Applying an edit distance to the matching of tree ring sequences in dendrochronology ☆,☆☆

Carola Wenk [1]

Institut für Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, Germany

Abstract

In dendrochronology wood samples are dated according to the tree rings they contain. The dating process consists of comparing the sequence of tree ring widths in the sample to a dated master sequence. Assuming that a tree forms exactly one ring per year a simple sliding algorithm solves this matching task.

But sometimes a tree produces no ring or even two rings in a year. If a sample sequence contains this kind of inconsistencies it cannot be dated correctly by the simple sliding algorithm. We therefore introduce a $\mathcal{O}(\alpha^2 mn + \alpha^4(m+n))$ algorithm for dating such a sample sequence against an error-free master sequence, where n and m are the lengths of the sequences. 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.
© 2003 Published by Elsevier B.V.

Keywords: Edit distance; Crossdating; Dendrochronology

1. Introduction

1.1. Dendrochronology

The tree ring structure in wood samples is important in many research areas, for instance in archeology, climatology, geomorphology and glaciology. The reason for that is that the growth of a tree and therefore its 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 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 horizontal cross-section of the stem of a tree visually consists of a number of concentric annuli, the tree rings. 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 at the beginning of the growth period and of thin, thickwalled cells built at the end. The first type of cell ensures the food supply to the shoots, whereas the other type accounts for the stability of the stem. Since the second type of cell 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. The width of a tree ring is the width of the annulus it describes in a horizontal cross-section of the stem.

In dendrochronology a wood sample is characterized by the sequence of its tree ring widths.[2] Such a sequence consists of positive real values each describing the width of one tree ring, in the same order as the tree rings occur in the sample. 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

In practice, before two tree ring width sequences are compared they have to be filtered. This so-called *standardization* process cleans the data from individual trends, which usually are long term trends. For instance with growing age the tree rings usually become thinner. Thus after standardization 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 logarithmic difference are used. In the sequel a tree ring sequence will thus be a standardized sequence of tree ring widths, which is a sequence of real values.

Assuming that the trees being considered have built exactly one ring each year, a crossdating can be performed by sliding the undated sample sequence along the dated master sequence, which is usually quite longer, starting and ending with a certain constant minimum overlap of e.g., 50 rings. At each position the distance (according to a predefined

---

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

distance measure) between the overlapping parts of the sequences is computed and the position yielding the best distance is proposed 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 sliding algorithm. Then the *t*-value (also known as *Student's t*) is defined as

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

where r is the correlation coefficient

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

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

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

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

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

with the signum function sign and the characteristic function

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

A sequential computation of all distances takes $\theta(nm)$ time, where m and n are the lengths of the master and the sample sequences, respectively. Considering (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 (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., [2,7,8]. 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 crossing position which yields the best distance. The results of the matching algorithm are always visually checked by a dendrochronologist.

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. It happens that due to bad growing conditions a tree does 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. Also mistakes during the measurement of very narrow rings can lead to missing rings. Moreover, climatic variations during a year 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 sample sequence which may contain inconsistencies against a clean master sequence is to split up the sample into shorter parts and to date each part on its own (either manually or using Cofecha [4]). Finally possible positions for missing or double rings are manually concluded. 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 a missing ring position date all to one year later) is then available, but a missing ring is not explicitly proposed.

1.4. Overview

The remaining part of the article is structured as follows: Section 2 concentrates on the edit distance as a distance measure. First a simple edit distance, similar to that between strings, is introduced. Then a first approach to restrict the number of possible missing or double rings is introduced, which however does not fully succeed. Last the α-edit distance is introduced, which guarantees to restrict the number of edit operations. In Section 3 we present the crossdating algorithm based on the α-edit distance, consider several heuristics, and finally present the enhanced crossdating algorithm including all heuristics. Section 4 is devoted to test results and Section 5 contains conclusions.

2. Edit distances

2.1. A simple edit distance

Let $A = a_0, \ldots, a_{n-1}$ and $B = b_0, \ldots, b_{m-1}$ be two standardized tree ring width sequences, where A may contain 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 into one (which compensates a double ring) into a sequence close to B. Closeness is defined by considering the sum of the squared differences. The transformations allowed are described by *transformation sequences* which are defined as follows:

τ	M		M		N	I	I	N
A	3	2	1	2	5			3
$\tau(A)$	5		3		5	4	4	3

Fig. 1. An example of a sequence A, a transformation sequence τ and the transformed sequence $\tau(A)$.

Definition 1. A *transformation sequence* is a sequence over the alphabet $\{I, M, N\}$, where I stands for *insert*, M for *merge*, and N for *identity operation*. For a transformation sequence τ we call the number of merge operations it contains γ_τ, the number of insert operations ι_τ and the number of identity operations ν_τ. We denote by $\mathcal{T}_{n,m}$ the set of all transformation sequences τ of length $|\tau| = m = \gamma_\tau + \iota_\tau + \nu_\tau$ such that $n = 2\gamma_\tau + \nu_\tau$.

Definition 2. Let $\tau = \tau_0, \ldots, \tau_{m-1} \in \mathcal{T}_{n,m}$ be a transformation sequence. Let $A = a_0, \ldots, a_{n-1}$ be a sequence of real values, possibly embedded into a sequence $\ldots, a_{-1}, A, a_n, \ldots$. Then the *transformation that τ effects on A* is defined as

$$\tau(A)_i = \begin{cases} a_{2\gamma_i + \nu_i} + a_{2\gamma_i + \nu_i + 1}, & \text{if } \tau_i = M, \\ a_{2\gamma_i + \nu_i}, & \text{if } \tau_i = N, \\ (a_{2\gamma_i + \nu_i - 1} + a_{2\gamma_i + \nu_i})/2, & \text{if } \tau_i = I, \end{cases} \qquad (6)$$

for $0 \leqslant i \leqslant m-1$, and where γ_i and ν_i denote the number of merge or identity operations in the prefix $\tau_0, \ldots, \tau_{i-1}$ of τ. If A is not embedded into a larger sequence such that either a_{-1} or a_n do not exist, then a transformation sequence with leading or ending insert operations, respectively, does not define a valid transformation.

The transformation τ effects on A can be best visualized by aligning A and τ and performing the transformation sequentially from left to right. Let for example be $A = 3, 2, 1, 2, 5, 3$ and consider a transformation sequence $\tau = MMNIIN$, see Fig. 1. Then the transformation merges 3 and 2 to 5 (M), it merges 1 and 2 to 3 (M), it does not change 5 (N), it inserts a ring which is done by taking the average $\frac{5+3}{2} = 4$ of the two surrounding rings (I), it inserts another ring in the same manner (I) and finally it keeps the last ring 3 (N).

Note that in Definition 2 the possibility that A is embedded into a another sequence is technically necessary, because otherwise leading or ending insert operations would not properly be defined. But these are necessary in the sequel, e.g., when considering partial overlap of the sample and the master, or in Lemma 2 where edit distances between prefixes are considered. If in the following a transformation sequence is considered which due to leading or ending insert operations does not define a valid transformation we will simply neglect it, in order not to overcomplicate the notation. However, this little sloppy notation does not affect the validity of the stated results. Furthermore, note that merge and insert operations operate only on the values given in the original sequence and not on those values that have been created by previous merge or insert operations. This definition has been chosen on the one hand with respect to the application where a modification of the original sequence is performed by taking the surrounding original values into account. On the other hand this definition enables us to subsequently derive a recurrence for the edit distance, whereas a different definition would cause problems in this point.

Definition 3. The *simple edit distance* $D_{\text{simp}}(A, B)$ is defined as

$$D_{\text{simp}}(A, B) = \begin{cases} \min_{\tau \in \mathcal{T}_{n,m}} \sum_{\upsilon=0}^{m-1} \big(\tau(A)_\upsilon - B_\upsilon\big)^2, & \text{if } \mathcal{T}_{n,m} \neq \emptyset, \\ \text{not defined}, & \text{otherwise.} \end{cases} \tag{7}$$

We call a transformation sequence which minimizes the sum *optimal*.

This notion of an edit distance is based on the edit distance for strings (see [3,12]) and on the dynamic time warping in speech recognition (see [6,10]). Note that strictly speaking an insert operation compensates only a ring which is actually missing, and it does not compensate a measurement error. The latter would be compensated by a split operation. But since measurement errors happen only with extremely narrow rings an insert operation is a good approximation for a split operation. Technically one could easily integrate split operations into the definition of the edit distance.

Lemma 1. $\mathcal{T}_{n,m} \neq \emptyset \Leftrightarrow n \leqslant 2m$.

Proof. Easy application of the equations $m = \gamma_\tau + \iota_\tau + \nu_\tau$ and $n = 2\gamma_\tau + \nu_\tau$ for a transformation sequence $\tau \in \mathcal{T}_{n,m}$.   □

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

Lemma 2.

$$D_{\text{simp}}(0, 0) = 0,$$

$$D_{\text{simp}}(i, j) = \min \left\{ \begin{array}{l} D_{\text{simp}}(i - 2, j - 1) + (a_{i-2} + a_{i-1} - b_{j-1})^2, \\ D_{\text{simp}}(i - 1, j - 1) + (a_{i-1} - b_{j-1})^2, \\ D_{\text{simp}}(i, j - 1) + \big((a_{i-1} + a_i)/2 - b_{j-1}\big)^2 \end{array} \right\} \tag{8}$$

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

Proof. Via a case distinction concerning the last character of the optimal transformation sequence.   □

Although the transformation space is exponentially big a dynamic programming approach allows to compute $D_{\text{simp}}(A, B)$ in $\theta(nm)$ time and space. This is accomplished by sequentially filling an $(n + 1) \times (m + 1)$ matrix in which cell (i, j) contains the value $D_{\text{simp}}(i, j)$. The condition $i \leqslant 2j$ and the symmetrical condition $(n - i) \leqslant 2(m - j)$ cut two corners off the matrix which represent undefined or for the computation of $D_{\text{simp}}(n, m)$ unnecessary values, respectively. According to Lemma 2 a value is computed out of at most three values. The value $D_{\text{simp}}(A, B) = D_{\text{simp}}(n, m)$ 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
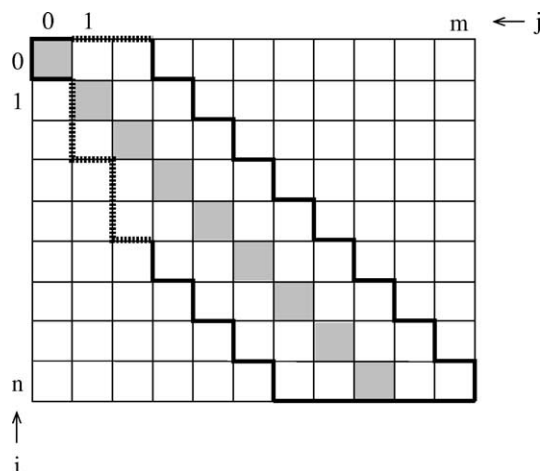
Fig. 2. Van Deusen's computation matrix with a strip of width $\alpha = 2$ to each side of the diagonal. The diagonal has been shaded.

which of the three possible cells contributed its value to the examined cell (either by re-calculating the sums or by checking a previously saved pointer to a cell). In this way a path of cells from cell (n, m) to cell $(0, 0)$ is constructed which obviously corresponds to a transformation sequence.

2.2. Van Deusen's edit distance

The transformation space over which the simple edit distance is minimized includes in particular transformations containing many edit operations. Transformations like this correspond to paths in the computation matrix with many non-diagonal sections. 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 Fig. 2.

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 be-tween a merge and an insert operation the conforming transformation path still stays inside a strip of width 1 around the diagonal.

2.3. α-edit distance

A straight forward improvement of Van Deusen's edit distance is the following notion which we call α-*edit distance*. This type of edit distance has been proposed for strings by Sankoff and Kruskal [5]. Since the number of edit operations (i.e., merges or inserts) con-tained in an optimal transformation should be small, the idea is to regard transformations and edit distances depending on the number of edit operations.

Definition 4. We denote by $\mathcal{T}_{n,m,\alpha}$ the set of all transformation sequences $\tau \in \mathcal{T}_{n,m}$ with $\iota_\tau + \gamma_\tau = \alpha$.

Definition 5. The α-*edit distance* is defined as

$$D(A, B, \alpha) = \begin{cases} \min_{\tau \in \mathcal{T}_{n,m,\alpha}} \sum_{\upsilon=0}^{m-1} \big(\tau(A)_\upsilon - B_\upsilon\big)^2, & \text{if } \mathcal{T}_{n,m,\alpha} \neq \emptyset, \\ \text{not defined}, & \text{otherwise.} \end{cases} \qquad (9)$$

Note that the α-edit distance as we defined it is the edit distance between the two sequences when only transformations containing *exactly* α edit operations are considered. It is also possible to define it with *up to* α edit operations as in [5]. However, our definition allows to select possibly suboptimal edit distances with a given number of edit operations in an easy way.

Lemma 3. $\mathcal{T}_{n,m,\alpha} \neq \emptyset \Leftrightarrow m \geqslant \alpha$ *and* $m - n = \alpha - 2\gamma$ *for a* $\gamma \in \{0, \dots, \alpha\}$.

Proof. Easy application of the equations $m = \gamma_\tau + \iota_\tau + \nu_\tau = n + \iota_\tau - \gamma_\tau$ and $\iota_\tau + \gamma_\tau = \alpha$ for a transformation sequence $\tau \in \mathcal{T}_{n,m,\alpha}$. $\quad\square$

We define $D(i, j, \alpha)$ to be the α-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 α-edit distance satisfies the following recurrence:

Lemma 4.

$$D(0, 0, 0) = 0,$$

$$D(i, j, \alpha) = \min \begin{cases} D(i - 2, j - 1, \alpha - 1) + (a_{i-2} + a_{i-1} - b_{j-1})^2, \\ D(i - 1, j - 1, \alpha) + (a_{i-1} - b_{j-1})^2, \\ D(i, j - 1, \alpha - 1) + \big((a_{i-1} + a_i)/2 - b_{j-1}\big)^2 \end{cases} \qquad (10)$$

for all $0 \leqslant i \leqslant n,\ 0 \leqslant j \leqslant m$
with $j \geqslant k\alpha$; *and* $j - i = \alpha - 2\gamma$ *for a* $\gamma \in \{0, \dots, \alpha\}$.

Proof. Via a case distinction concerning the last character of the optimal transformation sequence. $\quad\square$

The α-edit distance can be computed in a dynamic programming manner in $\mathcal{O}(\alpha^2 \cdot \min(n, m))$ time and space: The storage required is a part of an $(n+1) \times (m+1) \times (\alpha+1)$ box (see Fig. 3) in which cell (i, j, k) contains the value $D(i, j, k)$. Due to the condition $j - i = k - 2\gamma$ for a $\gamma \in \{0, \dots, k\}$ the defined values of $D(i, j, k)$ form diagonals inside the matrix. Note that for each $\gamma \in \{0, \dots, k\}$ 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 Fig. 3. According to Lemma 4 the value $D(i, j, k)$ is computed out of at most three values (see Fig. 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
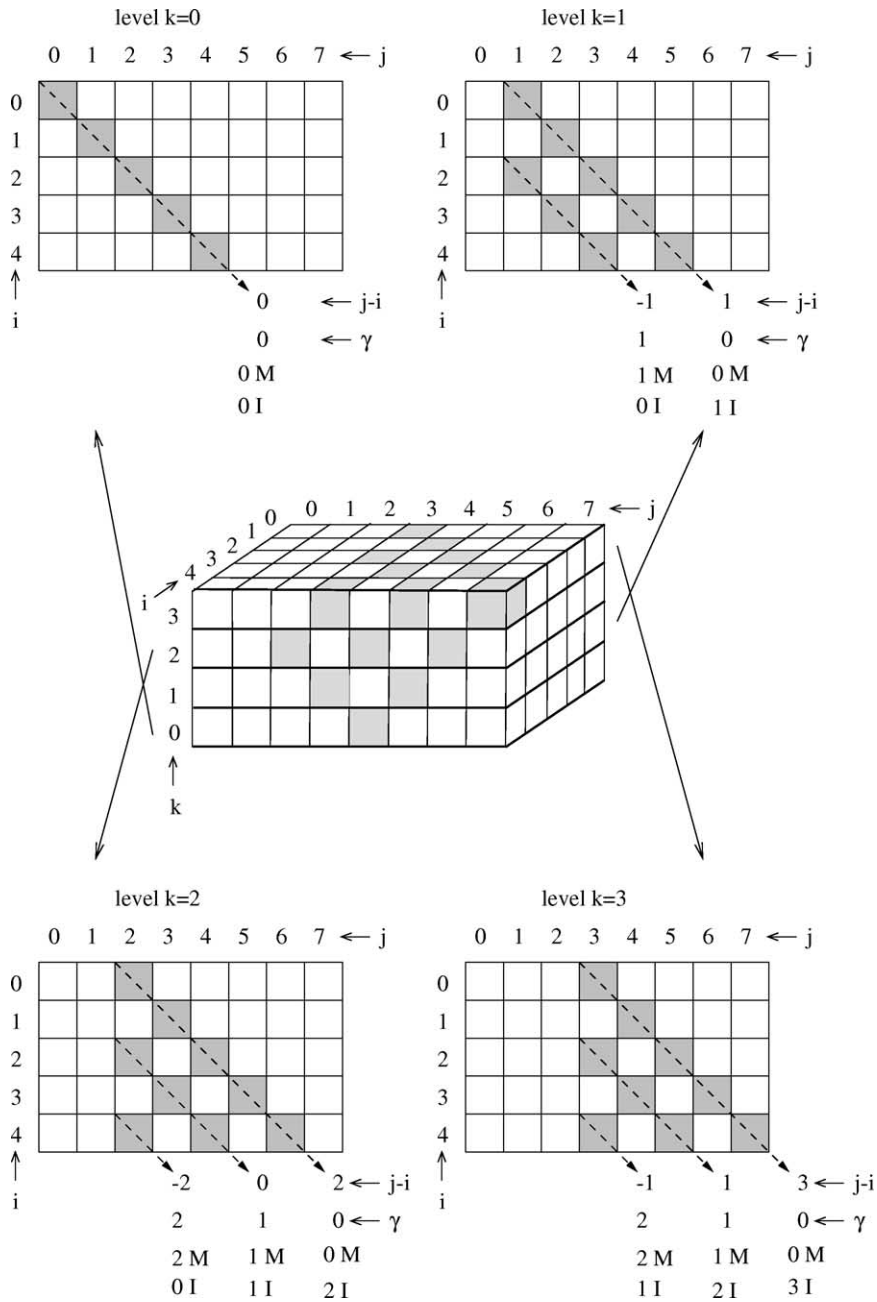
Fig. 3. Dynamic programming box needed for the computation of the 3-edit distance between two sequences of length 4 and 7. The defined cells have been shaded. The number of merge (M) and insert (I) operations corresponding to the diagonals have been written next to each diagonal.
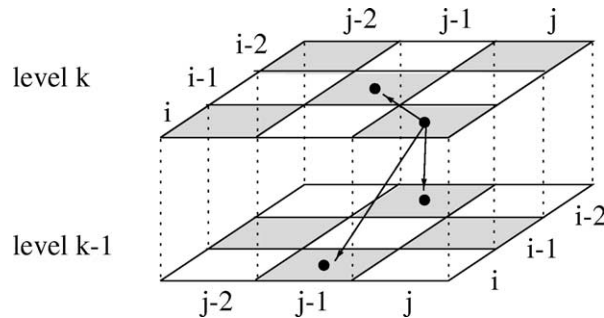
Fig. 4. Dynamic programming computation of the k-edit distance according to Lemma 4.

all k-edit distances between A and B with $0 \leqslant k \leqslant \alpha$. 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 $(\min(n, m) + 1) \sum_{k=0}^{\alpha}(k + 1) = \mathcal{O}(\alpha^2 \min(n, m))$ cells to be filled.

Theorem 1. *The α-edit distance between two strings of lengths n and m can be computed in time $= \mathcal{O}(\alpha^2 \min(n, m))$.*

3. Crossdating employing α-edit distances

The crossdating problem could be tackled in the following way, see, e.g., [3]: One could fill one big α-box for the sample and the whole master sequence, whereby all cells $D(0, j, 0)$, for $0 \leqslant j \leqslant m - 1$, have to be initialized with 0 in advance. The cells $D(n, j, k)$, for $0 \leqslant j \leqslant m - 1$ and $0 \leqslant k \leqslant \alpha$, then contain k-edit distances between the sample and different parts of the master sequence. This approach can be applied in cases where the best edit distance for each overlap position is searched. However, since due to the nature of the problem the dendrochronologists are also interested in suboptimal results, we follow a different approach here. For similar reasons [3] considers also algorithms that compute suboptimal alignments.

In our crossdating algorithm we apply the crossdating technique of sliding the sample sequence across the master sequence and computing distances between the overlapping parts at each crossing position. Of course the distance measure we apply is the α-edit distance. A parameter α, which denotes the maximum number of allowed edit operations, 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 being contained in a transformation sequence.

3.1. Crossdating by sample sliding

The crossdating algorithm is shown in Fig. 5. It basically slides the sample along the master sequence, computes all edit distances between the sample and a corresponding piece of the master sequence and finally sorts the computed results.

Let us 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 specific (coherent) piece of the master sequence.

Crossdating algorithm:

Standardization of the master and the sample sequence
For all overlap positions of the sample in the master
      Fill α-box
    For all cells in the last row and last column of each level
        Normalize edit distance
        Compute optimal transformation
        Store the distance, the transformation and the offset number in an
        overall result structure.
Sort all results in the result structure by decreasing normalized edit distance.
Display the best results (those with the smallest normalized edit distance).

Fig. 5. Crossdating algorithm.

A good way to visualize the following description is to associate the rows of each level of the α-box with the sample and the columns with the master piece. Assume the master piece starts at year y. Then the last row of each k-level, $0 \leqslant k \leqslant \alpha$, 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. In the case where 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 the last rows and last columns of each level in each α-box, we need to compare them in order to find an acceptable dating. A simple comparison is not meaningful, since the number of terms adding up to a k-edit distance in (9), $0 \leqslant k \leqslant \alpha$, varies according to the number and kind of edit operations performed and also according to a partial overlap of the sequences. So we normalize each edit distance by dividing by the number of added terms which is the length of the transformed sample sequence. Finally we sort the normalized edit distances and display the smallest normalized edit distances to the user, each together with an optimal transformation (i.e., the positions of possible missing and double rings) and the corresponding dating proposal (offset).

In order to decrease the space complexity we assume that there is exactly one optimal transformation per edit distance, otherwise we choose exactly one. Although theoretically there can be several optimal transformations associated with one edit distance, an exact equality of the sums is unlikely, because of the real valued input data. Then instead of storing in each cell of the α-box 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.

Theorem 2. *The crossdating algorithm shown in Fig. 5 needs $\mathcal{O}(\alpha^2 mn)$ time and $\mathcal{O}(\alpha^3(m+n))$ space.*

Proof. The standardization can be done in $\theta(m+n)$ time and space. The number of α-boxes to be filled is $\mathcal{O}(n+m)$, and since we need $\mathcal{O}(\alpha^2 \min(n,m))$ time to fill one α-box (see Theorem 1), we can fill them all 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

$$\mathcal{O}\left((m+n)\left(1+\sum_{k=1}^{\alpha}(k+1)k\right)\right) = \mathcal{O}\left(\alpha^3(m+n)\right).$$

Additionally there is $\mathcal{O}(\alpha^2 \min(n,m))$ space for one α-box needed to fill a box.

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 asymptotically running time.) So altogether the algorithm needs $\mathcal{O}(\alpha^2 mn)$ time and $\mathcal{O}(\alpha^3(m+n))$ space. □

3.2. Heuristics

Unfortunately, the results the edit-distance based crossdating algorithm computes do not, for several reasons, match the expectations of the dendrochronologists. One reason is, that there is no definition of a distance measure known that correctly models the differences and similarities between tree ring sequences. Neither the edit distance nor the t-value or the Gleichläufigkeitskoeffizient are therefore ultimate distance measures. In practice, the crossdating task is therefore performed by an experienced dendrochronologist who verifies and perhaps rejects the output of a crossdating tool. However, besides this general problem there are certain heuristics that improve on the practical performance of our algorithm, which we describe in the sequel. The goal is to tune the algorithm in that way, that the best match it computes is the match a dendrochronologist would find. This includes that the edit operations the algorithm proposes for the best result do also match the expectations of the dendrochronologist. We will call these edit operations *correct*, and others *incorrect*.

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. As a heuristic supplement we thus modify the edit distance in the following way: 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 so

that the opposite edit operation is not allowed when calculating the edit distances for these cells.

Furthermore, the simple comparison of all normalized edit distances (which means sorting them and taking the smallest as the best) proved 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), such that short sequences cause a better edit distance more easily than long sequences do. Since the t-value is length-dependent 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 instead of the normalized edit distance as the judging criterion. I.e., we compute the edit distance in order to find a good transformation, but in the end we sort the results with respect to the t-value.

So far the algorithm described 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 from the viewpoint of a dendrochronologist. A standard approach to that problem is to penalize edit operations either by a multiplicative or an additive term. Unfortunately this also affects edit operations at correct positions. Indeed, it seems that in respect to penalties incorrect edit operations are somehow more robust. We therefore decided not to penalize the edit operations, but we compare the obtained results in a heuristic post-processing 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. We call those results that contain "too many" edit operations *redundant*. By applying two *redundancy checks* we try to identify some possible redundant results and delete 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 redundancy check within the box plus another check concerning some neighboring boxes.

During the first redundancy check 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 occurrences 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 $e/e_{\text{comp}} < 0.9$. 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 redundancy check is established to eliminate those inter-box redundant results, which date the sequence incorrectly by a few years 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

---

Enhanced crossdating algorithm:

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

Fig. 6. Enhanced crossdating algorithm including all heuristics of Section 3.2.

being checked. Now the normalized edit distance e whose edit sequence probably contains an unnecessary prefix is deleted if $e/e_{\text{comp}} \geqslant 0.9$.

3.3. Enhanced crossdating algorithm

Theorem 3. *The enhanced crossdating algorithm shown in Fig.* 6*, which includes all heuristics of Section* 3.2*, needs* $\mathcal{O}(\alpha^2 mn + \alpha^4(m+n))$ *time and* $\mathcal{O}(\alpha^3(m+n))$ *space.*

Proof. The correlation coefficient and thus the t-value can be implicitly calculated during the box filling process at no extra cost asymptotically. The first redundancy 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 redundancy 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 redundancy checks there is asymptotically no more space needed. So altogether the algorithm needs $\mathcal{O}(\alpha^2 mn + \alpha^4(m+n))$ time and $\mathcal{O}(\alpha^3(m+n))$ space. $\square$

4. Test results

4.1. Implementation

The enhanced 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 dendrochronologist 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. That is, 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.

We have performed several tests which we present in the following two paragraphs: Tests with randomly generated missing or double rings and tests on data containing real missing rings 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 $\alpha = 2$, 3 and 4, a minimum overlap of 50 and a redundancy threshold of 10 (see end of Section 2.3). For each sample a random disturbance has been carried out 5 times. Some test results are shown in Tables 1, 2, 3 and 4. 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 have 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 was 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.[3] The *SET01* and *SET02* data are files which come with the crossdating program TSAP [9] (which does not search for missing or double rings during the crossdating). The other data was taken from the ITRDB,[4] 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 98% samples correctly on the average.[5] With $\alpha = 2$ the percentage of correct datings decreases by up to 33% (on the average only by 18% though), and the number of mistakenly found edit operations increases by up to 0.4 (on the average only by 0.1).

To see how our algorithm performs on sequences that contain missing rings, take a look at Table 2 which shows test results for four data sets 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 at most 51% of the cases. When allowing the algorithm to perform

---

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

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

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

Table 1
Test results without manipulation of the data

	date	date	k		date	date	k
	$\alpha=0$	$\alpha=2$			$\alpha=0$	$\alpha=2$	
kieftest	96%	81%	0.2	az052	100%	80%	0.0
germ001	98%	82%	0.0	az526	100%	95%	0.1
germ003	100%	94%	0.0	SET01	94%	63%	0.4
germ004	96%	75%	0.2	SET02	96%	63%	0.4
germ006	100%	88%	0.0	oh004	100%	72%	0.1
cana030	94%	78%	0.0	swed302	98%	86%	0.0

Table 2
Test results with a random deletion of one sample element and different values for α

α	kieftest			germ001			cana030			az052		
	date	date & edit	k	date	date & edit	k	date	date & edit	k	date	date & edit	k
0	25%	0%	0.0	51%	0%	0.0	27%	0%	0.0	48%	0%	0.0
1	77%	66%	1.0	94%	91%	1.0	87%	80%	1.0	92%	87%	1.0
2	76%	67%	1.1	92%	89%	1.0	83%	79%	1.0	92%	86%	1.0
3	70%	61%	1.2	91%	87%	1.0	79%	74%	1.0	92%	86%	1.0
4	65%	57%	1.4	90%	87%	1.0	72%	69%	1.0	91%	85%	1.0
5	60%	53%	1.5	89%	84%	1.0	66%	64%	1.2	91%	85%	1.0

Table 3
Test results with a random deletion or a random insertion of one sample element

	One random deletion				One random insertion			
	date	date	date & edit	k	date	date	date & edit	k
	$\alpha=0$		$\alpha=2$		$\alpha=0$		$\alpha=2$	
kieftest	25%	76%	67%	1.1	22%	72%	59%	1.0
germ001	51%	92%	89%	1.0	52%	93%	89%	1.0
germ003	58%	94%	83%	1.0	56%	94%	83%	1.0
germ004	47%	85%	78%	1.2	49%	87%	78%	1.0
germ006	21%	91%	84%	1.0	19%	91%	83%	1.0
cana030	27%	83%	79%	1.0	29%	76%	69%	1.0
az052	48%	92%	86%	1.0	47%	98%	89%	1.0
az526	36%	81%	74%	1.0	36%	83%	74%	1.0
SET01	26%	64%	53%	1.0	18%	65%	56%	1.0
SET02	33%	61%	44%	1.1	28%	60%	53%	1.1
oh004	47%	85%	83%	1.0	47%	87%	85%	1.0
swed302	40%	85%	69%	1.0	32%	75%	65%	1.1

some editations by choosing α, e.g., to be 2 or 3 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 many double or missing rings expected in a tree ring sequence, a small value of α (e.g., 2 or 3) should be sufficient most of the times.

Tables 3 and 4 show test results for different data collections with one random deletion, one random splitting and two random consecutive deletions. Although the percentages

Table 4
Test results with a random deletion of two consecutive sample elements

	date	date	date & edit	k	distance betw. edits
	$\alpha = 0$			$\alpha = 3$	
kieftest	14%	71%	58%	2.1	2.8
germ001	56%	91%	85%	2.0	2.2
germ003	54%	92%	74%	2.0	3.5
germ004	38%	83%	73%	2.1	3.0
germ006	16%	89%	78%	2.1	2.5
cana030	20%	77%	70%	2.0	2.6
az052	42%	92%	88%	2.0	2.0
az526	38%	82%	76%	2.0	2.6
SET01	26%	60%	53%	2.0	4.7
SET02	30%	58%	40%	2.1	3.2
oh004	43%	77%	73%	2.0	2.2
swed302	35%	80%	64%	2.0	2.9

for a correct dating with a correct editation vary from 40% to 89%, 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 possible missing or double rings than the standard crossdating methods do. Concerning two consecutive deletions, Table 4 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 occurrence of double rings. The reason for that is that there is a chance to visually identify a double ring on the wood (e.g., after some more preparation of the wood or using a better microscope), whereas for a missing ring there is not. We therefore restricted the tests on data with real inconsistencies to data containing missing rings.

Table 5 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 *date & edit* column contains the number of samples (and also the percentage relative to the *# of*

Table 5
Test results for data with real missing rings; $\alpha = 4$

	# of samples	ave. # of miss. rings	date	date & edit
wa067	19	2.53	$16 \cong 84\%$	$12 \cong 63\%$
wa069	13	2.44	$12 \cong 92\%$	$8 \cong 62\%$
wa072	19	4.76	$14 \cong 74\%$	$12 \cong 63\%$
wa079	23	2.22	$17 \cong 74\%$	$15 \cong 65\%$
breclav	7	4.40	$7 \cong 100\%$	$5 \cong 71\%$
chin04	13	4.66	$12 \cong 92\%$	$9 \cong 69\%$
az052	6	3.33	$5 \cong 83\%$	$4 \cong 67\%$
az526	14	3.71	$13 \cong 93\%$	$6 \cong 43\%$

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.

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 a $\theta((m + n) \log(n + m))$ algorithm. We presented a $\mathcal{O}(\alpha^2 mn + \alpha^4(m + n))$ crossdating algorithm which takes the possibility of up to α missing or 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. It 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 the evaluation of 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 offers some good datings, the program should be eligible to serve as an additional dating tool searching for missing and double rings.

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

References

[1] P.C. Van Deusen, A dynamic program for cross-dating tree rings, Canad. J. Forest Res. 20 (1989) 200–205.
[2] D.F. Elliott, K.R. Rao, Fast Transforms—Algorithms, Analyses, Applications, Academic Press, New York, 1982.

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

[4] R.L. Holmes, Computer-assisted quality control in tree-ring dating and measurement, Tree-Ring Bull. 43 (1983) 69–75.

[5] J.B. Kruskal, D. Sankoff, An anthology of algorithms and concepts for sequence comparison, in [6].

[6] J.B. Kruskal, D. Sankoff (Eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, Reading, MA, 1983.

[7] H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, Springer, Berlin, 1981.

[8] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, 2dn Edition, Cambridge University Press, Cambridge, 1992.

[9] F. Rinn, TSAP Reference Manual. Heidelberg, Berlin, http://ourworld.compuserve.com/homepages/frankrinn/.

[10] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoustics Speech Signal Process. 26 (1) (1978) 43–49.

[11] F.H. Schweingruber, Trees and Wood in Dendrochronology, Springer, Berlin, 1993.

[12] G.A. Stephen, String Searching Algorithms, World Scientific, Singapore, 1994.

Further reading

[1] C. Wenk, Algorithmen für das Crossdating in der Dendrochronologie, Master's Thesis, Freie Universität Berlin, Institut für Informatik, 1997.

[2] C. Wenk, Applying an edit distance to the matching of tree ring sequences in dendrochronology, in: M. Crochemore, M. Paterson (Eds.), Combinatorial Pattern Matching (CPM99), in: Lecture Notes Comput. Sci., Vol. 1645, 1999, pp. 223–242.