if $||S[u]| - |T[v]|| \leq d$ and $||L(S, u)| - |L(T, v)|| \leq d$. For a node $w$ of $T$, $T[w]$ is called a* $d$-descendant *of $T[v]$ if $w$ is a descendant of $v$ in $T$ and $|T[v]| - |T[w]| \leq d$. Symmetrically, $T[w]$ is called a* $d$-ancestor *of $T[v]$ if $w$ is an ancestor of $v$ in $T$ and $|T[w]| - |T[v]| \leq d$.*

The definition of $d$-relevance immediately yields the following lemma.

**Lemma 1.** *Let $S$, $T$ be two labeled ordered trees, and let $u$, $v$ be two nodes in $S$ and $T$ respectively. If there is an alignment between $S$ and $T$ which uses at most $d$ blank symbols (spaces) and consists of an alignment between $S[u]$ and $T[v]$ and an alignment between $\overline{S[u]}$ and $\overline{T[v]}$ then $(S[u], T[v])$ is $d$-relevant for $S$ and $T$.*

The next three lemmas will be useful for bounding the number of $d$-relevant pairs from above.

**Lemma 2.** *If the pairs $(S[u], T[v])$ and $(S[u], T[w])$ are $d$-relevant for two ordered trees $S$ and $T$, and $w$ is a descendant (or, ancestor) of $v$ in $T$ then $T[w]$ is a $2d$-descendant (or, $2d$-ancestor) of $T[v]$.*

*Proof.* Since $(S[u], T[v])$ is $d$-relevant, it holds that $||S[u]| - |T[v]|| \leq d$. Suppose that $T[w]$ is not a $2d$-descendant of $T[v]$ in $T$, i.e., $|T[v]| - |T[w]| > 2d$. Then we have $|S[u]| - |T[w]| = |S[u]| - |T[v]| + |T[v]| - |T[w]| > -d + 2d = d$, which contradicts the $d$-relevance of $(S[u], T[w])$. □

**Lemma 3.** *For a node $u$ of an ordered tree $S$, the number of $d$-ancestors of $S[u]$ is at most $d$.*

*Proof.* Assume that the number of $d$-ancestors of $S[u]$ is greater than $d$. By the pigeonhole principle there exists a $d$-ancestor $S[u']$ whose root $u'$ is located at distance greater than $d$ from $u$. But then $|S[u']| - |S[u]| > d$, which is a contradiction. □

**Lemma 4.** *Let $\{(S[u], T[v_i])\}_{i=0}^{l}$ be a sequence of distinct $d$-relevant pairs in two ordered trees $S, T$ such that for any $0 \leq i, j \leq l$, $v_i$ is not a descendant of $v_j$. Then, $l \leq 2d$ holds.*

*Proof.* We may assume w.l.o.g. that the sequence is ordered according to the left-right order in $T$. Since $(S[u], T[v_l])$ is $d$-relevant, $||L(S,u)| - |L(T,v_l)|| \leq d$ holds. On the other hand, we have $|L(T,v_l)| - |L(T,v_0)| \geq l$. Hence, if $l > 2d$ then $|L(S,u)| - |L(T,v_0)| > d$, which contradicts the $d$-relevance of $(S[u], T[v_0])$. □

By combining the three lemmas above, we obtain an upper bound on the number of $d$-relevant pairs of subtrees.

**Theorem 1.** *For two ordered trees $S, T$ and a node $u$ of $S$, the number of distinct $d$-relevant pairs of subtrees in which $u$ participates is $O(d^2)$. Consequently, there are $O(|S| \cdot d^2)$ $d$-relevant pairs of subtrees for $S$, $T$.*

*Proof.* Let $\{(S[u], T[v_i])\}_{i=0}^{l}$ be a maximal sequence of distinct $d$-relevant pairs of subtrees for two ordered trees $S, T$ such that for each $0 \leq i \leq l$ there is no $d$-relevant pair $(S[u], T[v])$, where $v$ is a descendant of $v_i$. It follows from Lemma 2 that for each $d$-relevant pair $(S[u], T[w])$, it either belongs to the sequence or $T[w]$ is a $2d$-ancestor of a member in the sequence. Hence, the number of $d$-relevant pairs in which $u$ participates is at most $(2d+1) \cdot (l+1)$ by Lemma 3. Now, it is sufficient to observe that $l$ cannot exceed $2d$ by Lemma 4. □

### 2.1 *d*-Relevant Pairs of Subforests

The dynamic programming algorithm of Jiang *et al.* recursively computes scores not only between pairs of subtrees of the input trees but also between some pairs of subforests of the trees. Therefore, in order to modify this algorithm, we need to generalize the concept of $d$-relevance for pairs of subtrees to include the aforementioned pairs of subforests. For this purpose, we introduce the following technical notations.

**Definition 3.** *For an ordered tree $S$ and a node $u$ of $S$, let $d_u$ be the degree of $u$ and denote the children of $u$ by $u_1, ..., u_{d_u}$, according to their left-to-right order. $S(u,i,j)$ refers to the ordered forest $S[u_i], ..., S[u_j]$, and $S(u)$ is short for $S(u, 1, d_u)$.*

Thus, $S(u)$ is the *complete ordered forest* obtained by removing $u$ and all edges incident to $u$ from $S[u]$. Also note that $S(u,i,i) = S[u_i]$.

**Definition 4.** *Let $S(u,i,j)$ be an ordered forest in an ordered tree $S$. $\overline{S(u,i,j)}$ stands for the ordered subtree of $S$ obtained by removing $S(u,i,j)$ and all edges incident to $S(u,i,j)$ from $S$. $L(S(u,i,j))$ denotes the set of leaves in $S$ that are to the left of the leaves of $S(u,i,j)$. The number of nodes in $S(u,i,j)$ is denoted by $|S(u,i,j)|$ and the cardinality of $L(S(u,i,j))$ by $|L(S(u,i,j))|$.*

Now, we are ready to generalize the concept of $d$-relevance as well as those of $d$-descendant and $d$-ancestor for pairs of nodes inducing full subtrees to include pairs of subforests of the form $(S(u,i,j),T(v,k,l))$.

**Definition 5.** *Let $d$ be a positive integer. For two ordered trees $S,T$ containing nodes $u$ and $v$ respectively, the pair of ordered subforests $(S(u,i,j),T(v,k,l))$ is called* $d$-relevant *if $||S(u,i,j)|-|T(v,k,l)|| \leq d$ and $||L(S(u,i,j))|-|L(T(v,k,l))|| \leq d$. For a node $w$ of $T$, $T(w,k',l')$ is called a* $d$-descendant *of $T(v,k,l)$ if $w$ is a descendant of $v$ in $T$ and $||T(w,k',l')| - |T(v,k,l)|| \leq d$. Symmetrically, $T(w,k',l')$ is called a* $d$-ancestor *of $T(v,k,l)$ if $w$ is an ancestor of $v$ in $T$ and $||T(w,k',l')| - |T(v,k,l)|| \leq d$.*

The definition of $d$-relevance of subforests immediately yields the following lemma analogous to Lemma 1.

**Lemma 5.** *Let $S,T$ be two labeled ordered trees, and let $S(u,i,j)$ and $T(v,k,l)$ be ordered forests in $S$ and $T$ respectively. If there is an alignment between $S$ and $T$ which uses at most $d$ blank symbols (spaces) and consists of an alignment between $S(u,i,j)$ and $T(v,k,l)$ and an alignment between $\overline{S(u,i,j)}$ and $\overline{T(v,k,l)}$ then $(S(u,i,j),T(v,k,l))$ is $d$-relevant for $S$ and $T$.*

The next three lemmata will be useful for bounding the number of $d$-relevant pairs of subforests from above. Their proofs are analogous to the corresponding proofs of Lemmata 2–4.

**Lemma 6.** *If the pairs $(S(u,i,j),T(v))$ and $(S(u,i,j),T(w))$ are $d$-relevant for two ordered trees $S,T$ and $w$ is a descendant (or, ancestor) of $v$ in $T$ then $T(w)$ is a $2d$-descendant (or, $2d$-ancestor) of $T(v)$.*

**Lemma 7.** *For a node $v$ of an ordered tree $T$, the number of $d$-ancestors of the form $T(w)$ of a forest $T(v)$ is at most $d$.*

**Lemma 8.** *Let $\{(S(u,i,j),T(v_q)\}_{q=0}^{l}$ be a sequence of distinct $d$-relevant pairs in two ordered trees $S,T$ such that for any $0 \leq q', q'' \leq l$, $v_{q'}$ is not a descendant of $v_{q''}$. Then, $l \leq 2d$ holds.*

By combining the three lemmas above, we obtain an upper bound on the number of $d$-relevant pairs $(S(u),T(v,k,l))$ and $(S(u,i,j),T(v))$ as in Theorem 1.

**Theorem 2.** *For two ordered trees $S,T$ and a node $u$ of $S$, the number of distinct $d$-relevant pairs of the form $(S(u,i,j),T(v))$ is $O(d^2(\deg(S))^2)$. Symmetrically, for a node $v$ of $T$, the number of distinct $d$-relevant pairs of the form $(S(u),T(v,k,l))$ is $O(d^2(\deg(T))^2)$. Consequently, there are $O(n \cdot d^2(maxdeg)^2)$ $d$-relevant pairs of the form $(S(u),T(v,k,l))$ or $(S(u,i,j),T(v))$ for $S,T$, where $n = \max\{|S|,\ |T|\}$.*

## 3 Constructing the *d*-Relevant Pairs

The test for $d$-relevance for a pair of subtrees can easily be accomplished in constant time after appropriate preprocessing. However, in order to speed up the at least quadratic algorithm of Jiang *at al.*, we cannot afford testing each possible subtree pair for $d$-relevance. Instead, we proceed as follows.

First, we compute all vectors $(|T[v]|, |L(T, v)|)$, where $v \in T$. This can be done in linear total time by using the Eulerian tour technique from [11]. We insert the vectors into a standard range search data structure, e.g., a layered range tree [6]. The construction of the data structure takes $O(|T| \cdot \log |T|)$ time. Then, for all $u$ in $S$ we compute the vectors $(|S[u]|, |L(S, u)|)$ in linear total time in the same way as above. For each $u$ in $S$, we query the data structure with the square centered at $(|S[u]|, |L(S, u)|)$ having side length $2d$. Each query takes $O(\log |S| + r)$ time, where $r$ is the number of reported vectors. Since each of the returned vectors is in one-to-one correspondence with a node $v$ such that the pair $(u, v)$ is $d$-relevant, $r = O(d^2)$ holds by Theorem 1.

Putting everything together, we obtain the following theorem.

**Theorem 3.** *For two ordered trees on at most $n$ nodes each and a non-negative integer $d$, all $d$-relevant pairs of subtrees can be reported in $O(n(\log n + d^2))$ time.*

We can use the same technique to precompute all pairs of $d$-relevant subforests. In fact, for our purposes it is sufficient to report all pairs of $d$-relevant subforests where at least one of the subforests is complete, i.e., is of the form $S(u)$ or $T(v)$. To report all $d$-relevant pairs of the form $(S(u), T(v, k, l))$, the number of vectors to insert into the layered range tree is $O(|T|(maxdeg)^2)$ since $O((maxdeg)^2)$ ordered forests of the form $T(v, k, l)$ originate from each node $v$ in $T$. Thus, the construction time becomes $O(|T| \cdot (maxdeg)^2 \cdot \log(|T| \cdot (maxdeg)^2)) = O(n \cdot (maxdeg)^2 \cdot \log n)$. The number of queries to the data structure is $O(|S|)$, and the query time is $O(\log(|T| \cdot (maxdeg)^2) + r) = O(\log n + r)$ time, where the sum of the $r$'s over $S$ is $O(nd^2 \cdot (maxdeg)^2)$ by Theorem 2. The reporting of $d$-relevant pairs of the form $(S(u, i, j), T(v))$ can be done symmetrically within the same (in terms of $n$) preprocessing and query time bounds.

Summing up, we obtain:

**Theorem 4.** *For two ordered trees on at most $n$ nodes each and a non-negative integer $d$, all $d$-relevant pairs of subforests, where at least one subforest is complete, can be reported in $O(n \cdot (maxdeg)^2 \cdot (\log n + d^2))$ time.*

## 4 The Fast Algorithm

Our fast algorithm for an optimal alignment between two ordered trees works under the assumption that there is an optimal alignment between the trees $S$, $T$ which uses at most $d$ blank symbols (spaces). First, we compute all $d$-relevant pairs of subtrees of $S, T$ as described in Section 3. As each $d$-relevant pair is reported, we insert it into a balanced binary search tree $\mathcal{B}_1$. Next, all

$d$-relevant pairs of subforests in which at least one subforest is complete are computed and inserted into a balanced binary search tree $\mathcal{B}_2$. According to Theorems 1 and 2, there are $O(nd^2(maxdeg)^2)$ $d$-relevant pairs of subtrees or subforests where at least one subforest is complete, so this preprocessing takes $O(n\cdot(maxdeg)^2\cdot(\log n+d^2)+n\cdot d^2(maxdeg)^2)\log(n\cdot d^2(maxdeg)^2)) = O(n\log n\cdot (maxdeg)^2\cdot d^2)$ time by Theorems 3 and 4. Then, we modify the algorithm of Jiang *et al.* recursively evaluating the score values (see [2]) solely for $d$-relevant pairs of subtrees or pairs of subtrees where one of the subtrees is empty.

This evaluation involves also recursive evaluation of the score values for $d$-relevant pairs of subforests where one of the forests is complete or empty. In fact, the recursive procedures in [2] include also intermediate terms with the scores values for pairs of subforests when none of the forests is complete or empty. However, these intermediate terms are eliminated by the composition of the aforementioned procedures, resulting in recursive formulas for the score values expressed in the form of maximum of some sums of score values for pairs of smaller subtrees or subforests where at least one of the subforests is complete or empty. Whenever the left handside is $d$-relevant in the application of such a formula, the components of the sum on the right hand side yielding the maximum, with the exception of the scores for the pairs including an empty subtree or subforest, also have to be $d$-relevant. Therefore, before an application of such a formula to an evaluation of a $d$-relevant pair, we simply test each of the components of the sums on the right handside, which is not a score for pair containing an empty subtree or subforest, for membership in $\mathcal{B}_1$ or $\mathcal{B}_2$. Such a membership query takes $O(\log n)$ time. If the test is positive we fetch the score value for the argument pair which should be evaluated by this time, otherwise we set that score value to minus infinity. The score values for pairs containing an empty subtree or subforest can be trivially precomputed in time $O(|S|+|T|)$. We conclude that the cost of determining the score for a $d$-relevant pair on the left handside of such a recursive formula on the basis of the scores for $d$-relevant pairs occuring on its right handside does not exceed the cost of determining the scores for this pair on the basis of the scores of pairs occuring on the right handside in the algorithm of Jiang *et al.* multiplied by $O(\log n)$.

Jiang *et al.* show that the cost of determining the score for a pair of subtrees or subforests by using the aforementioned formulas and already computed scores for pairs of smaller subtrees or subforests is $O(deg(z)\cdot(maxdeg)^2)$, where $z$ is a node in $S$ or $T$ which is either the root of the first subtree or the second subtree, or the parent of the roots of the trees in the first forest or the second forest. Hence, the corresponding cost for $d$-relevant pairs in our modification of this algorithm is $O(deg(z)\cdot(maxdeg)^2\cdot\log n)$. By Theorems 1 and 2, for a given node $z$ in $S$ or $T$, there are $O(d^2(maxdeg)^2)$ $d$-relevant pairs of subtrees of the form $(S[z],T[v])$ or $(S[u],T[z])$, or subforests of the form $(S(z,i,j),T[v])$ or $(S(u),T(z,l,k))$. Hence, our modified algorithm runs in $O(\sum_{z\in S\cup T} deg(z)\cdot (maxdeg)^2\cdot\log n\cdot d^2(maxdeg)^2)$ time, i.e., $O(nd^2(maxdeg)^4\log n)$ assuming the preprocessing has been done.

**Theorem 5.** *An optimal alignment of two ordered trees which uses at most $d$ blank symbols can be constructed in $O(n \log n \cdot (maxdeg)^4 \cdot d^2)$ time.*

Under the natural assumption that the score of a pair including at least one blank symbol is negative and by $\Omega(1)$ smaller than that of a pair consisting of two identical symbols, we immediately obtain the following lemma.

**Corollary 1.** *An optimal alignment of two ordered trees whose score is $O(d)$ apart from the score of the perfect alignment between the first tree and its copy can be constructed in $O(n \log n \cdot (maxdeg)^4 \cdot d^2)$ time.*

**Corollary 2.** *An optimal alignment of two ordered trees of bounded degree whose score is $O(d)$ apart from the score of the perfect alignment between the first tree and its copy can be constructed in $O(n \log n \cdot d^2)$ time.*

## 5 Final Remarks

An optimal alignment between two sequences whose score is at most $d$ apart from that of a perfect alignment between the first sequence and its copy can be constructed in $O(nd)$ time [8]. Since a sequence can be interpreted as a line ordered tree with node labels, a natural question arises: is it possible to lower the time complexity of our method, especially the exponent 2 of $d$?

Our method does not seem to generalize to include unordered trees directly. Simply, the proof of Lemma 4 relies on the ordering of the trees (i.e., on the sets $L(\ ,\ ))$. It is an interesting open problem whether a substantial speed-up in the construction of an optimal alignment between similar unordered trees of bounded degree is achievable.

In the construction of the $d$-relevant pairs we could use more sophisticated and more asymptotically efficient data structures for two dimensional range search on an integer grid [1]. However, this would not lead to an improvement of the overwhole asymptotic time complexity of our alignment algorithm.

## References

1. S. Alstrup, G.S. Brodal, T. Rauhe. New Data Structures for Orthogonal Range Searching. Proc. of 41st Annual Symposium on Foundations of Computer Science (FOCS 2000), 2000, pp. 198–207.
2. T. Jiang, L. Wang, and K. Zhang. Alignment of Trees - An Alternative to Tree Edit. Theoretical Computer Science, 143 (1995), pp. 137–148. (A preliminary version in Proc. of 5th Annual Symposium on Combinatorial Pattern Matching (CPM'94), Lecture Notes in Computer Science, Vol. 807, Springer, 1994, pp. 75–86.)
3. D. Keselman and A. Amir. Maximum agreement subtree in a set of evolutionary trees – metrics and efficient algorithms. Proc. of 35th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'94), 1994, pp. 758–769.
4. S.-Y. Le, R. Nussinov, and J.V. Maizel. Tree graphs of RNA secondary structures and their comparisons. Computers and Biomedical Research, 22 (1989), pp. 461–473.

5. P.A. Pevzner. Computational Molecular Biology: An Algorithmic Approach. The MIT Press, Cambridge, Massachusetts, 2000.
6. F. Preparata and M.I. Shamos. *Computational Geometry*. Springer-Verlag, New York, 1985.
7. D. Sankoff and J. Kruskal (Eds). *Time Warps, String Edits, and Macromolecules, the Theory and Practice of Sequence Comparison*. Addison Wesley, Reading Mass., 1983.
8. J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, 1997.
9. B. Shapiro. An algorithm for comparing multiple RNA secondary structures. Comput. Appl. Biosci. (1988) pp. 387–393.
10. K.C. Tai. The tree-to-tree correction problem. J. ACM 26 (1979), pp. 422-433.
11. R.E. Tarjan and U. Vishkin. Finding biconnected components and computing tree functions in logarithmic parallel time. SIAM Journal of Computing 14, 4 (1985), pp. 862–874.
12. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing 18, 6 (1989), pp. 1245–1262.