Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik

Diplomarbeit

Algorithmen für das Crossdating in der Dendrochronologie

von
Carola Wenk

Wissenschaftliche Leitung und Betreuung:
Prof. Dr. H. Alt

Dezember 1997

Zusammenfassung

In der Dendrochronologie werden Holzproben anhand ihrer Jahresringe datiert. Die für ein spezielles Jahrringcharakteristikum, wie die Jahrringbreite, aufgestellte Holzprobenjahrringfolge wird in einer datierten Referenzfolge durch eine eindimensionale Mustererkennung (*Crossdating*) lokalisiert und auf diese Weise datiert.

In dieser Arbeit wird das Crossdating von algorithmentheoretischer Sicht untersucht. Dazu werden zunächst bisher verwendete Crossdatingverfahren vorgestellt, effizientere Algorithmen für diese angegeben sowie ein neuer Crossdatingalgorithmus entwickelt.

Bisherige Crossdatingverfahren suchen in der Regel sukkzessive eine der Musterfolge möglichst ähnliche Folge an allen Positionen in der Referenzfolge. Diese sukkzessive Herangehensweise kann durch die Verwendung der schnellen Fourier Transformation (FFT) ersetzt und das Crossdating dadurch effizienter gestaltet werden. Die dabei verwendeten Abstandsbegriffe gehen jedoch davon aus, daß die Zeitskalen von Muster- und Referenzfolge übereinstimmen. Ungleichmäßigkeiten wie fehlende und doppelte Ringe werden nicht berücksichtigt und stellen eine Fehlerquelle für eine korrekte Datierung dar. Deshalb wird in dieser Arbeit ein Transformationsabstand vorgestellt, der Ungleichmäßigkeiten in der Musterfolge durch eventuell auftretende fehlende oder doppelte Jahresringe berücksichtigt. Weiterhin wird ein auf diesem Abtandsbegriff basierender neuer Crossdatingalgorithmus entwickelt, dessen Implementation ein Teil dieser Arbeit ist.

Für die Unterstützung bei dieser Arbeit bedanke ich mich bei:

Deutsches Archäologisches Institut (Eurasien-Abteilung)	Dr. U. Heußner
Freie Universität Berlin	Prof. Dr. Helmut Alt Christof Schultz
Technische Universität Berlin	Dipl.-Ing. Konstantin Lenz Robert Pohl

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Dendrochronologie

Schon vor mehr als vierhundert Jahren erkannte Leonardo da Vinci (1452 - 1519), daß Wachstumsringe von Bäumen einen jährlichen Charakter aufweisen. Seit der Antike wußte man zwar, daß Bäume wachsen, indem sie Schichten von frischem Holz unter der Rinde ausbilden, aber der jährliche Zusammenhang war noch unbekannt. Eine weiterführende Entdeckung Leonardo da Vincis war, daß die Breite eines solchen Jahresringes mit der in diesem Jahr vorhandenen Feuchtigkeit zusammenhing. So konnte er Aussagen über das Wetter vergangener Jahre, speziell über trockene und niederschlagsreiche Klimaperioden, machen.

Heutzutage werden Jahresringe zur Erforschung verschiedenster naturhistorischer Begebenheiten herangezogen, da sie abhängig von den jährlichen Umweltbedingungen gewachsen sind und deshalb eine Art Archiv der Umweltbedingungen der entsprechenden Jahre darstellen. Der Forschungszweig, der sich mit der Datierung von Jahresringen zur Beantwortung naturhistorischer Fragen beschäftigt, ist die *Dendrochronologie*. Als ihr Begründer gilt der amerikanische Astronom A.E. Douglass. Er stellte am Anfang dieses Jahrhunderts die einfachen dendrochronologischen Grundsätze in Beziehung zu anderen Wissenschaften, wie der Geschichtsforschung, der Klimatologie und der Astronomie. Auf dieser Basis entwickelte er die Dendrochronologie zu einer Wissenschaft, deren Namen er aus den griechischen Wörtern *dendron* (Holz), *chronos* (Zeit) und *logos* (die Wissenschaft von) zusammensetzte.

Eine wichtige Anwendung der Dendrochronologie liegt in der Erforschung des Klimas in der Vergangenheit. Da es instrumentelle Wetterbeobachtungen höchstens für die letzten dreihundert Jahre gibt, sind Jahrringe zusammen mit anderen natürlichen Umweltarchiven wie Seesedimenten oder Gletscherablagerungen ein willkommenes Mittel, um Aussagen über die jährliche Entwicklung des Klimas in der Vergangenheit zu machen. Daraus lassen

sich dann auch Prognosen über das zukünftige Klima herleiten.

In der Archäologie werden dendrochronologische Methoden zur Datierung von historischem Holz verwendet. Anhand der in dem Holz auftretenden Jahresringe können Entstehungsdaten von kunsthistorischen Objekten wie Möbeln, Musikinstrumenten oder Gemäldetafeln aber auch von historischen Gebäuden bis auf einige Jahre genau angegeben werden. Die vorhandenen Jahresringe werden dabei genau datiert, jedoch geht das Enstehungsdatum des entsprechenden kunsthistorischen Objekts bzw. des historischen Gebäudes nicht ganz genau daraus hervor, da bei vielen Holzbearbeitungsverfahren die Rinde sowie die äußeren Holzschichten abgeschlagen werden oder das Holz einige Jahre getrocknet wird. Deshalb gibt der letzte auf dem Holzstück vorhandene Jahrring nur bis auf einige Jahre genau das gesuchte Datum an.

Die Genauigkeit der Jahrringdatierungen führte zu einer Eichung der in den historischen Wissenschaften verbreiteten Methode der Radiokarbondatierung. Bei dieser Methode werden die Anteile des radioaktiven Kohlenstoffisotops C^{14} in der zu datierenden Probe gemessen und anhand der Halbwertzeit des Isotops das Alter der Probe bestimmt. Dabei wird aber davon ausgegangen, daß der Anteil des C^{14}-Isotops in der Luft in allen zur Untersuchung kommenden Jahrhunderten und Jahrtausenden weltweit konstant war. Die Dendrochronologie widerlegte diese Annahme und lieferte eine Eichung der bisherigen Radiokarbonzeitachse [49].

Die Dendrochronologie liefert in vielen anderen Bereichen Aussagen über die Umwelt in der Vergangenheit, wie etwa über die Ausdehnung und Bewegung von Gletschern, die Entstehung und Dynamik von Landmassen oder die Veränderungen von Flußverläufen und Wasserspiegeln. Einen guten Überblick bietet Schweingruber in [45] und [46].

Die Vorgehensweise der Dendrochronologie besteht darin, für jede zu untersuchende Holzprobe eine Jahrringfolge bezüglich eines bestimmten Jahrringcharakteristikums, wie etwa Breite oder Holzdichte, aufzustellen. Folgen solcher Art, deren Folgenglieder aufeinanderfolgenden Zeitpunkten entsprechen, werden auch als *Zeitreihen* bezeichnet.

Da Bäume bei gleichartigen Umweltbedingungen ein sehr ähnliches Wachstumsverhalten aufweisen, können solche Jahrringfolgen untereinander verglichen und einander entsprechende Jahrringmuster identifiziert werden. Ein solcher Vergleich zweier Jahrringfolgen wird als *Crossdating* bezeichnet und stellt den Kern der Dendrochronologie dar. Ist eine Jahrringfolge bereits datiert, d.h. es ist für jeden Jahrring sein Entstehungsjahr bekannt, eine andere Folge ist jedoch undatiert, so kann mit Hilfe des Crossdatings in beiden Folgen der gemeinsame Lebensabschnitt der zugrundeliegenden Bäume identifiziert und die zweite Folge aufgrund dessen datiert werden. Für eine eingehendere Erläuterung der Verfahrensweisen in der Dendrochronologie wird auf Kapitel 2 verwiesen.

1.2 Ziel und Aufbau dieser Arbeit

Beim Crossdating handelt es sich um eine approximative eindimensionale Mustererkennung. In dieser Diplomarbeit wird untersucht, inwiefern sich bisher verwendete Crossdatingverfahren durch Ergebnisse aus der Informatik verbessern lassen, und ob es bisher nicht genutzte Verfahren der approximativen Mustererkennung gibt, die sich für das Crossdating eignen.

Dazu werden die Grundlagen und Methoden der Dendrochronologie, speziell des Crossdatings, in Kapitel 2 erläutert. In Kapitel 3 werden in der Dendrochronologie häufig verwendete Crossdatingverfahren vorgestellt. Da die gebräuchlichen Crossdatingverfahren ein eventuelles Auftreten von fehlenden oder doppelten Ringen nicht berücksichtigen, wird in Kapitel 4 kurz auf spezielle Verfahren zur Lokalisierung solcher Unregelmäßigkeiten eingegangen.

Kapitel 5 beschäftigt sich mit Algorithmen für die *schnelle Fourier Transformation (FFT)* und beschreibt deren Anwendung auf die Crossdatingverfahren mittels Korrelations- und Gleichläufigkeitskoeffizienten zur Verringerung der Laufzeiten.

In Kapitel 6 wird ein Abstandsbegriff für Jahrringfolgen vorgestellt, der dem der Edit Distance für Zeichenketten oder dem Dynamic Time Warping in der Spracherkennung sehr ähnlich ist. Für diesen von Van Deusen [11] vorgeschlagenen Abstand wird durch eine andere Art der Einschränkung der Transformationsmenge ein von der Anzahl der Editieroperationen abhängiger Abstand definiert, der sich für die Anwendung in der Dendrochronologie besser eignet. Aufbauend auf diesem neuen Abstandsbegriff für Jahrringfolgen wird in Kapitel 7 ein Crossdatingverfahren vorgestellt, das Ungleichmäßigkeiten in der Musterjahrringfolge, wie fehlende und doppelte Ringe, berücksichtigt. Eine Implementierung dieses Crossdatingverfahrens in einem *C++* - Programm ist auf Diskette beigefügt sowie in Anhang C abgedruckt.

Kapitel 2

Grundlagen der Dendrochronologie

2.1 Jahrringwachstum

Die bei einem Querschnitt eines Baumstammes erkennbaren Ringe sind Wachstumsschichten des Baumes. Diese werden in dem teilungsfähigen Gewebe unterhalb der Rinde, dem Kambium, während der Vegetationszeit gebildet. In Gebieten mit jährlicher Vegetations- und Winterperiode werden solche Wachstumsschichten in der Regel jährlich angelegt und deshalb als *Jahres-* bzw. *Jahrringe* bezeichnet. Abbildung 2.1 zeigt eine schematische Darstellung eines Baumstammstückes mit einigen üblichen Bezeichnungen.

Am Anfang der Vegetationsperiode legen Bäume zur Nährstoffversorgung der Triebe große, dünnwandige Zellen an, während gegen Ende der Vegetationsperiode kleine, dickwandige Zellen zur Festigung des Stammes gebildet werden. Die zuerst genannte Art von Zellen, die man anhand ihrer hellen Färbung erkennen kann, wird als *Frühholz*, die letztere, deren Färbung dunkler ist, als *Spätholz* bezeichnet. Die Wachstumsschicht eines Jahres, also ein Jahrring, besteht demnach aus Frühholz und Spätholz, wobei je nach Baumart Früh- und Spätholz verschieden stark ausgeprägt sein können, und der Übergang mehr oder weniger fließend sein kann. Aufgrund des Helligkeitsunterschieds zwischen dem Spätholz eines Jahrrings und dem Frühholz seines Folgejahrrings können einzelne Jahrringe voneinander abgegrenzt werden, was in Abbildung 2.2 an einem Beispiel deutlich wird.

Das jährliche Wachstum eines Baumes ist abhängig von vielerlei Umwelteinflüssen, die sowohl von globaler als auch von individueller sowie von kurz- und von längerfristiger Art sein können. Die aus dem Wachstum hervorgehenden Jahresringe variieren, entsprechend der jährlich wechselnden Umweltbedingungen, in ihren verschiedenen Merkmalen wie Breite oder Holzdichte. Im wesentlichen hängt das Wachstum von der in dem betref-

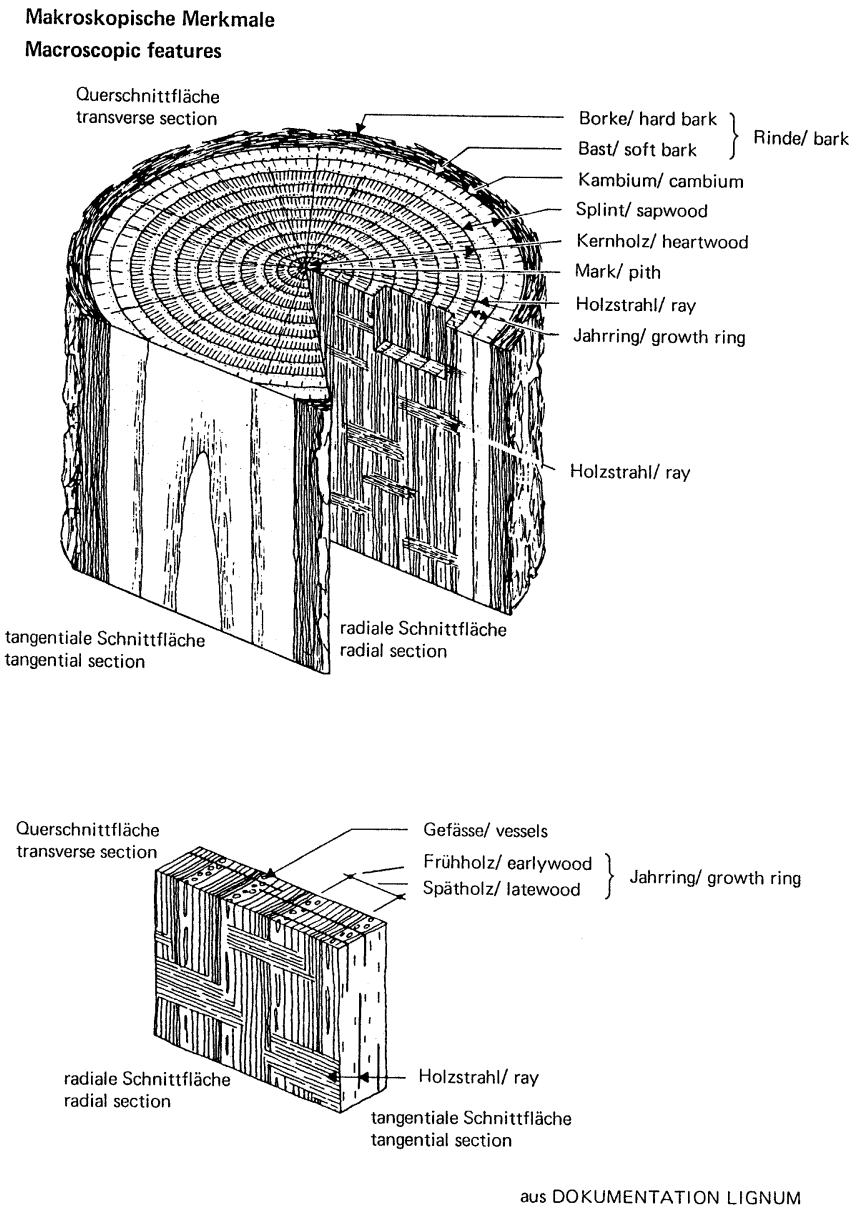

Abbildung 2.1: Makroskopische Merkmale eines Baumstammstückes (aus [45])

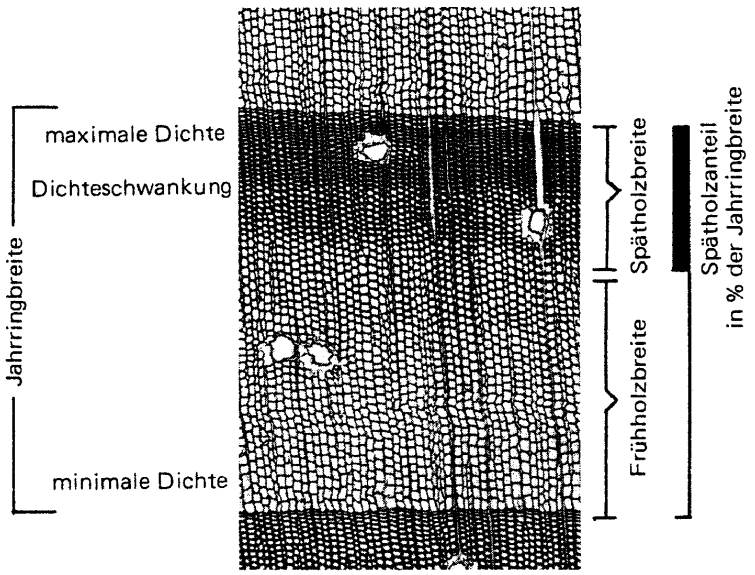

Abbildung 2.2: Verschiedene Merkmale eines Jahrrings (aus [45])

fenden Jahr herrschenden Witterung ab, insbesondere von Niederschlag und Temperatur. Aber auch das Klima der vorhergehenden Jahre hat durch länger wirksame Faktoren wie Knospenanlagen, Reservestoffbildung und Wurzelwachstum Einfluß auf das Wachstum. Abgesehen von der Witterung wirken sich viele weitere kurz- und längerfristige Umweltfaktoren aus. Einige dieser Faktoren sind forstliche Maßnahmen wie Düngen oder Holzeinschlag, aber auch natürliche wie Schädlingsbefall, Luft- und Wasserverunreinigungen oder Unterdrückung des Baumes durch andere Bäume. Weitere Erläuterungen dazu gibt zum Beispiel Schweingruber [46].

Bei Baumarten, die empfindlich auf Umwelteinflüsse reagieren, kann es vorkommen, daß in einem Jahr mit sehr schlechten Wachstumsbedingungen die Wachstumsschicht nicht um den gesamten Stammumfang bzw. überhaupt keine Wachstumsschicht ausgebildet wird. Eine solche Extremreaktion auf schlechte Wachstumsbedingungen tritt innerhalb eines depressiven Zeitraumes von einigen Jahren auf, wo die Wachstumsbedingungen im ganzen Zeitraum schlecht und deshalb die Jahresringe schmal ausgebildet sind.

Im Gegensatz zu einem solchen *fehlenden Jahrring* kann ein Baum unter gewissen Bedingungen zwei oder mehr Wachstumsschichten in einem Jahr ausbilden. Wenn zum Beispiel ein recht kalter Sommer von einem warmen Herbst gefolgt ist, kann es passieren, daß der Baum im Frühjahr Frühholz, im Sommer Spätholz und in dem zu warmen Herbst wiederum Frühholz gefolgt von Spätholz ausbildet. Die zwei so zustandegekommenen Wachstumsschichten sind teilweise nicht von zwei „echten“ Jahrringen zu unterscheiden. Eine solche Konstellation wird als *doppelter Jahrring* bezeichnet.

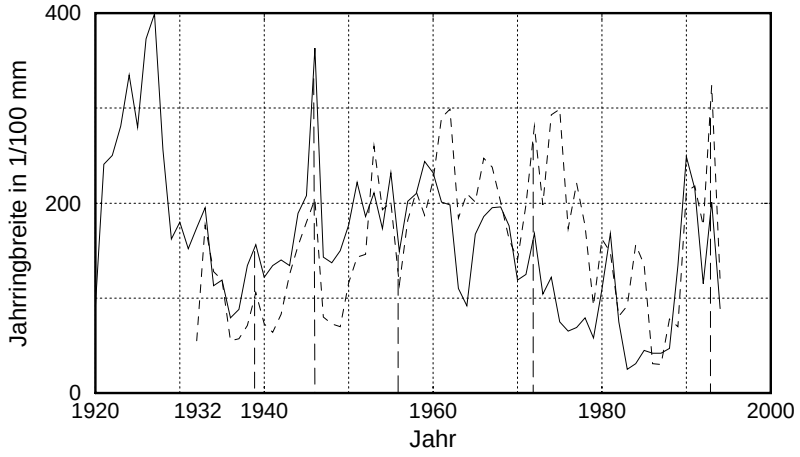

Abbildung 2.3: Jahrringbreitenkurven zweier nahe beieinander lebender Kiefern aus dem Berliner Umland. Daten aus dem Deutschen Archäologischen Institut, Eurasien-Abteilung.

Insgesamt bildet der in den Jahresringen eines jeden Baumes gespeicherte Wachstumsverlauf ein „biologisches Archiv“ früherer Umweltbedingungen. Die Auswertung dieser Archive zur Beantwortung naturhistorischer Fragen ist der Inhalt der Dendrochronologie.

2.2 Crossdating

Um den Wachstumsverlauf von Bäumen geeignet auswerten und vergleichen zu können, werden verschiedene Jahrringmerkmale betrachtet. Das augenscheinlichste Merkmal, die Jahrringbreite, ist mit wenig Aufwand meßbar und wird häufig verwendet. Andere Charakteristika sind zum Beispiel die Breite des Früh- oder Spätholzes oder die Holzdichte. In Abbildung 2.2 sind einige Merkmale aufgeführt. Welches Charakteristikum zur Untersuchung herangezogen wird, hängt von der jeweiligen Anwendung ab. In dieser Arbeit wird als Jahrringcharakteristikum die Jahrringbreite verwendet. Die Untersuchungen in dieser Arbeit lassen sich mit leichten Modifikationen auch auf einige andere Charakteristika übertragen; diese müssen dafür jedoch jahrringweise vorliegen.

Bäume reagieren in ihrem Wachstum auf gleichartige Umweltbedingungen sehr ähnlich. Die Wachstumsverläufe zweier Bäume derselben Spezies, die zur selben Zeit innerhalb eines gewissen Umkreises und deshalb unter ähnlichen klimatischen Bedingungen gewachsen sind, ähneln sich deshalb in gewisser Weise. Geübte Dendrochronologen können solche Gemeinsamkeiten teilweise direkt anhand der Jahresringe erkennen. In der Regel wird aber für jeden Baum bzw. Holzprobe eine Folge der Breiten seiner Jahresringe aufgestellt. Veranschaulicht wird eine solche Folge wie in Abbildung 2.3 als Kurve in einem Zeit-Jahrringbreiten-Koordinatensystem, indem die (x, y)-Koordinaten aufgetragen und von Jahr zu Jahr durch eine Linie verbunden werden.

In Abbildung 2.3 sind die Jahrringbreiten-Folgen von zwei nahe beieinander lebenden Kiefern aus einem Ort im Berliner Umland abgebildet. Der relative Verlauf von einem Jahr zum anderen (steigend oder fallend) sowie die Stellen der lokalen Minima und Maxima stimmen bei beiden Kurven zum größten Teil überein (z.B. 1939, 1946, 1956, 1972, 1993), ebenso der Langzeittrend mit niedrigen Werten um 1940, höheren Werten um 1960 und wieder niedrigen Werten um 1985.

Die absoluten Werte gleichen sich jedoch nicht völlig und der relative Verlauf sowie die lokalen Extrema stimmen nicht an allen Stellen überein. Diese Ungleichheiten rühren daher, daß sich auf jeden Baum zusätzlich zu globalen Umweltbedingungen, wie etwa der Witterung, auch viele individuelle Wachstumsbedingungen, wie z.B. Schädlingsbefall oder Konkurrenz, auswirken können.

Einen genauen Ähnlichkeitsbegriff solcher Kurven formal zu definieren ist schwierig, da viele verschiedene Faktoren auf die Bäume einwirken können. In der Praxis werden meistens in gewisser Form die oben übereinstimmenden Merkmale, also der relative Verlauf, lokale Extrema und kurz- bis längerfristige Trends, als Vergleichskriterien verwendet.

Zum Vergleich ist in Abbildung 2.4 die gestrichelte Kurve um ein Jahr nach rechts verschoben aufgetragen worden. Diese beiden Kurven haben offensichtlich wenig gemeinsam, da der Verlauf der lokalen Extrema nicht übereinstimmt.

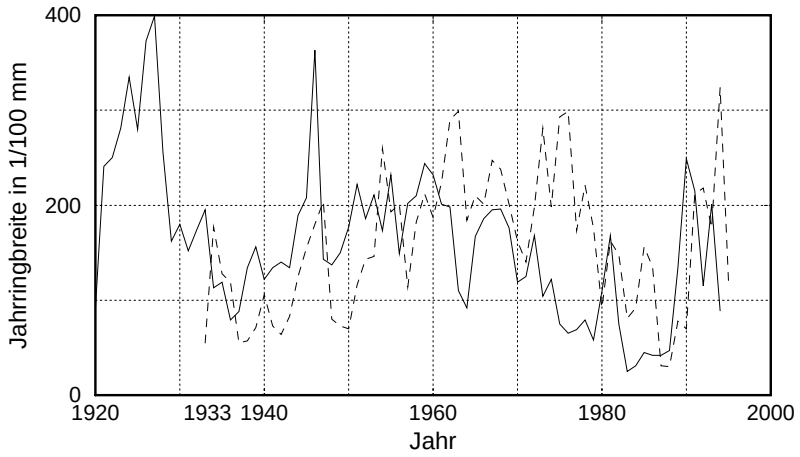

Abbildung 2.4: Jahrringbreitenkurven zweier nahe beieinander lebender Kiefern aus dem Berliner Umland mit der gestrichelten Kurve um ein Jahr nach rechts verschoben. Daten aus dem Deutschen Archäologischen Institut, Eurasien-Abteilung.

Der Verlauf der Jahrringbreiten eines Baumes ist in gewisser Weise für einen bestimmten Zeitabschnitt charakteristisch. Mit Hilfe geeigneter Vorverarbeitung der Jahrringdaten kann eine undatierte Jahrringbreiten-Folge in einer datierten Folge, sofern Spezies, Standort und Lebenszeitraum der Bäume miteinander harmonieren, durch Vergleich bzw.

Matching lokalisiert und somit datiert werden.

Wäre die in den Abbildungen 2.3 und 2.4 gestrichelt dargestellte Folge undatiert, so könnte sie anhand der anderen Folge datiert werden, indem sie nacheinander an allen möglichen Anfangsjahren aufgetragen und mit der anderen Folge verglichen würde. Mit dem Anfangsjahr 1932 würde dann die mit Abstand größte Übereinstimmung auftreten. Davon ausgehend, daß die Folge nur aus Jahrringbreiten von *aufeinanderfolgenden* Jahren besteht, wäre dann jedes Jahr der Folge bestimmt und diese somit *datiert*.

Diese Technik des Datierens einer Jahrringfolge durch den Vergleich mit einer datierten Jahrringfolge wird in der Dendrochronologie als *Crossdating* bezeichnet. Im mathematischen Sinne handelt es sich dabei um eine *approximative Mustererkennung* mit der undatierten Jahrringfolge als Muster, dessen ungefähres Auftreten in der datierten Referenzfolge gesucht ist.

Obwohl zwei Jahrringkurven augenscheinlich doch viele Unterschiede aufweisen, werden mit Hilfe einiger Vorverarbeitung der Daten, die Langzeittrends herausfiltern und Mittelwerte und Varianzen standardisieren, sehr gute Matching-Ergebnisse erzielt. Es sind zwar nicht immer alle Jahrringfolgen datierbar, da manche Folgen für eine gute Matchingaussage zu kurz oder nicht aussagekräftig genug sind, aber dennoch liefern die in Kapitel 3 beschriebenen Verfahren in den meisten Fällen ausreichend signifikante Matchingergebnisse. Ein gewisser Nachteil ist jedoch, daß keines der Verfahren fehlende oder doppelte Ringe berücksichtigt, wodurch eine „Verunreinigung“ der entsprechenden Chronologie bzw. schlicht keine Datierung einer Folge mit solchen Fehlbildungen möglich ist.

2.3 Jahrringchronologien

Wie bereits in Unterkapitel 2.1 erwähnt ist ein Baum sowohl individuellen als auch generellen Wachstumsfaktoren ausgesetzt. Je mehr individuelle Faktoren sich auf das Wachstum auswirken, desto schwieriger kann der Wachstumsverlauf mit dem anderer Bäume verglichen werden, da sich die individuellen Faktoren wie eine Art Störsignal in der Jahrringbreitenfolge wiederspiegeln. Deshalb ist es von Vorteil, pro Standort und Spezies eine Referenzfolge zur Verfügung zu haben, die zum größten Teil den gemeinsamen Wachstumsbedingungen entspricht, um bestmögliche Vergleiche durchführen zu können.

Eine solche Referenzfolge wird aus möglichst vielen Jahrringbreitenfolgen durch jährliche Mittelwertbildung gebildet und als *Jahrringchronologie* bezeichnet. Sie entsteht in der Regel durch sukkzessive Hinzunahme weiterer Jahrringbreitenfolgen, die zuvor mittels Crossdating an der bisherigen Jahrringchronologie datiert wurden. Häufig werden die Jahrringfolgen vor der Zusammenfassung zu einer Chronologie gewissen Transformationen, sogenannten Standardisierungen unterworfen, auf die in Unterkapitel 2.4 näher eingegangen wird.

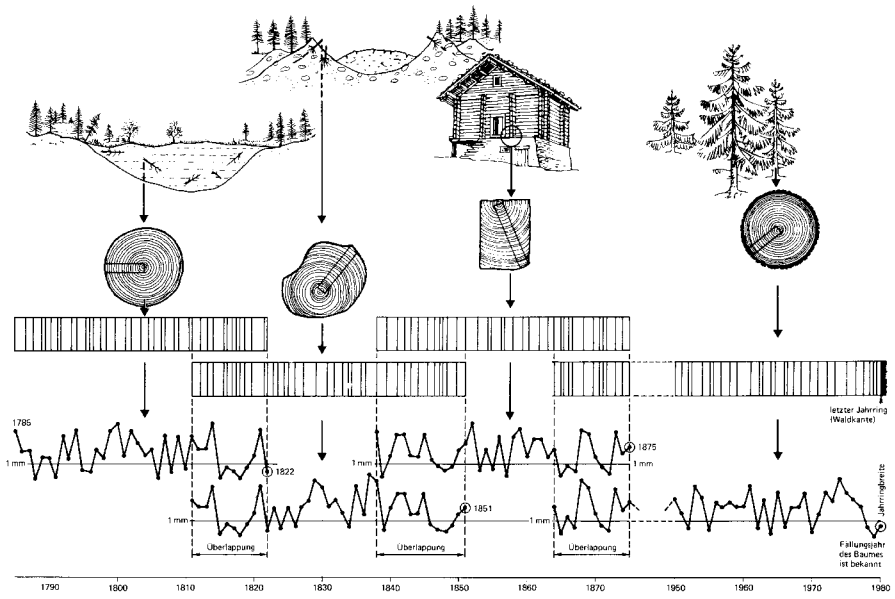

Abbildung 2.5: Schematische Darstellung des Überbrückungsverfahrens zur Erzeugung einer langen Jahrringchronologie (aus [45])

Um die Länge einer Jahrringchronologie nicht auf das Lebensalter der verwendeten Baumarten zu beschränken, sondern möglichst weit auszudehnen, wird das sogenannte *Überbrückungsverfahren* angewendet. Dazu wird in der Regel von einem datierten Holzstück der Gegenwart (z.B. von einem Baum, dessen Fälldatum bekannt ist) ausgegangen, und sukkzessiv ältere Holzproben zum Vergleich hinzugenommen, deren gemeinsamer Lebenszeitraum mindestens einige Jahrzehnte umfaßt. Zwei solche Jahrringbreitenfolgen überlappen sich, weshalb innerhalb dieses Überlappungsintervalls (*Brücke*) mittels des Crossdatings eine Datierung der einen Folge anhand der anderen vorgenommen werden kann. In Abbildung 2.5 ist das Überbrückungsverfahren anhand eines Beispiels verdeutlicht.

Auf diese Art und Weise können Jahrringchronologien ausgehend von der Gegenwart so weit schrittweise in die Vergangenheit verlängert werden, wie geeignetes Holz zur Verfügung steht. Solche Chronologien können dann mehrere tausend Jahre umfassen. Die dabei auftretenden Brücken sollten möglichst lang sein, um Fehldatierungen zu vermeiden.

Zur Realisierung dieses Überlappungsverfahrens wird beim Crossdating also nicht nur versucht, die ganze undatierte Jahrringfolge in der datierten Referenzfolge zu lokalisieren, sondern am Anfang der Referenzfolge auch alle Suffixe, die eine gewisse Mindestlänge überschreiten, sowie am Ende der Referenzfolge auch alle Präfixe, die ebenfalls die Mindestlänge überschreiten. Die Mindestlänge richtet sich dabei nach der Länge eines sinnvollen Überlappungsintervalls, was zwischen 20 und 100, aber oft bei etwa 50 Folgengliedern /

Jahren liegt. Ebenso ist es nicht sinnvoll, ganze Jahrringfolgen miteinander zu vergleichen, deren Länge kürzer als die sinnvolle Mindestlänge sind.

2.4 Standardisierung der Daten

2.4.1 Notwendigkeit von Standardisierungen

Jahrringfolgen werden vor einer weitergehenden Betrachtung fast immer bestimmten Transformationen unterworfen. Diese heben Schwankungen bestimmter Art in den Folgen hervor und schwächen andere wiederum ab mit dem Ziel, Mittelwerte und Varianzen der Folgen zu standardisieren. In der Sprache der Dendrochronologie werden solche Transformationen als *Standardisierungen* bzw. *Indexierungen* und die transformierten Folgen als *Indexfolgen* bezeichnet. Transformationen, die kurzfristige Schwankungen hervorheben und langfristige eliminieren, heißen *Hochpaßfilter*, und solche, die kurzfristige Schwankungen eliminieren und nur die Langzeittrends hervorheben, heißen *Tiefpaßfilter*.

Zum einen kann eine Zusammenfassung mehrerer unstandardisierter Folgen zu einer Chronologie zu Langzeittrends innerhalb der Chronologie führen, die als solche in keiner der Einzelfolgen enthalten sind. Solche künstlich entstandenen Langzeittrends können Vergleiche mit weiteren Jahrringfolgen verfälschen. Durch eine vorherige Standardisierung jeder Einzelfolge werden solche künstlichen Langzeittrends vermieden. Eine andere Möglichkeit künstliche Langzeittrends zu vermeiden besteht darin, eine möglichst hohe Anzahl von Einzelfolgen zu einer Chronologie zusammenzufassen, so daß sich unerwünschte Langzeittrends gegenseitig ausgleichen.

Zum anderen werden Standardisierungen für das Crossdating zweier Folgen benötigt. Für das Crossdating sind ausschließlich kurz- bis mittelfristige Schwankungen der Jahrringfolgen von Interesse, da diese häufig bei vielen Bäumen in ähnlicher Weise auftreten. Langfristige Schwankungen, die zum Beispiel vom zunehmenden Alter des Baumes (*Alterstrends*) oder von Konkurrenzbedingungen im Wachstum herrühren können, sind für einen Baum individuelle Merkmale, die die Vergleiche mit anderen Jahrringfolgen erschweren. Beispielsweise kann es ohne Eliminierung der Alterstrends schwierig sein, ältere mit jüngeren Bäumen zu Vergleichen. Deshalb werden für das Crossdating in der Regel extreme Hochpaßfilter verwendet, die oft nur Jahr-zu-Jahr-Trends übrig lassen.

2.4.2 Log-Standardisierung

In der Regel treten in Jahrringfolgen mehr schmale als breite Jahresringe auf, jedoch wirken Unterschiede zwischen breiten Jahresringen fälschlicherweise gravierender als zwischen schmalen Ringen. Deshalb wird bei dem Vergleich zweier Folgen sehr häufig ein Logarithmus der einzelnen Folgenwerte betrachtet.

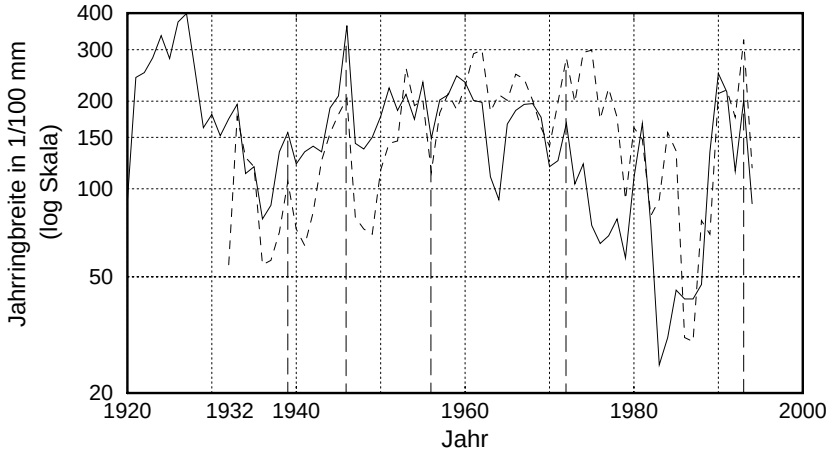

Abbildung 2.6: Jahrringbreitenkurven zweier Kiefern aus dem Berliner Umland mit logarithmischer y-Skala. Daten aus dem Deutschen Archäologischen Institut, Eurasien-Abteilung.

Für einen optischen Vergleich werden dafür die Folgen in einem Diagramm mit halblogarithmischer Skala aufgetragen. Durch diese logarithmische Darstellung bzw. Standardisierung werden Schwankungen innerhalb geringer Jahrringbreiten hervorgehoben und innerhalb breiter Jahrringe abgeschwächt. In Abbildung 2.6 sind die beiden Kurven aus Abbildung 2.3 mit einer logarithmischen y-Skala aufgetragen.

2.4.3 Standardisierung durch Glättungskurven

Viele verwendete Standardisierungsverfahren beruhen auf dem Prinzip, daß der extreme Langzeitverlauf einer Kurve mit Hilfe einer Ausgleichs- bzw. Glättungskurve ausgedrückt, und danach jeder Folgenwert durch den Wert der Ausgleichskurve an dieser Stelle geteilt wird. [1]

Für die Berechnung einer Ausgleichskurve gibt es deterministische und stochastische Verfahren. Bei den deterministischen Verfahren wird eine vorgegebene Kurve, zum Beispiel eine Gerade oder eine negativ exponentielle Funktion, der Jahrringfolge mit der Methode der kleinsten Quadrate angepaßt. Bei stochastischen Verfahren werden datenabhängigere Ausgleichskurven, wie gleitende Mittel oder Splines, verwendet. Die verschiedenen Möglichkeiten der Standardisierung mit Ausgleichskurven beschreiben Cook und Briffa [8].

Ein 1973 von Baillie und Pilcher [5] vorgestelltes Verfahren verwendet als Glättungskurve ein ungewichtetes gleitendes arithmetisches Mittel. Dabei wird ein Zeit-

[1] Bei Dichte-Betrachtungen wird bei Schweingruber [45] subtrahiert statt dividiert.

fenster, dessen Breite durch eine feste Anzahl von Jahrringwerten bestimmt ist, sukkzessiv über die Jahrringfolge geschoben („gleitend“), wobei die Werte innerhalb des Fensters gemittelt und dem mittleren Zeitpunkt des Fensters zugeordnet werden.

Ist die Jahrringfolge mit $(x_i)_i$ bezeichnet, so errechnet sich die Glättungskurve $(y_i)_i$ demnach durch

$$y_i := \frac{1}{2k+1} \sum_{j=i-k}^{i+k} x_j, \tag{2.1}$$

wenn $l = 2k + 1$ die Länge des Zeitfensters bezeichnet. Die standardisierte Folge $(z_i)_i$ berechnet sich durch jährliche Division durch die Werte der Glättungsfunktion, also durch

$$z_i = \frac{x_i}{y_i}. \tag{2.2}$$

Für die häufig verwendete Intervallänge von 5 liegt das gleitende Mittel sehr dicht an der Jahrringkurve an, weshalb in der standardisierten Kurve viele Langzeittrends entfernt sind. Übrig bleiben fast nur die Jahr-zu-Jahr-Trends. Diese Standardisierungsart wird als „Prozent des l-jährigen gleitenden Mittels“ bezeichnet. Häufig wird zusätzlich hinterher noch eine Log-Standardisierung durchgeführt. Dadurch wird laut Baillie und Pilcher [5] annähernd eine Normalverteilung der Daten erreicht, die zum Beispiel für den Crossdatingalgorithmus in Abschnitt 3.3.4 vorausgesetzt wird.

Allgemein kann durch eine Multiplikation der x_j mit Gewichten w_j und durch eine anschließende Division durch die Summe der w_j statt durch $2k+1$ ein gewichtetes gleitendes arithmetisches Mittel gewonnen werden. Von Cook und Briffa ([8]) wird das ungewichtete gleitende Mittel nicht empfohlen, obwohl es als ein einfach zu programmierendes Standardisierungsverfahren vielfach zu erfolgreichen Datierungen beigetragen hat.

2.4.4 Weitere Standardisierungen

Eine weitere Standardisierungsmöglichkeit ist die Bildung der Differenz der Folgenwerte von einem zum anderen Jahr (diskrete erste Ableitung). Diese ist einfach zu berechnen und stellt einen extremen Hochpaßfilter dar, der als Standardisierung zweier zu vergleichender Folgen verwendet werden kann. Statt der eigentlichen Folgenwerte können auch die Logarithmen der Folgenwerte zur jährlichen Differenzbildung herangezogen werden, was als Wuchswert-Formel bezeichnet wird.

Soll der Wert 0 als Jahrringbreitenwert zugelassen werden, können logarithmische Standardisierungen nicht verwendet werden. Um dies zu umgehen, kann vor einer Logarithmenbildung eine kleine Konstante, zum Beispiel 1, zum Folgenwert addiert, oder auch die von Van Deusen [11] vorgeschlagene Umkehrfunktion des Sinus hyperbolicus $\operatorname{arsinh} x = \ln\left(x + \sqrt{x^2+1}\right)$ verwendet werden.

Kapitel 3

Gebräuchliche Crossdatingverfahren

In diesem Kapitel werden weit verbreitete Crossdatingverfahren, die in mehreren Crossdatingprogrammen implementiert sind, beschrieben. Die Ergebnisse werden in der Regel mit statistischen Mitteln ausgewertet, auf die hier nur am Rande eingangen wird.

3.1 Problemstellung

Es seien zwei (Jahrringbreiten-) Folgen $a = a_0, \ldots, a_{m-1}$ und $b = b_0, \ldots, b_{n-1}$ gegeben. a sei die Referenzfolge bzw. Chronologie und b die in a zu lokalisierende Musterjahrringfolge. Die minimale Länge eines Überlappungsintervalls sei mit $minOvl$ bezeichnet.

Gesucht sind zu b sehr ähnliche, zusammenhängende Teilfolgen in a, unter denen der Benutzer durch visuelles Nachprüfen die eine b tatsächlich entsprechende Folge in a identifizieren und somit b datieren kann. Es handelt sich also um eine approximative Mustererkennung von b in a, für die mehrere mögliche Matchings von b in a zusammen mit einer Güte des Matchings ausgegeben werden sollen.

Die Art, wie sich eine zu b ähnliche Teilfolge in a von b unterscheiden kann, ist in diesem Kapitel ausschließlich auf Veränderungen der einzelnen Folgenwerte beschränkt, d.h., die Länge von b und einer zu b ähnlichen Folge ist gleich, und die jeweils i-ten Folgenglieder entsprechen einander. (Dadurch werden keine fehlenden oder doppelten Ringe betrachtet.) Welche Folgen als ähnlich betrachtet werden hängt von dem verwendeten Abstands- bzw. Ähnlichkeitsbegriff ab.

Zusätzlich werden am linken Rand von a auch Suffixe von b und am rechten Rand von a Präfixe von b für ein Matching betrachtet, wobei die Länge eines Suffixes bzw. Päfixes von

b gleich der Länge des zu vergleichenden Präfixes bzw. Suffixes in a, jedoch mindestens $minOvl$ sein muß. Anschaulich gesehen müssen also an den Rändern von a mindestens $minOvl$ Folgenglieder von b „überlappen“ und die restlichen $n - minOvl$ Folgenglieder links bzw. rechts von a „überhängen“ , wobei letztere nicht in den Vergleich einbezogen werden.

3.2 Grundlegender sukkzessiver Algorithmus

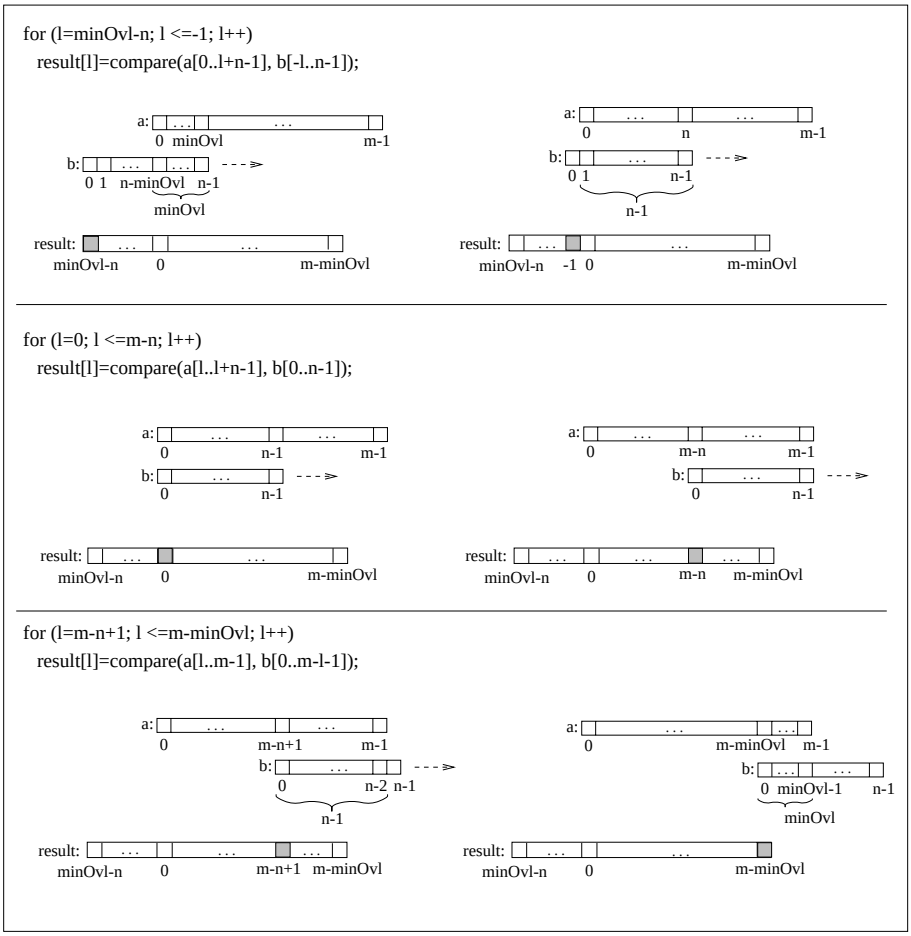

Abbildung 3.1: Sukkzessiver Basisalgorithmus für die meisten Crossdatingverfahren

Um die Position eines Matchings von b in a bestimmen zu können, muß für jede mögliche Überlappungsposition eine Kenngröße berechnet werden, die die Güte eines möglichen Matchings an dieser Stelle angibt. Eine solche Kenngröße kann ein Abstand zwischen zwei Folgen gleicher Länge oder eine statistische Kenngröße wie der Korrelationskoeffizient sein.

In dem sukkzessiven Algorithmus wird dazu anschaulich gesehen das Muster b von links nach rechts über die Referenzfolge a geschoben, wobei mindestens $minOvl$ Folgenglieder überlappen müssen. An jeder Stelle wird die Kenngröße zwischen den einander gegenüberstehenden (Teil-) Folgen berechnet und der Position / Index in r zugeordnet, an dem die Folge b mit ihrem ersten Folgenglied zum Vergleich angelegt wurde. Da mindestens $minOvl$ Folgenglieder überlappen müssen, werden den letzten $minOvl - 1$ Indizes keine Werte zugeordnet. Die teilweisen Überlappungen am Anfang von a werden durch Zuordnungen an die „künstlichen“ Indizes $minOvl - n, \ldots, -1$ realisiert.

Der Algorithmus ist in Abbildung 3.1 angegeben[1]. Die Folgen a, b sowie die Ergebnisfolge *result* sind als Felder realisiert, wobei das i-te Element eines Feld f mit $f[i]$ und das Teilfeld vom Index i bis zum Index j, $i \geq j$, mit $f[i..j]$ bezeichnet wird. Die Funktion *compare* berechnet die Kenngröße, die einen Ähnlichkeits- bzw. Abstandsbegriff zweier gleichlanger Folgen angibt.

Der Algorithmus gliedert sich in drei Schritte, die jeweils aus einer for-Schleife bestehen. Die zu den einzelnen Schritten abgebildeten Graphiken illustrieren den jeweils ersten und letzten Fall der for-Schleife. Im ersten Schritt werden alle unvollständigen Überlappungen an der linken Seite von a betrachtet. Dabei werden die mindestens $minOvl$-langen Präfixe von a mit den Suffixen gleicher Länge von b verglichen. Der zweite Schritt vergleicht b in voller Länge an allen möglichen Positionen in a. Im dritten Schritt werden ähnlich wie im ersten Schritt alle unvollständigen Überlappungen an der rechten Seite von a betrachtet. Dazu werden die Suffixe mit Mindestlänge $minOvl$ von a mit den Präfixen gleicher Länge von b verglichen.

Da a in den meisten Fällen die bereits datierte Referenzfolge ist, entspricht jeder Index in a einer Jahreszahl. Dann können die Zuordnungen der statistischen Größen zu Indizes in a gleichsam als Zuordnungen zu Jahreszahlen aufgefaßt werden.

Die den einzelnen Indizes bzw. Jahreszahlen zugeordneten Resultate werden dem Benutzer geeignet ausgegeben, wobei dieser ein Matching in der Regel visuell an den Jahrringfolgen oder direkt am Holz überprüft.

Ausgehend davon, daß die Berechnung eines Abstandes lineare Laufzeit in der Länge der zwei zu vergleichenden Folgen benötigt, ist die Laufzeit des sukkzessiven Algorithmus $\theta(mn - minOvl(minOvl - 1))$ bzw. für konstantes $minOvl$ $\theta(mn)$.

[1] Der an dieser Stelle sowie auch im folgenden verwendete Pseudo-Code ist an die Sprache C angelehnt.

3.3 Vergleich anhand linearer Korrelation

3.3.1 Der Korrelationskoeffizient

Seien $x = x_0, \ldots, x_{N-1}$ und $y = y_0, \ldots, y_{N-1}$ mit einem festen $N \in \{minOvl, \ldots, n-1\}$ die an einer Position in dem Algorithmus zu vergleichenden, sich gegenüberstehenden Folgen gleicher Länge.

Der *(empirische) Korrelationskoeffizient (Pearson's product-moment correlation coefficent)* wird häufig zur Beschreibung der Ähnlichkeit zweier Folgen verwendet und ist definiert als

$$r = \frac{\sum_{i=0}^{N-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \bar{x})^2 \sum_{i=0}^{N-1}(y_i - \bar{y})^2}} \tag{3.1}$$

$$= \frac{N\sum_{i=0}^{N-1} x_i y_i - \sum_{i=0}^{N-1} x_i - \sum_{i=0}^{N-1} y_i}{\sqrt{(N\sum_{i=0}^{N-1} x_i^2 - (\sum_{i=0}^{N-1} x_i)^2)(N\sum_{i=0}^{N-1} y_i^2 - (\sum_{i=0}^{N-1} y_i)^2)}} \tag{3.2}$$

wobei $\bar{x} = \frac{1}{N}\sum_{i=0}^{N-1} x_i$ und $\bar{y} = \frac{1}{N}\sum_{i=0}^{N-1} y_i$ die arithmetischen Mittel sind.

Der *(empirische) Korrelationskoeffizient* mißt den Grad der *linearen Abhängigkeit* von x und y. Die beiden Folgen sind *linear abhängig* voneinander, wenn alle Punkte (x_i, y_i) in einem Koordinatensystem mit den möglichen Werten von x als x-Achse und den möglichen Werten von y als y-Achse auf einer Geraden $y = \alpha x + \beta$ liegen. Je weniger x und y linear abhängig voneinander sind, desto mehr streuen die Punkte in einem solchen Koordinatensystem, und desto schwieriger ist, eine Gerade zu bestimmen, deren Abstand zu den Punkten minimal ist.

Der Wertebereich von r liegt zwischen -1 und $+1$ (jeweils einschließlich), wobei $|r| = 1$ ist genau dann, wenn beide Folgen linear mit Koeffizienten α und β voneinander abhängen. Das Vorzeichen von r entspricht dabei dem Vorzeichen von α. Ist α positiv, so steigt x genau dann, wenn y steigt, und fällt x genau dann, wenn y fällt [2]. Bei negativem α steigt x genau dann, wenn y fällt und umgekehrt [3]. Die Folgen werden bei $|r|$ nahe 1 entsprechend dem Vorzeichen von r als *stark positiv (negativ) korreliert* bezeichnet. Je mehr die Punkte

[2] D.h. für $1 \leq i \leq N-1$ gilt $sign(x_i - x_{i-1}) = sign(y_i - y_{i-1})$.
[3] D.h. für $1 \leq i \leq N-1$ gilt $sign(x_i - x_{i-1}) = -sign(y_i - y_{i-1})$.

(x_i, y_i) von einer Geraden abweichen, desto kleiner ist $|r|$, d.h. desto *schwächer* sind sie *korreliert*. Bei $|r| = 0$ sind die beiden Folgen *unkorreliert*, d.h. sie hängen nicht linear voneinander ab.

Beweise für diese Beobachtungen sowie Herleitungen für den Korrelationskoeffizienten sind in vielen Statistikbüchern, zum Beispiel in Yamane [57] und Worthing et al. [56] zu finden. Eine Herleitungsart verwendet die Methode der kleinsten Quadrate, um eine bestmögliche Anpassung der Stichprobenpaare an zwei sogenannte *Regressionsgeraden* zu finden. Diese Herleitungsmethode verdeutlicht den Zusammenhang des Korrelationskoeffizienten mit Geraden in der (x, y)-Ebene. Für Details zu diesem Ansatz wird auf Yamane [57] und Worthing et al. [56] verwiesen.

3.3.2 Interpretation linearer Korrelation in der Dendrochronologie

In der Dendrochronologie wird der zeitliche Verlauf zweier Folgen betrachtet, um diese zu vergleichen. Deshalb werden die Folgen wie in den vorigen Kapiteln zur Veranschaulichung in einem Zeit-Jahrringbreiten-Koordinatensystem aufgetragen. Eine lineare Abhängigkeit mit $r = 1$ zwischen zwei Folgen ist in einem solchen Koordinatensystem daran zu erkennen, daß sich die beiden Folgen nur durch eine Skalierung um den Faktor α und durch eine Verschiebung in der Ordinate um β unterscheiden. Der Richtungssinn, d.h. die Richtung der Steigung von Jahr zu Jahr (positiv, negativ), stimmt dann bei den beiden Kurven an jeder Stelle überein.

Da ein Baum durch für ihn individuell bessere Wachstumsbedingungen durchaus allgemein breitere (oder bei schlechteren Bedingungen schmalere) Jahrringe haben kann als ein in der Nähe wachsender Baum, können zwei gleichdatierte Jahrringbreitenfolgen um β Einheiten in Jahrringbreiten-Richtung verschoben sein. Höhere bzw. schwächere Sensitivität auf die Wachstumsbedingungen bei verschiedenen Bäumen schlägt sich in einem Skalierungsfaktor $\alpha > 0$ nieder. Ein Skalierungsfaktor kleiner 0 ist nicht möglich, da dies ein entgegengesetztes Verhalten des jährlichen Richtungssinns bedeuten würde. Je schwächer die lineare Abhängigkeit zweier Folgen voneinander ist, desto weniger stimmen der jährliche Richtungssinn, die Skalierung und die Verschiebung überein.

Diese Betrachtungen gelten jedoch nur, wenn die Jahrringfolgen jeweils *stationär* sind, d.h. keine Langzeittrends aufweisen, sondern gleichmäßig um einen festen Mittelwert variieren. Da Jahrringfolgen im allgemeinen Langzeittrends aufweisen, werden die Folgen a und b, bevor sie miteinander verglichen werden, speziellen Transformationen unterzogen, die die Langzeittrends eliminieren. Transformationen dieser Art wurden in Abschnitt 2.4 beschrieben.

3.3.3 Der t-Wert

Ein Nachteil von r besteht darin, daß r die Länge der Folgen nicht berücksichtigt. Häufig kann es deshalb bei kleinem N passieren, daß trotz eines hohen Wertes von r die Folgen

x und y nicht gleichdatiert sind. In der Dendrochronologie wird deshalb statt r der von r und N abhängige *t-Wert (Student's t)*

$$t = r\sqrt{\frac{N-2}{1-r^2}} \tag{3.3}$$

betrachtet. Dieser bezieht die Länge N der Folgen ein und drückt die Korrelation in einer Skalierung aus, die hohe Werte von r verstärkt darstellt, wie aus Abbildung 3.2 hervorgeht.

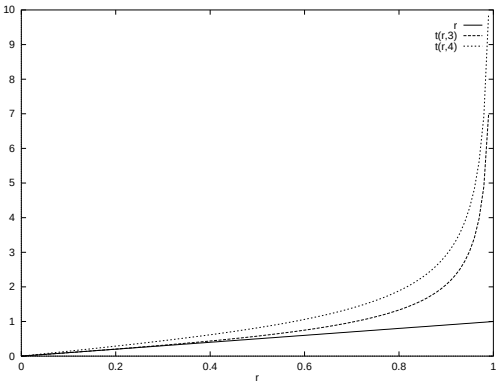

Abbildung 3.2: Veranschaulichung des Verlaufs des t-Wertes $t(r, N)$ aus Formel (3.3)

Abgesehen davon wird t häufig zur Berechnung der *statistischen Sicherheit* einer Datierung mit dem beobachteten Wert von r verwendet. t dient dabei als *Teststatistik* zur Überprüfung der Hypothese, daß die Ähnlichkeit der beiden Folgen, also das Auftreten eines strikt positiven Korrelationskoeffizienten, zufällig ist. Dazu wird die Wahrscheinlichkeit betrachtet, daß durch Zufall, also bei einer beliebigen, nicht übereinstimmenden Datierung der Folgen, ein mindestens so großer Korrelationskoeffizient wie der tatsächlich berechnete auftritt. Diese Wahrscheinlichkeit wird als *Signifikanz* des Tests bezeichnet, und stellt die Komplementärwahrscheinlichkeit zur statistischen Sicherheit dar.

Unter der Annahme, daß die Werte (x_i, y_i) der beiden Folgen an jedem Zeitpunkt i identisch bivariat normalverteilt und über die Zeit stochastisch unabhängig voneinander sind, kann aus t in konstanter Zeit die Signifikanz errechnet, bzw. in Tabellen nachgeschlagen werden.

Der Benutzer sollte in der Regel anhand des Wertes von t und der statistischen Sicherheit eine Entscheidung über ein Matching treffen. Eine möglichst hohe statistische Sicherheit gibt dann die Wahrscheinlichkeit an, daß ein Matching an der betrachteten Stelle, und damit eine Datierung, korrekt ist. Da hohe Werte von t hohen statistischen Sicherheiten entsprechen, kann eine alleinige Betrachtung des t-Wertes auch zu guten Ergebnissen führen.

Um annähernd die vorausgesetzte bivariate Normalverteilung der Daten zu erhalten werden diese vor der Berechnung der Korrelationskoeffizienten und der t-Werte standar-

disiert. Einige Standardisierungsverfahren wurden in Unterkapitel 2.4 angegeben. Häufig verwendet wird ein von Baillie und Pilcher [5] vorgeschlagenes 5-jähriges gleitendes Mittel als Glättungskurve mit anschließender Log-Transformation.

Die Annahme der zeitlichen Unabhängigkeit der Folgen sowie die Berechnung der einzelnen statistischen Sicherheiten ohne Berücksichtigung, daß der statistische Test in dem Matchingalgorithmus wiederholt durchgeführt wird, ist statistisch nicht völlig korrekt. Die Jahrringfolgen unterliegen einer gewissen Autokorrelation, und die wiederholte Durchführung des statistischen Tests müßte sich in der Berechnung der Wahrscheinlichkeiten auswirken. Anmerkungen und Verbesserungsvorschläge dazu finden sich zum Beispiel in [54] und [33].

Da nichtsdestoweniger dieses statistisch nicht völlig korrekte Verfahren mit Erfolg benutzt wird, und sich die Verbesserungsvorschläge nur auf die statistische Interpretation der Ergebnisse, also die Transformation der Daten sowie die Art der Signifikanzberechnung, nicht aber auf die Berechnung des Korrelationskoeffizienten ansich beziehen, wird in dieser Arbeit nicht auf diese Unstimmigkeiten eingegangen.

3.3.4 Korrelationsalgorithmus

Der aus der Verwendung des Korrelationskoeffizienten als Ähnlichkeitsmaß in dem Basisalgorithmus aus Unterkapitel 3.2 resultierende Algorithmus lautet wie folgt:

1. Berechnung einer geeigneten Standardisierung von a und b, wie etwa die jährliche Differenz der Logarithmen oder die Residuen des fünfjährigen gleitenden Mittels.
2. Sukkzessive Berechnung der Korrelationskoeffizienten an allen möglichen Überlappungen durch den Basisalgorithmus.
3. Umrechnung der Korrelationskoeffizienten in t-Werte und/oder in statistische Sicherheiten.

Die Schritte 1 und 3 hängen von der jeweiligen statistischen Auffassung und der daher verwendeten Standardisierungen und Sigifikanzberechnungen ab. Der Schritt 2 ändert sich dabei jedoch nicht, da die statistischen Untersuchungen nur dazu dienen, die berechneten Korrelationskoeffizienten geeignet auszuwerten.

Viele einfache Standardisierungen, wie etwa die jährliche Differenz der Logarithmen oder die Residuen des fünfjährigen gleitenden Mittels, können in Zeit linear zur Länge der zu standardisierenden Folge berechnet werden. Ausgehend von solchen Standardisierungen benötigt Schritt 1 eine Laufzeit von $\theta(n+m)$.

Die Berechnung eines t-Wertes nach Gleichung (3.3) sowie die Umrechnung dieses t-Wertes in eine Signifikanz ist in konstanter Zeit, also für alle $n+m-2minOvl+1$ Überlappungspositionen in $\theta(n+m-2minOvl)$ oder unabhängig von $minOvl$ in $\theta(n+m)$

möglich. Unter diesen Annahmen benötigen die Schritte 1 und 3 also insgesamt eine Laufzeit proportional zu $m + n$.

Da die Berechnung eines Korrelationskoeffizienten linear in der Überlappungslänge ist, benötigt Schritt 2 eine zu mn bzw. zu $mn - minOvl(minOvl - 1)$ proportionale Laufzeit, wie in Abschnitt 3.2 erläutert wurde. Für konstantes $minOvl$ ergibt sich damit eine quadratische Gesamtlaufzeit von $\theta(mn)$.

Die Laufzeit von Schritt 2 kann auf $\theta((m+n)log(m+n))$ reduziert werden, wie bereits 1984 von Munro [33] bemerkt wurde, indem alle Terme $\sum xy$ mit Hilfe der *schnellen Fourier Transformation (FFT)* und die anderen Summenterme rekursiv berechnet werden. Dieser Ansatz wird in Kapitel 5 vorgestellt.

3.4 Vergleich anhand des Richtungssinns

Eine weitere Möglichkeit, Jahrringfolgen zu vergleichen besteht darin, statt des Folgenwertes an einer Indexposition i den entsprechenden *Richtungssinn* der Folge an dieser Stelle zu betrachten. Der *Richtungssinn* sei dabei definiert als das Vorzeichen der Steigung vom i-ten zum $(i+1)$-ten Folgenglied. Für eine Folge $x_0, \ldots, x_{N-1}$ ergibt sich daraus eine zu betrachtende Folge $\bar{x} = x_0, \ldots, \bar{x}_{N-2}$ mit $\bar{x}_i = sign(|x_{i+1} - x_i|) \in \{-1, 0, 1\}$.

3.4.1 Der Gleichläufigkeitskoeffizient

Seien $x = x_0, \ldots, x_{N-1}$ und $y = y_0, \ldots, y_{N-1}$ die zwei zu vergleichenden (Teil-) Folgen gleicher Länge und $\bar{x}$ und $\bar{y}$ die dazugehörigen Richtungssinn-Folgen. Der *Gleichläufigkeitskoeffizient Glk* von x und y ist definiert als

$$\text{Glk} \quad = \frac{1}{N} \sum_{i=0}^{N-2} \chi(\bar{x}_i = \bar{y}_i), \tag{3.4}$$

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

Eine Prozentangabe dieser Art wurde 1943 von Huber [23] vorgeschlagen. Eckstein hat den Gleichläufigkeitskoeffizienten in [14] und [13] eingehend untersucht.

Auch für den Gleichläufigkeitskoeffizienten können Teststatistiken entwickelt werden, aus denen die statistische Sicherheit hervorgeht. Riemer [40] trifft dabei ähnlich wie in Abschnitt 3.3.3 Annahmen über die Verteilungen der Folgenwerte, während Eckstein [14] empirisch gewonnene Daten zur Berechnung der statistischen Sicherheit verwendet. Die statistische Sicherheit kann in konstanter Zeit aus der Überlappungslänge und dem Glk-Wert berechnet werden.

Da der Gleichläufigkeitskoeffizient nur die Richtung, nicht aber die Höhe der jährlichen Steigung berücksichtigt, gehen viele Informationen verloren. Er stellt deshalb ein recht grobes Ähnlichkeitsmaß dar und wird deshalb in der Praxis häufig nicht als einziges Matchingkritierum verwendet.

Dennoch hat der Gleichläufigkeitskoeffizient den Vorteil, daß er bei extremen Jahrringfolgen, die z.B. stark trendbehaft oder autokorreliert sind oder Ausreißer enthalten, zu guten Ergebnissen führt, wo der Korrelationskoeffizient versagt.

Laut Riemer [40] kommt Hollstein [21] zu dem Schluß, daß der Korrelationskoeffizient in der Datierungspraxis dem Gleichläufigkeitskoeffizienten grundsätzlich überlegen ist. Riemer stellt auch selbst fest, daß aus der Erfahrung vieler Anwender die Anwendung des Korrelationskoeffizienten mit vorangehender Standardisierung der Daten dem Gleichläufigkeitskoeffizienten vorzuziehen ist.

3.4.2 Gleichläufigkeitsalgorithmus

Im Gegensatz zum Korrelationsalgorithmus müssen die Daten nicht notwendigerweise vor der Berechnung der Gleichläufigkeitskoeffizienten standardisiert werden. Demnach ergibt sich der folgende Algorithmus:

1. Sukkzessive Berechnung der Gleichläufigkeitskoeffizienten an allen möglichen Überlappungen durch den Basisalgorithmus.
2. Berechnung der statistischen Sicherheiten.

Die einzelnen statistischen Sicherheiten lassen sich aus den Gleichläufigkeitskoeffizienten und den entsprechenden Überlappungslängen in konstanter Zeit berechnen, weshalb Schritt 2 insgesamt für alle $n + m - 2minOvl + 1$ Überlappungspositionen in $\theta(n+m-2minOvl)$ oder unabhängig von $minOvl$ in $\theta(n+m)$ Zeit möglich ist.

Die sukkzessive Berechnung aller Gleichläufigkeitskoeffizienten anhand des Basisalgorithmus aus Abbildung 3.2 benötigt eine quadratische Zeit $\theta(mn - minOvl(minOvl - 1)$ bzw. unabhängig von $minOvl$ eine Zeit von $\theta(mn)$, da die Berechnung eines Gleichläufkeitskoeffizienten linear in der Überlappungslänge ist. Damit ergibt sich eine quadratische Gesamtlaufzeit von $\theta(mn)$.

Durch die Anwendung der *schnellen Fourier Transformation (FFT)* kann die Laufzeit von Schritt 2 auf $\theta((m+n)log(m+n)$ reduziert werden. Dieser Ansatz wird in Kapitel 5 vorgestellt.

3.4.3 Der Datierungsindex

Eine in dem Handbuch des Programms TSAP [41] als von Burkhard Schmidt aus Köln vorgeschlagene Vergleichsmöglichkeit zwischen zwei Jahrringfolgen ist der *Datierungsindex D*. Dieser setzt sich aus dem t-Wert sowie aus dem Glk-Wert wie folgt zusammen:

$$D = (Glk - 50) * \bar{t}$$

mit $\bar{t} = \frac{t_{bp}+t_h}{2}$. Dabei bezeichnet t_{bp} den t-Wert nach einer Standardisierung mit der von Baillie und Pilcher [5] vorgeschlagenen Glättungskurve des 5-jährigen gleitenden arithmetischen Mittels. t_h bezeichnet den t-Wert nach einer Standardisierung mit der von Hollstein vorgeschlagenen Differenz der Logarithmen zweier aufeinanderfolgender Werte.

Um den Datierungsindex zu berechnen müssen demnach die t-Werte für zwei verschiedene Standardisierungen sowie der Gleichläufigkeitskoeffizient berechnet werden. Da die Berechnung der t-Werte sowie der Glk-Werte anhand des Basisalgorithmus eine Laufzeit von $\theta(mn)$ benötigt, stellt diese ebenfalls die Laufzeit für die Berechnung aller Datierungsindizes dar.

Wenn zur Berechnung der t-Werte sowie der Glk-Werte die die schnelle Fourier Transformation verwendenden Algorithmen aus Kapitel 5 benutzt werden, läßt sich die Laufzeit auf $\theta((m+n)log(m+n))$ reduzieren.

3.4.4 Weiserjahre

Ein *Weiserjahr (signature year, key year)* bezieht sich auf eine aus mehreren Jahrringfolgen zusammengesetzte Chronologie und soll ein für die Chronologie spezielles charakteristisches Jahr darstellen, das in möglichst vielen Jahrringfolgen auftritt.

In der Literatur gibt es unterschiedliche Definitionen für solche Weiserjahre, wovon einige bei Eckstein und Bauch [13] aufgeführt sind. So kann man ein Weiserjahr als ein Jahr definieren, in dem ein festgelegter, oder von der Anzahl der in der Chronologie enthaltenen Jahrringfolgen abhängiger, Prozentsatz der Jahrringfolgen in einer Chronologie den gleichen Richtungssinn aufweist. Möglich ist es auch, von diesen Jahren nur die lokalen Minima, Maxima oder beide als Weiserjahre anzusehen.

Handelt es sich bei der Referenzfolge um eine Chronologie, die aus hinreichend vielen Jahrringfolgen besteht, kann nun zusätzlich zur Betrachtung der Gleichläufigkeitskoeffizienten die Anzahl der übereinstimmenden Weiserjahre als ein weiteres Matching-Kriterium verwendet werden. Das patentierte Crossdatingverfahren von Heikkenen [20] verwendet diesen Ansatz.

Für Weiserjahre gibt es weitere unterschiedliche Definitionen und Begriffsbestimmungen, welche zum Beispiel bei Schweingruber [46] zu finden sind. An dieser Stelle wurde ein Weiserjahr wie bei Eckstein und Bauch [13] definiert, was der Definition eines *Weiserintervalls* bei Schweingruber [46] entspricht.

Kapitel 4

Vorhandene Verfahren zur Identifikation von fehlenden und mehrfachen Ringen

4.1 Verfahren zur Qualitätskontrolle

Um in der Praxis fehlende oder mehrfache Ringe in Folgen zu lokalisieren, werden häufig Verfahren angewendet, die *nach* dem Crossdating nach auffälligen Stellen in den datierten Folgen suchen. Diese Verfahren beruhen auf der Berechnung von Korrelationskoeffizienten auf kurzen Teilstücken der Folgen, so daß der Benutzer aus der Abfolge matchender Teilstücke auf Positionen fehlender oder mehrfacher Ringe schließen kann. Eine solche Qualitätskontrolle bereits datierter Jahrringfolgen führen der von Wendland [53] vorgeschlagene Algorithmus sowie das häufig in der Praxis verwendete Programm *Cofecha* von Holmes [22] aus.

Beide beruhen auf der Datierung vieler kurzer Teilfolgen der Musterfolge anhand der Berechnung von Korrelationskoeffizienten. Wendland vergleicht zwei Folgen derselben Länge, indem er alle zusammenhängenden Teilstücke fester Länge der Musterfolge betrachtet. *Cofecha* hingegen arbeitet auf einer Menge von (untereinander) datierten Folgen und vergleicht jeweils eine Folge mit der aus den restlichen Folgen gebildeten Mittelwertfolge. Dabei werden ebenfalls zusammenhängende Teilstücke fester Länge, jedoch nicht alle, betrachtet.

Diese Verfahren haben für den Vergleich zweier Folgen der Längen n und m eine quadratische Laufzeit von $\theta(nm)$.

4.2 Crossdatingverfahren

In mehreren Crossdatingprogrammen wurde das zur Qualitätskontrolle verwendete Prinzip, viele kurze Teilfolgen der Musterfolge in der Referenzfolge durch die Berechnung von Korrelationskoeffizienten zu lokalisieren, angewendet. Diese Verfahren gehen in der Regel davon aus, daß die Referenzfolge keine fehlenden oder mehrfachen Ringe enthält, und werden meist nur auf kurze Jahrringfolgen angewendet.

Zwei dieser Verfahren sind bei Mills [32] beschrieben, die dort für die Datierung kurzer Jahrringfolgen verglichen werden. Das Programm *cros-twenty* berechnet für alle Datierungen der Musterfolge Korrelationskoeffizienten auf Teilstücken (der Länge 20) in zu dem Algorithmus von Wendland [53] analoger Weise. Als Ergebnis wird für jedes Teilstück die beste Datierung angegeben. Das Programm *resample* verwendet zufällige, nicht notwendigerweise zusammenhängende Teilfolgen der Länge 20. Parker [36] berechnet in dem sogenannten *Shifting Unit Dating Program* ebenfalls Korrelationskoeffizienten auf zusammenhängenden Teilstücken fester Länge und gibt jeweils die drei besten Datierungen aus. Die Laufzeit dieser Crossdatingverfahren ist $\theta(nm)$.

Der von Van Deusen [11] vorgeschlagene Algorithmus zur Identifikation fehlender und doppelter Ringe basiert auf einer völlig anderen Herangehensweise: Er berechnet einen Transformationsabstand zwischen beiden Folgen, der mögliche Positionen von fehlenden und doppelten Ringen liefert. Eine sequentielle Berechnung dieses Transformationsabstandes kann als Crossdatingalgorithmus verwendet werden. Da der in Kapitel 6 vorgestellte Abstandsbegriff sowie der Crossdatingalgorithmus in Kapitel 7 auf dem Transformationsabstand von Van Deusen basieren, wird auf diesen in Kapitel 6 näher eingegangen.

Kapitel 5

Anwendungen der schnellen Fourier-Transformation auf das Crossdating

Mit Hilfe der *schnellen Fourier-Transformation (fast Fourier transform, FFT)* können sowohl die auf der Berechnung von Korrelationskoeffizienten als auch die auf der Berechnung von Gleichläufigkeitskoeffizienten basierenden Crossdatingalgorithmen effizienter gestaltet werden. In diesem Kapitel wird zuerst in Unterkapitel 5.1 die diskrete Fourier-Transformation erläutert und ein für die Anwendungen passender Algorithmus zur schnellen Fourier Transformation angegeben. In den Unterkapiteln 5.3 und 5.4 wird die FFT auf die Berechnung der Korrelationskoeffizienten sowie der Gleichläufigkeitskoeffizienten für das Crossdating angewendet.

5.1 Die schnelle Fourier-Transformation (FFT)

5.1.1 Die diskrete Fourier-Transformation (DFT)

Im folgenden sei N eine natürliche Zahl und $\xi := e^{\frac{2\pi i}{N}} \in \mathbb{C}$ eine primitive N-te Einheitswurzel, wobei $\mathbb{C}$ die komplexen Zahlen und i die imaginäre Zahl bezeichnen.

Definition 5.1 *Die Abbildung* $DFT : \mathbb{C}^N \to \mathbb{C}^N$, *die einen Vektor* $z = z_0, \ldots, z_{N-1} \in \mathbb{C}^N$ *auf einen Vektor* $DFT(z) \in \mathbb{C}^N$ *unter der Vorschrift*

$$DFT(z)_k := \sum_{j=0}^{N-1} z_j \xi^{jk} \tag{5.1}$$

abbildet, heißt diskrete Fourier-Transformation (DFT).

Für den nächsten Satz benötigen wir das folgende Lemma:

Lemma 5.2 *Es gilt*

$$\sum_{k=0}^{N-1} \xi^{kj} = \begin{cases} N\,, & \textit{für} \quad j \bmod N = 0 \\ 0\,, & \textit{sonst} \end{cases} \tag{5.2}$$

Beweis. Für $j \bmod N = 0$ sind alle $\xi^{kj} = 1$, also die Summe $= N$.

Sei nun $j \bmod N \neq 0$. Die Summenformel für die geometrische Summe führt zu $\sum_{k=0}^{N-1} \xi^{kj} = \frac{(\xi^N)^j - 1}{\xi^j - 1}$. Wegen $j \bmod N \neq 0$ ist $\xi^j \neq 1$ und der Nenner damit von Null verschieden. Da $\xi^N = e^{\frac{2\pi i N}{N}} = 1$ ist, folgt $\frac{(\xi^N)^j - 1}{\xi^j - 1} = 0$. □

Satz 5.3 *Die Abbildung $DFT^{-1} : \mathbb{C}^N \to \mathbb{C}^N$, die einen Vektor $z = z_0, \ldots, z_{N-1} \in \mathbb{C}^N$ auf einen Vektor $DFT^{-1}(z) \in \mathbb{C}^N$ unter der Vorschrift*

$$DFT^{-1}(z)_l := \frac{1}{N} \sum_{k=0}^{N-1} z_k \xi^{-kl} \tag{5.3}$$

abbildet, ist zur DFT invers.

Beweis. Durch Einsetzen der Definition der Abbildungen DFT und DFT^{-1} ineinander angewendet auf einen Vektor $z \in \mathbb{C}^N$ erhält man

$$DFT(DFT^{-1}(z))_l = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} (z_k \xi^{j(l-k)})$$

$$\text{und} \quad DFT^{-1}(DFT(z))_l = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} (z_k \xi^{j(k-l)})$$

Da k und l beide zwischen 0 und N liegen, ist $(k-l) \bmod N = 0$ genau dann, wenn $k = l$. Mit Lemma 5.2 folgt, daß der einzige von Null verschiedene Summand der jeweils äußeren Summe für $k = l$ auftritt und somit die jeweils l-te Koordinate gleich z_l ist. □

5.1.2 Algorithmen der schnellen Fourier-Transformation (FFT)

Die Algorithmen, die die DFT schnell berechnen, werden zusammengefaßt als *schnelle Fourier-Transformations-* oder kurz als *FFT-Algorithmen* bezeichnet. Eine gute Übersicht

FFTrek(N, x):
//Rekursionsabbruch
if($N == 1$) {
 return x;
} else{
 $x^g = (x_0, \ldots, x_{N-2})$;
 $x^u = (x_1, \ldots, x_{N-1})$;
 y =FFTrek($\frac{N}{2}, x^g$);
 z =FFTrek($\frac{N}{2}, x^u$);
 return DFTcombine(N, y, z); }

DFTcombine(N, x, y):
$\xi_N = e^{\frac{2\pi i}{N}}$;
$\xi = 1$;
for ($j = 0$; $j \leq \frac{N}{2} - 1$; $j++$) {
 $tmp = \xi * y_j$;
 $z_j = x_j + tmp$;
 $z_{j+\frac{N}{2}} = x_j - tmp$;
 $\xi = \xi * \xi_N$;
}
return z;

Abbildung 5.1: Rekursiver FFT-Algorithmus (aus Cormen et al. [10])

solcher Algorithmen bietet Nussbaumer [35]. An dieser Stelle werden nur einige FFT-Algorithmen erläutert, insbesondere solche, die für unsere Anwendung geeignet sind.

Alle FFT-Algorithmen basieren darauf, daß sich die Länge N des Eingabevektors $x = x_0, \ldots, x_{N-1}$ in möglichst viele Primfaktoren zerlegen läßt. Deshalb wird in den bekanntesten FFT-Algorithmen davon ausgegangen, daß die Länge N des Eingabevektors eine Zweierpotenz ist. An dieser Stelle wird ebenfalls diese Annahme getroffen und gegebenenfalls ein Eingabevektor mit Nullen auf Zweierpotenzlänge aufgefüllt. Für weitere FFT-Algorithmen und weitere ähnliche Tansformationsalgorithmen wird auf Nussbaumer [35], Press et al. [38] und Elliott et al. [15] verwiesen.

Der nachstehende Satz bildet den Kern des klassischen FFT-Algorithmus.

Satz 5.4 *Für einen Vektor $x = x_0, \ldots, x_{N-1} \in \mathbb{C}^n$, wobei N eine Zweierpotenz oder zumindest ein Vielfaches von 2 sei, hat die DFT folgende Eigenschaft:*

$$DFT(x)_k = DFT(x_g)_k + \xi^k DFT(x_u)_k \tag{5.4}$$

$$DFT(x)_{k+\frac{N}{2}} = DFT(x_g)_k - \xi^k DFT(x_u)_k \tag{5.5}$$

für $k = 0, \ldots, \frac{N}{2} - 1$. Dabei bezeichnen x_g und x_u die Teilfolgen aus a mit geraden bzw. ungeraden Indizes, also $x_g := x_0, x_2, \ldots, x_{N-2}$ und $x_u = x_1, x_3, \ldots, x_{N-1}$.

Beweis. Sei $k \in \{0, \ldots, \frac{N}{2} - 1\}$. Mit $\xi = e^{\frac{2\pi i}{N}}$ ist ξ^2 die entsprechende primitive $\frac{N}{2}$-te Einheitswurzel. Die Summe in der DFT kann nach geraden und ungeraden Indizes in x wie folgt aufgeteilt werden:

$$DFT(x)_k = \sum_{j=0}^{\frac{N}{2}-1} x_{2j}\xi^{2jk} + \xi^k \sum_{j=0}^{\frac{N}{2}-1} x_{2j+1}\xi^{2jk} = DFT(x_g)_k + \xi^k DFT(x_u)_k.$$

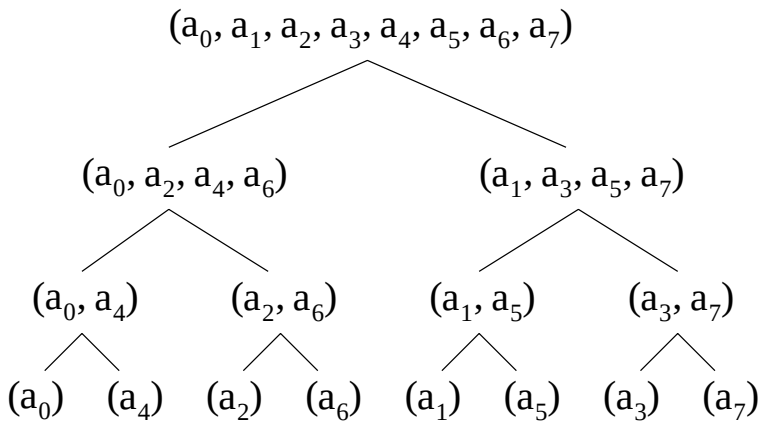

Abbildung 5.2: Rekursionsbaum für den FFTrek-Algorithmus für $n = 8$ (aus Cormen et al. [10]). In Klammern stehen die Eingaben für den jeweiligen rekursiven Aufruf, wobei ein linker Knoten dem ersten Rekursionsaufruf und ein rechter Knoten dem zweiten Rekursionsaufruf entspricht.

Analoges Aufteilen für Indizes $k + \frac{N}{2}$ führt zu

$$DFT(x)_{k+\frac{N}{2}} = \sum_{j=0}^{\frac{N}{2}-1} x_{2j}\xi^{2jk+jN} + \xi^{k+\frac{N}{2}} \sum_{j=0}^{\frac{N}{2}-1} x_{2j+1}\xi^{2jk+jN},$$

was unter Einbeziehung von $\xi^{\frac{N}{2}} = e^{\pi i} = -1$ und $\xi^N = 1$ zu

$$DFT(x)_{k+\frac{N}{2}} = \sum_{j=0}^{\frac{N}{2}-1} x_{2j}\xi^{2jk} - \xi^k \sum_{j=0}^{\frac{N}{2}-1} x_{2j+1}\xi^{2jk} = DFT(x_g) - \xi^k DFT(x_u)_k$$

führt. □

Ausgehend von diesem Satz läßt sich die DFT eines Vektors $x = x_0, \ldots, x_{N-1} \in \mathbb{C}^n$, wobei N eine Zweierpotenz sei, anhand des Algorithmus in Abbildung 5.1 rekursiv berechnen. Ein Beispiel eines Rekursionsbaumes ist in Abbildung 5.2 aufgezeigt, dabei seien die Ebenen mit 0 beginnend numeriert. In der l-ten Ebene des Rekursionsbaumes gibt es 2^l rekursive Aufrufe, die jeweils eine Eingabe der Länge $\frac{N}{2^l}$ haben und, bis auf rekursive Unteraufrufe, eine Laufzeit proportional zu der Eingabelänge benötigen. Da die Rekursionstiefe, also die Höhe des Baumes, $\log_2 N$ ist, beträgt die Laufzeit insgesamt $\theta(N + \sum_{l=1}^{\log_2 N} 2^l \frac{N}{2^l}) = \theta(N \log_2 N)$.

Da in jedem rekursiven Aufruf die Rückgabewerte beider rekursiver Aufrufe gespeichert werden, wird zeitweise doppelt soviel Speicherplatz wie die ursprüngliche Eingabe, also $\theta(2N)$ benötigt. Im folgenden wird ein iterativer Algorithmus vorgestellt, der die DFT im Platz des Eingabefeldes berechnet.

FFTit_base(N, x')**:**
for$(l = 1; l \leq \log_2 N; l++)\{$
  for$(k = 0; k \leq N - 2^l; k += 2^l)\{$
    $x'[k \ldots k + 2^l - 1]$ =DFTcombine$(2^{l-1},\ x'[k \ldots k + 2^{l-1} - 1],$
                                                $x'[k + 2^{l-1} \ldots k + 2^l - 1]);$
}

Abbildung 5.3: Iterativer FFT-Algorithmus zur Berechnung der DFT für einen Vektor x unter der Annahme, daß das Eingabefeld x' die Permutation der Elemente aus x enthält, die zur Berechnung der $DFT(x)$ benötigt wird. (aus Cormen et al. [10])

Im Gegensatz zur rekursiven Top-Down-Auswertung der Rekursionseigenschaft aus Satz 5.4 verwendet der iterative Algorithmus diese Eigenschaft in einem Bottom-Up-Ansatz. Bei einer Betrachtung des Rekursionsbaumes (Beispiel in Abbildung 5.2) wird deutlich, daß, sofern die Reihenfolge der Elemente des Eingabefeldes in den Blättern des Baumes bekannt ist, die DFT, durch wiederholte Anwendung der Eigenschaft aus Satz 5.4 auf einen Knoten mit je zwei Kinderknoten, von unten nach oben im Baum berechnet werden kann. Die DFT eines Knotens, dessen zwei Kinderknoten jeweils eine Eingabe der Länge 2^{l-1} haben, hat die Länge 2^l. Da jedes Element des gesamten Eingabefeldes in jeder Ebene des Baumes genau einmal vorkommt, kann bei der Berechnung der DFT eines Knotens anhand der zwei Kinderknoten das Ergebnis in den Plätzen der Eingaben gespeichert werden.

Die Permutation der Elemente eines Vektors x, die als Eingabe für den Algorithmus FFTit_base in Abbildung 5.3 benötigt wird, ist eine Anordnung der Elemente anhand der *bitweisen Umkehrung* der aufsteigenden Indizes. Das heißt, die aufsteigenden Indizes von 0 bis $N - 1$ müssen als Binärzahlen der Länge $\log_2 N$ betrachtet, die Bits jeder Zahl in ihrer Reihenfolge umgedreht, und danach wieder als Dezimalzahlen betrachtet werden. Die daraus resultierende Permutation der Indizes liefert die gewünschte Reihenfolge. Für $N = 8$ zum Beispiel sind die aufsteigend sortierten Indizes $0_{dez} = 000_{bin}$, $1_{dez} = 001_{bin}$, $2_{dez} = 010_{bin}$, $3_{dez} = 011_{bin}$, $4_{dez} = 100_{bin}$, $5_{dez} = 101_{bin}$, $6_{dez} = 110_{bin}$, $7_{dez} = 111_{bin}$. Die bitweise umgekehrte Reihenfolge, wie sie in den Blättern des Rekursionsbaumes in Abbildung 5.2 zu sehen ist, ist $0_{dez} = 000_{bin}$, $4_{dez} = 100_{bin}$, $2_{dez} = 010_{bin}$, $6_{dez} = 110_{bin}$, $1_{dez} = 001_{bin}$, $5_{dez} = 101_{bin}$, $3_{dez} = 011_{bin}$, $7_{dez} = 111_{bin}$.

Satz 5.5 *Die bitweise umgekehrte Anordnung der Elemente eines Vektors x der Länge N ist die korrekte Eingabe für den Algorithmus FFTit_base in Abbildung 5.3, so daß dieser die $DFT(x)$ berechnet.*

Beweis. Die korrekte Anordnung der Elemente aus x ist die der Blätter im Rekursionsbaum für FFTrek. Es muß also gezeigt werden, das die bitweise umgekehrte Reihenfolge die Reihenfolge der Blätter im Rekursionsbaum ist.

Bitrev(N, x)**:**
for$(j = 0; j \leq \frac{N}{2} - 1; j++)\{$
  if$(rev(j) \neq j)\{$
  swap$(x_j, x_{rev(j)})$;
}

Abbildung 5.4: Algorithmus zur Erstellung der bitweise umgekehrten Reihenfolge der Elemente eines Feldes x am Platz. Die Funktion $rev(j)$ liefert die Zahl, die durch Umkehrung der Bits in j entsteht. (Zum Beispiel $rev(4) = 1$)

FFTit(N, x)**:**
Bitrev(N, x);
FFTit_base(N, x);

Abbildung 5.5: Iterativer FFT-Algorithmus zur Berechnung der DFT eines Vektors x am Platz.

Beim ersten Aufruf von FFTrek, also bei der Wurzel des Baumes, gelangen Elemente mit geraden Indizes in den linken Teilbaum, der dem ersten rekursiven Unteraufruf entspricht, und Elemente mit ungeraden Indizes in den rechten Teilbaum, der dem zweiten rekursiven Unteraufruf entspricht. Da gerade Zahlen in binärer Schreibweise als letztes Bit eine 0 und ungerade Zahlen eine 1 aufweisen, stehen also im linken Teilbaum alle Elemente aus x, deren Indizes eine 0 und im rechten Teilbaum alle, deren Indizes eine 1 als letztes Bit aufweisen. Durch Wegstreichen des letzten Bits erhält man die Indexmenge $\{0, \ldots, \frac{N}{2} - 1\}$, die in den FFTrek-Aufrufen der beiden Kinderknoten verwendet wird. Eine rekursive Anwendung derselben Überlegung führt genau zu der bitweise umgekehrten Anordnung. □

In Abbildung 5.4 ist ein Algorithmus angegeben, der die bitweise umgekehrte Anordnung eines Feldes x am Platz vornimmt. Dieser basiert auf der Beobachtung, daß nur paarweise Elemente vertauscht werden müssen, da eine zweimalige bitweise Umkehrung einer Zahl wieder die Ausgangszahl ergibt. Den resultierenden iterativen FFT-Algorithmus zeigt Abbildung 5.5.

Bemerkung. Die FFT-Algorithmen können in leicht modifizierter Weise zur Berechnung der DFT^{-1} angewandt werden. Wie leicht einzusehen ist, genügt die DFT^{-1} fast derselben Bedingung wie die DFT in Satz 5.4, nur ist das Vorzeichen des Exponenten von ξ negativ. Ohne die Division durch N kann die DFT^{-1} mit einem FFT-Algorithmus berechnet werden, sofern in DFTcombine mit ξ^{-1} statt mit ξ multipliziert wird. Eine anschließende Division durch N liefert dann die DFT^{-1}.

5.2 Beziehungen zwischen der DFT und der zyklischen Korrelation

Die zyklische Faltung zweier Vektoren steht mit der zyklischen Korrelation in engem Zusammenhang. Deshalb werden beide im folgenden definiert und für uns nützliche Beziehungen zur DFT vorgestellt.

Definition 5.6 *Die* zyklische Faltung *zweier Vektoren* $x = x_0, \ldots, x_{N-1} \in \mathbb{C}^N$ *und* $y_0, \ldots, y_{N-1} \in \mathbb{C}^N$ *ist ein Vektor* $Falt(x,y) \in \mathbb{C}^N$*, dessen Elemente durch*

$$Falt(x,y)_l := \sum_{j=0}^{N-1} x_j y_{(l-j) \bmod N} \quad \textit{für } l = 0, \ldots, N-1 \tag{5.6}$$

definiert sind.
Die zyklische Korrelation *ist ein Vektor* $Korr(x,y) \in \mathbb{C}^N$*, dessen Elemente durch*

$$Korr(x,y)_l := \sum_{j=0}^{N-1} x_j y_{(l+j) \bmod N} \quad \textit{für } l = 0, \ldots, N-1 \tag{5.7}$$

definiert sind.

Das nachstehende Lemma liefert den Zusammenhang zwischen zyklischer Korrelation und zyklischer Faltung.

Lemma 5.7 *Für zwei Vektoren* $x, y \in \mathbb{C}^N$ *gilt*

$$Korr(x,y) = Falt(\overleftarrow{x}, y)\,, \tag{5.8}$$

wobei $\overleftarrow{x}$ *den durch* $\overleftarrow{x}_l := x_{N-l}$ *für* $l \in \{0, \ldots, N-1\}$ *definierten Vektor bezeichnet.*

Beweis. Sei $l \in \{0, \ldots, N-1\}$ beliebig. Es gilt $Falt(\overleftarrow{x}, y))_l = \sum_{j=0}^{N-1} x_{N-j}, y_{(l-j) \bmod N} = \sum_{j=0}^{N-1} x_j, y_{l+j \bmod N} = Korr(x,y)_l$. □

Die Faltung hat bezüglich der DFT eine wichtige Eigenschaft, die der folgende Satz zeigen wird:

Satz 5.8 *Für zwei Vektoren* $x = x_0, \ldots, x_{N-1} \in \mathbb{C}^N$ *und* $y_0, \ldots, y_{N-1} \in \mathbb{C}^N$ *und deren zyklische Faltung* $Falt(x,y)$ *gilt folgende Eigenschaft:*

$$Falt(x,y) = DFT^{-1}(DFT(x) \cdot DFT(y)), \tag{5.9}$$

wobei die Multiplikation der beiden Transformationsvektoren elementweise definiert sei.

Beweis. Sei der Vektor auf der rechten Seite der Gleichung mit d bezeichnet. Das Einsetzen der Definitionen der DFT und ihrer Inversen auf der rechten Seite führt, mit $l = 0, \ldots, N-1$ zu

$$\begin{aligned} d_l &= \frac{1}{N} \sum_{k=0}^{N-1} ((\sum_{\nu=0}^{N-1} x_\nu \xi^{\nu k}) \cdot (\sum_{\mu=0}^{N-1} y_\mu \xi^{\mu k})) \xi^{-lk} \\ &= \sum_{\nu=0}^{N-1} \sum_{\mu=0}^{N-1} (x_\nu y_\mu \frac{1}{N} \sum_{k=0}^{N-1} \xi^{k(\nu+\mu-l)}) \end{aligned}$$

Mit Lemma 5.2 folgt, daß die innere Summe nur für $(\nu + \mu - l) \bmod N = 0$ von Null verschieden, und zwar $= N$, ist. Deshalb gilt

$$d_l = \sum_{\nu=0}^{N-1} x_\nu y_{(l-\nu) \bmod N} = Falt(x, y)_l$$

□

Korollar 5.9 *Für die zyklische Korrelation zweier reeller Vektoren $x, y \in \mathbb{R}^N$ gilt*

$$Korr(x, y) = DFT^{-1}(\overline{DFT(x)} \cdot DFT(y)). \tag{5.10}$$

Dabei bezeichnet '$\_$' die komplexe Konjugation.

Beweis. Mit Lemma 5.7 und Satz 5.8 gilt

$$Korr(x, y) = Falt(\overleftarrow{x}, y) = DFT^{-1}(DFT(\overleftarrow{x}) \cdot DFT(y))$$

Da alle x_i reell, ξ eine n-te Einheitswurzel und $\xi^{-jk} = \overline{\xi^{jk}}$ sind, gilt

$$\begin{aligned} DFT(\overleftarrow{x})_k = \sum_{j=0}^{N-1} x_{N-j} \xi^{jk} &= \sum_{j=0}^{N-1} x_j \xi^{-jk} \\ &= \overline{\sum_{j=0}^{N-1} x_j \xi^{jk}} = \overline{DFT(x)} \end{aligned}$$

□

Um dieses Lemma auf den nicht-zyklischen Korrelationsbegriff, wie er für das Crossdating benötigt wird, anwenden zu können, müssen die Vektoren am Ende mit einer Anzahl von Nullen aufgefüllt werden. Darauf wird in Abschnitt 5.3.2 näher eingegangen.

5.3 Anwendung der FFT auf die Berechnung von Korrelationskoeffizienten

5.3.1 Sukkzessive Berechnung

Für eine Referenzjahrringfolge $a = (a_0, \ldots, a_{m-1}) \in \mathbb{R}^m$ und eine Musterjahrringfolge $b = (b_0, \ldots, b_{n-1}) \in \mathbb{R}^n$ können alle Korrelationskoeffizienten an jeder Überlappungsposition anhand des sukkzessiven Basisalgorithmus aus Unterkapitel 3.2 berechnet werden. Diese werden anhand der nachstehenden Formel berechnet und in dem Feld *result* gespeichert, das für die Indizes $minOvl - n, .., m - minOvl$ definiert ist.

$$result_l = \frac{L(l) \sum_{j=first(l)}^{last(l)} a_{l+j} b_j - \sum_{j=first(l)}^{last(l)} a_{l+j} - \sum_{j=first(l)}^{last(l)} b_j}{\sqrt{(L(l) \sum_{j=first(l)}^{last(l)} (a_{l+j})^2 - (\sum_{j=first(l)}^{last(l)} a_{l+j})^2)(L(l) \sum_{j=first(l)}^{last(l)} (b_j)^2 - (\sum_{j=first(l)}^{last(l)} b_j)^2)}}$$

$$\text{mit } first(l) = \begin{cases} -l & \text{, wenn } minOvl - n \leq l \leq -1 \\ 0 & \text{, sonst} \end{cases}$$

$$last(l) = \begin{cases} n-1 & \text{, wenn } minOvl - n \leq l \leq m - n \\ m-1-l & \text{, sonst} \end{cases}$$

$$\text{und } L(l) = last(l) - first(l) + 1$$

für $minOvl - n \leq l \leq m - minOvl$.

5.3.2 Effiziente Berechnung der Korrelation κ

Der bei der sukkzessiven Berechnung am zeitintensivsten zu berechnende Vektor ist $\kappa = \kappa(a, b)$, wobei

$$\kappa_l = \kappa(a, b)_l := \sum_{j=first(l)}^{last(l)} a_{l+j} b_j$$

für $minOvl - n \leq l \leq m - minOvl$. Eine sukkzessive Berechnung von κ benötigt eine Laufzeit von $\theta(mn)$ für konstantes $minOvl$. Durch Zurückführung auf die zyklische Korrelation mit Hilfe der FFT kann κ wie folgt effizient berechnet werden:

Bei der Berechnung von *result* und damit auch von κ_l müssen mindestens $minOvl$ Elemente von a und b überlappen, so daß höchstens $n - minOvl$ Elemente von b links oder rechts von a und höchstens $m - minOvl$ Elemente von a links oder rechts von b überhängen. Um κ auf die zyklische Korrelation zurückführen zu können, ohne daß überhängende Elemente das Ergebnis verfälschen, müssen als eine Art Pufferzone an das Ende von a

mindestens $n - minOvl$ Nullen und an das Ende von b mindestens $m - minOvl$ Nullen angehängt werden. Da für die Anwendung der FFT-Algorithmen aus Abschnitt 5.1.2 die Vektoren- bzw. Folgenlänge eine Zweierpotenz sein muß, sei n' die kleinste Zweierpotenz $\geq n+m-minOvl$; für diese gilt $n' \leq 2(n+m-minOvl)$. Die zu betrachtenden Vektoren $a', b' \in \mathbb{R}^{n'}$ entstehen aus a bzw. b durch das Auffüllen am Ende mit $n' - m$ bzw. $n' - n$ Nullen.

Satz 5.10 *Für die aus a und b wie oben beschrieben hervorgehenden Vektoren $a', b' \in \mathbb{R}^{n'}$ gilt:*

$$\kappa(a,b)_l = Korr(b', a')_{l \bmod n'}$$

für $minOvl - n \leq l \leq m - minOvl$.

Beweis. Sei $0 \leq l \leq m - n$. Dann gilt

$$\kappa(a,b)_l = \sum_{j=0}^{n-1} a_{j+l} b_j = \sum_{j=0}^{n-1} a'_{(j+l) \bmod n'} b'_j + \sum_{j=n}^{n'-1} a'_{(j+l) \bmod n'} b'_j = Korr(b', a')_l \ ,$$

da $a'_j = a_j$ für $0 \leq j \leq m-1$, $b'_j = b_j$ für $0 \leq j \leq n-1$ und $b'_j = 0$ für $n \leq j \leq n'-1$ gelten.

Sei nun $m - n + 1 \leq l \leq m - minOvl$. Dann gilt

$$\begin{aligned}
\kappa(a,b)_l &= \sum_{j=0}^{m-1-l} a_{j+l} b_j \\
&= \sum_{j=0}^{m-1-l} a'_{(j+l) \bmod n'} b'_j + \sum_{j=m-l}^{n'-l-1} a'_{(j+l) \bmod n'} b'_j + \sum_{j=n'-l}^{n'} a'_{(j+l) \bmod n'} b'_j \\
&= Korr(b', a')_l \ ,
\end{aligned}$$

da $a'_j = a_j$ für $m-n+1 \leq j \leq m-1$, $b'_j = b_j$ für $0 \leq j \leq n-1$, $a'_j = 0$ für $m \leq j \leq n'-1$ und $b'_j = 0$ für $n' - m + minOvl \leq j \leq n' - 1$.

Für $minOvl - n \leq l \leq -1$ gilt

$$\begin{aligned}
\kappa(a,b)_l &= \sum_{j=-l}^{n-1} a_{j+l} b_j \\
&= \sum_{j=0}^{-l-1} a'_{(j+l) \bmod n'} b'_j + \sum_{j=-l}^{n-1} a'_{(j+l) \bmod n'} b'_j + \sum_{j=n}^{n'-1} a'_{(j+l) \bmod n'} b'_j \\
&= \sum_{j=0}^{n'-1} a'_{(j+l+n') \bmod n'} b'_j = Korr(b', a')_{l+n'} = Korr(b', a')_{l \bmod n'} \ ,
\end{aligned}$$

da $a'_j = a_j$ für $0 \leq j \leq n-2$, $b'_j = b_j$ für $1 \leq j \leq n-1$, $a'_j = 0$ für $n' + minOvl - n \leq j \leq n'-1$ und $b'_j = 0$ für $n \leq j \leq n'-1$. □

Aufgrund von Korollar 5.9 kann $Korr(b', a')$ mit drei schnellen Fourier-Transformationen, n' komplexen Konjugationen und einer Division durch n' in $\theta(n' \log_2 n')$ $= \theta((n + m - minOvl) \log_2(n + m - minOvl))$, also für konstantes $minOvl$ in $\theta((n + m) \log_2(n + m))$ Zeit und $\theta(n + m)$ Platz berechnet werden. Wegen Satz 5.10 läßt sich κ aus $Korr(b', a')$ ablesen, wobei dafür nur $n + m - 2minOvl + 1$ Elemente von $Korr(b', a')$ benötigt werden. κ läßt sich demnach in $\theta((n+m) \log_2(n+m))$ Zeit und $\theta(n+m)$ Platz berechnen.

Der reell verwendete Speicherplatz kann geringfügig verbessert werden, indem ausgenutzt wird, daß die Vektoren aus reellen statt komplexen Zahlen bestehen. Da die FFT als Eingabe einen Vektor komplexer Zahlen erwartet, müssen für einen Vektor reeller Zahlen die Speicherplätze für die Imaginärteile mit Nullen aufgefüllt werden. Wie in [35] und [38] beschrieben ist, kann dies für die Berechnung zweier gleichlanger DFTs umgangen werden, indem der eine Eingabevektor als Imaginärteil des anderen Eingavektors gespeichert wird, die DFT von diesem zusammengesetzten Vektor berechnet, und anschließend die DFTs der ursprünglichen Folgen aus diesem Vektor in linearer Zeit unter Ausnutzung spezieller Eigenschaften der DFT errechnet werden. Auf diese Art und Weise halbiert sich der Platzaufwand für die Berechnung von $DFT(a')$ und $DFT(b')$, was jedoch asymptotisch keine Verbesserung darstellt.

Die Laufzeit kann ebenfalls geringfügig verbessert werden, indem in den drei FFT-Aufrufen die bitweise umgekehrten Anordnung der jeweiligen Eingabevektoren umgangen wird. Der Algorithmus FFTit aus Abbildung 5.5 wendet zuerst eine bitweise Umkehrung und danach den „eigentlichen“ Algorithmus FFTit_base an. Es gibt Algorithmen die die FFT als solche zuerst berechnen (siehe [35, 38]), so daß jedoch der Ergebnisvektor in bitweise umgekehrter Weise vorliegt, und anschließend eine bitweise umgekehrte Anordnung der Vektorelemente erfolgen muß. Ein Anwendung des letzteren Algorithmus auf die Berechnung von $DFT(a')$ und $DFT(b')$ ohne die bitweise Umkehrung, so daß die Ergebniselemente in bitweise umgekehrter Reihenfolge vorliegen, liefert dann, nach der komplexen Konjugation des einen Vektors und der elementweisen Multiplikation der Vektoren, eine passende Eingabe für den FFT_it_base-Algorithmus aus Abbildung 5.3. Dadurch entsteht eine Zeitersparnis von $\mathcal{O}(n')$, die sich jedoch nicht auf die asymptotische Laufzeit auswirkt. Ein Nachteil dieses Verfahrens ist, daß zwei verschiedene FFT-Algorithmen implementiert werden müssen.

5.3.3 Effiziente Berechnung der Korrelationskoeffizienten

Neben dem Vektor κ sind für die Berechnung des Korrelationskoeffizienten noch weitere Größen zu berechnen. Die Summen über alle b_j sowie über alle b_j^2 sollten einmal im voraus berechnet werden. Die Summen über a_{j+l} bzw. über a_{j+l}^2 können in $\mathcal{O}(n')$ Zeit

berechnet werden, indem jeweils auf die bereits berechnete Summe für $l-1$ zurückgegriffen wird. Dabei muß jedoch beachtet werden, daß ein solches Verfahren durch die vielen Subtraktionen sehr instabil sein kann.

Insgesamt können also alle Summen bis auf κ in allen zu berechnenden Korrelationskoeffizienten in Zeit und Platz $\theta(n+m)$ berechnet werden. Die Berechnung von κ benötigt Zeit $\theta((n+m)\log_2(n+m))$ und Platz $\theta(n+m)$, und pro Korrelationskoeffizient werden konstant viele weitere arithmetische Operationen ausgeführt. Insgesamt benötigt die Berechnung aller Korrelationskoeffizienten demnach eine Laufzeit von $\theta((n+m)\log_2(n+m))$ im Gegensatz zur sukkzessiven Berechnung mit Zeitaufwand $\theta(mn)$.

5.4 Anwendung der FFT auf die Berechnung von Gleichläufigkeitskoeffizienten

In diesem Unterkapitel wird eine Möglichkeit aufgezeigt, den Gleichläufigkeitskoeffizienten auf Korrelationskoeffizienten zurückzuführen, um so die FFT zur Berechnung der Glk-Werte zu verwenden.

5.4.1 Sukkzessive Berechnung

Werden anhand des sukkzessiven Basisalgorithmus aus Unterkapitel 3.2 die Gleichläufigkeitskoeffizienten berechnet, werden diese in dem Feld *result* gespeichert, das für die Indizes $minOvl-n,..,m-minOvl$ definiert ist.

$$result_l = \frac{1}{n} \sum_{j=first(l)}^{last(l)} \chi(\bar{a}_{j+l} = \bar{b}_j)$$

$$\text{wobei } first(l) = \begin{cases} -l & \text{, wenn } minOvl-n \le l \le -1 \\ 0 & \text{, sonst} \end{cases}$$

$$\text{und } last(l) = \begin{cases} n-2 & \text{, wenn } minOvl-n \le l \le m-n \\ m-l-2 & \text{, sonst} \end{cases}$$

für $minOvl-n \le l \le m-minOvl$. $\bar{a}$ und $\bar{b}$ bezeichnen die Richtungssinnvektoren, die durch Differenzenbildung aus a bzw. b entstehen.

5.4.2 Effiziente Berechnung des Vektors γ

Der Vektor γ der zu berechnenden Summen, mit

$$\gamma_l = \sum_{j=first(l)}^{last(l)} \chi(\bar{a}_{j+l} = \bar{b}_j) \tag{5.11}$$

für $minOvl - n \leq l \leq m - minOvl$ kann auf drei Korrelationsvektoren zurückgeführt werden. Dazu wird γ aufgeteilt in die drei Vektoren $\gamma^{(-1)}$, $\gamma^{(0)}$ und $\gamma^{(1)}$, die definiert sind durch

$$\gamma_l^{(k)} = \sum_{j=first(l)}^{last(l)} \chi(\bar{a}_{j+l} = \bar{b}_j = k) \tag{5.12}$$

für $k = -1, 0, 1$. Offensichtlich gilt

$$\gamma = \gamma^{(-1)} + \gamma^{(0)} + \gamma^{(1)}. \tag{5.13}$$

Jedes $\gamma^{(k)}$ kann als Korrelation κ der Folgen $\bar{a}^{(k)} \in \{0,1\}^{n-1}$ und $\bar{b}^{(k)} \in \{0,1\}^{n-1}$ mit $\bar{a}_j^{(k)} = \chi(\bar{a}_j = k)$ und $\bar{b}_j^{(k)} = \chi(\bar{b}_j = k)$ für $j = 0, \ldots, n-2$ betrachtet werden. Als Formel gilt

$$\gamma_l^{(k)} = \sum_{j=first(l)}^{last(l)} \bar{a}_{j+l}^{(k)} \bar{b}_j^{(k)} = \kappa(a^{(k)}, b^{(k)}). \tag{5.14}$$

Wie in Unterkapitel 5.3 kann jedes $\gamma^{(k)}$ mit Hilfe der FFT, in $\theta((n+m)\log_2(n+m))$ Zeit berechnet werden, so daß sich ganz γ in dieser Zeit berechnen läßt. Der zusätzlich benötigte Speicherplatz für die Folgen $a^{(k)}$ und $b^{(k)}$ ist $\theta(n+m)$.

5.4.3 Effiziente Berechnung der Gleichläufigkeitskoeffizienten

Da sich die Gleichläufigkeitskoeffizienten aus γ und einer Division berechnen, lassen sich alle Glk-Werte mit Hilfe der FFT in Zeit $\theta((n+m)\log_2(n+m))$ berechnen, im Gegensatz zur sukkzessiven Berechnung mit Zeitaufwand $\theta(mn)$. Der langsame Algorithmus ist jedoch sehr einfach und ohne zusätzlichen Speicheraufwand zu implementieren, während das $\theta((n+m)\log_2(n+m))$-Verfahren durch die Aufteilung in drei Vektoren und die Berechnung dreier FFTs verhältnismäßig kompliziert wird.

Kapitel 6

Zwei Transformationsabstände für Jahrringfolgen verschiedener Länge

Bäume bilden zwar meistens pro Jahr genau einen Jahrring aus, jedoch kann es unter bestimmten Umständen passieren, daß in einem Jahr kein Jahrring ausgebildet wird oder mehrere Jahrringe ausgebildet werden. Eine Folge mit solchen Unregelmäßigkeiten kann mit den üblichen Abstandsmaßen nicht korrekt in einer Referenzfolge lokalisiert werden, da die den Folgen zugrundeliegenden Zeitskalen nicht übereinstimmen.

In Unterkapitel 6.2 wird ein Abstandsbegriff vorgestellt, der mit Hilfe von lokalen Editierungen der Musterfolge mögliche Positionen fehlender und doppelter Ringe lokalisiert. Dieser im folgenden als *Transformationsabstand* bezeichnete Abstand wurde von Van Deusen [11] für Jahrringfolgen vorgeschlagen und wird ähnlich wie die *Edit-Distance* von Zeichenketten ([19], [48]) oder das *Dynamic Time Warping (DTW)* in der Spracherkennung ([43], [42]), definiert. Die bei Jahrringfolgen möglichen Editieroperationen sind die Zusammenfassung zweier Folgenglieder oder die Einfügung eines neuen Folgengliedes.

Um zu viele Einfügungen und Zusammenfassungen zu vermeiden, muß die Menge der zugelassenen Transformationen der Musterfolge eingeschränkt werden. Deshalb wird in Unterkapitel 6.3 ein weiterer Transformationsabstand definiert, der abhängig von der Anzahl der Editieroperationen berechnet wird. Dieser ist für die Anwendung in der Dendrochronologie besser geeignet, da in einer Jahrringfolge nur wenig doppelte oder fehlende Ringe auftreten.

6.1 Doppelte und fehlende Jahrringe

Die in den vorigen Kapiteln betrachteten Crossdatingalgorithmen gehen davon aus, daß in der Muster- und in der Referenzjahrringfolge die Elemente jeweils aufeinanderfolgenden Jahren entsprechen. Dabei wird jedoch nicht berücksichtigt, daß Bäume unter gewissen Umweltbedingungen in einem Jahr keine Wachstumsschicht (*fehlender Ring*) oder zwei Schichten (*doppelter Ring*) ausbilden können. Zwar können in einem Jahr auch mehr als zwei Ringe gebildet werden, jedoch kommt dies verhältnismäßig selten vor und wird deshalb im folgenden außer Acht gelassen. Weiterhin wird im folgenden davon ausgegangen, daß doppelte oder fehlende Ringe nur in der Muster- und nicht in der Referenzfolge auftreten können.

Bei einem Vergleich zweier Folgen ist es wichtig zu wissen, welche Indizes der beiden Folgen korrespondieren, das heißt, an welchen Stellen der beiden Folgen Elemente sinnvoll miteinander verglichen werden können. Ein Auftreten doppelter oder fehlender Ringe in der Musterfolge $x = x_0, \ldots, x_{N-1}$ verändert den normalerweise in Jahrringfolgen vorhandenen zeitlichen Ablauf in der Art, daß zwei aufeinanderfolgende Indizes nicht mehr zwei aufeinanderfolgenden Jahren entsprechen müssen. Ist x_i, x_{i+1} ein doppelter Ring, dann entsprechen i und $i+1$ demselben Jahr. Fehlt hingegen zwischen x_j und x_{j+1} ein Ring, dann entspricht j einem Jahr zwei Jahre vor dem Jahr für $j+1$.

Um die Folge x dem zeitlichen Ablauf aufeinanderfolgender Jahre anzugleichen, damit ein sinnvoller Vergleich mit einer Teilfolge $y = y_0, \ldots, y_{M-1}$ der Referenzfolge möglich ist, können folgende *Editieroperationen* auf x angewandt werden:

1. **Zusammenfassung:** Ist x_i, x_{i+1} ein doppelter Ring, dann werden die beiden Folgenglieder zu $x_i + x_{i+1}$ *zusammengefaßt*.
2. **Einfügung:** Fehlt zwischen x_j und x_{j+1} ein Ring, dann wird ein Folgenglied der Größe $\frac{x_j + x_{j+1}}{2}$ *eingefügt*.

Der Grund, daß bei einem doppelten Ring beide Folgenglieder durch Addition zusammengefaßt werden liegt darin, daß die Breite beider Jahresringe zusammen in etwa der Breite eines für dieses Jahr typischen Ringes entspricht, da die Vegetationszeit für beide Ringe insgesamt die gleiche ist wie bei anderen Bäumen für einen Ring. Bei anderen Jahrringcharakteristika als der Jahrringbreite muß eventuell anders zusammengefaßt werden.

Für einen fehlenden Ring wird oft in der Praxis eine 0 als Breite angegeben. Da dies bei der späteren Abstandsberechnung jedoch zu zu hohen lokalen Abständen an dieser Stelle führen kann, wird hier das von Van Deusen [11] vorgeschlagene arithmetische Mittel der umliegenden Jahrringbreiten als Breite eines einzufügenden Ringes verwendet. Auf diese Art und Weise erhält die Einfügung eines Ringes die gleiche Berechtigung wie das Unverändertlassen eines Ringes oder die Zusammenfassung zweier Ringe, da eine Einfügung nicht durch einen unnatürlich hohen lokalen Abstand bestraft wird.

6.2 Ein Transformationsabstand

6.2.1 Definition

Die Stellen der doppelten und fehlenden Ringe in x sind im allgemeinen nicht bekannt und stellen deshalb ein Hindernis in der Mustererkennung von x in der Referenzfolge dar. Um Stellen in x aufzufinden, die doppelten oder fehlenden Ringen entsprechen könnten und gleichzeitig einen Abstandsbegriff zu erhalten, kann der Transformationsabstand verwendet werden. Dieser wird, ähnlich wie die *Edit-Distance* bei Strings ([19], [48]) oder das *Dynamic Time Warping (DTW)* in der Spracherkennung ([43], [42]), definiert:

Definition 6.1 *Eine* Transformation τ *bildet eine Folge* $x = x_0, \ldots, x_{N-1}$ *anhand einer* Transformationsvorschrift *in die Folge* $\tau(x)$ *ab. Die Transformationsvorschrift für* τ*, die ebenfalls mit* τ *bezeichnet werden soll, ist eine Zeichenkette über dem Alphabet* $\{Z, E, I\}$ *für die* $N = 2z_\tau + id_\tau$ *gilt. Dabei bezeichnen* z_τ *die Anzahl des Buchstabens* Z *und* id_τ *die Anzahl des Buchstabens* I *in* τ *ist. Die Buchstaben entsprechen folgenden Elementaroperationen:*

- Z *Die* Zusammenfassung *zweier aufeinanderfolgender Elemente aus* x*.*
- E *Die* Einfügung *eines Elementes in* x*.*
- I *Das Übernehmen eines Elementes aus* x *(*Identitätsoperation*).*

Die ersten beiden Elementaroperationen werden zusammenfassend als Editieroperationen *bezeichnet.*

Durch τ *wird eine Abbildung von* x *beschrieben, indem* x *und* τ *gleichzeitig von links nach rechts durchgegangen werden und* x *anhand der den Buchstaben entsprechenden Elementaroperationen in* τ *verändert wird. Jedes Element von* x *wird dabei wegen* $N = 2z_\tau + id_\tau$ *genau einmal verwendet.*

	Z		Z		I	E	E	I
3	2	1	2	5				3
5			3	5	4	4	3	

So wird zum Beispiel eine Folge $3, 2, 1, 2, 5, 3$ anhand von $ZZIEEI$ wie folgt transformiert: 3 und 2 werden zu $3+2 = 5$ zusammengefaßt (Z), 1 und 2 werden zu $1 + 2 = 3$ zusammengefaßt (Z), 5 wird übernommen (I), danach werden zwei Elemente $\mu = \frac{5+3}{2} = 4$ eingefügt (EE) und die 3 wird übernommen (I), so daß insgesamt die Folge $5, 3, 5, 4, 4, 3$ entsteht. Insbesondere fassen bei einer Transformation zwei aufeinanderfolgende Zusammenfassungsoperationen nicht drei Elemente aus x zu einem, sondern hintereinander jeweils zwei zu einem zusammen. Steht vor oder nach einer Einfügung eine weitere Editieroperation, so berechnet sich der einzufügende Zahlenwert μ dennoch aus den umgebenden

Werten der Ausgangsfolge. Bei einer Einfügung am Anfang bzw. am Ende der Folge, sei μ als das erste bzw. letzte Element der Folge definiert. Die Bedingung $N = 2z_\tau + id_\tau$ versichert, daß alle Elemente von x einer Elementaroperation unterworfen werden.

Definition 6.2 *Es seien $N, M > 0$. Die Menge aller Transformationen, die eine Folge der Länge N in eine Folge der Länge M transformiert sei mit $\mathcal{T}_{N,M}$ bezeichnet. Für eine Transformation τ seien z_τ, e_τ und id_τ die Anzahlen der Zusammenfassungs-, Einfügungs- bzw. Identitätsoperationen.*

Lemma 6.3 *Es seien $N, M > 0$. Für $\tau \in \mathcal{T}_{N,M}$ gelten*

$$M = z_\tau + e_\tau + id_\tau \tag{6.1}$$
$$M = N + e_\tau - z_\tau \tag{6.2}$$

Beweis. Sei $\tau \in \mathcal{T}_{N,M}$. Jede Elementaroperation in τ entspricht genau einem Element in der Zielfolge der Länge M. Deshalb gilt $M = z_\tau + e_\tau + id_\tau$. Zusammen mit $N = 2z_\tau + id_\tau$ folgt daraus $M = N + e_\tau - z_\tau$. □

Satz 6.4 *Es seien $N, M > 0$. Es gilt $\mathcal{T}_{N,M} \neq \emptyset$ genau dann, wenn $N \leq 2M$ ist .*

Beweis. Sei $\mathcal{T}_{N,M} \neq \emptyset$ und $\tau \in \mathcal{T}_{N,M}$. Nach Lemma 6.3 gilt $M = N + e_\tau - z_\tau$ (I). Wegen $N = 2z_\tau + id_\tau$ kann τ höchstens $\lfloor \frac{N}{2} \rfloor$ Zusammenfassungen beinhalten (II). Zusammen ergeben (I) und (II) $N \leq 2M$. Seien umgekehrt N und M mit $N \leq 2M$ oder äquivalent dazu $\lceil \frac{N}{2} \rceil \leq M$ vorgegeben. Durch $\lceil \frac{N}{2} \rceil$ Zusammenfassungen mit anschließend $N - 2\lceil \frac{N}{2} \rceil$ Identitätsoperationen und $M - \lfloor \frac{N}{2} \rfloor$ Einfügungen erhält man eine die Bedingung $N = 2z_\tau + i_\tau$ erfüllende Transformation $\tau \in \mathcal{T}_{N,M}$. □

Definition 6.5 *Es seien $N, M \geq 0$. Der* Transformationsabstand D *von $x = x_0, \ldots, x_{N-1}$ und $y = y_0, \ldots, y_{M-1}$ ist definiert als*

$$D(x,y) = \begin{cases} 0 & , \textit{wenn } N{=}M{=}0 \\ \min\limits_{\tau \in \mathcal{T}_{N,M}} \sum\limits_{v=0}^{M-1} d(\tau(x)_v, y_v) & , \textit{wenn } \mathcal{T}_{N,M} \neq \emptyset \\ \textit{nicht definiert} & , \textit{sonst} \end{cases} \tag{6.3}$$

mit $d(c_1, c_2) = (c_1 - c_2)^2$. Für $N = 0$ bzw. $M = 0$ ist x bzw. y die leere Folge.

Eine Transformation, für die die Summe der Abstände das Minimum annimmt, heißt optimale Transformation.

Der Transformationsabstand ist eine natürliche Verallgemeinerung der Abstandsbegriffe für zwei Folgen gleicher Länge, die auf der Summierung elementweiser Abstände beruhen. Mit der in Definition 6.5 angegebenen lokalen Abstandsfunktion $d(c_1, c_2) = (c_1 - c_2)^2$ verallgemeinert der Transformationsabstand den quadrierten euklidischen Abstand. Für d kann aber auch ein anderer Abstandsbegriff, wie etwa der Betrag der Differenz der beiden Werte, verwendet werden.

Lemma 6.6 *Wegen Satz* 6.4 *ist der Transformationsabstand von* $x_0, \ldots, x_{N-1}$ *und* $y = y_0, \ldots, y_{M-1}$ *genau dann definiert, wenn* $0 \leq N \leq 2M$ *gilt und* N, M *nicht beide gleich* 0 *sind.*

Für eine Transformation von x nach y gibt es keine kanonisch inverse Transformation von y nach x, da eine Einfügung nicht invers zu einer Zusammenfassung ist. Deshalb ist der Transformationsabstand unsymmetrisch, d.h. es gilt im allgemeinen $D(x, y) \neq D(y, x)$.

6.2.2 Rekursionsformel

Da es exponentiell viele Transformationen gibt, die $x = x_0, \ldots, x_{N-1}$ in eine Folge der Länge M transformieren, kann der Transformationsabstand zwischen x und $y = y_0, \ldots, y_{M-1}$ nicht effizient durch die Bildung aller möglichen Transformationen und die Minimierung derer Abstände berechnet werden. Durch das Prinzip der *dynamischen Programmierung* können sowohl der Transformationsabstand als auch eine den Abstand bestimmende Transformation mit Zeit- und Platzaufwand $\mathcal{O}(NM)$ berechnet werden.

Satz 6.7 *Für den* Transformationsabstand D *zweier Jahrringfolgen* $x = x_0, \ldots, x_{N-1}$ *und* $y = y_0, \ldots, y_{M-1}$ *gilt folgende rekursive Formel:*

$$\begin{aligned} D(0,0) &= 0 \\ D(i,j) &= \min^{\star} \left\{ \begin{array}{l} D(i-2, j-1) + d(x_{i-2} + x_{i-1}, y_{j-1}), \\ D(i-1, j-1) + d(x_{i-1}, y_{j-1}), \\ D(i, j-1) + d(\mu(i-1), y_{j-1}) \end{array} \right\} \end{aligned} \tag{6.4}$$

$$\textit{für alle } 0 \leq i \leq N,\ 0 \leq j \leq M \textit{mit } i \leq 2j$$

Dabei ist $D(i,j)$ *eine abkürzende Schreibweise für* $D(x[0..i-1], y[0..j-1])$. $\min^{\star}$ *bezeichnet die Minimumbildung, in der nicht definierte Werte außer Acht gelassen werden und* $\mu(i-1)$ *sei der Wert eines in* x *einzufügenden Ringes hinter der Position* $i-1$ *(vgl. Unterkapitel 6.1).*

Lemma 6.8 *Für* $i, j > 0$ *nimmt* $D(i,j)$ *keinen anderen Wert als einen der drei zur Minimumbildung im Satz betrachteten Werte an.*

Beweis. Seien $i, j > 0$ mit $i \leq 2j$ und τ eine optimale Transformation von $x[0..i-1]$ in eine Folge der Länge j. Nach Definition muß der letzte Buchstabe der Transformation von τ einer Einfügung, Zusammenfassung oder Identität entsprechen.

Im Falle der Einfügung wird ein Element $\mu(i-1)$ hinter x_{i-1} eingefügt, so daß $\mu(i-1)$ das $(j-1)$-te Element der transformierten Folge $\tau(x)$ ist. Deshalb wird bei der Aufsummierung zur Berechnung des Abstandes für das jeweils letzte Element von $\tau(x)$ und y der Wert $d(\mu(i-1), y_{j-1})$ summiert. Die Transformation ohne das letzte Element muß, sofern sie eine gültige Transformation repräsentiert, eine optimale Transformation für $x[0..i-1]$ und $y[0,..,j-2]$ sein. Eine gültige Transformation repräsentiert sie genau dann, wenn $i \leq 2(j-1)$ gilt. Angenommen, diese Ungleichung ist erfüllt, die Transformation jedoch nicht optimal, dann würde eine optimale Transformation mit einer nachfolgenden Einfügeoperation einen kleineren Abstand von $x[0..i-1]$ zu $y[0..j-1]$ liefern. Dies wäre jedoch ein Widerspruch zur Optimalität der ursprünglich betrachteten Transformation. Im Falle einer Einfügung als letzte Elementaroperation ist also $D(i,j) = D(i,j-1) + d(\mu(i-1), y_{j-1})$, sofern $i \leq 2(j-1)$ gilt. Ansonsten ist eine Einfügung als letzte Elementaroperation nicht möglich.

Im Falle einer Zusammenfassung als letzte Elementaroperation in der betrachteten optimalen Transformation werden x_{i-1} und x_{i-2} zu einem Element zusammengefaßt. Dies trägt $d(x_{i-2} + x_{i-1}, y_{j-1})$ zur Abstandssumme bei. Die Transformation ohne das letzte Element muß aufgrund analoger Argumentation wie oben eine optimale Transformation für die jeweils restlichen Folgenglieder sein, sofern $i-2 \leq 2(j-1)$ gilt. Deshalb ist in diesem Fall $D(i,j) = D(i-2,j-1) + d(x_{i-2} + x_{i-1}, y_{j-1})$, wenn $i-2 \leq 2(j-1)$. Ansonsten ist eine Zusammenfassung als letzte Elementaroperation nicht zugelassen.

Ist die letzte Elementaroperation eine Identitätsoperation, wird x_{i-1} bei der Transformation beibehalten, so daß $d(x_{i-1}, y_{j-1})$ zur Summe beiträgt. Sofern $i-1 \leq 2(j-1)$ ist, gilt $D(i,j) = D(i-1,j-1) + d(x_{i-1}, y_{j-1})$, ansonsten ist eine Identitätsoperation als letzte Elementaroperation nicht zugelassen.

Da das letzte Element der Transformationsvorschrift eine der drei Elementartransformationen ist, kann $D(i,j)$ keinen anderen Wert als $D(i-2,j-1) + d(x_{i-2} + x_{i-1}, y_{j-1})$, $D(i-1,j-1) + d(x_{i-1}, y_{j-1})$ oder $D(i,j-1) + d(\mu(i-1), y_{j-1})$ annehmen. □

Lemma 6.9 *Für alle i, j mit $0 < i \leq N$, $0 < j \leq M$ und $i \leq 2j$ ist mindestens einer der Werte $D(i-2,j-1)$, $D(i-1,j-1)$ oder $D(i,j-1)$ definiert.*

Beweis. Nach Lemma 6.6 ist der Transformationsabstand $D(\bar{i}, \bar{j})$ genau für $0 \leq \bar{i} \leq 2\bar{j}$ definiert. Für $\bar{i} = i-2$ und $\bar{j} = j-1$ muß deshalb $0 \leq i-2 \leq 2j-2$, für $\bar{i} = i-1$ und $\bar{j} = j-1$ muß $0 \leq i-1 \leq 2j-2$ und für $\bar{i} = i$ und $\bar{j} = j-1$ muß $0 \leq i \leq 2j-2$ gelten. Da diese Intervalle für i das Intervall $0 \leq j \leq 2j$ lückenlos überdecken, ist mindestens einer der drei Werte definiert. □

Beweis von Satz 6.7. Nach dem Lemma 6.8 kann $D(i,j)$ nur einen der drei betrachteten Werte annehmen. Da jeder der drei Werte einer Transformation von $x[0..i-1]$ entspricht, für $D(i,j)$ aber eine optimale Transformation gesucht ist, muß $D(i,j)$ das Minimum $\min^\star$ der drei Werte sein. Da nach Lemma 6.9 mindestens einer der drei Werte definiert ist, ist die Minimumbildung ebenfalls definiert. □

6.2.3 Berechnung des Abstandes mit Hilfe der dynamischen Programmierung

Die Rekursionsgleichung liefert den Ansatz zur Berechnung des Transformationsabstandes mit Hilfe der *dynamischen Programmierung*. Im Gegensatz zu einer einfachen Top-Down-Auswertung der Rekursion, die durch mehrfache Berechnung derselben Teilergebnisse zu einer exponentiellen Laufzeit führt, verfolgt die dynamische Programmierung einen effizienteren Bottom-Up-Ansatz.

Alle $D(i,j)$ werden systematisch für *aufsteigende* Werte von i und j, beginnend bei $i=0$ und $j=0$, berechnet und abgespeichert, so daß am Ende das erwünschte Resultat $D(N,M)$ vorliegt. Die Wertebereiche für i und j, nämlich $0 \le j \le M$ und, abhängig von j, $0 \le i \le \min(N, 2j)$, gehen aus den Bedingungen des Satzes 6.7 hervor. Durch die in Satz 6.10 aufgestellte zusätzliche Bedingung werden diese noch verkleinert.

Satz 6.10 *Bei der Berechnung von $D(N,M)$ anhand der Rekursionsgleichung* (6.4) *wird nur auf $D(i,j)$ mit $N-i \le 2(M-j)$ zurückgegriffen.*

Beweis. Sei $\tau \in \mathcal{T}_{N,M}$ eine optimale Transformation für $D(N,M)$. Zur Berechnung von $D(N,M)$ mit Hilfe der Rekursionsgleichung werden solche $D(i,j)$ benötigt, für die die zugehörige optimale Transformation $\tau_1 \in \mathcal{T}_{i,j}$ ein Präfix von τ sein kann. Für den entsprechenden Suffix $\tau_2 \in \mathcal{T}_{N-i,M-j}$ von τ liefert die für eine Transformation notwendige Bedingung aus Satz 6.4 $N-i \le 2(M-j)$. □

Die zusätzliche Bedingung aus Satz 6.10 ergibt umgeformt $i \ge N-2M+2j$, und damit den von j abhängigen Bereich

$$minI(j) := \max(0, N-2M+2j) \le i \le \min(N,2j) := maxI(j) \tag{6.5}$$

für i. Die für die Berechnung von $D(i,j)$ zu betrachtenden Werte $D(\bar{i},\bar{j})$ mit $(\bar{i}=i-2, \bar{j}=j-1)$, $(\bar{i}=i-1, \bar{j}=j-1)$ und $(\bar{i}=i, \bar{j}=j-1)$ sind nur definiert, falls $\bar{i}$ und $\bar{j}$ ebenfalls in den Wertebereichen für i und j liegen, d.h. wenn $0 \le \bar{j} \le M$ und $minI(\bar{j}) \le \bar{i} \le maxI(\bar{j})$ gelten.

Den Algorithmus zur Berechnung des Transformationsabstandes zeigt Abbildung 6.1. Die Berechnung erfolgt in Teilen einer $(N+1)\times(M+1)$-Matrix, deren Zeilen- und Spaltennumerierung jeweils den Bereichen der Indizes i und j entspricht. Am Anfang wird das

Transformationsabstand($x[0..N-1],\ y[0..M-1]$)**:**
// Initialisierung
$D[0,0] = 0$;

// Spaltenweises Füllen der Matrix mit Hilfe der Rekursionsgleichung
for $(j = 1;\ j \leq M;\ j++)\{$
  for $(i =$minl$(j);\ i \leq$ maxl$(j);\ i++)\{$
    if $($minl$(j-1) \leq i-2 \leq$maxl$(j-1))\{$
      // Zusammenfassung
      $doubl = D[i-2, j-1] + d(x[i-2] + b[i-1], y[j-1])$;
    $\}$
    if (minl$(j-1) \leq i-1 \leq$maxl$(j-1)$)$\{$
      // Identität
      $match = D[i-1, j-1] + d(x[i-1], y[i-1])$;
    $\}$
    if $($minl$(j-1) \leq i \leq$maxl$(j-1))\{$
      // Einfügung
      $miss = D[i, j-1] + d(\mu(i-1), y[i-1])$;

    $\}$
    $min = \min^{\star}(doubl, match, miss)$;
  $\}$ // for j
$\}$ // for i
return D;

Abbildung 6.1: DP-Algorithmus zur Berechnung des Transformationsabstandes zweier Jahrringfolgen $x_0, \ldots, x_{N-1}$ und $y_0, \ldots, y_{M-1}$. Dabei seien minl(j) und maxl(j) wie in Formel (6.5) definiert.

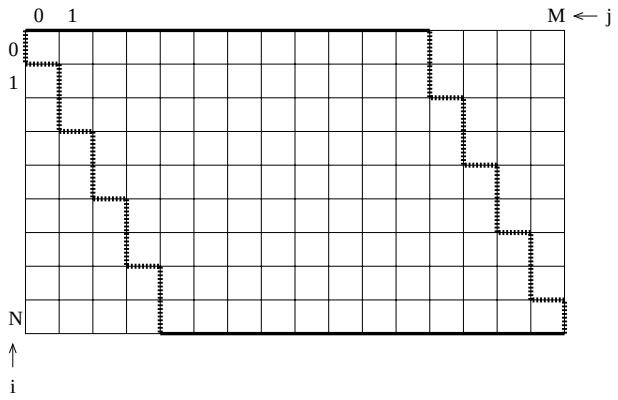

Abbildung 6.2: Matrix für die Berechnung des Transformationsabstandes anhand des Algorithmus aus Abbildung 6.1. Der verwendete Bereich der Matrix ist eingerahmt dargestellt, wobei die linke gestrichelte Begrenzungslinie der Bedingung $i \leq 2j$ und die rechte gestrichelte Begrenzungslinie der Bedingung $(N - i) \leq 2(M - j)$ entspricht.

Feld $(0, 0)$ mit der Anfangsbedingung $D(0, 0) = 0$ initialisiert. Danach werden zeilenweise für aufsteigende i und j innerhalb der Wertebereiche alle $D(i, j)$-Werte berechnet und in die Matrix an der entsprechenden Stelle eingetragen, so daß das Resultat am Ende in Zelle (N, M) steht. Abbildung 6.2 zeigt, welche Zellen der Matrix verwendet werden.

Die Anzahl der verwendeten Zellen in der Matrix ist

$$\sum_{j=0}^{M} (\min(N, 2j) - \max(0, N - 2M + 2j)) = NM - \lceil \frac{N^2}{2} \rceil + M + 1 = \mathcal{O}(NM)$$

Da, bis auf die Initialisierung von $D(0, 0)$, für jeden $D(i, j)$-Wert drei Werte betrachtet und deren Minimum gebildet werden muß, ist die Laufzeit des Algorithmus proportional zur Anzahl der verwendeten Zellen. Der verwendete Speicherplatz besteht bis auf eine konstante Anzahl von Hilfsvariablen hauptsächlich aus den verwendeten Zellen der Matrix, sofern nur für die verwendeten Zellen Speicherplatz reserviert wird. Der Algorithmus benötigt demnach $\mathcal{O}(NM)$ Zeit und Platz.

Die vollständig ausgefüllte Berechnungsmatrix wird für die Berechnung einer dem Abstand entsprechenden optimalen Transformation benötigt. Hierauf wird in Abschnitt 6.2.5 eingegangen. Wenn nur der Abstand, ohne eine zugehörige optimale Transformation, berechnet werden soll, muß nicht die gesamte Matrix gespeichert werden. Bei einem spaltenweisen bzw. zeilenweisen Ausfüllen der Matrix genügt es, immer nur die zwei zuletzt ausgefüllten Spalten bzw. Zeilen im Speicher zu halten. Auf diese Art und Weise reduziert sich der Speicherplatz auf $\theta(min(M, N))$.

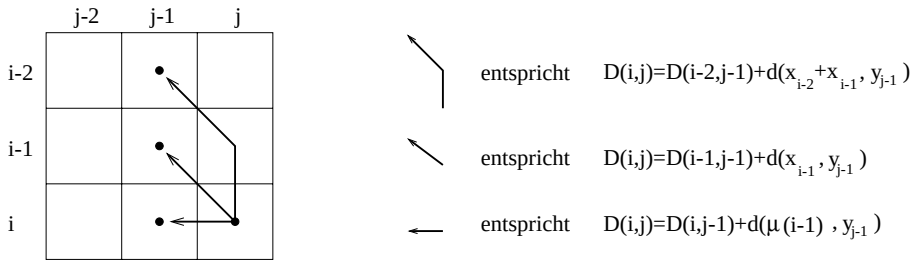

Abbildung 6.3: Mögliche von einem Knoten (i, j) ausgehende Kanten in einem Berechnungsgraphen

6.2.4 Berechnungsgraph

Die rekursive Berechnung des Transformationsabstandes kann anhand eines *Berechnungsgraphen* verdeutlicht werden. Die Knoten entsprechen dabei den Zellen der Berechnungsmatrix. Zwischen zwei Knoten k_1, k_2 gibt es genau dann eine gerichtete Kante $k_1 \to k_2$, wenn sich in einem Rekursionsschritt, also bei der Minimierung über die drei möglichen Werte, $D(k_1)$ aus $D(k_2)$ berechnet.

Für einen Knoten gibt es demnach drei mögliche ausgehende Kanten, wie in Abbildung 6.3 ersichtlich ist, von denen mindestens eine vorhanden sein muß, aber auch zwei oder alle drei vorhanden sein können.

6.2.5 Berechnung einer optimalen Transformation anhand der Berechnungsmatrix

Aus dem Beweis für Lemma 6.8 geht hervor, daß eine optimale Transformation in kanonischer Weise mit der Rekursivität des Transformationsabstandes zusammenhängt:

- Ist $D(i,j) = D(i-2, j-1) + d(x_{i-2} + x_{i-1}, y_{j-1})$, dann ist jede optimale Transformation für $x[0..i-3]$ und $y[0..j-2]$ mit einer anschließenden *Zusammenfassung* eine optimale Transformation für $x[0..i-1]$ und $y[0..j-1]$.
- Ist $D(i,j) = D(i-1, j-1) + d(x_{i-1}, y_{j-1})$, dann ist jede optimale Transformation für $x[0..i-2]$ und $y[0..j-2]$ mit einer anschließenden *Identitätsoperation* eine optimale Transformation für $x[0..i-1]$ und $y[0..j-1]$.
- Ist $D(i,j) = D(i, j-1) + d(\mu(i-1), y_{j-1})$, dann ist jede optimale Transformation für $x[0..i-1]$ und $y[0..j-2]$ mit einer anschließenden *Einfügung* eine optimale Transformation für $x[0..i-1]$ und $y[0..j-1]$.

Jede Elementaroperation entspricht demnach genau einer möglichen Kante im Berechnungsgraphen, so daß ein Pfad im Berechnungsgraphen von (N, M) nach $(0, 0)$ eine Trans-

Transformation$(D[N+1][M+1], x[0..N-1], y[0..M-1])$**:**

```
// Ausgabe einer optimalen Transformationsvorschrift
// Initialisierung
i=N;
j=M;

// Zurückverfolgung eines Berechnungsweges
while ((i ≥ 0)&&(j ≥ 0))) {
   if (D(i,j) == D(i-1,j-1) + d(x[i-1], y[j-1])) {
      // Identität
      print('I');
      i=i-1;
      j=j-1;
   } else if (D(i,j) == D(i-2,j-1) + d(x[i-2] + x[i-1], y[j-1])) {
      // Zusammenfassung
      print('Z');
      i=i-2;
      j=j-1;
   } elsif (D(i,j) == D(i,j-1) + d(μ(i-1), y[j-1])) {
      // Einfügung
      print('E');
      j=j-1;
   }

}
```

Abbildung 6.4: Algorithmus zur Berechnung und Ausgabe einer optimalen Transformation aus der Berechnungsmatrix D des Transformationsabstandes der Jahrringfolgen $x_0, \ldots, x_{N-1}$ und $y_0, \ldots, y_{M-1}$. Zugriffe auf nicht definierte Teile der Matrix D innerhalb einer *if*-Abfrage sollen so verstanden werden, daß die *if*-Abfrage *false* liefert.

formationsvorschrift definiert.

Eine Transformationsvorschrift wird deshalb durch eine Zurückverfolgung der erfolgten Berechnungsschritte anhand der Matrix D berechnet (siehe Abbildung 6.4). Ausgehend von der Zelle (N, M) wird eine vorhandene Kante des Berechnungsgraphen zurückverfolgt und als entsprechende Elementaroperation interpretiert.

Die Zurückverfolgung der einzelnen Berechnungsschritte kann erleichtert werden, indem während des Berechnungsalgorithmus für jedes betrachtete Feld (i, j) der Berechnungsmatrix abgespeichert wird, welche der drei Zellen $(i-1, j-2)$, $(i-1, j-1)$ und $(i, j-1)$ zur Minimumbildung beigetragen haben. Dadurch erhöht sich der reelle jedoch nicht der asymptotische Speicheraufwand, da es sich nur um konstanten zusätzlichen Platz pro Zelle handelt.

Die Länge einer Transformation, also die Anzahl aller in ihr enthaltenen Elementaroperationen, ist nach Satz 6.3 gleich M. Der Algorithmus zur Berechnung einer Transformation benötigt konstante Zeit für jede auszugebende Elementaroperation, also $\theta(M)$.

6.2.6 Notwendigkeit der Einschränkung der Transformationsmenge

Der in diesem Unterkapitel definierte Transformationsabstand liefert ohne eine Einschränkung der zugelassenen Transformationsmenge keine geeigneten Informationen für das Crossdating. Der Grund dafür liegt darin, daß Jahrringfolgen wenig fehlende und doppelte Ringe ausbilden. Der Transformationsabstand zwischen zwei Folgen wird jedoch über alle möglichen Transformationen minimiert, d.h. es können möglicherweise viele Zusammenfassungen und Einfügungen in einer optimalen Transformation vorhanden sein, was den Abstand zwar minimiert, jedoch vom dendrochronologischen Standpunkt her wenig sinnvoll ist. Wünschenswert ist deshalb die Berechnung von Transformationsabständen mit einer eingeschränkten Menge zugelassener Transformationen, die wenig Zusammenfassungen und Einfügungen enthalten.

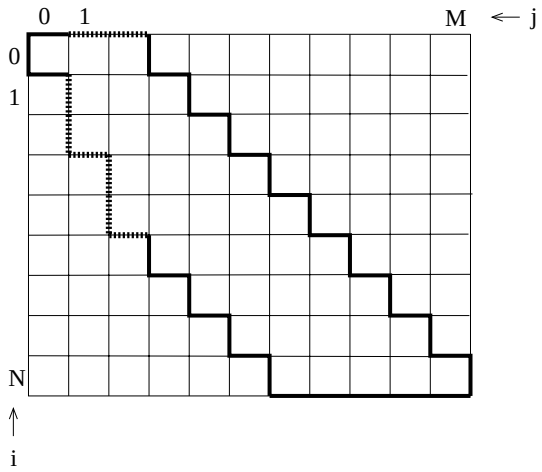

Abbildung 6.5: Berechnungsmatrix für die Berechnung des Transformationsabstandes unter der Einschränkung der Transformationsmenge auf Transformationen τ, die für alle Präfixe τ' die Bedingung $|z_{\tau'} - e_{\tau'}| \leq \alpha = 2$ erfüllen.

Van Deusen definiert in [11] einen Transformationsabstand wie in Abschnitt 6.2.1 erläutert. Die Menge der zugelassenen Transformationen schränkt er jedoch ein, indem er in der Berechnungsmatrix nur die Zellen berechnet, die auf der bei $(0, 0)$ beginnenden Diagonalen $j - i = 0$ sowie auf je α Nebendiagonalen liegen. Dadurch werden nur solche Transformationen τ betrachtet, die für alle Präfixe τ' die Zusatzbedingung $|z_{\tau'} - e_{\tau'}| \leq \alpha$ erfüllen. Insbesondere sind hierin auch die Transformationen enthalten, die $\leq \alpha$ Editieroperationen enthalten. Abbildung 6.5 zeigt eine solche Berechnungsmatrix, in der der zu berechnende Bereich für $\alpha = 2$ eingezeichnet ist. Die gestrichelten Linien verdeutlichen

das α. Die Länge einer unter diesen Bedingungen mit y vergleichbaren Folge ist durch diese Zusatzbedingung auf mindestens $|x| - \alpha$ und höchstens $|x| + \alpha$ begrenzt.

Transformationen, die der obigen Zusatzbedingung genügen, können aber dennoch weitaus mehr als α Editieroperationen enthalten, solange sich lokal die Anzahl der Zusammenfassungen und Einfügungen ausgleicht. Ein von der Anzahl der Editieroperationen abhängiger Transformationsabstand wird in Unterkapitel 6.3 entwickelt. Ein auf diesem Abstand basierender Crossdatingalgorithmus, dessen Implementation ein Teil dieser Diplomarbeit ist, wird in Kapitel 7 vorgestellt.

6.3 Ein von der Anzahl der Editieroperationen abhängiger Transformationsabstand

Für eine Auswertung der Ähnlichkeit zweier Jahrringfolgen anhand des Transformationsabstandes ist die Anzahl der in einer optimalen Transformation enthaltenen Editieroperationen ein wichtiges Kriterium. Ein geringer Transformationsabstand, dessen zugehörige optimale Transformation viele Irregularitäten in die Jahrringfolge x hineininterpretiert ist sicherlich nicht so überzeugend, wie ein etwas größerer Abstand, dessen zugehörige Transformation sehr wenig Irregularitäten in x erkennen läßt. Deshalb wird in diesem Unterkapitel der Transformationsabstand abhängig von der Anzahl der in der optimalen Transformation enthaltenen Editieroperationen berechnet. Ein solcher Abstandsbegriff wird bei Kruskal und Sankoff [28] für den Vergleich zweier Zeichenketten vorgeschlagen.

Als Crossdatingalgorithmus werden dann in Kapitel 7 für eine vorgegebene Höchstanzahl α der Editieroperationen zwischen x und allen möglichen zusammenhängenden Teilfolgen in y die Abstände berechnet.

6.3.1 Definition

Definition 6.11 *Für den Vergleich zweier Jahrringfolgen $x = x_0, \ldots, x_{M-1}$ und $y = y_0, \ldots, y_{N-1}$ sei $D(i,j,k)$ der Transformationsabstand $D(i,j)$ der Präfixe $x[0..i-1]$ und $y[0..j-1]$ unter ausschließlicher Betrachtung solcher Transformationen, die genau k Editieroperationen enthalten. Es ist*

$$D(i,j,k) = \begin{cases} 0 & , \textit{wenn } i{=}j{=}k{=}0 \\ \min\limits_{\tau \in \mathcal{T}_{i,j,k}} \sum\limits_{v=0}^{j-1} d(\tau(x)_v, y_v) & , \textit{wenn } \mathcal{T}_{i,j,k} \neq \emptyset \\ \textit{nicht definiert} & , \textit{sonst} \end{cases} \tag{6.6}$$

wobei $\mathcal{T}_{i,j,k}$ die Menge der Transformationen ist, die eine Folge der Länge i in eine Folge der Länge j mit genau k Editieroperationen transformiert und $d(c,d) = (c-d)^2$.

Der Abstand $D(N,M,k)$ wird im folgenden als k-Abstand von x und y bezeichnet.

Satz 6.12 *Seien $i, j, k \geq 0$. Es gilt $\mathcal{T}_{i,j,k} \neq \emptyset$ genau dann, wenn die Bedingungen*

1. *$j \geq k$ und*
2. *$j - i = k - 2\nu$ für ein $\nu \in \{0, \ldots, k\}$*

erfüllt sind.

Beweis. Sei $\tau \in \mathcal{T}_{i,j,k}$. Für die dadurch definierten i und j gilt nach Satz 6.3 $j = i + e_\tau - z_\tau$. Eine Umformung dieser Gleichung unter Berücksichtigung von $e_\tau = k - z_\tau$ führt zu $j - i = k - 2z_\tau$. Da z_τ zwischen 0 und k liegt gilt die zweite Bedingung. Die erste Bedingung folgt ebenfalls aus Satz 6.3 durch $j = e_\tau + z_\tau + id_\tau = k + id_\tau \geq k$.

Seien umgekehrt die Parameter i, j und k gegeben, so daß sie die zwei Bedingungen erfüllen. Für $z_\tau := \nu = \frac{k-j+i}{2}$, $e_\tau := k - z_\tau$ und $id_\tau = j - k$, die aufgrund der beiden Bedingungen definiert und nicht negativ sind, gilt $2z_\tau + id_\tau = i$. z_τ Zusammenfassungen, e_τ Einfügungen und id_τ Identitätsoperationen in beliebiger Reihenfolge definieren deshalb eine Transformation τ mit k Editieroperationen, die eine Folge der Länge i in eine Folge der Länge $k + id_\tau = j$ transformiert. □

Korollar 6.13 *Der k-Abstand $D(i, j, k)$ ist genau dann definiert, wenn*

1. *$i \geq 0$,*
2. *$0 \leq k \leq j$ und*
3. *$j - i = k - 2\nu$ für ein $\nu \in \{0, \ldots, k\}$*

gelten.

6.3.2 Rekursionsformel

Für $D(i, j, k)$ gilt eine ähnliche Rekursionsformel wie für $D(i, j)$.

Satz 6.14 *Für $D(i, j, k)$ gilt folgende rekursive Formel:*

$$\begin{aligned} D(0,0,0) &= 0 \\ D(i,j,k) &= \min{}^{\star} \left\{ \begin{array}{l} D(i-2, j-1, k-1) + d(x_{i-2} + x_{i-1}, y_{j-1}), \\ D(i-1, j-1, k) + d(x_{i-1}, y_{j-1}), \\ D(i, j-1, k-1) + d(\mu(i-1), y_{j-1}) \end{array} \right\} \end{aligned} \tag{6.7}$$

für alle $i, j, k \geq 0$
mit $j \geq k$ und $j - i = k - 2\nu$ für ein $\nu \in \{0, \ldots, k\}$.

Lemma 6.15 *Für alle $i, j, k \geq 0$, jedoch nicht alle gleich 0, die den Bedingungen aus Satz 6.14 genügen, ist mindestens einer der zur Minimumbildung betrachteten Rekursionswerte definiert.*

Beweis. Aus den Bedingungen aus Korollar 6.13 ergibt sich, daß für i, j, k mit

(I) $i \geq 2$, $j \geq k \geq 1$ und $\nu \geq 1$

$D(i-2, j-1, k-1)$ definiert ist. $D(i-1, j-1, k)$ ist definiert, falls

(II) $i \geq 1$ und $j \geq k+1 \geq 1$

gelten. Wenn

(III) $i \geq 0$, $j \geq k \geq 1$ und $\nu \leq k-1$

sind, existiert $D(i, j-1, k-1)$.

Es bleibt zu zeigen, daß für jedes Tripel (i, j, k) einer der drei Fälle (I), (II) oder (III) erfüllt und somit mindestens einer der drei Werte definiert ist. Sei $i = 0$. Aus $j - i = k - 2\nu$ und $j \geq k$ folgen $\nu = 0$ und $j = k$. Wenn $k = 0$ ist, dann ist $(i, j, k) = (0, 0, 0)$. Für $k > 0$ gilt (III). Sei nun $i = 1$. Es folgen $\nu = 0$ und $j = k + 1$, weshalb (II) gilt. Sei $i \geq 2$. Für $j = k$ folgt $i = 2\nu$ und daraus (I), und für $j > k$ folgt direkt (II). □

Beweis von Satz 6.14. Der Unterschied zur Rekursionsformel (6.4) aus Satz 6.7 besteht in dem zusätzlichen Parameter k, der Bedingung $j \geq k$ und, statt $i \leq 2j$, $j - i = k - 2\nu$ für ein $\nu \in \{0, \ldots, k\}$.

Die Wahl des jeweils dritten Parameters bei den drei zur Minimierung betrachteten Transformationsabständen geht aus der Betrachtung der zugehörigen optimalen Transformationen hervor: Der Beweis geht wie im Beweis von Satz 6.7 auf die Betrachtung einer optimalen Transformation zurück. Im ersten Fall ist die letzte Elementaroperation eine Zusammenfassung, so daß die restliche Transformation genau $k - 1$ Editieroperationen enthalten muß, um mit der letzten Zusammenfassung insgesamt auf k Editieroperationen zu kommen. Im zweiten Fall ist die letzte Elementaroperation eine Identitätsoperation, so daß die k Editieroperationen in der restlichen Transformation enthalten sein müssen. Im dritten Fall handelt es sich bei der letzten Elementaroperation um eine Einfügung, so daß in der Rekursion $k - 1$ als dritter Parameter eingeht.

Nach Lemma 6.15 existiert mindestens einer der drei zur Minimierung betrachteten Werte. Aus Satz 6.12 folgt, daß $D(i, j, k)$ genau für i, j, k mit den Bedingungen aus dem Satz existiert. □

6.3.3 Berechnung des Abstandes mit Hilfe der dynamischen Programmierung

6.3.3.1 Berechnungsquader

Die Berechnung des α-Abstandes zweier Folgen $x[0..N-1]$ und $y[0..M-1]$ kann anhand der Rekursionsformel (6.7) mit Hilfe der dynamischen Programmierung in Teilen eines $(N+1)\times(M+1)\times(\alpha+1)$-Quaders durchgeführt werden. Eine Zelle (i,j,k) des Quaders, wobei $0 \le i \le N$, $0 \le j \le M$ und $0 \le k \le \alpha$ sind, enthält dabei den Wert $D(i,j,k)$. Für die Berechnung einer Zelle (i,j,k), die sich in der k-ten Ebene des Quaders befindet, wird auf die Zelle $(i-1,j-1,k)$ in derselben Ebene sowie auf die Zellen $(i-2,j-1,k-1)$ und $(i,j-1,k-1)$ in der $(k-1)$-ten Ebene zurückgegriffen. Abbildung 6.6 veranschaulicht dies anhand der entsprechenden Kanten im Berechnungsgraphen.

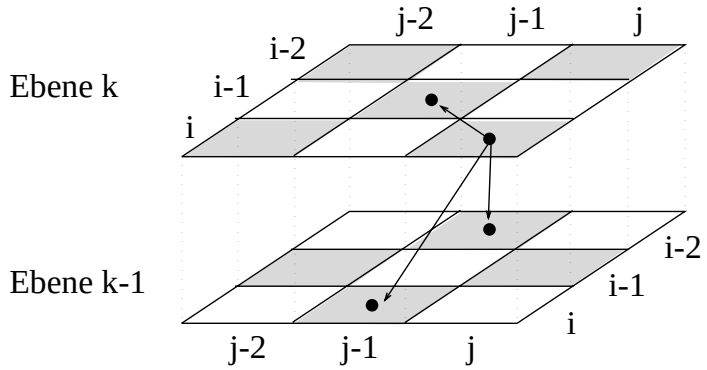

Abbildung 6.6: Mögliche von (i,j,k) ausgehende Kanten im Berechnungsgraphen. Für die Berechnung von $D(i,j,k)$ wird auf zwei Zellen der $(k-1)$-ten und auf eine Zelle der k-ten Ebene des Berechnungsquaders zurückgegriffen.

Die Zellen (i,j,k) des Quaders, für die die entsprechenden Abstände $D(i,j,k)$ überhaupt definiert sind, müssen nach Satz 6.12 den Bedingungen $j \ge k$ und $j-i = k-2\nu$ für ein $\nu \in \{0 \ldots k\}$ genügen.

In Abbildung 6.7 ist ein Berechnungsquader für $N=4$, $M=7$ und $\alpha=3$ abgebildet, in dem die Zellen mit definierten Abständen grau eingezeichnet sind. Durch die Bedingung $j-i=k-2\nu$ für ein $\nu \in \{0,\ldots,k\}$ sind nur die eingezeichneten Diagonalen $j-i$ belegt. Wegen $j \ge k$ werden nur Kästchen ab der j-ten Spalte verwendet.

Satz 6.16 *Es seien $k \ge 0$ und eine Diagonale $d = k-2\nu$ für ein $\nu \in \{0,\ldots,k\}$ vorgegeben. Für alle $D(i,j,k)$ mit $j-i=d$ ist die Anzahl der Zusammenfassungen z und der Einfügungen e der betrachteten Transformationen gleich und läßt sich durch*

$$z = \nu \tag{6.8}$$

$$\textit{und} \quad e = k-\nu \tag{6.9}$$

berechnen.

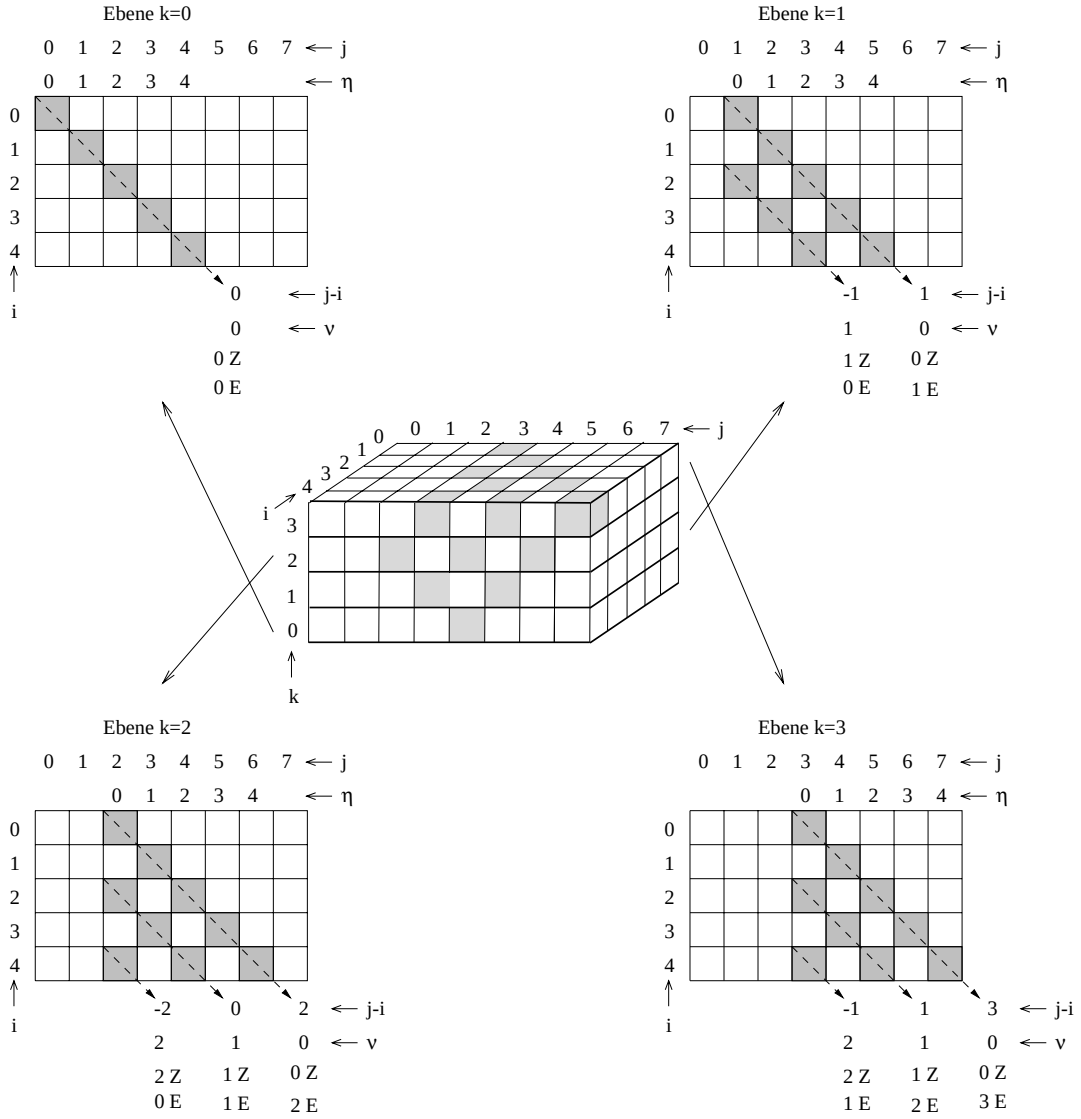

Abbildung 6.7: Berechnungsquader für die Berechnung des 3-Transformationsabstandes für zwei Folgen x der Länge 4 und y der Länge 7. Alle definierten Speicherzellen sind grau eingezeichnet. Die den Diagonalen entsprechenden Anzahlen der Zusammenfassungen (Z) und Einfügungen (E) sind für jede Diagonale angegeben.

Beweis. Aus dem Beweis für Satz 6.12 geht hervor, daß für eine Transformation τ, die für die Berechnung eines Kästchens (i, j) in der k-ten Ebene betrachtet wird, $j - i = k - 2z_\tau$ gilt. Für jedes Kästchen läßt sich deshalb aus $j - i$ und k die Anzahl der Zusammenfassungen z und Einfügungen e der für $D(i, j, k)$ betrachteten Transformationen durch $z = \frac{k-(j-i)}{2}$ und $e = k - z$ berechnen. Für die Kästchen auf der Diagonalen d sind alle $j - i = d$ gleich, so daß sich z und e durch die behaupteten Formeln berechnen lassen. □

Die Ebene sowie die Diagonale, auf der ein Kästchen liegt, bestimmen also die Anzahl der vorhandenen Einfügungs- und Zusammenfassungsoperationen in den für dieses Kästchen betrachteten Transformationen. In dem Beispiel in Abbildung 6.7 sind die Anzahlen der Einfügungs- und Zusammenfassungsoperationen für jede Diagonale angegeben.

Eine Berechnung des Abstandes $D(N, M, \alpha)$ kann nun durch systematisches Füllen des Berechnungsquaders, also für aufsteigende i, j und k, durchgeführt werden. In Unterabschnitt 6.3.3.2 wird eine andere Indizierung des Berechnungsquaders vorgestellt, die es erlaubt, nur die definierten Kästchen des Quaders abzuspeichern.

6.3.3.2 Indizierung des Berechnungsquaders mit (k, ν, η)

Um beim Füllen des Berechnungsquaders nur die Kästchen abzuspeichern, für die die entsprechenden Abstände überhaupt definiert sind, ist eine Indizierung der Kästchen statt mit (i, j, k) mit (k, ν, η) sinnvoll, wobei k wie vorher die Ebene, ν die durch

$$j - i = k - 2\nu \quad \text{mit} \quad 0 \leq \nu \leq k \tag{6.10}$$

festgelegte Diagonale und

$$\eta := i - 2\nu \tag{6.11}$$

die Position in der Diagonalen bezeichnen. Das folgende Lemma und die zwei folgenden Sätze stellen Beziehungen zwischen diesen beiden Darstellungen her.

Lemma 6.17 *Für (i, j, k) und (k, ν, η) gelten folgende Beziehungen:*

1. $i = 2\nu + \eta$
2. $j = k + \eta$

und

3. $\nu = \frac{i-j+k}{2}$
4. $\eta = j - k$

Beweis. Alle Gleichungen folgen direkt aus den Definitionen $\eta := i - 2\nu$ und $j - i = k - 2\nu$. □

Satz 6.18 *Die folgenden Bedingungen charakterisieren genau alle definierten Zellen in einem Berechnungsquader der Größe* $(N+1) \times (M+1) \times (\alpha+1)$*:*

1. $0 \leq k \leq \alpha$
2. $0 \leq \nu \leq \min(k, \lfloor \frac{N}{2} \rfloor) =: maxNu(k)$
3. $0 \leq \eta \leq \min(N - 2\nu, M - k) =: maxEta(k, \nu)$

Beweis. Durch ein Einsetzen der Definitionen für ν und η wird deutlich, daß $\eta \geq 0$ zu $j \geq k$, $\eta \leq M - k$ zu $j \leq M$, $\eta \leq N - 2\nu$ zu $i \leq N$ und $\nu \geq 0$ zu $i \geq 0$ äquivalent sind. $\nu \leq \frac{N}{2}$ folgt aus $0 \leq i \leq N$ und $k \leq j \leq M$. □

Der folgende Satz gibt Aufschluß darüber, wie die Rekursionsgleichung für die Berechnung des Transformationsabstandes bei einer Indizierung des Berechnungsquaders mit (k, ν, η) verwendet wird. Als Vergleich zu Abbildung 6.6 sind in Abbildung 6.8 die für die Berechnung einer Zelle (k, ν, η) benötigten Zellen in (k, ν, η)-Indizierung angegeben.

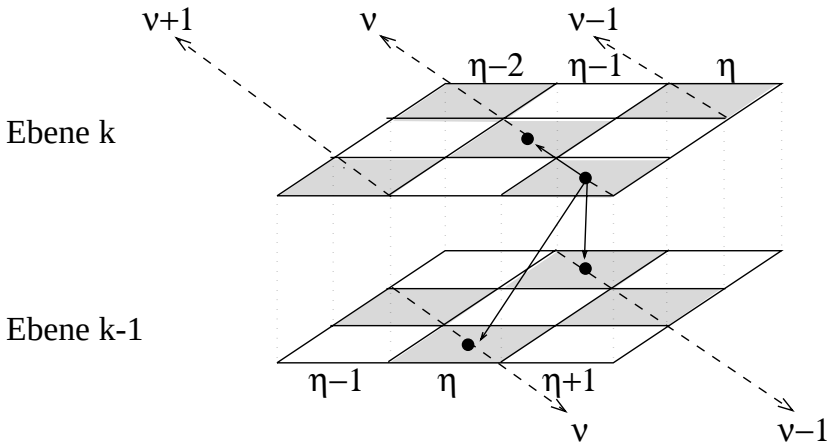

Abbildung 6.8: Mögliche von einer Zelle (k, ν, η) ausgehende Kanten im Berechnungsgraphen.

Satz 6.19 *Für die zwei Darstellungen* (i, j, k) *und* (k, ν, η) *gelten folgende Entsprechungen:*

$$\begin{array}{lcl} (i-2, j-1, k-1) & \longleftrightarrow & (k-1, \nu-1, \eta) \\ (i-1, j-1, k) & \longleftrightarrow & (k, \nu, \eta-1) \\ (i, j-1, k-1) & \longleftrightarrow & (k-1, \nu, \eta) \end{array}$$

Beweis. Um eine Entsprechung $(\bar{i}, \bar{j}, \bar{k}) \longleftrightarrow (\bar{k}, \bar{\nu}, \bar{\eta})$ zu zeigen, wird zweimal das Lemma 6.17 verwendet. Das erste Mal werden die Gleichungen 1) und 2) auf i, j, k, ν, η und das zweite Mal die Gleichungen 3) und 4) auf $\bar{i}, \bar{j}, \bar{k}\, \bar{\nu}, \bar{\eta}$ angewendet.

Sei ($\bar{i} = i-2, \bar{j} = j-1, \bar{k} = k-1$). Nach dem Lemma gelten $\bar{i} = 2\nu+\eta-2$, $\bar{j} = k+\eta-1$ und $\bar{k} = k-1$. Eine weitere Verwendung des Lemmas liefert $\bar{\nu} = \nu - 1$ und $\bar{\eta} = \eta$, also insgesamt ($\bar{k} = k-1, \bar{\nu} = \nu - 1, \bar{\eta} = \eta$).

Sei nun ($\bar{i} = i-1, \bar{j} = j-1, \bar{k} = k$). Die erste Anwendung des Lemmas liefert $\bar{i} = 2\nu + \eta - 1$, $\bar{j} = k + \eta - 1$ und $\bar{k} = k$. Die zweite Anwendung führt zu $\bar{\nu} = \nu$ und $\bar{\eta} = \eta - 1$, also insgesamt zu ($\bar{k} = k, \bar{\nu} = \nu, \bar{\eta} = \eta - 1$).

Sei nunmehr ($\bar{i} = i, \bar{j} = j-1, \bar{k} = k-1$). Das Lemma liefert $\bar{i} = 2\nu + \eta$, $\bar{j} = k + \eta - 1$ und $\bar{k} = k - 1$. Die zweite Anwendung liefert $\bar{\nu} = \nu$ und $\bar{\eta} = \eta$, und insgesamt ($\bar{k} = k-1, \bar{\nu} = \nu, \bar{\eta} = \eta$). □

Um den Berechnungsquader effizient zu füllen, kann also statt der Indizierung mit (i, j, k) ebenso die Indizierung (k, ν, η) verwendet werden. Sofern keine Verwechslungsmöglichkeit besteht werden sowohl die Bezeichnung $D(i, j, k)$ als auch $D(k, \nu, \eta)$ verwendet, deren Unterschied aus der Benennung der Parameter hervorgeht.

6.3.3.3 Algorithmus zum Füllen des Berechnungsquaders

Der Algorithmus zum Füllen des Berechnungsquaders für zwei Folgen $x = x_0, \ldots, x_{N-1}$ und $y = y_0, \ldots, y_{M-1}$ und vorgegebenes $\alpha \geq 0$ ist in Abbildung 6.9 angegeben. Durch die Indizierung mit (k, ν, η) wird nur für die wirklich definierten Speicherzellen Platz reserviert.

Die Wertebereiche für k, ν und η aus Satz 6.18 spiegeln sich in den for-Schleifen wieder. Nach Satz 6.19 müssen für die Berechnung von $D(k, \nu, \eta)$ anhand der Rekursionsformel (6.7) die Werte $D(\bar{k}, \bar{\nu}, \bar{\eta})$ mit

1. $\bar{k} = k-1$, $\bar{\nu} = \nu - 1$, $\bar{\eta} = \eta$ (Zusammenfassung),
2. $\bar{k} = k$, $\bar{\nu} = \nu$, $\bar{\eta} = \eta - 1$ (Identität) und
3. $\bar{k} = k-1$, $\bar{\nu} = \nu$, $\bar{\eta} = \eta$ (Einfügung)

betrachtet werden. Diese sind nur definiert, falls $\bar{k}$, $\bar{\nu}$ und $\bar{\eta}$ innerhalb der durch $\bar{k}$, $\bar{\nu}$ und $\bar{\eta}$ definierten Wertebereiche liegen. Vor einem Zugriff auf $D(\bar{k}, \bar{\nu}, \bar{\eta})$ wird deshalb geprüft, ob die Indizes $\bar{k}$,$\bar{\nu}$,$\bar{\eta}$ innerhalb der Grenzen liegen. Die Anzahl der Kästchen ist genau die Anzahl der Schleifendurchläufe, also

$$\sum_{k=0}^{\alpha} \sum_{\nu=0}^{\min(k, \lfloor \frac{N}{2} \rfloor)} (\min(N - 2\nu, M - k) + 1)$$

Da nach Definition immer $\nu \leq k$ und $\min(N - 2\nu, M - k) \leq \min(N, M)$ gelten, läßt sich

Berechnungsquader($x[0..N-1]$, $y[0..M-1]$, α)**:**

```
//Speicherplatzreservierung
cell ***D=new **cell[α + 1];
for (k = 0; k ≤ α; k++) {
  D[k] =new *cell[maxNu(k) + 1];
  for (ν = 0; ν ≤maxNu(k); ν++) {
    D[k, ν] =new cell[maxEta(k, ν)];
  }
}

// Initialisierung
D[0, 0, 0] = 0;

// Füllen des Berechnungsquaders
for (k = 0; k ≤ α; k++) {
  for (ν = 0; ν ≤maxNu(k)); ν++) {
    for (η = 0; η ≤maxEta(k, ν); η++) {
      if(k > 0) {
        if ((ν > 0)&&(ν − 1 ≤maxNu(k − 1))&&(η ≤maxEta(k − 1, ν − 1))) {
          doubl = D(k − 1, ν − 1, η) + d(x[i(ν, η) − 1] + x[i(ν, η)], y[j(k, ν)]);
        }
        if ((ν ≤maxNu(k − 1))&& (η ≤maxEta(k − 1, ν))) {
          miss = D(k − 1, ν, η) + d(μ(i(ν, η)), y[j(k, ν)]);
        }
      }
      if (η > 0) {
        match = D(k, ν, η − 1) + d(x[i(ν, η)], y[j(k, ν)]);
      }
      D[k, ν, η] = min*(doubl,miss,match);
    }
  }
}
return D;
```

Abbildung 6.9: DP-Algorithmus zum Füllen des Berechnungsquaders. Die Bereiche der Indizes k, ν und η, speziell die Funktionen maxEta und maxNu, beziehen sich auf Satz 6.18. Die Funktionen $i(\nu, \eta)$ und $j(k, \nu)$ seien durch die Beziehungen aus Lemma 6.17 definiert.

die Anzahl der Kästchen nach oben abschätzen durch

$$\sum_{k=0}^{\alpha}\sum_{\nu=0}^{k}(\min(N,M)+1) = (\min(N,M)+1)(\frac{\alpha^2}{2}+\frac{3\alpha}{2}+1) = \mathcal{O}(\alpha^2\min(N,M)).$$

Da die Laufzeit des Algorithmus proportional zur Anzahl der Kästchen ist, sind sowohl die Platz- als auch die Zeitkomplexität $\mathcal{O}(\alpha^2\min(N,M))$.

Anhand des ausgefüllten Berechnungsquaders können nun für jeden interessierenden Abstand eine oder auch alle optimalen Transformationen zurückverfolgt und ausgegeben werden. Eine Zurückverfolgung einer optimalen Transformation benötigt eine Laufzeit proportional zur Länge der Transformation, also zu dem jeweiligen $j = k + \eta$.

6.3.3.4 Speicherung von „Abkürzungen“ im Berechnungsquader

Für eine optimale Transformation genügt es, für jede in ihr enthaltene Editieroperation den Typ (Zusammenfassung oder Einfügung) sowie die Position in der Musterfolge anzugeben. Auf einem optimalen Pfad müssen dafür nur die Zellen des Quaders betrachtet werden, in denen ein Ebenenwechsel stattfindet, da die Ebene die Anzahl der Editieroperationen angibt. Für eine Zelle (k, ν, η), die im Berechnungsgraph eine ausgehende Kante zu einer Zelle der Ebene $k-1$ besitzt, gibt das aktuelle $i = 2\nu + \eta$ die Position in der Musterfolge an. Handelt es sich um eine Einfügung, wird nach dem i-ten Folgenglied ein Element eingefügt, handelt es sich um eine Zusammenfassung, werden das $(i-1)$-te und das i-te Folgenglied zusammengefaßt.

Sofern es genügt, für jeden Abstand nur eine optimale Transformation zu berechnen, kann die Laufzeit für die Berechnung einer optimalen Transformation verkürzt werden, indem in jeder Zelle der Matrix ein Verweis auf die nächste Zelle auf dem optimalen Pfad, bei der ein Ebenenwechsel stattfindet, abgespeichert wird. Bei einer Zurückverfolgung des Berechnungsweges können dann diese Verweise als Abkürzungen im optimalen Pfad benutzt werden. Die Berechnung einer optimalen Transformation für eine Zelle in der Ebene k benötigt dann eine Laufzeit von $\theta(k)$.

Ein solcher Verweis kann während des Füllens des Berechnungsquaders leicht mitberechnet werden: Als Verweis auf eine andere Zelle genügt es, den Parameter η dieser Zelle sowie den Typ der Editieroperation zu speichern. Die Parameter ν und k gehen daraus hervor, daß aufeinanderfolgende Identitätsoperationen immer in derselben Diagonale in derselben Ebene stehen, und eine Editieroperation immer einen Ebenenwechsel bedeutet, wobei sich bei einer Einfügeoperation das ν nicht verändert, während es sich bei einer Zusammenfassung um eins verringert. Der in Abbildung 6.10 angegebene Pseudocode ist im Berechnungsquader-Algorithmus aus Abbildung 6.9 am Ende des innersten Schleifenrumpfes, also direkt nach der Zuweisung des Minimums an die Zelle, einzufügen und beschreibt die Speicherung eines Verweises.

```
if(D[k,ν,η]==match){
  D[k][ν][η].type='I';
  D[k][ν][η].η=D[k][nu][η − 1].η;
} else if(D[k,ν,η]==doubl){
  D[k][ν][η].type='Z';
  D[k][ν][η].η=η;
} else if(D[k,ν,η]==miss){
  D[k][ν][η].type='M';
  D[k][ν][η].η=η;
}
```

Abbildung 6.10: Speicherung eines Verweises auf die nächste Zelle auf dem optimalen Pfad.

Durch die einander ausschließende (if - else if)-Konstruktion wird, falls es von einer Zelle mehrere ausgehende Kanten gibt, eine Identitätsoperation ('I') gegenüber einer Zusammenfassung ('Z'), und eine Zusammenfassung gegenüber einer Einfügung ('E') vorgezogen. Diese Reihenfolge ist willkürlich; eine andere Reihenfolge oder aber eine zufällige Auswahl sind ebenso möglich. Auf diese Art wird, falls es für einen Abstand mehrere optimale Transformationen gibt, genau eine ausgewählt. In der Regel wird es relativ selten vorkommen, daß zwei verschiedene Elementaroperationen bei einer Zelle genau denselben Abstand liefern.

Die Zurückverfolgung einer Transformation anhand solcher Abkürzungen ist in Abbildung 6.11 angegeben. Der Algorithmus speichert für jede Editieroperation die Position i und den Typ in dem Feld edits, was er am Ende zurückliefert. Falls es mindestens eine Editieroperation für die Ausgangszelle gibt, d.h., wenn $k > 0$ ist, wird die Transformation zurückverfolgt. Dazu werden die drei Laufvariablen $k2$, $\nu 2$ und $\eta 2$ benötigt. Am Anfang wird, je nachdem, ob die Ausgangszelle schon eine Editieroperation enthält oder nicht, $\eta 2$ initialisiert. $\nu 2$ und $k2$ werden mit ν bzw. k initialisiert, da sich bei der ersten Editieroperation die Diagonale und die Ebene nicht ändern. In der for-Schleife durchlaufen $(k2, \nu 2, \eta 2)$ die Editieroperationszellen der optimalen Transformation. Dabei werden $k2$ in jedem Schritt und $\nu 2$ nur bei einer Zusammenfassung um eins verringert (vgl. Satz 6.19).

6.3.3.5 Einschränkungen des Berechnungsquaders

Ein vollständig ausgefüllter $(N+1) \times (M+1) \times (\alpha+1)$-Berechnungsquader beinhaltet meistens mehr Informationen, als wirklich benötigt werden: So enthält er beispielsweise in den jeweils letzten Zeilen und Spalten alle definierten k-Abstände für $0 \leq k \leq \alpha$ zwischen allen Präfixen der ersten Folge und der gesamten zweiten Folge und zwischen der gesamten ersten Folge und allen Präfixen der zweiten Folge. Ist das Ziel jedoch nur die Berechnung eines guten Abstandsbegriffs zwischen zwei Folgen, wird man in der Regel nur an allen k-Abständen der gesamten Folgen, also an allen $D(N, M, k)$ für $0 \leq k \leq \alpha$ interessiert

TransformationMitAbkürzungen(D, k, ν, η):

```
if(k > 0){
  //Initialisierung der Laufvariablen
  if(D[k][ν][η].arrow =='I'){
    η2 = D[k][ν][η].η;
  } else{
    η2 = η;
  }
  ν2 = ν;
  k2 = k;
  while(k2 > 0){
    edits[k2 − 1]=(i(ν2, η2), D[k2][ν2][η2].arrow);
    if(D[k2][ν2][η2].arrow =='Z'){
      ν2 − −;
    }
    k2 − −;
    η2 = D[k2][ν2][η2].η;
  }
  return edits;
}
return null;
```

Abbildung 6.11: Algorithmus zur Zurückverfolgung einer optimalen Transformation für die Zelle (k, ν, η) anhand des mit Abkürzungen versehenen Berechnungsquaders D. Der Algorithmus liefert ein Feld der Länge k von Editieroperationen, bzw. null bei $k = 0$ zurück.

sein.

Je nach Anwendung kann man daher den Berechnungsquader einschränken, so daß keine unnötigen Zellen berechnet werden. Für die alleinige Berechnung des Abstandes $D(N, M, \alpha)$ zum Beispiel werden nur solche (i, j, k) benötigt, für die eine optimale Transformation $\tau_1 \in \mathcal{T}_{i,j,k}$ ein Präfix einer optimalen Transformation $\tau \in \mathcal{T}_{N,M,\alpha}$ sein kann. Der entsprechende Suffix $\tau_2 \in \mathcal{T}_{N-i,M-j,\alpha-k}$ muß dann ebenfalls eine Transformation sein, was nach Satz 6.12 genau dann der Fall ist, wenn

1. $M - j - N + i = \alpha - k - 2\bar{\nu}$ für ein $\bar{\nu} \in \{0, \ldots, \alpha - k\}$
2. und $M - j \geq \alpha - k$

gelten. Umgeformt in (k, ν, η)-Schreibweise ergibt dies die Bedingungen

1. $\frac{M-N-\alpha+2k}{2} \leq \nu \leq \frac{\alpha+N-M}{2}$ und
2. $\eta \leq M - \alpha$.

Sollen alle $D(N, M, \alpha')$ für $0 \leq \alpha' \leq \alpha$ berechnet werden, müssen diese Bedingungen für mindestens ein α' erfüllt sein, was insgesamt die Bedingungen

1. $\frac{M-N-\alpha+2k}{2} \leq \nu \leq \frac{\alpha+N-M}{2}$ und
2. $\eta \leq M$

ergibt.

Solche zusätzlichen Bedingungen schränken die Wertebereiche der Variablen k, ν oder η ein. Diese Einschränkungen können sich jedoch nur auf den jeweils minimalen und maximalen Wert der Variable beziehen, da ansonsten eine Berechnung anhand der Rekursionsgleichung nicht mehr möglich wäre. Deshalb können beim Füllen des Berechnungsquaders problemlos möglicherweise kleinere Wertebereiche für die Variablen sowohl bei der Speicherplatzreservierung als auch bei dem Durchgehen der for-Schleifen, verwendet werden. Die Laufzeit und der verwendete Speicherplatz verringern sich dann natürlich entsprechend der kleineren Wertebereiche.

Kapitel 7

Ein Crossdatingalgorithmus basierend auf k-Abständen

In diesem Kapitel wird ein neuer Crossdatingalgorithmus vorgestellt, der eine Musterjahrringfolge, die fehlende und doppelte Ringe enthalten kann, in einer von Irregularitäten freien Referenzfolge lokalisiert. Der Algorithmus verwendet den k-Transformationsabstand aus Kapitel 6.

7.1 Grundlegende Idee

Seien $a = a_0, \ldots, a_{m-1}$ die Referenzjahrringfolge, $b = b_0, \ldots, b_{n-1}$ die in a zu lokalisierende Musterjahrringfolge, $minOvl$ die minimale Überlappungslänge und α eine obere Schranke für die Anzahl zugelassener Editieroperationen.

Der k-Transformationsabstand kann als Abstandsbegriff für das Crossdating herangezogen werden, indem an allen Überlappungspositionen von b in a alle möglichen k-Transformationsabstände zwischen b und zusammenhängenden Teilfolgen in a mit bis zu α Editieroperationen berechnet werden. Um die Abstände sinnvoll untereinander vergleichen zu können, muß danach jeder Abstand durch die Länge $j = k + \eta$ der entsprechend optimal transformierten Musterfolge geteilt werden (*Längennormierung*).

Anhand der Abstände allein kann jedoch noch keine Datierungsentscheidung getroffen werden. Zwar geht aus der Position des Abstandes in einem Berechnungsquader die Anzahl der Einfügungen und Zusammenfassungen in jeder optimalen Transformation hervor (vgl. Satz 6.16), jedoch kommt es oft vor, daß zur objektiven Minimierung des Abstandes subjektiv gesehen unnötige Editieroperationen nötig sind. Zum Beispiel folgt oft kurz nach einer Einfügung eine Zusammenfassung, oder umgekehrt, oder eine Editieroperation steht ganz am Anfang oder ganz am Ende der Folge. Zur Bewertung der Abstände müssen

also die optimalen Transformationen vorliegen, so daß subjektiv über die Notwendigkeit der Editieroperationen entschieden werden kann.

Zur besseren Auswertung sollten alle Abstände sortiert werden, so daß sie dann aufsteigend zusammen mit den entsprechenden Datierungen und den Stellen der Einfügungen und Zusammenfassungen in den optimalen Transformationen für b ausgegeben werden können.

7.2 Berechnung aller Folge-Präfix- und Präfix-Folge-Abstände in einem Berechnungsquader

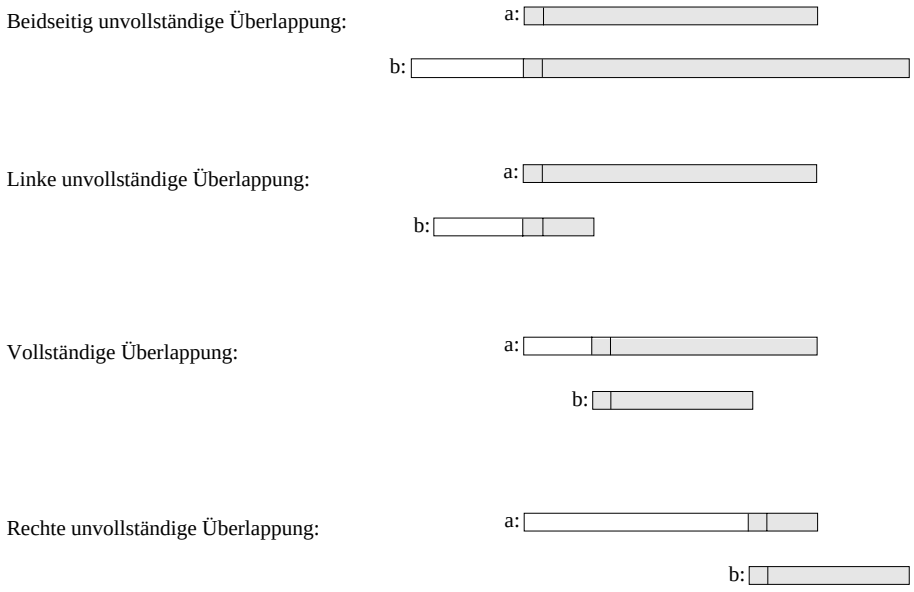

Abbildung 7.1: Konstellationen, die beim Anlegen von b in a auftreten können. Für die Berechnung aller Transformationsabstände, die b auf dasselbe Anfangsjahr datieren, müssen nur die grau eingezeichneten Suffixe betrachtet werden.

Für die Berechnung aller möglichen Transformationsabstände an allen Überlappungspositionen von b in a können alle Abstände, die b auf dasselbe Anfangsjahr datieren, zusammengefaßt in einem Berechnungsquader berechnet werden. Dafür wird anschaulich gesehen die Folge b an einer Position in a, die das Anfangsjahr bestimmt, angelegt, und nur die Suffixe beider Folgen ab der ersten gemeinsamen Überlappungsposition betrachtet. Die vier Fälle, die dabei auftreten können, zeigt Abbildung 7.1. Bei einer linken sowie bei einer beidseitig unvollständigen Überlappung von b in a muß nur der Suffix x von b betrachtet werden, der mit a überlappt, und mit $a =: y$ verglichen werden. Bei einer vollständigen sowie einer rechten unvollständigen Überlappung von b in a muß die ganze Folge $b =: x$ mit dem durch das Anfangsjahr bestimmten Suffix y von a verglichen werden.

Für die so definierten Teilfolgen $x = x_0, \ldots, x_{N-1}$ von b und $y = y_0, \ldots, y_{M-1}$ von a, die in Abbildung 7.1 grau eingezeichnet sind, sind die k-Abstände zwischen x und allen Präfixen von y (*Folge-Präfix-Abstände*) sowie zwischen allen Präfixen von x und der gesamten Folge y (*Präfix-Folge-Abstände*), sofern sie existieren, von Interesse. Die *Folge-Präfix-Abstände* entsprechen allen definierten $D(N, j, k)$ für $0 \leq j \leq M$ und $0 \leq k \leq \alpha$ und die *Präfix-Folge-Abstände* allen definierten $D(i, M, k)$ für $0 \leq i \leq N$ und $0 \leq k \leq \alpha$.

Da ein k-Abstand mit $0 \leq k \leq \alpha$ nur existieren kann, wenn die Längendifferenz der zu vergleichenden Folgen α nicht überschreitet, seien im folgenden N_e und M_e definiert durch

$$\begin{aligned} N_e = N \quad \text{und} \quad M_e = N + \alpha \quad &, \text{falls} \quad M > N + \alpha \\ N_e = M + \alpha \quad \text{und} \quad M_e = M \quad &, \text{falls} \quad M < N - \alpha \\ N_e = N \quad \text{und} \quad M_e = M \quad &, \text{falls} \quad |M - N| \leq \alpha \; . \end{aligned}$$

Bei der Berechnung eines k-Abstandes mit Hilfe der dynamischen Programmierung anhand der Rekursionsgleichung (6.7) wird auf die k- und $(k-1)$-Abstände von Präfixen zurückgegriffen, die im Berechnungsquader bereits gespeichert wurden. Deshalb können alle Folge-Präfix- und Präfix-Folge-Abstände in einem Berechnungsquader berechnet werden, indem der Berechnungsquader bishin zu den größten benötigten Indizes, also mit allen $D(i, j, k)$ gefüllt wird, die den folgenden Bedingungen genügen:

1. $0 \leq k \leq \alpha$
2. $0 \leq i \leq N_e$
3. $k \leq j \leq M_e$
4. $j - i = k - 2\nu$ für ein $\nu \in \{0 \ldots k\}$

Da die Länge einer sinnvoll vergleichbaren Folge vom Benutzer mit *minOvl* angegeben ist, sollte die transformierte Musterfolge ebenfalls eine Mindestlänge von *minOvl* haben. Um dies zu garantieren, muß für jede mögliche Transformation $\tau \in \mathcal{T}_{i,j,k}$ die Ungleichung $N_e - z_\tau + e_\tau \geq minOvl$ gelten. Unter Berücksichtigung von Lemma 6.17 ergibt dies eine Einschränkung des Berechnungsquaders mit der zusätzlichen Bedingung

$$\nu \leq \frac{N_e - minOvl + k}{2} \tag{7.1}$$

Die zu berechnenden Zellen müssen also bei einer (k, ν, η)-Indizierung folgenden Bedingungen genügen:

1. $0 \leq k \leq \alpha$
2. $0 \leq \nu \leq \min(k, \frac{N_e}{2}, \frac{N_e - minOvl + k}{2}) =: \text{maxNu}(k)$
3. $0 \leq \eta \leq \min(N_e - 2\nu, M_e - k) =: \text{maxEta}(k, \nu)$

Ein solcher Berechnungsquader kann nun mit dem Algorithmus aus Abschnitt 6.3.3.3, unter der Berücksichtigung der einschränkenderen Bedingung für ν, gefüllt werden, was Zeit- und Platz $\mathcal{O}(\alpha^2 \min(M_e, N_e))$ benötigt.

7.3 Sukkzessive Berechnung aller Ergebnisabstände inklusive je einer optimalen Transformation

Die Berechnung aller möglichen Transformationsabstände an allen Überlappungspositionen von b in a erfolgt sukkzessiv für alle möglichen Datierungen von b in a. Dafür wird wie bei dem sukkzessiven Algorithmus aus Abschnitt 3.2 die Folge b an allen Positionen in a angelegt und für jede Position der entsprechende Präfixquader (vgl. Abschnitt 7.2) berechnet. Da der Benutzer zumindest für die insgesamt besten Abstände die optimalen Transformationen wissen möchte, könnten alle Präfixquader abgespeichert werden, so daß eine Zurückverfolgung der Berechnung jederzeit für jeden Abstand möglich wäre. Ein solcher Algorithmus würde Zeit und Platz

$$\mathcal{O}(\sum_{l=minOvl-n}^{-1} \alpha^2 \min(n+l, m) + \sum_{l=0}^{m-minOvl} \alpha^2 \min(n, m-l))$$

benötigen. Die Summen ergeben ausgerechnet $\alpha^2(mn - minOvl^2 + minOvl)$, und damit eine Laufzeit und einen Platzverbrauch von $\mathcal{O}(\alpha^2(mn - minOvl^2 + minOvl)) = \mathcal{O}(\alpha^2 mn)$.

Zumindest der Platzverbrauch kann jedoch reduziert werden, indem die Transformationen vorher berechnet und nicht jeder Präfixquader abgespeichert wird. Dazu wird direkt nach dem Füllen eines Berechnungsquaders für jeden Ergebnisabstand *eine* optimale Transformation zurückverfolgt, deren Editieroperationen in einem Feld der Länge k gespeichert werden, wobei k die Anzahl der Editieroperationen ist. Jeder Abstand wird dann zusammen mit dem Feld der Editieroperationen der optimalen Transformation in einem zusätzlichen Ergebnisfeld gespeichert. Sind für alle Abstände die Editieroperationen berechnet und gespeichert, wird der Präfixquader gelöscht.

Die Einschränkung auf nur eine optimale Transformation pro Abstand muß natürlich nicht getroffen werden. Es könnten auch alle optimalen Transformationen für einen Abstand berechnet werden, so daß statt einem Feld der Länge k eine Liste von Feldern der

```
Ergebnisse(D, result, resultIndex,l):
  for(k = 0; k <= alpha; k++){
    for(ν = 0; ν <= maxNu(k); ν++){
      η = maxEta(k, ν);
      //Speichere Ergebnis in result
      result[rIndex].editOps= TransformationMitAbkürzungen(D, k, ν, η);
      result[rIndex].value=D[k][ν][η]/(k + η);
      result[rIndex].l=l;
      rIndex++;
    }
  }

Ergebnisfeld(a[0..m − 1], b[0..n − 1], minOvl, α):
  resultIndex=0;
  //Linke und beidseitig unvollständige Überlappungen
  for (l=minOvl-n; l < 0; l++){
    D =Berechnungsquader(b[−l..n − 1], a[0..m − 1], minOvl, α);
    Ergebnisse(D,result,resultIndex,l);
  }

  //Vollständige und rechte unvollständige Überlappungen
  for (l = 0; l ≤ m − minOvl; l++){
    D =Berechnungsquader(b[0..n − 1], a[l..m − 1], minOvl, α);
    Ergebnisse(D,result,resultIndex,l);
  }
  return result;
```

Abbildung 7.2: Algorithmus zur Berechnung aller Präfixquader an allen Überlappungspositionen einer Referenzfolge a mit einer Musterfolge b.

Länge k abgespeichert werden müßte. Daß bei der Berechnung einer Zelle mindestens zwei der drei verschiedenen Berechnungsarten genau denselben Abstand liefern, und deshalb mehr als eine optimale Transformation für einen Abstand existiert, ist jedoch eher unwahrscheinlich. Falls der Fall doch auftritt, wird genau eine optimale Transformation ausgewählt.

Da für jeden Ergebnisabstand nur eine optimale Transformation zurückverfolgt wird, können wie in Unterabschnitt 6.3.3.4 beschrieben, beim Füllen des Präfixquaders Abkürzungen im optimalen Pfad gespeichert werden, die die Berechnung einer optimalen Transformation in Laufzeit $\theta(k)$ ermöglichen.

Der hier beschriebene Algorithmus ist als Pseudocode in Abbildung 7.2 angegeben. Die sukkzessive Berechnung aller Präfixquader erfolgt in der Prozedur Ergebnisfeld. Beim Füllen eines Berechnungsquaders sollen dabei die Abkürzungen anhand des Algorithmus aus Abschnitt 6.3.3.4 mitberechnet werden. Für jeden gefüllten Quader wird die Prozedur Ergebnisse aufgerufen, in der für alle Ergebnisabstände des Quaders, also alle $D(k, \nu, maxEta(k, \nu))$, ein neues Element des Ergebnisfeldes result gefüllt wird. Ein Element des Ergebnisfeldes enthält den bereits längennormierten Abstand (value), das Feld der Editieroperationen der optimalen Transformation (edits) und den Index l aus dem sukkzessiven Algorithmus, der die Datierung angibt. Die Längennormierung erfolgt dabei durch eine Division durch $k + maxEta(k, \nu)$. Die optimale Transformation wird mit dem Algorithmus TransformationMitAbkürzungen aus Abschnitt 6.3.3.4 berechnet. Für jeden Ergebnisabstand wird ein weiteres Feld in result belegt und der Index resultIndex, der am Ende hinter das letzte belegte Feld zeigt, um eins inkrementiert.

Da jede Ebene k eines Berechnungsquaders höchstens $k + 1$ Diagonalen hat und $m + n - 2minOvl + 1$ Berechnungsquader gefüllt werden, gibt es höchstens

$$(m + n - 2minOvl + 1) \sum_{k=0}^{\alpha} (k + 1) = (m + n - 2minOvl + 1) \frac{\alpha^2 + 3\alpha + 2}{2}$$

Ergebnisabstände, die in result gespeichert werden müssen. Da die Speicherung einer optimalen Transformation $\theta(k)$ an Speicherplatz und Laufzeit kostet, muß für das Speichern einer Zelle (k, ν, η) im result-Feld $\theta(k + 1)$ Speicherplatz und auch Laufzeit verwendet werden. Für alle Ergebnisabstände ergibt dies insgesamt eine Zeit- und Platzkomplexität von

$$\mathcal{O}(\,(m + n - 2minOvl + 1)(1 + \sum_{k=1}^{\alpha} (k + 1)k)\,) = \mathcal{O}((m + n)\alpha^3) \quad .$$

Die Berechnung aller Ergebnisabstände mit dem Ergebnisfeld-Algorithmus hat unter diesen Bedingungen eine Laufzeit von $\mathcal{O}(\alpha^2 mn + (m + n)\alpha^3) = \mathcal{O}(\alpha^2 mn)$ und eine Platzkomplexität von $\mathcal{O}((m + n)\alpha^3 + \alpha \min(n, m)^2)$.

7.4 Crossdatingalgorithmus

Die Jahrringfolgen enthalten, wie in den Kapiteln 1 und 2 erwähnt wurde, in der Regel Langzeittrends. Eine Berechnung der Transformationsabstände ohne vorherige Standardisierung der Folgen (vgl. Unterkapitel 2.4) wird deshalb selten zu befriedigenden Ergebnissen führen. Eine Standardisierung der Folgen ist also notwendig. Der komplette Crossdatingalgorithmus verläuft daher in drei Schritten:

1. Standardisierung beider Folgen
2. Berechnung aller Ergebnisabstände mit Ergebnisfeld
3. Sortieren des Ergebnisfeldes mit anschließender Ausgabe

Die Standardisierung beider Folgen kann in $\theta(n+m)$ Zeit und Platz erfolgen. Schritt 2 hat eine Laufzeit von $\mathcal{O}(\alpha^2 mn)$ und benötigt $\mathcal{O}((m+n)\alpha^3+\alpha\min(n,m)^2)$ Platz. Das Sortieren des Ergebnisfeldes dauert mit einem geeigneten Algorithmus $\mathcal{O}(((m+n)\alpha^2)\log((m+n)\alpha^2))$, und die Ausgabe aller Abstände $\mathcal{O}((m+n)\alpha^3+\alpha\min(n,m)^2)$. Die Laufzeit insgesamt ist deshalb $\mathcal{O}(\alpha^2 mn)$ und der Platzverbrauch $\mathcal{O}((m+n)\alpha^3+\alpha\min(n,m)^2)$.

Durch die Standardisierung der Folgen im voraus entsprechen die Kosten der Editieroperationen nicht den ursprünglich beabsichtigten Kosten. So wird bei einer Zusammenfassung statt des standardisierten Wertes der Summe zweier Jahrringbreiten die Summe der standardisierten Werte verwendet und bei einer Einfügung statt des standardisierten Wertes des arithmetischen Mittels der umliegenden Werte das arithmetische Mittel der standardisierten Werte. Van Deusen [11] umgeht dies, indem er während des Füllens der Berechnungsmatrix implizit eine Standardisierung der Daten mitberechnet. Dies ist jedoch nur für spezielle Standardisierungen möglich, die auf Daten *vor* dem zu standardisierenden Wert zugreifen, wie etwa die Differenz von Logarithmen oder eine einfache Logarithmenstandardisierung.

7.5 Das Crossdatingprogramm dpcross

Der Crossdatingalgorithmus wurde in dem auf Diskette beigefügten sowie in Anhang C abgedruckten Programm *dpcross* in der Programmiersprache $C++$ implementiert. Zur graphischen Auswertung der Ergebnisse wird das Programm *gnuplot* verwendet.

Als Standardisierungsverfahren stehen einerseits die Standardisierung anhand eines ungewichteten gleitenden Mittels sowie die Berechnung der elementweisen Differenz von Logarithmen zur Verfügung. Es können jedoch auch bereits standardisierte Folgen als Eingabe verwendet werden. Die Wahl der Standardisierung sowie die Einstellung von Parametern wie α oder *minOvl* erfolgt über eine Konfigurationsdatei, die in Anhang A näher erläutert wird.

```
11:43 wenk@trick% dpcross ../data/archinst/kieftestmw56.2cl ../data/archinst/kieftest56.2cl
Vorverarbeitung: floatave; halfwidth=2, dolog=true
alpha=1, minOvl=50, nRaw=114, mRaw=151
factor=100, w_E=1, w_I=1, w_Z=1, resultNum= 10

Anzahl der Ergebnisse: 497
Bekannte Datierung: 1881
Jahr: 1881-1993   Kosten: 4.29988 Laenge: 113; Z 1992;
Jahr: 1880-1994   Kosten: 4.3121 Laenge: 115; E 1880;
Jahr: 1882-1994   Kosten: 4.34169 Laenge: 113; Z 1883;
Jahr: 1881-1994   Kosten: 4.34674 Laenge: 114;
Jahr: 1881-1995   Kosten: 4.61073 Laenge: 115; E 1994;
Jahr: 1794-1906   Kosten: 5.08527 Laenge: 62; Z 1896;
Jahr: 1795-1907   Kosten: 5.41404 Laenge: 63; Z 1846;
Jahr: 1794-1907   Kosten: 5.4479 Laenge: 63;
Jahr: 1793-1907   Kosten: 5.49892 Laenge: 63; E 1845;
Jahr: 1794-1908   Kosten: 5.56953 Laenge: 64; E 1907;
Anzeige mit anderer resultNum (j/n)? n

Gnuplot aller aufgefuehrten Matchings (j/n)? j

<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
<return> für den nächsten Graph
Nochmaliges Gnuplot (j/n)? n

Fertig (j/n)? (Wenn nein, dann nochmaliges Anzeigen der Daten.) j

Lösche Verzeichnis work mit rm -f work/*

11:44 wenk@trick% ▉                                              ~/dpcross/program
```

Abbildung 7.3: Beispiel für einen Programmablauf. Hinter „Kosten:“ steht der jeweilige längennormierte Abstand. „Laenge“ gibt die Länge der transformierten Folge an. Die Daten sind eine Jahrringbreitenfolge und eine Referenzfolge von Kiefern aus dem Berliner Umland. (Daten aus dem Deutschen Archäologischen Insitut, Eurasien-Abteilung) Die graphische Veranschaulichung des ersten Matching-Resultats ist in Abbildung 7.4 und die des sechsten Resultats in Abbildung 7.5 abgebildet.

Als eine einfache Verallgemeinerung des Abstandsbegriffs wird in dem Programm der bezüglich der Editieroperationen *gewichtete Transformationsabstand*

$$D(i,j,k) = \begin{cases} 0 & \text{, wenn i=j=k=0} \\ \min\limits_{\tau \in \mathcal{T}_{i,j,k}} \sum\limits_{v=0}^{j-1} \mathsf{w\_}\tau_v d(\tau(x)_v, y_v) & \text{, wenn } \mathcal{T}_{i,j,k} \neq \emptyset \\ \text{nicht definiert} & \text{, sonst} \end{cases} \tag{7.2}$$

berechnet. Dabei ist für jeden Elementaroperationstyp ein Gewicht definiert, und zwar

1. w_E für eine Einfügung,
2. w_I für eine Identitätsoperation und
3. w_Z für eine Zusammenfassung.

In der Rekursionsformel wird der gewichtete Transformationsabstand wie folgt berech-

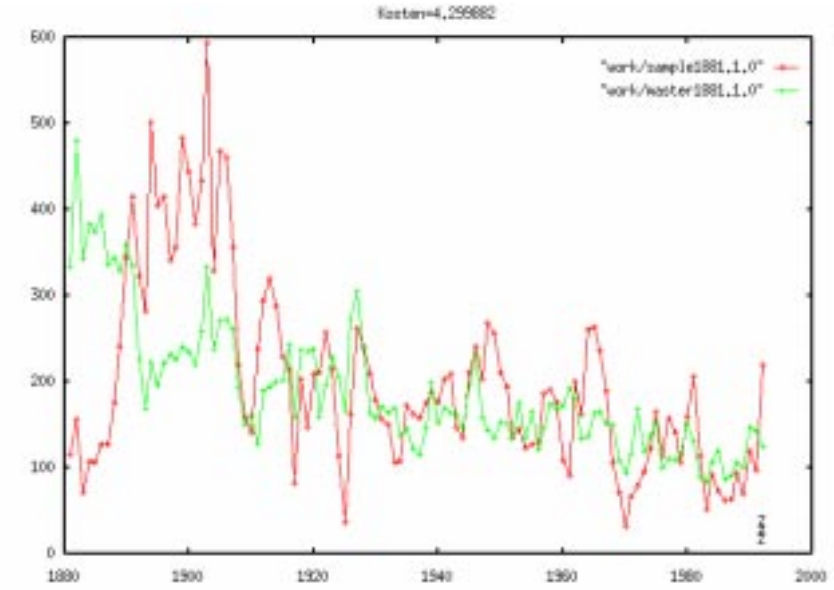

Abbildung 7.4: Graphische Darstellung der ersten Datierung aus dem Beispiel-Programmablauf aus Abbildung 7.3 mit Hilfe des Programms *gnuplot*.

net:

$$D(i,j,k) \quad = \quad \overset{\star}{\min} \left\{ \begin{array}{l} D(i-2,j-1,k-1) + \mathrm{w\_}Z \cdot d(x_{i-2} + x_{i-1}, y_{j-1}), \\ D(i-1,j-1,k) + \mathrm{w\_}I \cdot d(x_{i-1}, y_{j-1}), \\ D(i,j-1,k-1) + \mathrm{w\_}E \cdot d(\mu(i-1), y_{j-1}) \end{array} \right\} \tag{7.3}$$

Für w_E=w_Z=w_I=1 entspricht dies dem in Unterkapitel 6.3 definierten k-Abstand. Mit den Gewichtungsfaktoren kann die Auswahl der Elementaroperationen in gewisser Weise beeinflußt werden. Hohe Faktoren für Editieroperationen und ein niedriger Faktor für eine Identitätsoperation müßten den Algorithmus zu weniger Editierungen veranlassen. Die Wahl solcher Gewichtungen ist jedoch subjektiv. Unabhängig von den Gewichtungen wird zur Längennormierung jeder Abstand durch die Länge der optimalen Transformation, also durch die Anzahl der Elementaroperationen, geteilt.

Ein Beispiel für einen Programmablauf wird in den Abbildungen 7.3, 7.4 und 7.5 gezeigt. Als Musterfolge wurde eine bereits datierte Jahrringbreitenfolge einer Kiefer aus dem Berliner Umland und als Referenzfolge eine aus Kiefern aus demselben Umkreis gewonnene Mittelwertfolge verwendet (Daten aus dem Deutschen Archäologischen Institut, Eurasien-Abteilung) [1]. Der Parameter α wurde auf 1 und $minOvl$ auf 50 gesetzt. Die ersten fünf Datierungsvorschläge datieren die Folge, bis auf subjektiv gesehen unnötige Editieroperationen, auf die korrekte Datierung 1881. Beim ersten und beim fünften Er-

[1] Weitere Daten sind bei der International Tree-Ring Data Bank (ITRDB) unter http://www.ngdc.noaa.gov/paleo/treering.html erhältlich.

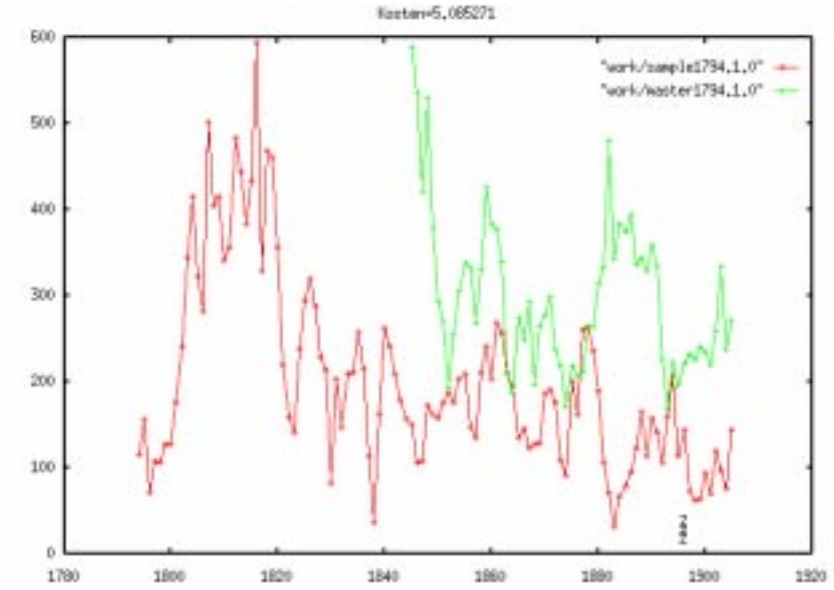

Abbildung 7.5: Graphische Darstellung der sechsten Datierung aus dem Beispiel-Programmablauf aus Abbildung 7.3 mit Hilfe des Programms *gnuplot*.

gebnis schlägt das Programm eine Editieroperation ganz am Ende der Folge vor, während es beim zweiten und dritten Ergebnis ganz am Anfang der Folge editiert. Die Datierung des dritten Ergebnisses auf 1882 entspricht durch die Zusammenfassung im Jahr 1883 eigentlich einer Datierung auf 1881. Eine Datierung auf das Jahr 1881 ohne Editierungen wird im vierten Ergebnis vorgeschlagen. Die restlichen Datierungsvorschläge fallen durch die relativ kurze Länge der transformierten Folgen auf. In Abbildung 7.4 ist das erste Ergebnis mit Hilfe des Programms *gnuplot* graphisch dargestellt, indem die transformierte Musterfolge sowie das entsprechende Referenzfolgenstück als Kurven aufgetragen und die Stellen der Einfügungen und Zusammenfassungen markiert werden. Zum Vergleich zeigt Abbildung 7.5 die graphische Darstellung des sechsten Ergebnisses.

Beispiele für das Verhalten des Programms bei künstlich gelöschten bzw. verdoppelten Ringen zeigen die Abbildungen 7.6 und 7.7. Die Musterfolge aus dem ersten Beispiel wurde dafür künstlich verändert. In Abbildung 7.6 wurde im Jahr 1925 ein im Verhältnis zu den umliegenden Werten auffallend niedriger Wert gelöscht. Die Datierungsvorschläge sind nun nicht mehr so einheitlich wie im ersten Beispiel, jedoch ist der erste Vorschlag bis auf die genaue Position der Einfügung korrekt. Für das Beispiel in Abbildung 7.7 wurde ein relativ hoher Wert im Jahr 1913 durch zwei zu diesem Wert aufsummierende Werte ersetzt. Wie bei der künstlichen Löschung eines Elements sind auch hier die Datierungsvorschläge uneinheitlich, jedoch ist wiederum der erste Datierungsvorschlag korrekt.

```
11:45 wenk@trick% dpcross ../data/archinst/kieftestmw56.2cl ../data/archinst/kieftest56E.2cl
Vorverarbeitung: floatave; halfwidth=2, dolog=true
alpha=1, minOvl=50, nRaw=113, mRaw=151
factor=100, w_E=1, w_I=1, w_Z=1, resultNum= 10

Anzahl der Ergebnisse: 494
Bekannte Datierung: 1881
Jahr: 1881-1994   Kosten: 3.25657 Laenge: 114; E 1919;
Jahr: 1909-1995   Kosten: 3.62479 Laenge: 87; Z 1931;
Jahr: 1941-1995   Kosten: 4.10106 Laenge: 55; Z 1951;
Jahr: 1926-1995   Kosten: 4.13678 Laenge: 70; E 1941;
Jahr: 1922-1995   Kosten: 4.16679 Laenge: 74; Z 1934;
Jahr: 1928-1995   Kosten: 4.19445 Laenge: 68; Z 1964;
Jahr: 1883-1994   Kosten: 4.40824 Laenge: 112; Z 1884;
Jahr: 1940-1995   Kosten: 4.54917 Laenge: 56; Z 1976;
Jahr: 1907-1995   Kosten: 4.55028 Laenge: 89; E 1922;
Jahr: 1940-1995   Kosten: 4.60933 Laenge: 56;
Anzeige mit anderer resultNum (j/n)? 
```

Abbildung 7.6: Datierungsergebnisse für dieselben Folgen aus Abbildung 7.3, wobei in der Musterfolge der Wert im Jahr 1925 gelöscht wurde.

```
11:46 wenk@trick% dpcross ../data/archinst/kieftestmw56.2cl ../data/archinst/kieftest56Z.2cl

Vorverarbeitung: floatave; halfwidth=2, dolog=true
alpha=1, minOvl=50, nRaw=115, mRaw=151
factor=100, w_E=1, w_I=1, w_Z=1, resultNum= 10

Anzahl der Ergebnisse: 500
Bekannte Datierung: 1881
Jahr: 1881-1994   Kosten: 4.61552 Laenge: 114; Z 1914;
Jahr: 1793-1906   Kosten: 5.08527 Laenge: 62; Z 1896;
Jahr: 1794-1907   Kosten: 5.41404 Laenge: 63; Z 1846;
Jahr: 1793-1907   Kosten: 5.4479 Laenge: 63;
Jahr: 1792-1907   Kosten: 5.49892 Laenge: 63; E 1845;
Jahr: 1793-1908   Kosten: 5.56953 Laenge: 64; E 1907;
Jahr: 1783-1898   Kosten: 5.57977 Laenge: 54; E 1870;
Jahr: 1788-1903   Kosten: 5.69587 Laenge: 59; E 1871;
Jahr: 1790-1903   Kosten: 5.7439 Laenge: 59; Z 1893;
Jahr: 1789-1904   Kosten: 5.94687 Laenge: 60; E 1872;
Anzeige mit anderer resultNum (j/n)? 
```

Abbildung 7.7: Datierungsergebnisse für dieselben Folgen aus Abbildung 7.3, wobei in der Musterfolge der Wert im Jahr 1913 durch zwei auf diesen Wert aufsummierende Werte ersetzt wurde.

Kapitel 8

Ausblick

Das Programm dpcross ist ein aus dendrochronologischer Sichtweise ungewohntes Crossdatingprogramm, da es in eine Musterjahrringfolge oft mehrere fehlende und doppelte Ringe hineininterpretiert. Sofern bei den Matchingergebnissen zwischen subjektiv unnötigen und plausibel erscheinenden Editieroperationen unterschieden wird, erscheinen die Datierungsvorschläge von dpcross durchaus verwendbar. Dies sollte jedoch von Dendrochronologen in Tests untersucht werden.

Da in dpcross davon ausgegangen wird, daß fehlende und doppelte Ringe nur in der Muster- und nicht in der Referenzfolge auftreten können, ist es in der Praxis hauptsächlich für archäologische Zwecke einsetzbar, bei denen bereits eine qualitativ hochwertige Referenzfolge vorliegt. Die Konstruktion einer Referenzfolge aus mehreren Jahringfolgen ist mit dpcross nur begrenzt möglich, da in diesem Fall auch die Referenzfolge möglicherweise Unregelmäßigkeiten enthalten kann. Als Alternative kann wie bisher nach einer Referenzfolgenbildung ein Qualitätskontrollprogramms wie Cofecha eingesetzt werden.

Die bisherige Implementation von dpcross ist auf Jahrringbreitenfolgen beschränkt. Eine Anwendung auf andere Jahrringcharakteristika ist jedoch, sofern diese jahrringweise vorliegen, durch einfache Umgestaltung der Zusammenfassungs- und Einfügungsoperationen möglich.

Möglicher Forschungsbedarf besteht für die gleichzeitige Datierung einer Menge von Jahrringfolgen eines Standorts untereinander mit impliziter Konstruktion einer Referenzfolge aus diesen Folgen. Bisher werden die Folgen sukkzessiv datiert und zu einer Referenzfolge zusammengefaßt, und anschließend wird in der Regel ein Qualitätskontrollprogramm angewendet.

Anhang A

Programmkonfiguration

Das Programm **dpcross** erwartet als Kommandozeilenparameter die Namen der Referenzfolgen- sowie der Musterfolgendatei; der Aufruf erfolgt also in der Form dpcross <referenzdatei> <musterdatei>. Optional können als weitere Kommandozeilenparameter α und die Anzahl resultNum der auszugebenden besten Ergebnisabstände angegeben werden. Die Referenzfolgendatei muß im Zwei-Spalten-Format vorliegen, wobei die erste Spalte den Jahreszahlen und die zweite Spalte den jeweiligen Jahrringbreiten entspricht. Die Musterfolgendatei kann entweder im Ein-Spalten-Format, in dem in jeder Zeile eine Jahrringbreite steht, oder im Zwei-Spalten-Format sein. Das Zwei-Spalten-Format dient zur Bestätigung oder Überprüfung bereits bestehender Datierungen des Musters, daher wird der Inhalt der ersten Spalte, der Aufschluß über eine mögliche Datierung geben könnte, ignoriert.

Zur weitergehenden Konfiguration des Programms steht die Konfigurationsdatei **dpcross.cfg** zur Verfügung. Ein Beispiel einer solchen Datei zeigt Abbildung A.1, in der die Standardwerte für die Parameter angegeben sind. Eine Zeile wird als Kommentarzeile aufgefaßt, wenn am Anfang der Zeile zwei '/' stehen. Ein Parameter wird gesetzt, indem der Parametername gefolgt von einem '=' und dem zu setzendem Wert in eine Zeile geschrieben wird, wobei keine Leerzeichen dazwischen stehen dürfen. Wird ein bestimmter Parameter nicht in der Konfigurationsdatei gesetzt, wird ein Standardwert angenommen.

Der Parameter preprocess gibt an, ob die Daten vorverarbeitet, also standardisiert werden sollen. Dabei stehen eine Standardisierung anhand eines gleitenden Mittels (floatave), wie in Abschnitt 2.4.3 angegeben, oder anhand der Differenz von Logarithmen (logdiff) zur Verfügung. Bei einer anderen Angabe wird keine Standardisierung durchgeführt. Im Falle der Standardisierung anhand eines gleitenden Mittels stehen die weiteren Parameter halfwidth und dolog zur Verfügung. halfwidth gibt die abgerundete halbe Anzahl der zu mittelnden Jahre an, also zum Beispiel muß bei 5 zu mittelnden Jahren halfwidth=2 gesetzt werden. Der boolesche Parameter dolog, der die Werte true oder false annehmen kann, gibt an, ob der natürliche Logarithmus der transformierten Werte als endgültige Transfor-

```
//Vorverarbeitung mit gleitendem Mittel (floatave) oder Differenz von
//Logarithmen (logdiff)?
preprocess=floatave
//floatave: Halbe Anzahl (abgerundet) der zu mittelnden Jahre:
halfwidth=2
//floatave: Logarithmus:
dolog=true
//logdiff: Faktor:
logmult=5
//Minimale "Uberlappung:
minOvl=50
//Alpha:
alpha=1
//Anzahl der auszugebenden Ergebnisse
resultNum=10
//Multiplikation jedes Abstandes mit factor
factor=100
//Multiplikation einer Einf"ugeoperation mit w_E
w_E=1
//Multiplikation einer Identit"atsoperation mit w_I
w_I=1
//Multiplikation einer Zusammenfassungsoperation mit w_Z
w_Z=1
//Aufruf zum L"oschen des Verzeichnisses "work"
//delDir=null , wenn nicht gel"oscht werden soll
delDir=rm -f work/*
```

Abbildung A.1: Konfigurationsdatei für das Programm **dpcross**. Kommentare werden durch zwei '/' am Anfang der Zeile markiert. Parameter werden durch einen Ausdruck der Form <parametername>=<parameterwert> in einer Zeile gesetzt.

mation betrachtet werden soll oder nicht. Bei der Standardisierung mittels der Differenz von Logarithmen berechnet sich die transformierte Folge y aus der Ursprungsfolge x durch die Formel $y_i = \log(\mathsf{logmult} * x_i) - \log(\mathsf{logmult} * x_{i+1})$.

Der Parameter minOvl legt die minimale Überlappungslänge und alpha die maximale Anzahl von Editieroperationen fest. Beide Parameter müssen mindestens gleich dem Minimum der Längen der zwei Eingabefolgen sein und werden gegebenenfalls auf diesen Wert gesetzt. resultNum gibt an, wieviele Ergebnisabstände ausgegeben werden sollen. Da das Feld der Ergebnisabstände sortiert wird, werden die resultNum-kleinsten Abstände ausgegeben. Mit dem Parameter factor kann man einen Faktor angegeben, mit dem jeder Ergebnisabstand multipliziert werden soll. Dies dient ausschließlich der besseren Übersicht.

Nach dem Programmablauf werden die Dateien im Arbeitsverzeichnis „work“ gelöscht. Der entsprechende Aufruf kann in der Konfigurationsdatei in dem Parameter delDir angegeben werden. Wird dieser Parameter nicht oder mit dem Wert null angegeben, wird das Verzeichnis nicht gelöscht.

Anhang B

Programmdokumentation

Das Programm ist in der Programmiersprache $C++$ geschrieben. Es besteht aus drei Hauptklassen, einigen Hilfsklassen und einer main-Methode. Die drei Hauptklassen sind

- die Klasse box, die einen Präfixquader darstellt,
- die Klasse resultArray, die das Ergebnisfeld beinhaltet und berechnet
- und die Klasse array, die ein dynamisches Feld mit den zwei möglichen Standardisierungsverfahren implementiert.

Für jede der drei Hauptklassen gibt es gleichnamig Implementations- (.cc) und Definitionsdateien (.h). Für die Hilfsklassen ist außerdem eine Datei misc.h und für die main-Methode eine Datei main.cc vorhanden. Alle Dateien sind im Anhang C abgedruckt.

B.1 Die Datei misc.h

In der Datei misc.h sind mehrere Klassen definiert, auf die die Hauptklassen zugreifen. Der Verbund editOp definiert eine Editieroperation die, wie in Unterabschnitt 6.3.3.4 erläutert ist, durch den Wert i und den Typ type der Editieroperation bestimmt ist. Der Typ einer Editieroperation wird dabei durch Konstanten kodiert.

Die Klasse cell repräsentiert genau eine Zelle des Berechnungsquaders. Sie enthält als Komponenten den jeweiligen Transformationsabstand value, den Typ der genau einen Elementaroperation, die einer von dieser Zelle ausgehenden Kante des Berechnungsgraphen entspricht (arrow) und das eine Abkürzung im Quader definierende eta.

Die Klasse resultElement beschreibt einen Ergebnisabstand mit Zusatzinformationen. Sie beinhaltet den längentransformierten Abstand value, den die Datierung angebenden

Parameter whichbox, sowie die die ursprüngliche Zelle charakterisierenden Variablen k, nu und $length = k + eta$. Eine weitere Komponente ist das Feld edits mit Elementen vom Typ editOp, in dem eine optimale Transformation anhand der in ihr enthaltenen Editieroperationen abgelegt wird. edits hat dabei immer die Länge k.

Mit Hilfe der Klasse stillWorkingClass wird während der Berechnung angezeigt, daß die Berechnung im Gange ist.

B.2 Die Klasse box

In der Klasse box wird ein Präfixquader berechnet. Der Quader wird in einem dreidimensionalen Feld entry abgespeichert, das aus Objekten der Klasse cell besteht.

Die meisten Variablen, Methoden und Parameter entsprechen den in Unterkapitel 6.3 und in Kapitel 7 verwendeten Begriffen und Algorithmen. Im Konstruktor der Klasse box werden N_e und M_e berechnet und der benötigte Speicherplatz reserviert, welcher im Destruktor wiederum freigegeben wird. In der Methode fill wird der Präfixquader mittels des Algorithmus aus Abschnitt 6.3.3.3 anhand der in Abschnitt 7.2 angegebenen Wertebereiche für k, ν und η gefüllt. Für jede Zelle wird dabei eine Abkürzung, wie in Abschnitt 6.3.3.4 beschrieben ist, gespeichert.

In der Methode computeResults werden aus dem gefüllten Berechnungsquader unter Anwendung der Ergebnisse-Prozedur aus Unterkapitel 7.3 alle benötigten Abstände mit jeweils genau einer optimalen Transformation gespeichert. Die Speicherung findet dabei in einem Feld result mit Elementen der in misc.h definierten Klasse resultElement statt. Das Feld wird in der Klasse boxArray gespeichert, und in jedem Präfixquader werden weitere Ergebnisabstände hinzugefügt. Die Variable rIndex zeigt dabei auf bzw. hinter das letzte definierte Element in result. Für jeden Ergebnisabstand wird dabei eine Transformation mittels des Algorithmus TransformationMitAbkürzungen aus Unterabschnitt 6.3.3.4 berechnet und die einzelnen Editieroperationen in der Komponente edits des Ergebniselementes gespeichert.

B.3 Die Klasse resultArray

Die Klasse resultArray wird im Hauptprogramm genau einmal instanziiert. Sie ist für die Verwaltung und Berechnung aller Ergebniselemente zuständig. Ein Ergebniselement ist ein Objekt der Klasse resultElement und beinhaltet einen Ergebnisabstand, die zugehörige optimale Transformation sowie Informationen zu der ursprünglichen Zelle und dem Präfixquader. resultArray enthält ein Feld namens result, das aus Objekten der Klasse resultElement besteht. Da es höchstens $(m + n - 2minOvl + 1)\frac{\alpha^2+3\alpha+2}{2}$ Ergebnisabstände geben kann (s. Unterkapitel 7.3), wird das Feld im Konstruktor mit dieser Größe angelegt.

In der Methode fill werden anhand des Ergebnisfeld-Algorithmus aus Abschnitt 7.3 alle Ergebnisabstände mit zugehörigen optimalen Transformationen berechnet. In jedem Schritt wird dabei ein neuer Berechnungsquader **b** der Klasse box initialisiert und gefüllt. Das Feld result wird durch die Aufrufe von **b**− >**computeResults** sukkzessive aufgefüllt, wobei die Komponente rIndex immer auf das nächste freie Element in result zeigt. Um anzuzeigen, daß die Berechnung läuft, wird nach jedem Schleifendurchgang stillWorking.print() aufgerufen.

Mit der Methode sort wird das Ergebnisfeld nach aufsteigenden Abstandseinträgen value sortiert. Dabei wird das in ANSI-C vorhandene Quicksort qsort verwendet. Quicksort hat zwar im schlechtesten Fall eine quadratische Laufzeit, benötigt jedoch im Mittel eine Laufzeit von $\theta(n \log n)$ für Felder der Größe n.

Die Methode show gibt die in result gespeicherten und vorher mit sort sortierten Ergebnisse aus. Zur Übersichtlichkeit werden nur die resultNum ersten, also dem Abstand nach kleinsten, Ergebnisse ausgegeben. Für ein Ergebnis wird zuerst die entsprechende Jahreszahl ausgegeben, die sich durch Addition des Anfangsjahres masstartyear der Referenzfolge und des Parameters whichbox, der die Position des Berechnungsquaders im sukkzessiven Algorithmus angibt, berechnet. Weiterhin werden die Kosten, also der Abstand value, die Länge der optimal transformierten Folge sowie die Positionen (in Kalenderjahren, bezogen auf die Datierung) der Editieroperationen der einen abgespeicherten optimalen Transformation ausgegeben. Der Typ einer Editieroperation wird dabei mit Z für eine Zusammenfassung und E für eine Einfügung angegeben.

Eine graphische Auswertung der Ergebnisse ist mit Hilfe des Programm *gnuplot* möglich. Die dafür benötigten Dateien werden in der Methode writeTransformedData in das Verzeichnis *work* geschrieben. Für jedes der resultNum ersten Ergebnisse in result werden drei Dateien angelegt: Eine Datei im Zwei-Spalten-Format für die transformierte Musterfolge, eine Datei im Zwei-Spalten-Format für das entsprechende Teilstück der Referenzfolge sowie eine Datei mit *gnuplot*-Aufrufen zum Anzeigen der ersten beiden Dateien und zur Markierung der Stellen der Einfügungen und Zusammenfassungen. Die Dateinamen bestehen aus den Präfixen sample, master bzw. plot, und dem gemeinsamen Suffix <Jahr>.<Anzahl der Zusammenfassungen>.<Anzahl der Einfügungen>. Die Anzahl der Zusammenfassungen ist dabei das entsprechende ν und die Anzahl der Einfügungen $k - \nu$ (vgl. Satz 6.16). In die Datei namens result werden der *gnuplot* Programmaufruf sowie die Namen aller plot-Dateien geschrieben. Diese wird mit einem Systemaufruf, der zur Zeit für Unix-Systeme angegeben ist und für andere Betriebssysteme eventuell geändert werden muß, ausführbar gemacht und später gegebenenfalls ausgeführt.

B.4 Die Klasse array

In der Klasse array ist ein dynamisches Feld A von double-Werten implementiert, das zur Speicherung der Muster- und Referenzdaten dient. Im Konstruktor wird ein Feld der

übergebenen Größe size angelegt. Bei der Zuweisung von Werten mit der Methode set wird dabei gegebenenfalls das Feld in Schritten der Größe increase erweitert.

Die Methode mu gibt für einen gegebenen Index i das arithmetische Mittel der Werte A[i-1] und A[i+1] zurück. Falls i das erste bzw. letzte Element des Feldes bezeichnet wird das erste bzw. letzte Element des Feldes zurückgegeben.

Die Methoden floating_average und log_diff stellen die beiden zur Verfügung stehenden Standardisierungsverfahren dar. In floating_average wird die im Paramter raw übergebene Folge anhand des gleitenden Mittels (vgl. Abschnitt 2.4.3) mit einer Breite von $2 * \text{halfwidth} + 1$ standardisiert und im Objekt-eigenen Feld A gespeichert. Falls der Parameter dolog wahr ist, wird der natürliche Logarithmus jedes mit dem gleitenden Mittel standardisierten Wertes gespeichert. In der Methode log_diff wird die Differenz von natürlichen Logarithmen der Eingabefolge raw gebildet, wobei vor der Anwendung des Logarithmus jeder Wert mit dem Faktor logmult multipliziert wird. Die in A gespeichert Ergebnisfolge ist dabei um eins kürzer als die Ursprungsfolge raw.

B.5 Die main-Methode

In der main-Methode wird der in Unterkapitel 7.5 angegebene Programmablauf realisert. Zunächst werden aus der gegebenenfalls existierenden Konfigurationsdatei dpcross.cfg die Programmparameter eingelesen (vgl. Anhang A). Danach werden die Parameter von der Kommandozeile, also die Dateinamen der Referenz- und der Musterfolge sowie gegebenenfalls α und $resultNum$ eingelesen. Die in der Konfigurationsdatei angegebenen Werte für α und $resultNum$ werden dabei gegebenenfalls überschrieben.

Danach werden die Muster- und Referenzdaten aus den angegebenen Dateien in die Instanzen sampleraw bzw. masterraw der Klasse array eingelesen. Wie in Anhang A beschrieben, muß die Musterfolge im Ein- bzw. Zwei-Spalten-Format und die Referenzfolge im Zwei-Spalten-Format vorliegen. Die Anfangsdaten werden in den Variablen samstartyear und masstartyear gespeichert; dabei dient die Datierung samstartyear nur zum Vergleich. Gegebenenfalls werden die Daten standardisiert, wobei die standardisierte Musterfolge in dem array sample und die standardisierte Referenzfolge in master abgespeichert wird. Falls einige der Parameter halfwidth, logmult, minOvl und alpha vom Benutzer auf undefinierte Werte oder garnicht gesetzt wurden, werden diese auf Standardwerte gesetzt.

Die Berechnung und Ausgabe der Abstände erfolgt nun durch das Anlegen einer Instanz results der Klasse resultArray und dem Aufruf der Methoden results.fill, results.sort und results.show. Der Methode results.fill werden dabei die standardisierten Daten oder, wenn nicht standardisiert wurde, die eingelesenen Daten übergeben. Vor der Ausgabe der Ergebnisse werden zur Information die Länge der zur Abstandsberechnung verwendeten Folgen, einige Parameter sowie zum Vergleich eine bekannte Datierung der Musterfolge, falls vorhanden, ausgegeben. Der Aufruf von results.show wird so oft wiederholt, wie der

Benutzer den Parameter resultNum ändern möchte. Anschließend werden die resultNum ersten Ergebnisse mit result.writeTransformedData in Dateien geschrieben und gegebenenfalls mit *gnuplot* angezeigt. Die Ausgabe der Ergebnisse kann beliebig oft wiederholt werden. Am Ende werden die Ergebnisdateien im Verzeichnis *work* gelöscht. Falls der Anwender die Ergebnisdateien anderweitig verwenden möchte, sollte er diese während des Programmablaufs an einer anderen Stelle speichern oder in der Konfigurationsdatei den Löschstring delDir umsetzen.

Anhang C

Programmcode

C.1 main.cc

```
#include "box.h"
#include "resultArray.h"

int main(int argc, char *argv[]){
   // Name der Konfigurationsdatei
  char configfilename[20]="dpcross.cfg";

  // Dateinamen der Eingabedateien
  char *samname, *masname;

  // L"angen des Musters und der Referenzfolge
  unsigned samsize, massize;

  // Anfangsjahre (beim Muster nur zum Vergleich)
  unsigned masstartyear, samstartyear=0;

  // Dateiname der ausf"uhrbaren gnuplot-Datei
  char resultname[]="work/result";

  // minimale "Uberlappung
  unsigned minOvl=50;

  // Maximale Anzahl von Editieroperationen
  unsigned alpha=1;
```

```
// Gewichtsfaktoren f"ur die einzelnen Elementaroperationen
double w_E=1., w_Z=1., w_I=1.;

// Faktor zur Multiplikation mit jedem Abstand
// (f"ur eine "ubersichtlichere Ausgabe)
double factor=100.;

// Nicht-standardisierte Daten
array sampleraw(100,100), masterraw(100,100);

// Faktor zur Multiplikation bei log.-Standardisierung
double logmult=5;

// Anzahl der auszugebenden Ergebnisse
unsigned resultNum=10;

// Halbe Breite des gleitenden Mittels
unsigned halfwidth=2;

// Befehl zum L"oschen des work-Verzeichnisses
char *delDir=new char[256];
strcpy(delDir,"rm -f work/*");

unsigned i=0, dummy;
int ok=1;
unsigned dofloatave=1,  dolog=1, dologdiff=0;
double tmp;
FILE *file;
char buffer[256], *s;


/*
 * Einlesen der Konfigurationsdatei
 */
if((file=fopen(configfilename,"r"))!=NULL){
  s=fgets(buffer,256,file);
  while(s!=NULL){
    if(strncmp(buffer,"\\",2)!=0){
      //Kein Kommentar
      if(strncmp(buffer,"preprocess",10)==0){
        if(strncmp((buffer+11),"floatave",8)==0){
          dologdiff=0;
          //Gleitendes Mittel
          dofloatave=1;
```

```
        } else if(strncmp((buffer+11),"logdiff",7)==0){
          dofloatave=0;
          //Differenz von Logarithmen
          dologdiff=1;
        } else{
          dofloatave=0;
          dologdiff=0;
        }
      } else if(strncmp(buffer,"halfwidth",9)==0){
        sscanf(buffer+10, "%i",&halfwidth);

      } else if(strncmp(buffer,"dolog",5)==0){
        if(strncmp((buffer+6),"true",4)==0){
          dolog=1;
        } else{
          dolog=0;
        }

      } else if(strncmp(buffer,"logmult",7)==0){
        sscanf(buffer+8, "%lf",&logmult);

      } else if(strncmp(buffer,"alpha",5)==0){
        sscanf(buffer+6, "%u",&alpha);

      } else if(strncmp(buffer,"minOvl",6)==0){
        sscanf(buffer+7, "%u",&minOvl);

      } else if(strncmp(buffer,"resultNum",9)==0){
        sscanf(buffer+10, "%u",&resultNum);

      } else if(strncmp(buffer,"factor",6)==0){
        sscanf(buffer+7, "%lf",&factor);

      } else if(strncmp(buffer,"w_E",3)==0){
        sscanf(buffer+4, "%lf",&w_E);

      } else if(strncmp(buffer,"w_I",3)==0){
        sscanf(buffer+4, "%lf",&w_I);
      } else if(strncmp(buffer,"w_Z",3)==0){
        sscanf(buffer+4, "%lf",&w_Z);

      } else if(strncmp(buffer,"delDir",6)==0){
strcpy(delDir,buffer+7);
if(!strncmp(delDir,"null",4)){
```

```
  delete delDir;
  delDir=NULL;
}
      }

    }
    s=fgets(buffer,256,file);
  }
  fclose(file);
}//if

/*
 * Einlesen der Dateinamen (sowie optional alpha und
 * resultNum) von der Kommandozeile
 */
if(argc<3){
  cerr<<"Bitte Master- und Samplename angeben."<<endl;
  exit(1);
} else{
  masname=argv[1];
  samname=argv[2];
  if(argc>=4){
    alpha=atoi(argv[3]);
  }
  if(argc>=5){
    resultNum=atoi(argv[4]);
  }
}

/*
 * Einlesen der Musterdaten
 */
file=fopen(samname,"r");
if(file==NULL){
  cerr<<"Fehler beim Oeffnen der Musterdatei "<<samname<<endl;
  exit(1);
}

i=0;
//Einlesen der ersten Zeile, um auf Ein- oder Zwei-Zeilen-Format zu pruefen.
fgets(buffer,256,file);

//Zwei-Zeilen-Format
if(sscanf(buffer, "%u%lf", &dummy, &tmp)==2){
```

```
  samstartyear=dummy;

  sampleraw.set(i++,tmp);
  while(ok!=EOF){
    ok=fscanf(file, "%u", &dummy);
    fscanf(file, "%lf", &tmp);
    if(ok!=EOF)
      sampleraw.set(i++,tmp);
  }

//Ein-Zeilen-Format
} else{
  sampleraw.set(i++,dummy);
  while(ok!=EOF){
    ok=fscanf(file, "%lf", &tmp);
    if(ok!=EOF)
      sampleraw.set(i++,tmp);
  }

}
samsize=i;
fclose(file);

/*
 * Einlesen der Referenzdaten
 */
ok=1;
i=0;
file=fopen(masname,"r");
if(file==NULL){
  cerr<<"Error opening master file "<<masname<<endl;
  exit(1);
}
ok=fscanf(file, "%u", &masstartyear);
fscanf(file, "%lf", &tmp);
if(ok!=EOF)
  masterraw.set(i++,tmp);
while(ok!=EOF){
  ok=fscanf(file, "%u", &dummy);
  fscanf(file, "%lf", &tmp);
  if(ok!=EOF)
   masterraw.set(i++,tmp);
}
fclose(file);
```

```
massize=i;


/*
 * Berechnung der standardisierten Daten
 */
// standardisierte Daten
array sample(samsize,100);
array master(massize,100);

unsigned minLast=min(sampleraw.last,masterraw.last);

if(dofloatave){
  //Berechnung einer kleineren 'halfwidth', falls noetig
  if((2*halfwidth+1>minLast)||(halfwidth==0)){
    halfwidth=(minLast-1)/2;
    cout<<"halfwidth geaendert auf "<<halfwidth<<endl;
  }
  sample.floating_average(sampleraw, halfwidth, (dolog==1));
  master.floating_average(masterraw, halfwidth, (dolog==1));
  minLast=min(sample.last,master.last);
} else if(dologdiff){
  if(logmult<=0){
    logmult=1.;
    cout<<"logmult geaendert auf "<<logmult<<endl;
  }

  sample.log_diff(sampleraw, logmult);
  master.log_diff(masterraw, logmult);
  minLast=min(sample.last,master.last);
}

//Berechne kleineres minOvl, wenn noetig
if(minOvl>minLast){
  minOvl=minLast;
  cout<<"minOvl geaendert auf "<<minOvl<<endl;
}

//Berechne kleineres alpha, wenn noetig
if(alpha>minLast){
  alpha=minLast;
  cout<<"alpha geaendert auf "<<alpha<<endl;
}
```

```
/*
 * Berechnung und Ausgabe der Abst"ande
 */
if(dofloatave){
  cout<<"Vorverarbeitung: floatave; halfwidth="<<halfwidth
      <<", dolog="<<((dolog=0)?"false":"true")<<endl;
} else if(dologdiff){
  cout<<"Vorverarbeitung: logdiff; logmult="<<logmult<<endl;
} else{
  cout<<"Keine Vorverarbeitung."<<endl;
}
cout<<"alpha="<<alpha<<", minOvl="<<minOvl<<", nRaw="<<sampleraw.last
    <<", mRaw="<<masterraw.last<<endl;
cout<<"factor="<<factor<<", w_E="<<w_E<<", w_I="<<w_I<<", w_Z="
    <<w_Z<<", resultNum= "<<resultNum<<endl;

resultArray results((dofloatave||dologdiff)?sample.last:sampleraw.last,
    (dofloatave||dologdiff)?master.last:masterraw.last,
    alpha,minOvl);
results.fill((dofloatave||dologdiff)?sample:sampleraw,
    (dofloatave||dologdiff)?master:masterraw,factor,w_E, w_I,w_Z);
results.sort();

char answer='n';
char finished='j';

do{
  do{
    if((answer=='j')||(answer=='J')){
      cout<<"resultNum: ";
      cin>>resultNum;
      cout<<endl;
    }
    if(samstartyear!=0)
      cout<<"Bekannte Datierung: "<<samstartyear<<endl;
    results.show(resultNum,masstartyear);
    cout<<"Anzeige mit anderer resultNum (j/n)? ";
    cin>>answer;
    cout<<endl;
  } while((answer=='j')||(answer=='J'));

  results.writeTransformedData(masstartyear,resultNum,sampleraw,masterraw,
 resultname);
```

```
    cout<<"Gnuplot aller aufgefuehrten Matchings (j/n)? ";
    cin>>answer;
    cout<<endl;
    do{
      if((answer=='j')||(answer=='J')){
        system(resultname);
        cout<<"Nochmaliges Gnuplot (j/n)? ";
        cin>>answer;
        cout<<endl;
      }
    } while((answer=='j')||(answer=='J'));
    cout<<"Fertig (j/n)? (Wenn nein, dann nochmaliges Anzeigen der Daten.) ";
    cin>>finished;
    cout<<endl;
  }while((finished!='j')&&(finished!='J'));

  cout<<"L"osche Verzeichnis work mit "<<delDir<<endl;
  system(delDir);
}
```

C.2 box.h

```
#ifndef BOX_H
#define BOX_H

#include "misc.h"

class box{
  public:
  cell ***entry;
  unsigned M_e,N_e,alpha,minOvl;

  box(unsigned N, unsigned M, unsigned alpha, unsigned minOvl);
  ~box();
  static int i(unsigned nu, unsigned eta);
  static int j(unsigned k, unsigned eta);
  unsigned maxNu(unsigned k);
  unsigned maxEta(unsigned k, unsigned nu);
  static double d(double a, double b);
  void fill(array& sample, array& master, unsigned sstart, unsigned mstart,
    double w_E, double w_I, double w_Z);
```

```
  void computeResults(resultElement *result, unsigned& rIndex,
 unsigned whichBox, double factor);
};


#endif //BOX_H
```

C.3 box.cc

```
#include "box.h"

//------------------------------------------------------------------------/
//----------------------------     box       --------------------------/
//------------------------------------------------------------------------/
box:: box(unsigned N, unsigned M, unsigned alpha, unsigned minOvl){
  this->alpha=alpha;
  this->minOvl=minOvl;

  N_e=N;
  M_e=M;
  if(M>N+alpha)
    M_e=N+alpha;
  else if(M<N-alpha)
    N_e=M+alpha;

  entry=new cell**[alpha+1];
  for(unsigned k=0; k<=alpha; k++){
    entry[k]=new cell*[(maxNu(k)+1)];
    for (unsigned nu=0; nu<=maxNu(k); nu++){
      entry[k][nu]=new cell[maxEta(k,nu)+1];
    }
  }

  entry[0][0][0].value=0.;
}

//------------------------------------------------------------------------/
//----------------------------    ~box     --------------------------/
//------------------------------------------------------------------------/
box:: ~box(){
  for(unsigned k=0; k<=alpha; k++){
    for (unsigned nu=0; nu<=maxNu(k); nu++){
```

```
      delete []entry[k][nu];
    }
    delete []entry[k];
  }

  delete []entry;
}

//------------------------------------------------------------------------/
//----------------------------       i       --------------------------/
//------------------------------------------------------------------------/
int box:: i(unsigned nu, unsigned eta){
  //-1 , da das Feld bei 0 beginnt
  return (2*nu+eta-1);
}

//------------------------------------------------------------------------/
//----------------------------       j       --------------------------/
//------------------------------------------------------------------------/
int box:: j(unsigned k, unsigned eta){
  //-1 , da das Feld bei 0 beginnt
  return (k+eta-1);
}

//------------------------------------------------------------------------/
//----------------------------       d       --------------------------/
//------------------------------------------------------------------------/
double box:: d(double a, double b){
  return sqr(a-b);
}

//------------------------------------------------------------------------/
//----------------------------  maxNu       --------------------------/
//------------------------------------------------------------------------/
unsigned box:: maxNu(unsigned k){
  return min(k,min((unsigned)(N_e/2),(unsigned)((N_e-minOvl+k)/2)));
}

//------------------------------------------------------------------------/
//----------------------------  maxEta      --------------------------/
//------------------------------------------------------------------------/
unsigned box:: maxEta(unsigned k, unsigned nu){
  return  min(N_e-2*nu,M_e-k);
}
```

```
//-------------------------------------------------------------------------/
//----------------------------   fill        --------------------------/
//-------------------------------------------------------------------------/
void box:: fill(array& sample, array& master, unsigned sstart,
                      unsigned mstart,double w_E, double w_I, double w_Z){
  unsigned char defined;

  unsigned k,nu,eta;
  double miss=0., match=0., doubl=0., min;

  for(k=0; k<=alpha; k++){
    for(nu=0; nu<=maxNu(k); nu++){
      for(eta=((k==0)&&(nu==0))?1:0; eta<=maxEta(k,nu); eta++){
        defined=NONE;
        if(k>0){
          if((nu>0)&&((nu-1)<=maxNu(k-1))&&(eta<=maxEta(k-1,nu-1))){
            doubl=entry[k-1][nu-1][eta].value
                  +w_Z*d(sample[sstart+i(nu,eta)-1]+sample[sstart+i(nu,eta)],
                  master[mstart+j(k,eta)]);
            defined |= DOUBL;
    min=doubl;
          }
          if((nu<=maxNu(k-1))&&(eta<=maxEta(k-1,nu))){
            miss=entry[k-1][nu][eta].value
                 +w_E*d(sample.mu(sstart+i(nu,eta)),master[mstart+j(k,eta)]);
            defined |= MISS;
    min=miss;
  }
        }
        if(eta>0){
          match=entry[k][nu][eta-1].value
                +w_I*d(sample[sstart+i(nu,eta)],master[mstart+j(k,eta)]);
          defined |= MATCH;
  min=match;
        }

if(((defined & DOUBL) !=0)&&(doubl<min)){
  min=doubl;
}
if(((defined & MISS) !=0)&&(miss<min)){
  min=miss;
}
if(((defined & MATCH) !=0)&&(match<min)){
```

```
  min=match;
}

        entry[k][nu][eta].value=min;

//Speichern einer Abkuerzung
        if(((defined & MATCH) !=0)&&(min==match)){
          entry[k][nu][eta].arrow=MATCH;
  entry[k][nu][eta].eta=entry[k][nu][eta-1].eta;
        } else if(((defined & DOUBL) !=0)&&(min==doubl)){
          entry[k][nu][eta].arrow=DOUBL;
  entry[k][nu][eta].eta=eta;
        } else if(((defined & MISS) !=0)&&(min==miss)){
          entry[k][nu][eta].arrow=MISS;
  entry[k][nu][eta].eta=eta;
       }
      } //for eta
    } //for nu
  } //for k
}

//-------------------------------------------------------------------------/
//---------------------     computeResults    ---------------------------/
//-------------------------------------------------------------------------/
void box:: computeResults(resultElement *result, unsigned& rIndex,
 unsigned whichBox, double factor){

  unsigned k,nu,eta,k2,nu2,eta2;
  for(k=0; k<=alpha; k++){
    for(nu=0; nu<=maxNu(k); nu++){
      eta=maxEta(k,nu);

      //Speichere Ergebnis in result
      result[rIndex].length=eta+k;
      result[rIndex].value=factor*entry[k][nu][eta].value
                  /((double)result[rIndex].length);
      result[rIndex].whichBox=whichBox;
      result[rIndex].k=k;
      result[rIndex].nu=nu;

      //Verfolge Transformation zurueck
      if(k>0){
        result[rIndex].edits=new editOp[k];
        if(entry[k][nu][eta].arrow==MATCH){
```

```
  eta2=entry[k][nu][eta].eta;
} else{
  eta2=eta;
}
nu2=nu;
k2=k;
while(k2>0){
  result[rIndex].edits[k2-1].i=i(nu2,eta2);
  result[rIndex].edits[k2-1].type=entry[k2][nu2][eta2].arrow;
  if(entry[k2][nu2][eta2].arrow==DOUBL){
    nu2--;
  }
  k2--;
  eta2=entry[k2][nu2][eta2].eta;
}
      }
      rIndex++;

    } //for nu
  } //for k
}
```

C.4 resultArray.h

```
#ifndef RESULTARRAY_H
#define RESULTARRAY_H

#include "misc.h"
#include "box.h"

class resultArray{
  public:
  resultElement *result;
  box *b;
  stillWorkingClass stillWorking;
  unsigned rIndex, n, m, boxnum, alpha, minOvl;

  resultArray(unsigned n, unsigned m, unsigned alpha, unsigned minOvl);
  ~resultArray();
  void fill(array& sample, array& master,double factor,
double w_E, double w_I, double w_Z);
  void sort();
```

```
  void show(unsigned resultNum, unsigned masstartyear);
  void writeTransformedData(unsigned masstartyear,unsigned resultNum,
      array& sampleraw, array& masterraw,char* resultname);
};


#endif //RESULTARRAY_H
```

C.5 resultArray.cc

```
#include "resultArray.h"

//------------------------------------------------------------------------/
//-----------------------      resultArray      --------------------------/
//------------------------------------------------------------------------/
resultArray:: resultArray(unsigned n, unsigned m,unsigned alpha,
  unsigned minOvl){
  rIndex=0;
  this->n=n;
  this->m=m;
  this->alpha=alpha;
  this->minOvl=minOvl;
  boxnum=n+m-2*minOvl+1;

  result=new resultElement[boxnum*((alpha*(alpha+1))/2+alpha+1)];
  result[0].length=0;
  result[0].value=0.;
}

//------------------------------------------------------------------------/
//-----------------------     ~resultArray    --------------------------/
//------------------------------------------------------------------------/
resultArray:: ~resultArray(){

  for(unsigned i=0; i<rIndex; i++){
    if(result[i].k>0)
      delete []result[i].edits;
  }
  delete[] result;
}
```

```
//------------------------------------------------------------------------/
//-----------------------         fill       ---------------------------/
//------------------------------------------------------------------------/
void resultArray:: fill(array& sample, array& master, double factor,
double w_E, double w_I, double w_Z){
  for (unsigned ii=0; ii<boxnum; ii++){
    //linke unvollstaendige Ueberlappungen
    if(ii<n-minOvl){
      b=new box(minOvl+ii,m,alpha,minOvl);
      b->fill(sample, master, n-minOvl-ii,0,w_E,w_I,w_Z);
      b->computeResults(result,rIndex,ii,factor);
    //vollstaendige und rechte unvollstaendige Ueberlappugen
    } else {
      b=new box(n,m+n-ii-minOvl,alpha,minOvl);
      b->fill(sample, master,0,ii-n+minOvl,w_E,w_I,w_Z);
      b->computeResults(result,rIndex,ii,factor);
    }
    delete b;
    stillWorking.print();
  }
  cout<<"\b "<<endl;
  cout<<"Anzahl der Ergebnisse: "<<rIndex<<endl;
}

//------------------------------------------------------------------------/
//-----------------------         sort       ---------------------------/
//------------------------------------------------------------------------/
void resultArray:: sort(){
  qsort((void *)result, rIndex, sizeof(resultElement),
        (int (*)(const void*,const void*))(resultElement::compare));
}

//------------------------------------------------------------------------/
//-----------------------         show       ---------------------------/
//------------------------------------------------------------------------/
void resultArray:: show(unsigned resultNum, unsigned masstartyear){
  unsigned year;

  for(unsigned ii=0; ii<min(rIndex,resultNum); ii++){
     year=masstartyear+result[ii].whichBox-n+minOvl;

     unsigned lastYear=year+result[ii].length-1;
      //Im Falle einer linken unvollstaendigen Ueberlappung addiere
      //die Anzahl der nicht ueberlappenden Daten
```

```
    if(result[ii].whichBox<n-minOvl){
      lastYear+=n-minOvl-result[ii].whichBox;
    }
    cout<<"Jahr: "<<year<<"-"<<lastYear;
    cout<<"   Kosten: "<<result[ii].value<<" Laenge: "<<result[ii].length
        <<";";
    if(year<masstartyear)
      year=masstartyear;
    for(unsigned kk=0; kk<result[ii].k; kk++){
      year+=result[ii].edits[kk].i;
      if(result[ii].edits[kk].type==DOUBL){
 year--;
      } else{
 year++;
      }
     if(kk>0){
 year-=(result[ii].edits[kk-1].i);
      }
      if(result[ii].edits[kk].type==DOUBL){
 cout<<" Z "<<year<<";";
      } else {//MISS
         cout<<" E "<<year<<";";
      }
    }//for kk
    cout<<endl;
    //}//if
  } //for ii
}

//------------------------------------------------------------------------/
//--------------------   writeTransformedData  -------------------------/
//------------------------------------------------------------------------/
void resultArray:: writeTransformedData(unsigned masstartyear,
    unsigned resultNum, array& sampleraw, array& masterraw, char* resultname){

  // Dateinamenst"amme f"ur die Ergebnisdateien
  char masstring[]="work/master", samstring[]="work/sample";
  char plotstring[]="work/plot";

  char samfilename[30], masfilename[30],plotname[30];
  FILE *samfile, *masfile, *plotfile, *resultfile;
  unsigned year,y,y2;

  if((resultfile=fopen(resultname,"w"))==0){
```

```
  cerr<<"Fehler beim Schreiben einer Ergebnisdatei. "<<endl;
  cerr<<"Eventuell mu"s das Verzeichnis \"work\" angelegt werden."<<endl;
  exit(0);
}
fprintf(resultfile,"gnuplot ");
for(unsigned ii=0; ii<min(resultNum,rIndex); ii++){
    year=masstartyear+result[ii].whichBox-n+minOvl;

    sprintf(samfilename,"%s%u.%u.%u",samstring,year,result[ii].nu,
                      result[ii].k-result[ii].nu);
    sprintf(masfilename,"%s%u.%u.%u",masstring,year,result[ii].nu,
                        result[ii].k-result[ii].nu);
    sprintf(plotname,"%s%u.%u.%u",plotstring,year,result[ii].nu,
                      result[ii].k-result[ii].nu);

    //Das erste, in die Datei zu schreibende sample-Element ist das 0.
    unsigned  s=0;

    //Das erste master-Element, was in die Datei geschrieben wird ist
    //  das 0., im Falle einer linken unvollstaendigen Ueberlappung
    //  das (ii-n+minOvl)., sonst
    unsigned m=(result[ii].whichBox<n-minOvl)?0:result[ii].whichBox-n+minOvl;

    samfile=fopen(samfilename,"w");
    masfile=fopen(masfilename,"w");
    plotfile=fopen(plotname,"w");
    fprintf(resultfile,"%s ",plotname);

    //Loesche alle Labels in gnuplot
    fprintf(plotfile,"set nolabel\n");
    fprintf(plotfile,"set title \"Kosten=%f\"\n",result[ii].value);

    //Im Falle einer linken unvollstaendigen Ueberlappung schreibe die
    //nicht ueberlappenden Daten in die Datei
    if(result[ii].whichBox<n-minOvl){
      for(unsigned iii=0; iii<n-minOvl-result[ii].whichBox; iii++){
        fprintf(samfile,"%u\t%f\n",
          masstartyear-(n-minOvl-result[ii].whichBox-iii),sampleraw[s++]);
      }
    }

    y=masstartyear+m;
    //y=year;
    //y2 ist das Jahr, an dem die naechste Editieroperation stattfindet
```

```
for(unsigned kk=0; kk<result[ii].k; kk++){
   y2=y+result[ii].edits[kk].i;
   if(result[ii].edits[kk].type==DOUBL){
     y2--;
   } else{ //MISS
     y2++;
   }
   if(kk>0){
     y2-=(result[ii].edits[kk-1].i+1);
  }

  for(; y<y2; y++){
            fprintf(samfile,"%u\t%f\n",y,sampleraw[s++]);
    //if(y2>=masstartyear)
              fprintf(masfile,"%u\t%f\n",y,masterraw[m++]);
  }

  if(result[ii].edits[kk].type==DOUBL){
    fprintf(plotfile,"set label \"Z\" at %u,17 center\n",y);
    fprintf(plotfile,"set label \"Z\" at %u,27 center\n",y);
    fprintf(plotfile,"set label \"Z\" at %u,37 center\n",y);

    fprintf(samfile,"%u\t%f\n",y,(sampleraw[s]+sampleraw[s+1]));
    s+=2;
    //if(y2>=masstartyear)
        fprintf(masfile,"%u\t%f\n",y,masterraw[m++]);
  } else {//MISS
    fprintf(plotfile,"set label \"E\" at %u,17 center\n",y);
    fprintf(plotfile,"set label \"E\" at %u,27 center\n",y);
    fprintf(plotfile,"set label \"E\" at %u,37 center\n",y);
    fprintf(samfile,"%u\t%f\n",y,sampleraw.mu(s-1));
    //if(y2>=masstartyear)
              fprintf(masfile,"%u\t%f\n",y,masterraw[m++]);
  }
  y=y2+1;
        }//for kk

      unsigned lastS=result[ii].length-result[ii].k+result[ii].nu;
      //Im Falle einer linken unvollstaendigen Ueberlappung addiere
      //die Anzahl der nicht ueberlappenden Daten
      if(result[ii].whichBox<n-minOvl){
lastS+=n-minOvl-result[ii].whichBox;
      }
      while(s<lastS){
```

```
        fprintf(samfile,"%u\t%f\n",y,sampleraw[s++]);
        fprintf(masfile,"%u\t%f\n",y++,masterraw[m++]);
      }

    fprintf(plotfile,"plot \"%s\" with linespoints, \"%s\" with linespoints",
       samfilename,masfilename);
    fprintf(plotfile,"\npause -1 \"<return> f"ur den n"achsten Graph\"");
    fclose(plotfile);

    fclose(samfile);
    fclose(masfile);
    // }//if

  } //for ii

  fclose(resultfile);
  char command[20];
  sprintf(command,"chmod u+x %s",resultname);
  system(command);
}
```

C.6 misc.h

```
#ifndef MISC_H
#define MISC_H

#include<fstream.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include "array.h"

#define min(a,b) ((a)<(b)?(a):(b))
#define max(a,b) ((a)>(b)?(a):(b))
#define sqr(a) ((a)*(a))

// Konstanten
const unsigned short NONE=0;
const unsigned short MISS=1;
const unsigned short MATCH=2;
const unsigned short DOUBL=4;
```

```
struct editOp{
  unsigned i;
  unsigned short type;
};


class cell{
  public:
  double value;
  unsigned short arrow;
  unsigned eta;

  cell(){
    value=0.;
    arrow=NONE;
    eta=0;
  }
};

class resultElement{
public:
  double value;
  unsigned whichBox,k,nu,length;
  editOp *edits;

  static int compare(const resultElement *r1, const resultElement *r2){
    if(r1->value < r2->value)
      return -1;
    else if(r1->value > r2->value)
      return 1;
    else
      return 0;
  }
};


class stillWorkingClass{
  unsigned i;

  //0 is '/', 1 is '\'
  unsigned lastchar;

public:
```

```
  stillWorkingClass(){
  i=0;
  lastchar=0;
  }

  void print(){
    if(i==0){
      lastchar=1;
      cout<<"/"<<flush;
    } else{
      if(lastchar==0){
        lastchar=1;
        cout<<"\b\\"<<flush;
      } else{
        lastchar=0;
        cout<<"\b/"<<flush;
      }
    }
    i++;
  }
};


#endif //MISC_H
```

C.7 array.h

```
#ifndef ARRAY_H
#define ARRAY_H

#include<iostream.h>
#include<stdlib.h>
#include<math.h>

class array{
  unsigned size;
  unsigned increase;
  double *A;

  public:
  unsigned last; // 1+Index des rechtesten belegten Array-Platzes
  array(unsigned size=100, unsigned increase=100);
```

```
  ~array();
  double operator[](unsigned i);
  double mu(int i);
  void set(unsigned i, double value);
  void floating_average(array& raw, unsigned halfwidth,bool dolog);
  void log_diff(array& raw, double logmult);
};


#endif //ARRAY_H
```

C.8 array.cc

```
#include "array.h"

//-------------------------------------------------------------------------/
//------------------------     array      -----------------------------/
//-------------------------------------------------------------------------/
array:: array(unsigned size, unsigned increase){
  if((A=(double *)malloc(size*sizeof(double)))==NULL){
    cerr<<"Fehler bei der Bereitstellung von Speicherplatz."<<endl;
    exit(0);
  }
  this->increase=increase;
  this->size=size;
  last=0;
}

//-------------------------------------------------------------------------/
//------------------------    ~array      -----------------------------/
//-------------------------------------------------------------------------/
array:: ~array(){
  free(A);
}

//-------------------------------------------------------------------------/
//------------------------   operator[]   -----------------------------/
//-------------------------------------------------------------------------/
double array:: operator[](unsigned i){
  if(i<last){
    return(A[i]);
  } else{
```

```
    cout<<"Index i="<<i<<" >= "<<last<<"=last"<<endl;
    exit(1);
  }
}

//-------------------------------------------------------------------------/
//----------------------------      mu       --------------------------/
//-------------------------------------------------------------------------/
double array:: mu(int i){
  if(i<(int)last){
    if(i==(int)last-1)
       return A[i];
    else if(i==-1)
       return A[0];
    else{
      return ((double)(A[i]+A[i+1])/2.);
    }
  } else{
    cout<<"Index i="<<i<<" >= "<<last<<"=last"<<endl;
    exit(1);
  }
}

//-------------------------------------------------------------------------/
//------------------------       set      -----------------------------/
//-------------------------------------------------------------------------/
void array:: set(unsigned i, double value){
  if(i<size){
    A[i]=value;
  } else{
    unsigned k;
    for(k=1; i>=size+k*increase; k++)
      ;

    void *v;
    if((v=realloc(A,(size+k*increase)*sizeof(double)))==NULL){
      cerr<<"Fehler bei der Bereitstellung von Speicherplatz."<<endl;
      exit(0);
    }
    A=(double *)v;
    size+=k*increase;
    A[i]=value;
  }
  if(i+1>last)
```

```
    last=i+1;
}


//-------------------------------------------------------------------------/
//------------------------  floating_average  ----------------------------/
//-------------------------------------------------------------------------/
void array:: floating_average(array& raw, unsigned halfwidth, bool dolog){
// Prozent des (2*halfwidth+1)-gleitenden Mittels,
// eventuell der Logarithmus davon
  unsigned i, left, right, mid=0;
  double sum=0, factor, newvalue;

  factor=2*halfwidth+1;

  // linke unvollstaendige gleitende Mittel
  for(i=0; i<=halfwidth; i++)
    sum+=raw[i];
  for(i=0; i<halfwidth; i++)
    sum+=raw[0];

  newvalue=factor*(double)raw[mid]/(double)sum;
  if(dolog)
    newvalue=log(newvalue);
  set(mid,newvalue);

  left=0;
  mid=0;
  right=halfwidth;

  for(mid=1; mid<halfwidth; mid++){
    sum-=raw[0];
    sum+=raw[++right];

    newvalue=factor*(double)raw[mid]/(double)sum;
    if(dolog)
      newvalue=log(newvalue);
    set(mid,newvalue);

  }
  sum-=raw[0];
  sum+=raw[++right];
  // Jetzt sind genau 2*halfwidth+1 Jahre in 'sum' aufsummiert.
```

```
  double sumOfNewvalues=0.;

  // vollstaendige gleitende Mittel
  newvalue=factor*(double)raw[mid]/(double)sum;
  if(dolog)
    newvalue=log(newvalue);
  set(mid,newvalue);

  while(right<raw.last-1){ //raw.last-1 ist das letzte Element von 'raw'
    sum-=raw[left++];
    sum+=raw[++right];
    mid++;
    newvalue=factor*(double)raw[mid]/(double)sum;
    if(dolog)
      newvalue=log(newvalue);
    set(mid,newvalue);
    sumOfNewvalues+=newvalue;
  }

  // rechte unvollstaendige gleitende Mittel
  for(mid++; mid<=raw.last-1; mid++){
    sum-=raw[left++];
    sum+=raw[last-1];

    newvalue=factor*(double)raw[mid]/(double)sum;
    if(dolog)
      newvalue=log(newvalue);
    set(mid,newvalue);
  }
}


//------------------------------------------------------------------------/
//-----------------------      log_diff      --------------------------/
//------------------------------------------------------------------------/
void array:: log_diff(array& raw, double logmult){
  for(unsigned i=0; i<raw.last-1; i++){
   set(i,(log(logmult*(double)raw[i])-log(logmult*(double)raw[i+1])));
  }

}
```

Literaturverzeichnis

[1] Roland W. Aniol. Tree-ring analysis using Catras. *Dendrochronologia*, 1:45–53, 1983.

[2] R.M. Argent. *Dendroclimatological investigation of river red gum.* PhD thesis, University of Melbourne, 1995. Kapitel 10.

[3] Robert Argent and Rob Wilson. The tree ring faq. http://aqua.civag.unimelb.edu.au/~argent.

[4] M.G.L. Baillie. *Tree-Ring Dating and Archaeology.* Croom Helm London, 1982.

[5] M.G.L. Baillie and J.R. Pilcher. A simple crossdating program for tree-ring research. *Tree-Ring Bulletin*, 33:7–14, 1973.

[6] I.N. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik.* B.G. Teubner, 25 edition, 1991.

[7] Paolo Cherubini and Wolfgang Zierhofer. Bäume als Zeugen der Klimageschichte. *Spektrum der Wissenschaft*, pages 122–124, November 1996.

[8] Edward R. Cook and K. Briffa. Data analysis. In Edward R. Cook and Leonardas A. Kairiukstis, editors, *Methods of Dendrochronology - Applications in the Environmental Sciences*, pages 97–162. Kluwer Academic Publishers and International Institute for Applied Systems Analysis, Dordrecht, Niederlande, 1990.

[9] Edward R. Cook and Leonardas A. Kairiukstis, editors. *Methods of Dendrochronology - Applications in the Environmental Sciences.* Kluwer Academic Publishers and International Institute for Applied Systems Analysis, Dordrecht, Niederlande, 1990.

[10] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1994.

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

[12] A.E. Douglass. Crossdating in dendrochronology. *Journal of Forestry*, 39:825–831, 1941.

[13] D. Eckstein and J. Bauch. Beitrag zur Rationalisierung eines dendrochronologischen Verfahrens und zur Analyse seiner Aussagesicherheit. *Forstwissenschaftliches Zentralblatt*, 88:230–250, 1969.

[14] Dieter Eckstein. *Entwicklung und Anwendung der Dendrochronologie zur Altersbestimmung der Siedlung Haithabu*. PhD thesis, Universität Hamburg, 1969.

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

[16] Harold C. Fritts. Computer programs for tree-ring research. *Tree-Ring Bulletin*, 25:2–7, 1963.

[17] H.C. Fritts. *Tree Rings and Climate*. Academic Press, 1976.

[18] Klaus Gollmer. *Prozeßdiagnose mit Mitteln der Mustererkennung und Anwendung in der Biotechnologie*. VDI Verlag, 1996.

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

[20] Herman John Heikkenen. Tree-ring patterns: A key-year technique for crossdating. *Journal of Forestry*, pages 302–305, May 1984.

[21] E. Hollstein. *Mitteleuropäische Eichenchronologie*. Phillip von Zabern, Mainz am Rhein, 1980.

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

[23] Bruno Huber. Über die Sicherheit jahrringchronologischer Datierung. *Holz als Roh- und Werkstoff*, 6:263–268, 1943.

[24] Bruno Huber. Dendrochronologie. In *Handbuch der Mikroskopie in der Technik*, volume 5, pages 171–211. 1970.

[25] International Tree-Ring Data Bank (ITRDB). ITRDB program library. ftp.ngdc.noaa.gov; http://www.ngdc.noaa.gov/paleo/treering.html.

[26] Michele Kaennel and Fritz H. Schweingruber. *Multilingual Glossary of Dendrochronology*. Paul Haupt Publishers, Bern, Schweitz, 1995.

[27] Peter Klein and Dieter Eckstein. Die Dendrochronologie und ihre Anwendung. *Spektrum der Wissenschaft*, pages 56–68, Januar 1988.

[28] 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.

[29] Peter Ian Kuniholm and Bernd Kromer et al. Anatolian tree rings and the absolute chronology of the eastern Mediterranean, 2220-718bc. *Nature*, 381:780–783, June 1996.

[30] Roy Lawrance and Robert A. Wagner. An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery*, 22(2):177–183, 1975.

[31] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

[32] Coralie Mills. Dating short tree-ring sequences: an evaluation of three statistical procedures. In *Tree Rings and Environment: Proceedings of the international dendrochronological symposium in Ystad, Schweden*, pages 225–229, 1992.

[33] Martin A.R. Munro. An improved algorithm for crossdating tree-ring series. *Tree-Ring Bulletin*, 44:17–27, 1984.

[34] Hermann Ney. Dynamic programming as a technique for pattern recognition. In *Proceedings of the 6th International Conference on Pattern Recognition in Munich*, pages 1119–1125, 1982.

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

[36] M.L. Parker. Dendrochronological techniques used by the geological survey of canada. Technical report, Geological Survey of Canada, 1971. paper 71-25.

[37] Robert Pohl. *Corina - The Cornell Ring Analysis Program.* The Malcolm and Carolyn Wiener Laboratory for Aegean and Near Eastern Dendrochronology, Department of the History of Art and Archaelogy, Cornell University, Ithaca. Quellcode und Handbuch.

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

[39] Lawrence R. Rabiner, Aaron E. Rosenberg, and Stephen E. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-26(6):575–582, 1978.

[40] Thomas Riemer. Über die Varianz von Jahrringbreiten. Statistische Methoden für die Auswertung der jährlichen Dickenzuwächse von Bäumen unter sich ändernden Lebensbedingungen. Berichte des Forschungszentrums Waldökosysteme, Reihe A, Band 121, 1994. Dissertation Universität Göttingen.

[41] Frank Rinn. *TSAP Reference Manual.* Heidelberg.

[42] 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.

[43] 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.

[44] Rainer Schwabe. Elementare Wahrscheinlichkeitstheorie. Skriptum zur Vorlesung im Wintersemester 1990/91, Freie Universität Berlin.

[45] F.H. Schweingruber. *Der Jahrring.* Paul Haupt, Bern, 1983.

[46] F.H. Schweingruber. *Jahrringe und Umwelt - Dendroökologie.* Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft, Birmensdorf, 1993.

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

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

[49] R.E. Taylor, Minze Stuiver, and Paula J. Reimer. Development and extension of the calibration of the radiocarbon time scale: Archaeological applications. *Quaternary Science Reviews (Quaternary Geochronology)*, 15:655–668, 1996.

[50] Esko Ukkonen and Derick Wood. Approximate string matching with suffix automata. *Algorithmica*, 10:353–364, 1993.

[51] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[52] Ynjiun P. Wang and Theo Pavlidis. Optimal correspondence of string subsequences. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12. 1990.

[53] Wayne M. Wendland. An objective method to identify missing or false rings. *Tree-Ring Bulletin*, 35:41–47, 1975.

[54] T.M.L. Wigley, P.D. Jones, and K.R. Briffa. Cross-dating methods in dendrochronology. *Journal of Archaeological Science*, 14:51–64, 1987.

[55] Brian Williams. *Biostatistiscs.* Chapman & Hall, 1993.

[56] Archie G. Worthing and Joseph Geffner. *Treatment of Experimental Data.* John Wiley, Chapman & Hall, 1943.

[57] Taro Yamane. *Statistik - Ein einführendes Lehrbuch*, volume 1. Fischer, 1976.