Applying an Edit Distance to the Matching of Tree Ring Sequences in Dendrochronology *

Carola Wenk
Institut für Informatik
Freie Universität Berlin
Takustr. 9, D-14195 Berlin
wenk@inf.fu-berlin.de

Abstract

In dendrochronology wood samples are dated according to the tree rings they contain. The dating process is a one dimensional matching in which the sequence of tree ring widths in the sample is compared to a dated master sequence. Assuming that a tree forms one ring per year a consecutive piece in the master sequence is searched which has the same length than and is similar to the sample sequence. A brute force algorithm takes $\theta(mn)$ time where n and m are the lengths of the sample and the master sequences, respectively.

Yet sometimes a tree produces no ring or even two rings in a year. A sample sequence containing this kind of inconsistencies cannot be dated correctly by the simple dating algorithm mentioned above. We therefore introduce a $\mathcal{O}(\alpha^4(m+n)+\alpha^2 mn)$ algorithm for dating such a sample sequence against an error-free master sequence. Our algorithm takes into account that the sample might contain up to α missing or double rings and suggests possible positions for these kind of inconsistencies. This is done by employing an *edit distance* as the distance measure.

1 Introduction

1.1 Dendrochronology

The tree ring structure in wood samples is important in many research areas, like for instance in archaeology, climatology, geomorphology and glaciology. The reason for that is that the growth of a tree and therefore its tree rings depend on the environmental conditions that the tree has been exposed to, so that the tree rings build an archive of these environmental conditions. The science that deals with the dating of tree rings in

*Part of a research project supported by Deutsche Forschungsgemeinschaft, grant AL 253/4-2

order to answer questions related to natural history is called dendrochronology. The name is derived from the greek words *dendron* (wood), *chronos* (time) and *logos* (the science of).

A tree ring is a growth layer that the tree forms under its bark during the vegetation period. It consists of big, thinwalled cells that are built in the beginning of the growth period and also of thin, thickwalled cells built at the end. The first type of cells ensures the food supply of the shoots, whereas the other type accounts for the stability of the stem. Since the second type of cells looks much darker than the first type it is possible to visually detect the border between two successive tree rings. In areas with an annual vegetation and winter period a tree usually adds exactly one tree ring per year.

In dendrochronology a wood sample is usually characterized by the sequence of its tree ring widths [1]. Since trees growing under similar conditions (especially climatic conditions like rainfall) build similar tree rings it is possible to successfully compare certain tree ring sequences. In fact, the usual way of dating tree ring sequences in dendrochronology is to compare the undated sequence to a dated sequence. This procedure, called *crossdating*, is a fundamental task in dendrochronology.

1.2 Crossdating

Assuming that the trees being considered have built exactly one ring each year, a crossdating can be performed by sliding the sample along the master sequence starting and ending with a certain constant minimum overlap of e.g. 50 rings. At each position a distance measure between the overlapping parts of the sequences is computed and the position yielding the best distance is considered as the correct dating position. The most common distance measures are the *t-value*, and the so-called *Gleichläufigkeitskoeffizient* (percentage of slope equivalence).

Let $x = x_0, \ldots, x_{N-1}$ and $y = y_0, \ldots, y_{N-1}$ be the two sequences being compared in one step of the algorithm. Then the t-value (*Student's t*) is defined by

$$t = r\sqrt{(\frac{N-2}{1-r^2})} \tag{1}$$

where r is the correlation coefficient

[1]Depending on the application there are also other tree ring characteristica than the width used (see e.g. [11]), but in this article we will regard tree ring widths only.

$$r = \frac{\sum_{i=0}^{N-1}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i-\bar{x})^2 \sum_{i=0}^{N-1}(y_i-\bar{y})^2}} \tag{2}$$

$$= \frac{N\sum_{i=0}^{N-1} x_i y_i - \sum_{i=0}^{N-1} x_i - \sum_{i=0}^{N-1} y_i}{\sqrt{(N\sum_{i=0}^{N-1} x_i^2 - (\sum_{i=0}^{N-1} x_i)^2)(N\sum_{i=0}^{N-1} y_i^2 - (\sum_{i=0}^{N-1} y_i)^2)}} \tag{3}$$

with the arithmetic means $\bar{x}$ and $\bar{y}$. The *Gleichläufigkeitskoeffizient Glk* is the percentage of slope equivalence of the two sequences,

$$\mathrm{Glk} = \frac{1}{N-1}\sum_{i=0}^{N-2}\chi(x_{i+1}-x_i = y_{i+1}-y_i) \tag{4}$$

$$= \frac{1}{N-1}\sum_{k=-1}^{1}\sum_{i=0}^{N-2}\chi(x_{i+1}-x_i = k)\cdot\chi(y_{i+1}-y_i = k) \tag{5}$$

with the characteristic function $\chi(a=b) = \begin{cases} 1\,, & \text{if} \quad a=b \\ 0\,, & \text{if} \quad a\neq b \end{cases}$.

A sequential computation of all distances takes $\theta(nm)$ time, where n and m are the lengths of the master and the sample sequences, respectively. Considering formula 3 a non-sequential computation of all correlation coefficients depends on the efficient calculation of the *correlation terms* $\sum x_i y_i$, since all other terms can be computed in linear time. Note that the inner sum in formula 5 is also a correlation term. Employing the Fast Fourier Transform (FFT) all such correlation terms can be computed in time $\theta((n+m)log(n+m))$ instead of the brute force $\theta(nm)$, see e.g. [6],[7],[2]. Due to the discretization of the slope of the sequences the Gleichläufigkeitskoeffizient usually gives less information than the t-value.

Since the data is very noisy it usually does not suffice to simply date the sample sequence according to the overlapping position which has the best distance. The results of the matching algorithm are always visually checked by a dendrochronologist.

Before the above described comparison can be made, the tree ring sequences have to be filtered. This so-called *standardization* process cleans the data from individual trends, which usually are long term trends, so that only general trends which occur in several tree ring sequences remain. Typically high-pass filters like the percentage of a five-year running mean or the difference of logarithms are used.

1.3 Missing and double rings

However, the assumption made above that a tree ring sequence contains exactly one value per year is not always true. First of all mistakes during the measurement of the ring widths happen, especially if the rings are very thin. But due to bad growing conditions a tree might also not build a ring around the whole stem or even not at all, which can result in a *missing ring* in the tree ring sequence. Furthermore climatic changes can cause a tree to build two rings a year, a *double ring*.

If the sequences to be compared contain missing or double rings most matching algorithms do not produce satisfying results since they do not take into account the transposition in time which is caused by a missing or a double ring. The usual approach to date a probably inconsistencies containing sample sequence against a clean master sequence is to split up the sample into shorter parts, to date each part on its own (either manually or using Cofecha [4]) and finally to manually conclude a possible position for a ring insertion or merging. Cofecha [4] is a quality control tool which checks a set of dated samples for mutual dating consistency by splitting up each sequence into small pieces and comparing these to the other sequences. This leads to a lot of information to be evaluated. The information needed to deduce a possible missing ring (i.e. when the pieces to the right of the missing ring position date all to one year later) is then available, but a missing ring is not explicitly proposed.

2 Edit distances in an α-box

2.1 A simple edit distance

Assume there are given two standardized tree ring width sequences $A = a_0, ..., a_{n-1}$ and $B = b_0, ..., b_{m-1}$, where A is subject to probably containing missing or double rings (representing the sample sequence) whereas B is known to be a clean reference sequence (representing a part of the master sequence). In order to get a notion of how much A differs from B we look for a transformation transforming A by inserting rings (which compensates a missing ring) or merging two rings to one (which compensates a double ring) into a sequence close to B. Closeness is defined by taking the sum of the squared distances. The transformations allowed are described by *transformation sequences* over the alphabet $\{I, M, N\}$ where I stands for *insert* M for *merge* and N for *identity operation*.

τ	M	M	N	I	I	N
A	$\underbrace{3\ 2}$	$\underbrace{1\ 2}$	5			3
τ(A)	5	3	5	4	4	3

Let for example be $A = 3, 2, 1, 2, 5, 3$ and a transformation sequence $\tau = MMNIIN$. The transformation is performed sequentially from left to right by merging 3 and 2 to 5 (M), merging 1 and 2 to 3 (M), not changing 5 (N), inserting a ring which is done by taking the average $\mu = \frac{5+3}{2} = 4$ of the two surrounding rings (I), inserting another ring in the same manner (I) and finally not changing the last ring 3 (N).

The *simple edit distance* $D_{simp}(A, B)$ is defined by

$$D_{simp}(A,B) = \begin{cases} \min\limits_{\tau \in \mathcal{T}_{n,m}} \sum\limits_{v=0}^{m-1} (\tau(A)_v - B_v)^2 & \text{, if } \mathcal{T}_{n,m} \neq \emptyset \\ \text{not defined} & \text{, otherwise} \end{cases} \tag{6}$$

where $\mathcal{T}_{n,m}$ contains all transformation sequences (which we identify with a transformation each) that transform a sequence of length n into a sequence of length m. We call a transformation sequence which minimizes the sum *optimal*. For a transformation $\tau \in \mathcal{T}_{n,m}$ let γ_τ be the number of merge operations, ι_τ the number of insert operations and ν_τ the number of identity operations in τ. Then from the definition of τ follows $n = 2 * \gamma_\tau + \nu_\tau$ as well as $m = \gamma_\tau + \iota_\tau + \nu_\tau = n + \iota_\tau - \gamma_\tau$. Making use of these properties it is easy to show that $\mathcal{T}_{n,m}$ is non-empty if and only if $n \leq 2m$. This notion of an edit distance is based on the edit distance for strings (see [3], [12]) or on the dynamic time warping in speech recognition (see [10], [9]) respectively.

The profit of taking the sum of the squared distances as a minimization criterion is the existence of a recurrence relation which leads to an efficient computation of the simple edit distance. Define $D_{simp}(i, j) := D_{simp}(A[0..i-1], B[0..j-1])$ to be the simple edit distance of the prefixes of A and B for $i \leq 2j$. Then the definition of the transformations by transformation sequences implies the existence of the following recurrence:

$$\begin{aligned} D_{simp}(0,0) &= 0 \\ D_{simp}(i,j) &= \min \left\{ \begin{array}{l} D_{simp}(i-2, j-1) + (a_{i-2} + a_{i-1} - b_{j-1})^2, \\ D_{simp}(i-1, j-1) + (a_{i-1} - b_{j-1})^2, \\ D_{simp}(i, j-1) + d(\mu(i-1) - b_{j-1})^2 \end{array} \right\} \end{aligned} \tag{7}$$

$$\text{for all } 0 \leq i \leq n,\ 0 \leq j \leq m \text{ with } i \leq 2j.$$

Although the transformation space is exponentially big a dynamic programming approach allows to compute $D_{simp}(A, B)$ in $\theta(nm)$ time and space. This is accomplished by sequentially filling an $(n+1) \times (m+1)$ matrix (see figure 1) in which the cell (i, j) contains the value $D_{simp}(i, j)$. The condition $i \leq 2j$ and the symmetrical condition $(n-i) \leq 2(m-j)$ cut two corners off the matrix which represent undefined or for the computation of $D_{simp}(n, m)$ unnecessary values, respectively. According to recurrence (7) a value is computed out of at most three values (see figure 2). The value $D_{simp}(A, B) = D_{simp}(n, m)$ is placed in cell (N, M).

An optimal transformation can be retrieved from the filled matrix by backtracking the performed computation. This is done by starting in cell (n, m) and recursively checking which of the three possible cells contributed its value to the examined cell (either by recalculating the sums or by checking a previously saved pointer/arrow to a cell). In this way a path of cells (or arrows, see figure 2) from cell (n, m) to cell $(0, 0)$ is constructed which obviously corresponds to a transformation sequence.

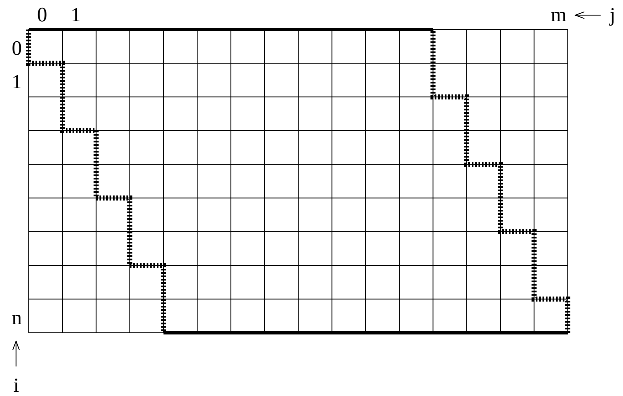

Figure 1: Matrix for the dynamic programming computation of the simple edit distance $D_{simp}(A, B)$. The left cut-off is caused by the condition $i \leq 2j$, the right by $(n - i) \leq 2(m - j)$.

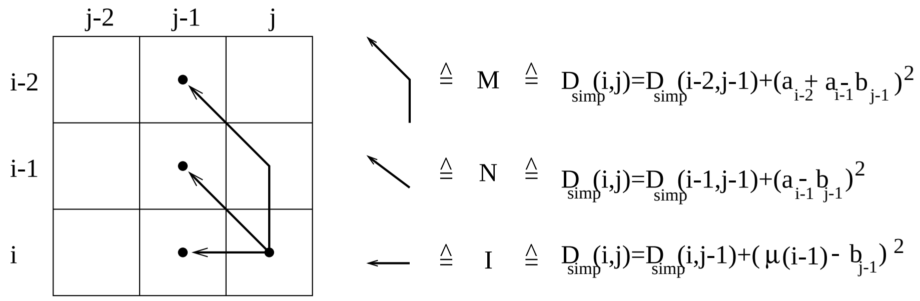

Figure 2: Dynamic programming computation of the simple edit distance according to the recurrence relation (7).

2.2 Van Deusen's edit distance

The transformation space over which the simple edit distance is minimized includes in particular transformations with many editations. Transformations like this correspond to paths in the computation matrix with many non-diagonal arrows. Since a tree ring sequence usually contains only very few missing or double rings, Van Deusen [1] reduced the transformation space by allowing only those paths in the matrix which stay inside a given strip of constant width around the diagonal starting at $(0,0)$; see figure 3.

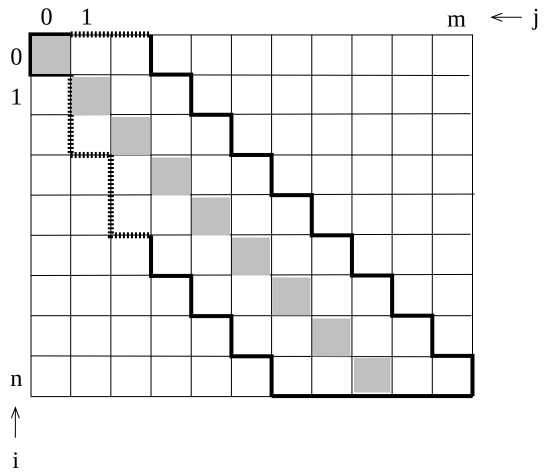

Figure 3: Van Deusen's computation matrix with a strip of width $\alpha = 2$ to each side of the diagonal. The diagonal has been shaded.

The width of a strip is given by a parameter α which denotes the width on each side of the diagonal. The in this way reduced transformation space contains only transformations in which the edit operations are locally balanced. Yet there are still transformations with many edit operations possible. For instance if the transformation sequence alternates between a merge and an insert operation the conforming transformation path still stays inside a strip of width 1 around the diagonal.

2.3 α-box edit distance

A straight forward improvement of Van Deusen's edit distance is the following notion which we will call *α-box edit distance* or *k-edit distance*. This type of edit distance has been proposed for strings by Sankoff and Kruskal [5] in 1983. Since the number of edit operations contained in an optimal transformation is supposed to be small, the idea is to regard transformations and edit distances depending on the number of edit operations.

We therefore define the k*-edit distance* as follows:

$$D(A,B,k) = \begin{cases} \min\limits_{\tau \in \mathcal{T}_{n,m,k}} \sum\limits_{v=0}^{m-1} (\tau(A)_v - B_v)^2 & \text{, if } \mathcal{T}_{n,m,k} \neq \emptyset \\ \text{not defined} & \text{, otherwise} \end{cases} \tag{8}$$

where $\mathcal{T}_{n,m,k}$ is the set of all transformation sequences transforming a sequence of length n into a sequence of length m using exactly k edit operations. Let again γ_τ be the number of merge operations, ι_τ the number of insert operations and ν_τ the number of identity operations in τ. Then from the definition of τ follow $m = \gamma_\tau + \iota_\tau + \nu_\tau = n + \iota_\tau - \gamma_\tau$ and $\iota_\tau + \gamma_\tau = k$. It is then easy to show that $\mathcal{T}_{n,m,k}$ is non-empty if and only if $m \geq k$ and $m - n = k - 2\gamma$ for a $\gamma \in \{0, \ldots, k\}$.

We define $D(i,j,k)$ to be the k-edit distance between the prefixes $A[0..i-1]$ and $B[0..j-1]$. Just as in the case of the simple edit distance the k-edit distance satisfies a recurrence relation:

$$\begin{aligned} D(0,0,0) &= 0 \\ D(i,j,k) &= \min \left\{ \begin{array}{l} D(i-2, j-1, k-1) + (a_{i-2} + a_{i-1} - b_{j-1})^2, \\ D(i-1, j-1, k) + (a_{i-1} - b_{j-1})^2, \\ D(i, j-1, k-1) + d(\mu(i-1) - b_{j-1})^2 \end{array} \right\} \end{aligned} \tag{9}$$

$$\begin{array}{l} \text{for all } \ 0 \leq i \leq n, \ 0 \leq j \leq m \\ \text{with } \ j \geq k \ \text{ and } \ j - i = k - 2\gamma \ \text{ for a } \ \gamma \in \{0, \ldots, k\}. \end{array}$$

Due to the condition $j - i = k - 2\gamma$ for a $\gamma \in \{0, \ldots, k\}$ the defined values of $D(i,j,k)$ form diagonals inside the matrix (see figures 4 and 5). For each $\gamma \in \{0, \ldots, k\}$ there is one corresponding diagonal in level k. All edit distances in one diagonal contain the same number of merge and insert operations as shown in figure 5.

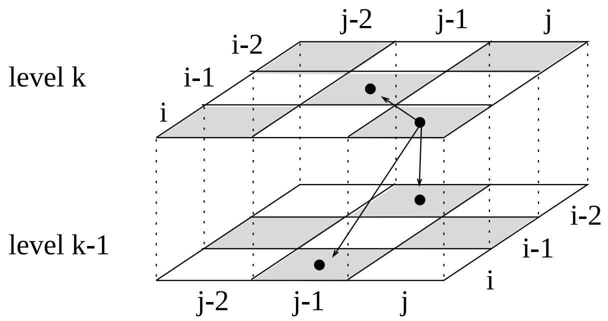

Figure 4: Dynamic programming computation of the k-edit distance according to the recurrence relation (9). A projection of the two levels onto one level results in the matrix shown in figure 2.

The α-edit distance can be computed in a dynamic programming manner in $\theta(\alpha^2 \min(n,m))$ time and space. The storage required is a part of an $(\alpha+1) \times (n+1) \times (m+1)$ box (see figure 5) in which cell (i,j,k) contains the value $D(i,j,k)$. According

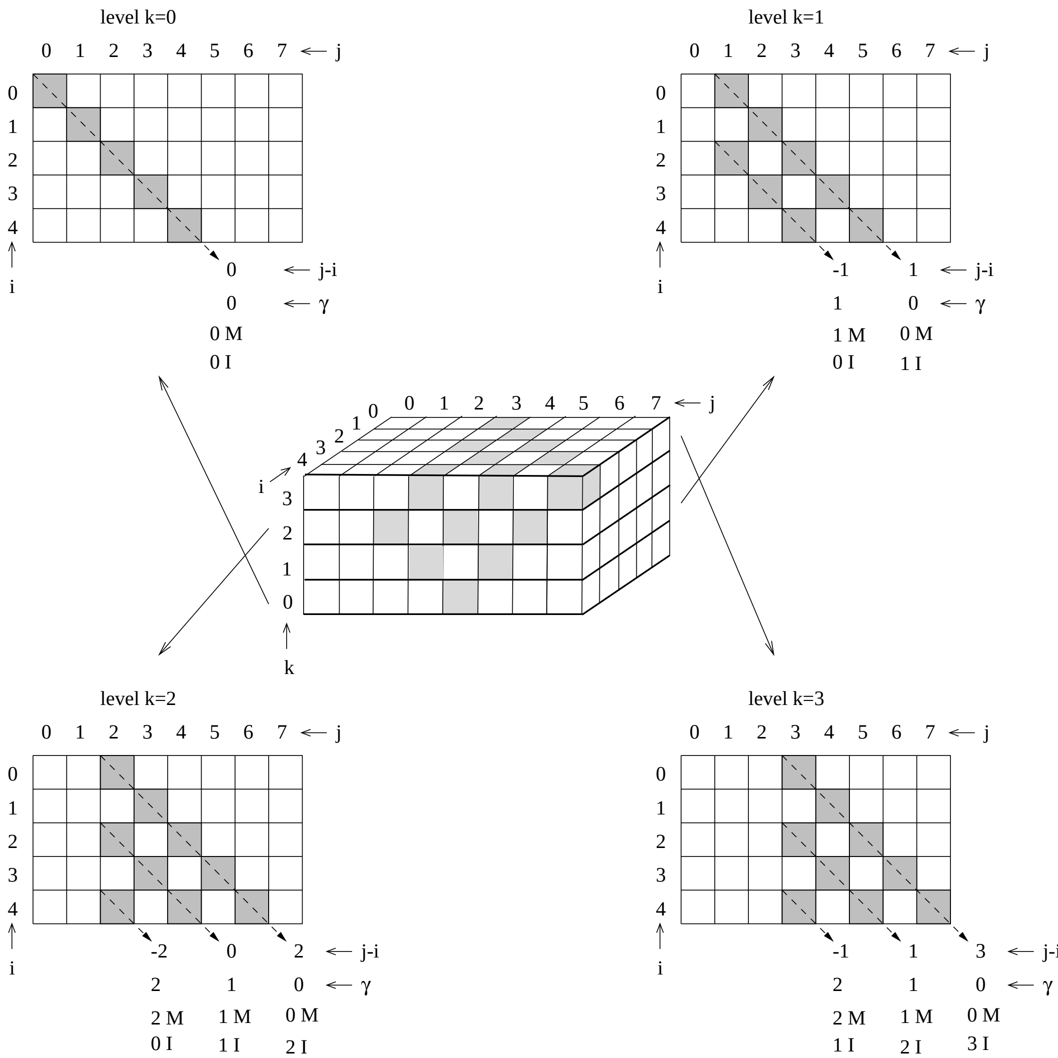

Figure 5: Dynamic programming box needed for the computation of the 3-edit distance between two sequences of length 4 and 7. The defined cells have been shaded. The number of merge (M) and insert (I) operations corresponding to the diagonals have been written next to each diagonal.

to recurrence (9) the value $D(i, j, k)$ is computed out of at most three values (see figure 4) whereby a change of the k-level is performed only in the case of an edit operation (merge or insert). The computation is carried out by filling the diagonals level by level, thereby touching each cell only a constant number of times, until finally filling cell (n, m, α). Note that this computation box (α-box) contains especially all k-edit distances between A and B with $0 \leq k \leq \alpha$. In each k-level there are at most $k + 1$ diagonals and each diagonal contains at most $\min(n, m) + 1$ cells. Therefore there are $(\min(n, m) + 1) \sum_{k=0}^{\alpha} (k + 1)$ $= (\min(n, m) + 1) \frac{(\alpha+1)(\alpha+2)}{2} = \mathcal{O}(\alpha^2 \min(n, m))$ cells to be filled.

Often an optimal transformation contains two opposite edit operations (insert-merge or merge-insert) almost successively. A pair of edit operations like this has no global effect on the edited sequence but only a local effect during the short time interval between the two edit operations. Given a threshold (e.g. 10) for the minimum number of years between two opposite edit operations, 10 cells on the diagonal after an edit operation are marked such that the opposite edit operation is not allowed when calculating the edit distances for these cells.

Although theoretically there can be several optimal transformations associated with one edit distance (this corresponds to more than one arrow pointing to one cell), we always choose exactly one optimal transformation per edit distance (or cell), since due to the real valued input data an exact equality of the sums is unlikely. Then instead of storing in each cell a pointer to the cell which contributed its value to the sum, we collapse a path of diagonal pointers to one pointer (*shortcut*) directly pointing to the position where the transformation path changes the k-level. That way a traceback of the transformation path of a cell in level k needs $\theta(k)$ time and a transformation can be saved for later use in $\theta(k)$ space.

3 Crossdating employing k-edit distances

Crossdating is usually performed by sliding the sample sequence across the master sequence and computing distance measures between the overlapping parts at each crossing position. The same is done in the algorithm presented here using the k-edit distance and therefore taking into account possibly missing or double rings. The parameter α must be specified by the user in advance. As a postprocessing step a new heuristic is applied in order to further restrict the number of edit operations contained in a transformation sequence.

3.1 Simple crossdating by sample sliding

Let's take a closer look at the α-box which has to be filled in order to compute the α-edit distance between the sample and a certain (coherent) piece of the master sequence. Assume the master piece starts at year y. Then the last row of each k-level, $0 \leq k \leq \alpha$, contains all k-edit distances between the sample and all prefixes of the master piece, while each last column contains all k-edit distances between the master piece and all prefixes of the sample. If the master piece is a suffix of the master sequence (or if its length is the length of the sample plus α), the α-box contains all possible k-edit distances between the sample and those master pieces which date the sample to year y.

When we slide the sample across the master, at each position computing an α-box for the sample and the suffix of the master sequence, especially the last row and last column of each level, we obtain all information we need to date the sample: The last rows contain results comparing the sample to the master sequence at different offsets, where each offset is represented in one α-box. The last columns are interesting only in the case that the sample partly overlaps the end of the master sequence, so that a prefix of the sample must be considered. Otherwise, that is in the case that the master piece is longer than the sample, the last column degenerates to one cell which is already part of the last row.

After having computed all edit distances in all last rows and last columns of each level in each α-box, we need to compare them in order to find a good dating. A simple comparison is not meaningful, since the number of terms summing up to a k-edit distance in formula (8) varies according to the number and kind of edit operations performed and also according to a partly overlap of the sequences. So we need to normalize the edit distance by dividing by the number of summed up terms which is the length of the transformed sample sequence. Yet a simple comparison of all normalized edit distances (which means sorting them and taking the smallest as the best) turned out not to be useful. The reason for that is that the normalization removes the information about the length of the sequence (i.e. the number of summands), so that shorter sequences cause a better edit distance more easily than longer sequences do.

Since the t-value is length-dependant and a commonly used distance measure in dendrochronology (which dendrochronologists are familiar with) we take the t-value between the transformed sequence and the master piece as a judging criterion. The correlation coefficient can be implicitly calculated during the matrix filling process at asymptotically no extra cost. So we sort all t-values of the last row and last column of each box and output the biggest t-values, each together with an optimal transformation (i.e. the positions of possibly missing and double rings) and the corresponding dating proposal (offset) to the user.

3.2 Heuristic postprocessing of the results

The algorithm described so far simply sorts all results in the end without taking the number of edit operations into account. Therefore the best results will often contain too many edit operations. A standard approach to that problem is to penalize edit operations either by a multiplicative or an additive term, but unfortunately this also affects edit operations at correct positions. Indeed, it seems that incorrect edit operations are somehow more robust concerning penalties. We therefore decided not to penalize the edit operations, but we compare the obtained results in a heuristic postprocessing step. We store all edit distances of all last rows and last columns of each level of each α-box in an overall result structure. Then usually a good dating appears several times among the best results. Those similar results then differ only in some edit operations, whereby they usually share some edit operations (the correct ones) and include some more edit operations which improve the edit distance a little but which are incorrect.

The heuristic we developed then tries in two phases to identify some possibly redundand results and deletes those from the overall result structure. For each edit distance result (that is an edit distance in the last row or column of a box-level) we do a redundance check within the box plus another check concerning some neighboring boxes.

In the first check phase it is tested, if the normalized distance is significantly smaller than each normalized edit distance associated with each last cell on the diagonals on the transformation path. This captures the idea that one good match often appears several times, where the different occurences share some correct edit operations and include some more incorrect edit operations. An additional edit operation should therefore be admitted only if it improves (hence decreases) the edit distance significantly. A normalized edit distance e is said to be significantly smaller than the normalized edit distance e_{comp}, if $e/e_{comp} < 0.9$. If an edit distance did not pass this check it is deleted from the overall result structure and not compared to other edit distances anymore.

The second check phase is established to eliminate those inter-box redundand results, which date the sequence a few years wrong according to superfluous edit operations at the beginning of the transformation sequence. Call the subsequence of the transformation sequence which includes only the insert and merge operations an *edit sequence*. For every prefix of the edit sequence the time transposition it induces is calculated (a merge operation corresponds to a transposition one year to the left, an insert operation one year to the right). The α-box at the transposed position is checked if it contains an edit distance e_{comp} with an edit sequence equal to the remaining suffix of the edit sequence being checked. Now the normalized edit distance e whose edit sequence probably contains an unnecessary prefix is deleted if $e/e_{comp} \geq 0.9$.

3.3 Crossdating algorithm

Summing up, the crossdating algorithm works as follows:

Crossdating algorithm:

Standardization of the master and the sample sequence (see par. 1.2)
For all overlap positions of the sample in the master
    Fill α-box (see par. 2.3)
    For all cells in the last row and last column of each level
        Normalize edit distance
        Compute optimal transformation
        Redundance check 1: Check with norm. edit distances on transformation path.
        If edit distance is not redundand:
            Store the distance, the t-value, the transformation and the offset number
            in an overall result structure.
Redundance check 2: Remove inter-box-redundand results.
Sort all results in the result structure by decreasing t-value.
Display the best results (those with the highest t-values).

The standardization can be done in $\theta(m+n)$ time and space. The number of α-boxes to be filled is $\mathcal{O}(n+m)$, so that they can all be filled in $\mathcal{O}(\alpha^2 mn)$ time. We do not need to store all α-boxes since we need only the last row and the last column of each level of every α-box. There is one last row or last column entry for each diagonal, so that we only have to count the diagonals of which there are at most $(k+1)$ in every k-level. So we have $\mathcal{O}((m+n)\sum_{k=0}^{\alpha}(k+1)) = \mathcal{O}(\alpha^2(m+n))$ edit distances to be stored. But for each edit distance we also store its corresponding optimal transformation which needs $\theta(k)$ space, so that the space which is altogether needed to store all results sums up to $\mathcal{O}((m+n)(1+\sum_{k=1}^{\alpha}(k+1)k)) = \mathcal{O}(\alpha^3(m+n))$. Additionally there is $\mathcal{O}(\alpha^2 \min(n,m))$ space for one α-box needed to fill a box.

The first redundance check needs $\theta(k)$ time for each edit distance which sums up to $\mathcal{O}((m+n)(1+\sum_{k=1}^{\alpha}(k+1)k)) = \mathcal{O}(\alpha^3(m+n))$ altogether. The second redundance check needs $\mathcal{O}(k^2)$ time each, hence together $\mathcal{O}((m+n)(1+\sum_{k=1}^{\alpha}(k+1)k^2)) = \mathcal{O}(\alpha^4(m+n))$. For the redundance checks there is asymptotically no more space needed. The sorting of all results takes $\mathcal{O}(\alpha^2(m+n)\log(\alpha^2(m+n)))$ time. (Since we are interested in some of the best results only we actually do not have to sort all results, but sorting does not affect the asymptotical running time.) So altogether the algorithm needs $\mathcal{O}(\alpha^4(m+n)+\alpha^2 mn)$ time and $\mathcal{O}(\alpha^3(m+n))$ space.

4 Test results

4.1 Implementation

The crossdating algorithm has been implemented in C++ in a command-line-oriented Unix environment. A program executable can be obtained from the author.

In practice a crossdating program outputs several good matchings (e.g. the best 5 or 10), and the dendrochronologists visually checks if one of them represents the correct dating. Likewise the program we have implemented allows the user to subsequently evaluate the results according to different criteria. I.e. once the results have been computed, the best results in a certain time interval, those having a bigger minimum overlap or those for a lower value for α can be queried. The computation time of such modified result queries by scanning the list of the sorted results is linear in the number of results, thus $\mathcal{O}(\alpha^2(m+n))$. Note that our program especially computes all those t-values that are computed during the simple sliding algorithm. They can be accessed by querying the results for $\alpha = 0$. Our program therefore generalizes the simple sliding algorithm.

Several tests have been performed which we will present in the following three paragraphs: Tests with randomly generated missing or double rings, tests on data containing real missing rings and finally runtime tests. However, the automatized tests do not cover the interactive program properties.

4.2 Randomly generated disturbances

The program was tested on collections of already dated samples. In each collection one sample was randomly disturbed by deleting or splitting up some values, and this sample was then tried to date against the mean sequence of the remaining sequences in the collection.

Tests were performed mainly for the parameter values $\alpha = 2$, a minimum overlap of 50 and a redundancy threshold of 10 (see end of paragraph 2.3). For each sample a random disturbance has been carried out 5 times. Some test results are shown in the tables 1, 2, 3, 4 and 5. The column *date* shows the percentage of those data sets in the collection for which the correct dating has been found. *Date & edit* shows the percentage of those data sets for which the correct dating and the correct type of editation in an interval of radius 10 around the correct position has been found. The column k shows the average number of proposed edit operations for those results for which the correct date and editation has been found. For standardization the percentage of a five-year running mean (also called *floating average*; in the following tables abbreviated with *float-ave*) or the difference of logarithms (abbreviated with *log*) were used.

The data used was supplied by the following sources: The *kieftest* data is tree ring width data from a German pine, which was supplied by Deutsches Archäologisches Institut[2]. The *SET01* and *SET02* data are files which come with the crossdating program TSAP [8] (which does not search for missing or double rings during the crossdating). The other data was taken from the ITRDB[3], where the *germ* data sets are from German oaks, the *cana* data from Canadian white spruce, the *az* data from Arizona where *az526* is ponderosa pine, the *oh* data from white oak from Ohio and the *swed* data from scotch pine from Sweden.

Table 1 shows how many samples of each tested collection the algorithm dates correctly when no random disturbances have been performed. With $\alpha = 0$ this equals a simple sliding algorithm concerning t-values which dates 97% samples correctly on the average[4]. With $\alpha = 2$ the percentage of correct datings decreases by up to 33% (on the average only by 15% though), and the number of mistakenly found edit operations increases by up to 0.9 (on the average only by 0.4). Although our algorithm therefore does not perform really good on sequences that do not contain missing or double rings, it produces much

[2]We like to thank Dr. K.-U. Heußner from Deutsches Archäologisches Insitut, Eurasien-Abteilung, Im Dol 2-6, D-14195 Berlin.

[3]The International Tree-Ring Data Bank (ITRDB) is located at http://www.ngdc.noaa.gov/paleo/treering.html.

[4]On the average means here averaged over the tested collections shown in the tables.

more promising results on sequences that do.

	date	date	k	date	date	k
	$\alpha = 0$	$\alpha = 2$		$\alpha = 0$	$\alpha = 2$	
kieftest	96 %	80 %	0,3	96 %	86 %	0,5
germ001	98 %	80 %	0,2	98 %	83 %	0,2
germ003	100 %	94 %	0,1	100 %	88 %	0,1
germ004	96 %	79 %	0,7	96 %	88 %	0,9
germ006	100 %	88 %	0,4	100 %	94 %	0,0
cana030	94 %	72 %	0,3	94 %	89 %	0,0
az052	100 %	80 %	0,1	100 %	80 %	0,4
az526	100 %	100 %	0,1	100 %	95 %	0,8
SET01	94 %	69 %	0,7	94 %	63 %	0,5
SET02	96 %	71 %	0,8	96 %	63 %	0,5
oh004	100 %	72 %	0,1	100 %	79 %	0,1
swed302	98 %	82 %	0,0	98 %	89 %	0,1

Float-ave preprocessing Log preprocessing

Table 1: Test results without manipulation of the data.

To see that, let's take a look at table 2 which shows test results for the *germ001* data with one randomly deleted element and for different values of α. Setting $\alpha = 0$ (again, this equals the simple sliding algorithm) the correct date is found in only 51% or 41% of the cases (for floating-average or log preprocessing, respectively). When allowing the algorithm to perform some editations by choosing α slightly greater than 1, the chance for a correct dating increases dramatically. But the farther α is away from the correct number of edit operations needed, the more false edit operations are performed and the more the chance for a correct date decreases. But since there are not suspected many double or missing rings in a tree ring sequence, a small value of α (e.g. 2 or 3) should be sufficient most of the times.

α	date	date & edit	k	date	date & edit	k
0	51 %	0 %	0,0	41 %	0 %	0,0
1	94 %	91 %	1,0	95 %	91 %	1,0
2	93 %	89 %	1,0	92 %	87 %	1,0
3	81 %	77 %	1,4	84 %	76 %	1,4
4	82 %	78 %	1,5	84 %	76 %	1,4
5	85 %	78 %	2,2	85 %	76 %	1,9

Float-ave preprocessing Log preprocessing

Table 2: Test results concerning the *germ001* data with a random deletion of one sample element and different values for α.

	date	date	date & edit	k	date	date	date & edit	k
	$\alpha = 0$	$\alpha = 2$			$\alpha = 0$	$\alpha = 2$		
kieftest	24 %	73 %	67 %	1,2	28 %	76 %	69 %	1,2
germ001	51 %	92 %	89 %	1,0	41 %	92 %	87 %	1,0
germ003	58 %	94 %	83 %	1,0	53 %	92 %	81 %	1,0
germ004	47 %	84 %	76 %	1,2	50 %	87 %	78 %	1,2
germ006	21 %	91 %	85 %	1,0	26 %	88 %	81 %	1,0
cana030	27 %	72 %	69 %	1,0	30 %	90 %	80 %	1,0
az052	44 %	97 %	94 %	1,0	42 %	95 %	95 %	1,0
az526	38 %	91 %	79 %	1,0	35 %	96 %	83 %	1,1
SET01	34 %	60 %	54 %	1,0	25 %	59 %	49 %	1,1
SET02	37 %	54 %	48 %	1,2	33 %	59 %	53 %	1,2
oh004	47 %	85 %	83 %	1,0	42 %	81 %	80 %	1,0
swed302	43 %	83 %	72 %	1,0	39 %	89 %	79 %	1,0

Float-ave preprocessing Log preprocessing

Table 3: Test results with a random deletion of one sample element.

	date	date	date & edit	k	date	date	date & edit	k
	$\alpha = 0$	$\alpha = 2$			$\alpha = 0$	$\alpha = 2$		
kieftest	22 %	75 %	63 %	1,2	28 %	75 %	58 %	1,2
germ001	52 %	92 %	88 %	1,0	42 %	93 %	88 %	1,0
germ003	56 %	94 %	83 %	1,0	52 %	92 %	79 %	1,0
germ004	48 %	88 %	78 %	1,1	50 %	88 %	74 %	1,1
germ006	19 %	90 %	81 %	1,0	26 %	88 %	85 %	1,0
cana030	29 %	72 %	66 %	1,0	34 %	82 %	71 %	1,0
az052	47 %	98 %	88 %	1,0	42 %	98 %	80 %	1,0
az526	36 %	83 %	74 %	1,0	34 %	85 %	66 %	1,2
SET01	18 %	60 %	53 %	1,0	16 %	59 %	56 %	1,1
SET02	28 %	57 %	48 %	1,1	32 %	59 %	52 %	1,1
oh004	47 %	87 %	85 %	1,0	43 %	85 %	83 %	1,0
swed302	34 %	74 %	67 %	1,1	35 %	79 %	69 %	1,0

Float-ave preprocessing Log preprocessing

Table 4: Test results with a random splitting of one sample element.

	date	date	date & edit	k	distance betw. edits	date	date	date & edit	k	distance betw. edits
	$\alpha = 0$	$\alpha = 3$				$\alpha = 0$	$\alpha = 3$			
kieftest	13 %	67 %	55 %	2,2	2,7	21 %	73 %	56 %	2,1	2,5
germ001	56 %	91 %	85 %	2,0	2,2	54 %	89 %	78 %	2,0	2,2
germ003	54 %	91 %	74 %	2,0	3,5	58 %	89 %	68 %	2,0	3,5
germ004	38 %	80 %	79 %	2,1	3,1	49 %	85 %	69 %	2,1	2,7
germ006	16 %	85 %	75 %	2,0	2,5	21 %	84 %	66 %	2,0	2,5
cana030	20 %	71 %	66 %	2,0	2,8	34 %	86 %	73 %	2,0	3,0
az052	44 %	94 %	91 %	2,0	1,8	42 %	91 %	88 %	2,0	1,7
az526	44 %	89 %	79 %	2,0	2,2	48 %	92 %	78 %	2,1	2,3
SET01	24 %	55 %	49 %	2,0	2,8	34 %	55 %	39 %	2,2	3,1
SET02	30 %	51 %	39 %	2,1	3,1	38 %	52 %	38 %	2,1	2,4
oh004	45 %	79 %	76 %	2,0	2,3	49 %	76 %	72 %	2,0	2,2
swed302	34 %	77 %	65 %	2,0	2,7	48 %	81 %	65 %	2,0	2,6

Float-ave preprocessing Log preprocessing

Table 5: Test results with a random deletion of two consecutive sample elements.

Tables 3, 4 and 5 show test results for different data collections with one random deletion, one random splitting and two random consecutive deletions, respectively. Although the percentages for a correct dating with a correct editation vary from 38% to 95%, the percentages are usually extremely higher than those for a dating with $\alpha = 0$. However, as can be seen in the column *date* for $\alpha > 0$, the program finds the correct date more often than the correct date plus the correct editation, because it proposes some wrong, additional or not enough edit operations. In fact, if the program finds the correct date, it usually proposes most of the editations at an almost correct position and skips necessary editations only if they are too close to the beginning or the end of the sequence. In any case the results of the program give more information to the user about possibly missing or double rings than the standard crossdating methods do. Concerning two consecutive deletions, table 5 shows that if two missing rings have been found, they lie only about 2 or 3 years apart.

4.3 Real missing rings

In this paragraph we present results from tests performed on data containing real missing rings. Test data like this is widely available because many dendrochronologists mark a missing ring as a ring with width 0. Test data for double rings is rather hard to find because dendrochronologists usually do not record the occurence of double rings. The reason for that is that a double ring can (with some more preparation of the wood or a better microscope) often visually be identified on the wood, whereas a missing ring cannot. We therefore restricted the tests on data with real inconsistencies to data containing missing rings.

Table 6 shows test results for collections of samples where some samples contain missing rings. During the tests α has been chosen to be 4. The *breclav* data was supplied by Deutsches Archäologisches Institut, the others are available at the ITRDB. The *wa* data is from a subalpine larch from Washington State, the *chin* data is Armand's pine from China, and the *az* data is from Arizona, as mentioned above. The column *# of samples* contains the number of samples in the collection that contain missing rings. The *ave. # of miss. rings* column shows the average number of missing rings contained in the samples. The *data & edit* column contains the number of samples (and also the percentage relative to the *# of samples*) that the algorithm dates correctly with the correct number and position (with a tolerance of 10) of insertions. The master sequences to date against are built out of those samples in each collection that do not contain missing rings.

	# of samples	ave. # of miss. rings	date	date & edit	date	date & edit
wa067	19	2,42	$16 \cong 84\%$	$12 \cong 63\%$	$17 \cong 89\%$	$14 \cong 74\%$
wa069	13	1,08	$12 \cong 92\%$	$9 \cong 69\%$	$13 \cong 100\%$	$10 \cong 77\%$
wa072	19	2,32	$16 \cong 84\%$	$13 \cong 68\%$	$16 \cong 84\%$	$13 \cong 68\%$
wa079	23	2,22	$17 \cong 74\%$	$15 \cong 65\%$	$18 \cong 78\%$	$17 \cong 74\%$
breclav	7	1	$7 \cong 100\%$	$5 \cong 71\%$	$7 \cong 100\%$	$6 \cong 86\%$
chin04	13	2,92	$12 \cong 92\%$	$9 \cong 69\%$	$13 \cong 100\%$	$11 \cong 85\%$
az052	6	3,33	$5 \cong 83\%$	$4 \cong 67\%$	$5 \cong 83\%$	$4 \cong 67\%$
az526	14	3,71	$13 \cong 93\%$	$6 \cong 43\%$	$13 \cong 93\%$	$7 \cong 50\%$

Float-ave preprocessing   Log preprocessing

Table 6: Test results for data with real missing rings.

4.4 Runtime tests

The crossdating algorithm has been implemented in $C++$ in a Unix environment and has been tested on a Sparc Ultra machine. The runtime the program needs for fixed $\alpha = 2$ is illustrated in figure 6, and for a fixed sample length $n = 300$ in figure 7.

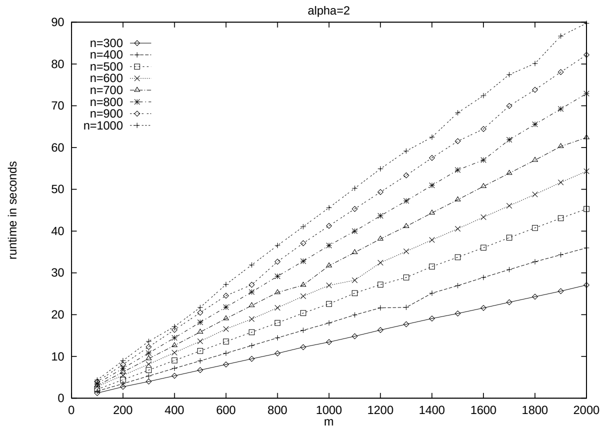

Figure 6: Runtime tests with $\alpha = 2$ and n and m the lengths of the sample and master sequence, respectively.

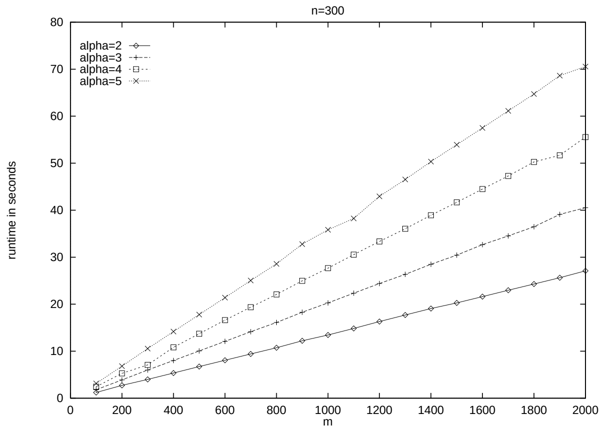

Figure 7: Runtime tests with $n = 300$ fixed.

5 Implementation and Conclusions

We investigated the problem of matching tree ring width sequences (crossdating) which is stated in dendrochronology. Assuming that a tree forms exactly one ring each year the matching can be performed by an easy $\theta(mn)$ algorithm. We presented a $\mathcal{O}(\alpha^4(m+n) + \alpha^2 mn)$ crossdating algorithm which takes the possibility of missing and double rings into account by employing an edit distance as a distance measure.

The algorithm has been implemented and tested. The tests show that the dating quality of the algorithm varies depending strongly on the input data and is best when α equals the number of inconsistencies to be found. It is therefore not possible to date a tree ring sequence according solely to the first dating proposition of the algorithm. However, in the usual dating process results of an automatic matching are taken as dating propositions only and are always visually verified by a dendrochronologist. Since our program allows to evaluate the computed results for some different parameter settings, e.g. for a lower value for α, in rather fast $\mathcal{O}(\alpha^2(m+n))$ time, and it usually proposes some good datings, the program should definitely be eligible to serve as an additional dating tool searching for missing and double rings.

For further research it is interesting to investigate if it is possible to compare several tree ring sequences at once in order to produce a mean sequence (*master sequence*, *chronology*). Another question is if similar matching techniques based on the edit distance can be applied to other environmental archives like sea or glacier sediments, where certain environmental events cause different distortions of the underlying sequences.

References

[1] Paul C. Van Deusen. A dynamic program for cross-dating tree rings. *Canadian Journal of Forest Research*, 20:200–205, 1989.

[2] Douglas F. Elliott and K. Ramamohan Rao. *Fast Transforms - Algorithms, Analyses, Applications.* Academic Press, 1982.

[3] Dan Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, 1997.

[4] Richard L. Holmes. Computer-assisted quality control in tree-ring dating and measurement. *Tree-Ring Bulletin*, 43:69–75, 1983.

[5] Joseph B. Kruskal and David Sankoff. An anthology of algorithms and concepts for sequence comparison. In David Sankoff and Joseph B. Kruskal, editors, *Time Warps, String Edits, and Mocromolecules: The Theory and Practice of Sequence Comparison*, chapter 10. Addison-Wesley Publishing Company, 1983.

[6] H.J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms.* Springer Verlag, 1981.

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 2. edition, 1992.

[8] Frank Rinn. *TSAP Reference Manual.* Heidelberg. http://ourworld.compuserve.com/homepages/frankrinn/.

[9] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26(1):43–49, 1978.

[10] David Sankoff and Joseph B. Kruskal, editors. *Time Warps, String Edits, and Mocromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley Publishing Company, 1983.

[11] F.H. Schweingruber. *Trees and Wood in Dendrochronology.* Springer-Verlag, 1993.

[12] Graham A. Stephen. *String Searching Algorithms.* World Scientific, 1994.

[13] Carola Wenk. Algorithmen für das Crossdating in der Dendrochronologie. Master's thesis, Freie Universität Berlin, Institut für Informatik, 1997.