Diplomarbeit

Tree-Coffee: Multiples Sequenz-Struktur-Alignment von RNA für vorgegebene Strukturen mit Hilfe einer Konsistenzerweiterung

Albert-Ludwigs-Universität Freiburg
Fakultät für angewandte Wissenschaften
Lehrstuhl für Bioinformatik

Joachim Krempel
Matrikelnr.: 1762156

Gutachter:
Prof. Dr. Rolf Backofen
Dr. Sebastian Will

21.01.2009

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Ort, Datum

Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Diese Diplomarbeit beschäftigt sich mit dem Thema des multiplen Sequenz-Struktur-Alignments von RNA[1], anhand eines neu entwickelten Algorithmus genannt „Tree-Coffee“. Sequenz-Struktur-Alignment bedeutet, dass zur Bestimmung des Alignments nicht nur die Nukleotidebene betrachtet wird, sondern auch die Ähnlichkeit der gegebenen Strukturen in diese Berechnung eingearbeitet wird. Ziel dieses Verfahrens ist es einen neuen Ansatz in diesem Bereich der algorithmischen Lösungen zur gegebenen Problemstellung zu verwirklichen.

Die daraus resultierenden Ergebnisse werden mit Hilfe eines Benchmarks analysiert, wodurch ein erster Eindruck gewonnen werden kann, wie gut dieser Ansatz in der Praxis funktioniert. Insbesondere werden sich im Laufe dieser Arbeit auch einige Schwachstellen offenbaren, die sich negativ auf die Ergebnisse auswirken.

1.1 Motivation

> „Seit Jahrzehnten wurden RNA Moleküle nur als Befehlsempfänger der DNA[2] angesehen, die genetische Informationen in Proteine umwandeln. Entdeckungen der letzten Zeit zeigen auf, dass eine Unterklasse der RNA viele Funktionen der Zelle kontrollieren. [...] Es gibt sogar Hinweise darauf, dass bestimmte RNA Moleküle dabei helfen eine Zelle ihrer Bestimmung zuzuweisen, indem sie während der Entwicklung Gene aktivieren oder deaktivieren.“

Diese Erkenntnis wurde im Jahre 2002 als Durchbruch des Jahres gefeiert. [Cou02]

Seit diesem Zeitpunkt wurde der Erforschung der RNA und ihrer Bedeutung ein immer größerer Stellenwert zugesprochen. Ein Ergebnis aus der verstärkten Forschung auf diesem Gebiet besteht darin, dass die Funktion eines RNA Moleküls zu großen Teilen von seiner Struktur abhängig ist. Als Annahme aus dieser Tatsache resultiert die Vermutung über eine stärkere Konserviertheit der Struktur eines Moleküls gegenüber ihrer Sequenzbasen. Um dieser Möglichkeit Rechnung zu tragen, ist es notwendig, in einem Alignment nicht nur die reinen Mutationswahrscheinlichkeiten der Basen zu betrachten, sondern auch die strukturellen Gegebenheiten in dieses Alignment einfließen zu lassen.

Als Konsequenz daraus ergeben sich die Bemühungen schnelle und effiziente Algorithmen zu entwickeln, die sich mit dieser Problematik auseinandersetzen. „Tree-Coffee“ ist einer davon und hat es sich zum Ziel gesetzt, diese Aufgabenstellung mit Hilfe fester Eingabestrukturen und einer Konsistenzerweiterung zu lösen.

[1] engl. RiboNucleic Acid
[2] engl. DeoxyriboNucleic Acid

1.2 Ziele und Vorteile des Algorithmus

Das Ziel von „Tree-Coffee“ besteht also darin, ein multiples Sequenz-Struktur-Alignment für eine beliebige Anzahl von Sequenz- und Strukturpaaren zu berechnen. Dabei soll der Einfluss der Strukturen während der ganzen Bearbeitung des Alignments erhalten bleiben. Auf diese Weise lässt sich sicherstellen, dass die wichtigen Faktoren für die Funktionalität und die Konservierung der Sequenzen während des gesamten Prozesses Einfluss auf das Ergebnis nehmen.

Dies alleine ist jedoch nicht der Grund wieso ein Sequenz-Struktur-Alignment gewählt wurde. Im Vergleich zu einem reinen Algorithmus, der nur auf Sequenzebene arbeitet, bringt die Verwendung einer Sequenz-Struktur-Paarung mehr nützliche Informationen mit sich. Bei der Verwendung von multiplen Sequenzalignment-Algorithmen zeigt sich eine Verschlechterung der Ergebnisse, sobald die paarweise Sequenzidentität (APSI[3]), der zu alignierenden Sequenzen, abnimmt. Erfahrungswerte zeigen, dass ein Sequenz-Struktur-Alignment für APSI-Werte $\leq 60\%$ bessere Ergebnisse liefert als ein Sequenzalignment. Eine Herausforderung ist es nun die Mehrkosten bezüglich Laufzeit und Speicherplatzbedarf, die für die Durchführung eines Sequenz-Struktur-Alignments benötigt werden, in einem vertretbaren Rahmen zur Verbesserung der Ergebnisse zu halten. „Tree-Coffee“ wird versuchen einen guten Kompromiss zwischen Laufzeit und Speicher einzugehen, ohne dabei spürbare Einbußen bezüglich der Qualität der Alignments zu erhalten.

1.3 Einordnung von Tree-Coffee

In diesem Abschnitt wird kurz auf andere Algorithmen eingegangen, die sich mit dem Thema des multiplen Alignments beschäftigen. Insbesondere wird es von Interesse sein Unterschiede aufzuzeigen, um einen Überblick zu erhalten, in welchem Bereich „Tree-Coffee“ eine andere Strategie verfolgt, als die hier aufgezählten Verfahren.

Es wird dabei nicht darum gehen diese Algorithmen in ihrer ganzen Komplexität vorzustellen, sondern eher einen kurzen Einblick in den Ablauf zu erhalten ohne auf die genaue Realisierung einzugehen.

Zusätzlich zu Sequenz-Struktur-Alignment Ansätzen wird hierbei auch ein Sequenzalignment Algorithmus betrachtet, um im späteren Verlauf dieser Arbeit bei der Betrachtung der Testergebnisse einen schematischen Einblick zu besitzen und somit die Vergleichsmöglichkeit zwischen diesen beiden Arten des Alignments zu haben.

[3] engl. average pairwise sequence identity

1.3.1 ClustalW

Bei ClustalW handelt es sich um einen Algorithmus zur Berechnung eines multiplen Sequenzalignments, der mit Hilfe von Sequenzgewichtung, positionsspezifischen Gapkosten und der Auswahl einer den Sequenzen angepassten Gewichtsmatrix, die Qualität des progressiven Alignments verbessern will [THG94]. Der Ansatz eines progressiven Alignments besteht im wesentlichen darin, die Sequenzen paarweise zu analysieren, um Informationen zu diesen zu erhalten. Daraufhin wird mit diesen Informationen eine Ordnung erstellt, die angibt in welcher Reihenfolge die Bildung des Alignments am effizientesten wäre, bevor letztlich das multiple Alignment erstellt wird.

Aus diesem Grund lässt sich ClustalW in 3 Verarbeitungsschritte aufteilen:

- Paarweises Alignment aller Sequenzpaare zur Erstellung einer Distanzmatrix.
- Aufbauen eines phylogenetischen Baumes aus dieser Distanzmatrix.
- Alignment entlang des erstellten Baumes.

Nach der Berechnung aller Distanzen, welche entweder mit Hilfe einer schnellen annähernden Methode oder eines genaueren dynamischen Programmieransatzes durchgeführt werden können, wird der phylogenetische Baum mittels eines „Neighbour-Joining“-Verfahrens [SN87] aufgestellt. Anhand des Guide-Trees[4] wird nun zusätzlich noch jeder Sequenz ein Gewicht zugeordnet, das sich in Abhängigkeit der Distanz zur Wurzel des Baumes berechnet. Nun beginnt die Berechnung des multiplen Alignments. Um die Qualität dabei zu verbessern fließen einige Verbesserungen in dieses ein. Für die erste Optimierung werden die berechneten Sequenzgewichte verwendet, indem man sie als einfache Multiplikationsfaktoren bei der Bewertung von Positionen einwirken lässt. Eine weitere Idee besteht darin, die initialen Gapkosten während des Alignments zu manipulieren. Dies geschieht anhand von verschiedenen Informationen wie der Ähnlichkeit, der aktuell zu alignierenden Sequenzen, die Länge, respektive der Längenunterschied zwischen ihnen und positionsspezifischen Gapkosten. Dies ist nur ein kleiner Überblick über die Kriterien anhand derer die Kosten modifiziert werden. Für jeden weiteren Schritt des progressiven Alignments wird anhand der Distanz zwischen den betrachteten Sequenzen (Sequenzgruppen) eine Wahl über die zu verwendende Bewertungsmatrix getroffen. Desweiteren ist es möglich, divergente Sequenzen unterhalb einer bestimmten Sequenzidentität zu allen anderen, auszusetzen und erst im späteren Verlauf zu betrachten, um Fehler die dadurch entstehen könnten zu vermeiden. Durch alle diese eingeführten Verfahren zur Verbesserung des progressiven Alignments erreicht ClustalW gute bis sehr gute Ergebnisse für die berechneten Alignments.

[4] hier: synonym für phylogenetischen Baum

1.3.2 Der Sankoff-Algorithmus

Das parallele Alignieren und Falten von Sequenzen behandelte Sankoff in einer seiner Veröffentlichungen [San85]. Sankoff versuchte dabei mit der Kombination eines Alignment- und eines Faltungsalgorithmus noch während des Alignments eine gemeinsame Struktur der Sequenzen vorherzusagen. Der Teil des Alignments versucht dabei die Distanz zwischen den alignierten Sequenzen zu minimieren, während versucht wird die stabilste Struktur zu finden, also diejenige Struktur mit der geringsten freien Energie. Dieses Verfahren benötigt jedoch schon für das paarweise Alignment sehr viel Zeit und das obwohl bereits bei der Strukturberechnung Einbußen in der Genauigkeit zu Gunsten der Geschwindigkeit getroffen wurden. Der Algorithmus von Sankoff hat in der Theorie gute Ansätze gezeigt, aber ist in der Praxis nur schwer in dieser Form umsetzbar. Stattdessen wurde sein Algorithmus mit Modifikationen weiterentwickelt und ist noch heute von Bedeutung. Dazu zählen zum Beispiel die Algorithmen FOLDALIGN [HLSG05] oder PMmulti [HBS04]

1.3.3 RNA-forester

Einen anderen Ansatz zur Berechnung eines multiplen Alignments verfolgt RNA-forester [HTGK03]. Ähnlich des Verfahrens von Jiang et al. zur Alignierung von geordneten Bäumen [JWZ95] werden bei RNA-forester die Sequenz-Struktur-Paare als Wälder von Bäumen interpretiert. Durch die unterschiedliche Darstellung von Knoten für gebundene Basen und freie Basen ist es dabei möglich in einem Baum die komplette Information über die Sequenz und ihre Struktur zu erhalten. Ziel des Algorithmus ist es nun die gegebenen Bäume zu alignieren, und dabei einen „Superbaum“ zu erstellen, der das vollständige multiple Alignment darstellt. Dabei wird damit begonnen zwei Bäume zu betrachten und das beste Alignment zu erstellen.

Nach der Berechnung aller paarweisen Alignments wird mittels eines progressiven Ansatzes das multiple Alignment erstellt.

1.3.4 MARNA

Ein von Sven Siebert und Rolf Backofen entwickeltes Verfahren zur Berechnung eines multiplen Alignments wurde in der Publikation: „MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons“ [SB05] vorgestellt.

In einem ersten Schritt erstellt MARNA für alle Paare von Sequenzen und Strukturen anhand einer geringfügig modifizierten Bewertungsfunktion, welche von Jiang et al. im Jahr 2002 entwickelt wurde [JLMZ02], eine Bewertung. Zur Durchführung dieser paarweisen Bewertung aller Paare steht es dem Benutzer frei, neben den Sequenzen Strukturen anzugeben, oder diese von MARNA berechnen zu lassen. Dabei besteht die Möglichkeit, zwischen einer Struktur mit minimaler freier Energie, oder der Verwendung von sub-optimalen Strukturen, zu wählen. Die sub-optimalen Strukturen werden mit Hilfe von RNAsubopt aus dem Vienna RNA package [uPS98] beziehungsweise RNAshapes,

einem von Giegerich et al. entwickelten Algorithmus [GVR04] berechnet. Wählt der Anwender diese zweite Möglichkeit wird nicht nur eine Struktur pro Sequenz verwendet, sondern ein so genanntes Ensemble von sub-optimalen Strukturen. Auf diesem Wege ist es MARNA möglich, mehrere Sequenz-Struktur-Alignments in Abhängigkeit der Anzahl vorhandener Strukturen für jede Sequenzpaarung durchzuführen. Mit Hilfe dieser paarweisen Alignments werden Bibliotheken erstellt, in denen alle realisierten Kanten des Alignments und ein dazugehöriges Kantengewicht gespeichert werden.

Zur Berechnung eines einzelnen multiplen Alignments werden diese paarweisen Bewertungsergebnisse an „T-Coffee“ übergeben, das sich um die weitere Berechnung kümmert. Da in den erstellten Bibliotheken auch Informationen über die verwendeten Strukturen eingeflossen sind lässt sich aus diesem multiplen Alignment auf einfache Weise eine Konsensusstruktur ableiten, indem eine Struktur erstellt wird, die die Anzahl der konservierten Basen maximiert. Das Ergebnis liefert also ein multiples Alignment und die dazugehörige Konsensusstruktur für die alignierten Sequenzen. Eine Implementation des MARNA Algorithmus kann auf der Bioinformatikwebseite der Universität Freiburg unter http://biwww2.informatik.uni-freiburg.de/Software/MARNA/index.html gefunden werden [SB03].

1.3.5 Schlussfolgerung

Durch die hier gezeigten Beispiele ist zu erkennen, dass es viele Möglichkeiten gibt, sich dem Thema Alignment zu nähern. Es ist möglich Alignments nur aufgrund der vorhandenen Sequenzen zu erstellen, oder auch ihre Strukturen hinzu zu ziehen. Während sich Sankoff durch die Idee der simultanen Strukturberechnung deutlich von Tree-Coffee, das feste Eingabestrukturen erwartet, unterscheidet verfolgt RNA-forester durch die Methode des Alignierens von Bäumen eine völlig andere Strategie zur Berechnung des Alignments.

Der hier vorgestellte Algorithmus lässt sich am ehesten noch mit MARNA vergleichen. Ein wichtiger Unterschied liegt darin, dass Tree-Coffee die Transformation der Kantengewichte nicht nur auf den sequentiellen, sondern auch auf den strukturellen Kanten durchführt. Zusätzlich bildet MARNA das multiple Alignment nicht länger auf Sequenz-Struktur-Ebene, wohingegen Tree-Coffee auch bei dieser Berechnung weiterhin die gegebenen Strukturen mit aligniert. MARNA verwendet also die Strukturinformationen nur für die Gewichtung der Kanten und nicht zusätzlich noch zur Bildung des multiplen Alignments. Dies stellt einen Vorteil von Tree-Coffee dar, da die Strukturinformationen bis zum Ende des Algorithmus verwendet werden, um das beste Ergebnis zu erzielen. Zu diesem Zweck erstellt Tree-Coffee während des multiplen Alignments Konsensusstrukturen für die Alignments, um diese in späteren Schritten wieder zu verwenden.

Aufgrund der Gemeinsamkeiten zwischen MARNA und Tree-Coffee wird dieser Vergleich bei den Testergebnissen von großem Interesse sein und zeigen ob der Ansatz von Tree-Coffee gegenüber MARNA im Vorteil ist.

1.4 Aufbau der Arbeit

Nachdem nun einige Motive dargelegt wurden, aus welchem Grund man sich mit dem Thema Alignment von Sequenzen, oder hier im speziellen Sequenz-Struktur-Alignment, beschäftigt, wird ein kurzer Ausblick darauf gegeben was in den nächsten Kapiteln folgen wird.

Zuerst werden einige grundlegende Informationen rund um den biologischen Hintergrund der RNA aufgeführt. Dabei werden die verschiedenen Arten der RNA und ihre Funktionen etwas ausführlicher behandelt, um einen tieferen Einblick zu gewähren, aus welchen Gründen die Erforschung der RNA von großer Wichtigkeit ist. Bei Tree-Coffee handelt es sich, wie bereits erwähnt, um einen Algorithmus zur Realisierung eines Sequenz-Struktur-Alignments, weshalb auch grundlegende Kenntnisse über Strukturen von Sequenzen nötig sind. Hierzu werden die verschiedenen Abstraktionsebenen einer Struktur vorgestellt und erläutert. Nach Betrachtung biologischer Grundlagen wird erläutert worin genau die Bedeutung von Alignments liegt und weswegen diesen in dem Bereich der Bioinformatik eine so hohe Relevanz zugesprochen wird.

Im Anschluss an die Vermittlung der biologischen Grundlagen gilt es die formalen Grundlagen und Definitionen darzulegen, die im weiteren Verlauf der Arbeit vermehrt verwendet werden. Hinzukommend wird der „T-Coffee“ [NHH00] Algorithmus vorgestellt, in Anlehnung dessen die Idee zu Tree-Coffee entstand, der im darauf folgenden Kapitel ausführlich erläutert wird.

Die Betrachtung der Testergebnisse des entwickelten Algorithmus im Vergleich zu einigen anderen Verfahren, die das selbe Ziel verfolgen, wird als nächster Punkt betrachtet. Neben der Präsentation der Ergebnisse steht dabei auch die Analyse derer und ein erster Kommentar über die Erfüllung der Erwartungen bezüglich des Programms im Mittelpunkt. Etwaige Beanstandungen an den Resultaten werden im letzten Abschnitt dieser Diplomarbeit diskutiert.

2 Biologische Grundlagen rund um die RNA

In diesem Kapitel werden die biologischen Fakten zum Thema RNA vorgestellt. Von Interesse ist es hier einen Einblick zu gewähren, weshalb die RNA in der Forschung seit einigen Jahren eine große Aufmerksamkeit zugesprochen bekommt.

Diese kleine Einführung in die Thematik wird in zwei Teilen erfolgen. Zuerst wird die Beschaffenheit der RNA, ihre Entstehung und ihr Aufbau erläutert werden, woraufhin eine kurze Übersicht der verschiedenen Arten und den ihnen zugesprochenen Funktionen folgen wird. In einem zweiten Teil wird sich alles um den Bereich der Strukturen von RNA Molekülen drehen. Zu diesem Zweck werden die unterschiedlichen Abstraktionsebenen erläutert und ein Bezug zwischen der Struktur und ihrer biologischen Bedeutung hergestellt.

2.1 Bestandteile und Aufbau

Die Ribonukleinsäure, kurz RNA, entsteht aus der chemischen Zusammensetzung vieler Nukleotiden. Diese Verbindung wird als Polymer bezeichnet, ein aus vielen Bauteilen zusammengesetztes Molekül, oder im Falle von RNA und DNA speziell als Oligonukleotid, einem Polymer aus einer Vielzahl von Nukleotiden. Ein einzelnes Nukleotid besteht dabei aus folgenden Teilen: (Abbildung 2.1)

- Base
- Ribose (Zucker)
- Phosphatgruppe

Die Verbindung der einzelnen Nukleotide zu einem Polymer erfolgt mit Hilfe einer phosphodiester-Bindung zwischen dem Phosphat des ersten Moleküls und der Hydroxyl-Gruppe an Position 3' des Zuckers. Auf diese Weise entwickelt sich ein so genannter „Backbone“ entlang dessen die einzelnen Basen der Nukleotide aufgereiht sind. Anschaulich lässt sich dies am besten beschreiben durch eine lange Linie an deren einzelnen Positionen eine Base steht. Von essentieller Bedeutung für die Funktionalität des Moleküls sind hierbei seine Größe und die Anordnung der Basen entlang des Backbones.

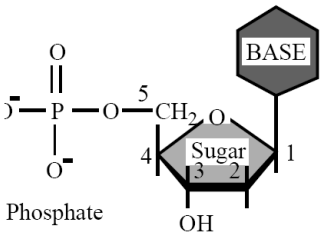

Abbildung 2.1: Allgemeine Darstellung eines Nukleotids
Quelle: [CB00]

2.1.1 Vorkommende Basen innerhalb eines RNA Moleküls

Ein RNA Molekül kann die vier Basen Adenin, Cytosin, Guanin und Uracil in beliebiger Anzahl und Anordnung enthalten. Im weiteren Verlauf dieser Arbeit werden diese Basen zur Vereinfachung durch A, C, G und U repräsentiert.

Durch einfach Aufzählung der vorkommenden Basen vom Start bis zum Ende des Moleküls lässt sich die genaue Zusammensetzung des Polymers beschreiben. Diese Aufzählung nennt man Sequenz der RNA. Ein Beispiel: „AGUCGUA“ stünde dabei für eine RNA bestehend aus sieben Basen in der gezeigten Reihenfolge.
Der chemische Aufbau der einzelnen Basen, und des gesamten RNA Moleküls, ermöglicht die Bindungen zwischen einzelnen Basen durch Wasserstoffbrücken oder andere schwächere Mechanismen wie der van-der-Waals-Kräfte, wodurch sich eine Struktur entwickelt.

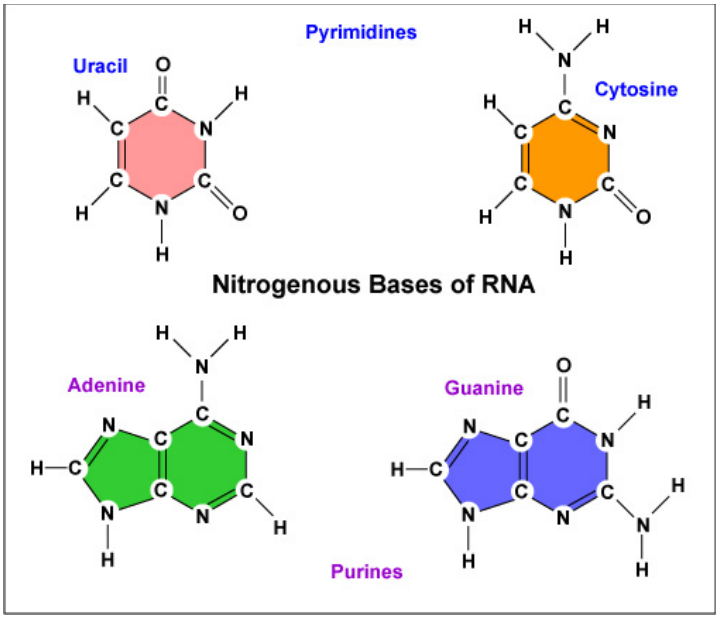

Abbildung 2.2: Die vier Basen der RNA
Quelle: http://student.ccbcmd.edu/~gkaiser/biotutorials/dna/images/RNAbases.jpg

2.2 Untergruppen und ihre Funktion

RNAs lassen sich durch ihre Funktionen in Gruppen aufteilen, von denen hier exemplarisch einige aufgelistet und erläutert werden (Aus [CB00] und [Edd01]).

- *mRNA* - Bezeichnet die Gruppe der „messenger“ RNAs, die aus der DNA umgesetzt, und für die Synthese der Proteine benötigt wird.
- *tRNA* - Die „transfer“ RNA dient dem Zweck, eine Aminosäure der Proteinsynthese zuzuführen während die Informationen der mRNA übersetzt werden.
- *rRNA* - Ein Bestandteil der Ribosome ist die „ribosomale“ RNA, welche die Fähigkeit besitzt Informationen der mRNA in Aminosäuren umzuwandeln und diese an die tRNA weiterzugeben.
- *snoRNA* - Diese Gruppe der RNAs ist dazu bestimmt, ortsspezifische Modifikationen an anderen RNA-Typen vorzunehmen.
- *miRNA* - Zur Familie der nicht-kodierenden RNAs zählend, sind die „micro“ RNA Regulatoren für kodierende RNAs.

Aufgrund der entdeckten Vielfalt in der Familie der RNAs, insbesondere in dem Bereich der nicht-kodierenden Moleküle, wurden die Forschungsbemühungen stark erweitert und damit begonnen die wahren Ausmaße der RNA für einen lebenden Organismus zu erkennen.

2.3 Ebenen der Darstellung einer RNA

In der Biologie stehen einige Abstraktionsebenen zur Darstellung der Struktur eines RNA Moleküls zur Verfügung, welche durch die Bindungsmöglichkeiten einzelner Basen untereinander ausgebildet werden. Im Folgenden werden nun drei Arten dieser Darstellungen anhand von kleinen Beispielen erläutert werden.

2.3.1 Primärstruktur

Eine der einfachsten Möglichkeiten ein RNA Molekül zu beschreiben, ist mit der Primärstruktur gegeben. Dabei handelt es sich im eigentlichen Sinne nicht um die Beschreibung einer Struktur, sondern vielmehr um die Aufzählung der vorkommenden Basen der betreffenden RNA entlang des Backbones.

Beispiel 2.1. `GUAGAUAUAGUUUACAAAAAACAUUAGACUGUGAAUCUAA`

Wie zu erkennen ist, beinhaltet diese Darstellung keinerlei Strukturinformationen über vorhandene Bindungen zwischen den Basen, sondern dient lediglich dazu die Bestandteile der RNA zu beschreiben.

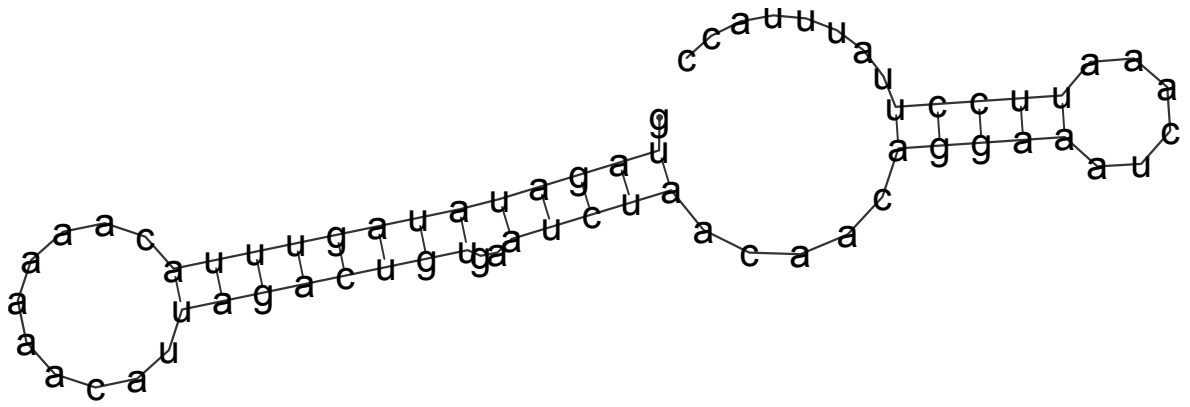

Abbildung 2.3: Darstellung einer Sekundärstruktur
Erstellt mit RNAplot aus dem Vienna Package [uPS98]

2.3.2 Sekundärstruktur

Der Sinn der Sekundärstruktur besteht darin Informationen über die realisierten Bindungen der Basen aufzuzeigen. In Abbildung 2.3 ist eine grafische Darstellung der Struktur einer RNA gezeigt. Um nicht immer mit dieser grafischen Beschreibung zu arbeiten, stehen einige andere Notationsmöglichkeiten für diese Strukturebene zur Verfügung.

1. Dot-Bracket-Notation
2. Kantenlisten
3. Baumdarstellung

In dieser Arbeit wird die Dot-Bracket-Notation verwendet werden, mit deren Hilfe sich anhand einer einfachen Zeichenfolge aus '(', ')' und '.' entlang der einzelnen Basen die Beschreibung der Sekundärstruktur ermöglichen lässt. In Verbindung mit der Primärstruktur lässt sich so die Beschaffenheit der RNA und der entwickelten Bindungen zwischen den Basen effizient darstellen.

Beispiel 2.2. `.((((((((((((.........))))))))..))))).`

Das Beispiel zeigt die Sekundärstruktur der in der Primärstruktur vorgestellten RNA. Die Anforderungen an die Struktur liegen darin, dass die Anzahl der Symbole, der Anzahl der Basen der RNA entspricht. '.' stellt hierbei eine freie Base dar wohingegen ein Paar aus '(' und ')' die Bindung zweier Basen symbolisiert. Daraus ergibt sich auch die Tatsache, dass die Anzahl öffnender und schließender Klammern gleich sein muss. In der Bioinformatik werden die meisten strukturbezogenen Berechnungen mit Hilfe der Sekundärstruktur durchgeführt, da sie oftmals für die benötigten Zwecke ausreicht und effiziente Berechnungen darauf möglich sind. Zusätzlich gibt es zur Erstellung von Sekundärstrukturen, auf Basis der Primärstruktur einfache Algorithmen wie beispielsweise die Vorhersage anhand der Minimierung der freien Energie nach Zuker [ZS81].

Abbildung 2.4: Tertiärstruktur einer tRNA
Quelle:
http://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/3d_tRNA.png/602px-3d_tRNA.png

2.3.3 Tertiärstruktur

Mit der Tertiärstruktur können nicht nur die Bindungen zwischen Basen ausgedrückt werden, sondern auch die räumliche Anordnung des gesamten RNA Strangs aufgrund der geformten Bindungen gezeigt werden. Die Berechnung oder experimentelle Feststellung einer solchen Struktur ist schwierig und nur mit viel Zeitaufwand zu bewerkstelligen. Für Algorithmen in der Bioinformatik wie Alignments sind die Tertiärstrukturen also von einer praktikablen Verwendung weit entfernt.

2.4 Bedeutung der Struktur

Die gebildete Struktur einer RNA ist von großer Bedeutung für ihre Funktion. So sorgt zum Beispiel die L-förmige Tertiärstruktur einer tRNA (Abbildung 2.4) für die Voraussetzungen einer erfolgreichen Interaktion mit den Ribosomen. Durch diesen entscheidenden Einfluss der Struktur resultiert auch die Annahme über eine stärkere Konserviertheit dieser, im Vergleich zur Basenkomposition einer RNA, während des Evolutionsprozesses. Aufgrund dieser Theorie ist es möglich durch Sequenz-Struktur-Alignments auch Sequenzen mit geringer Sequenzidentität zu vergleichen, um Gemeinsamkeiten festzustellen.

3 Alignment

Angenommen, es wird eine Anzahl von Zeichenketten betrachtet und Ziel ist es, dabei herauszufinden wie groß die Ähnlichkeit dieser ist, oder wie viele Manipulationen an einer Kette vorzunehmen sind, um sie in eine andere zu überführen, dann hilft ein Alignment bei der Lösung dieser Aufgabe.

3.1 Was versteht man unter Alignment

Aufgabe eines Alignments ist es also, die Ähnlichkeit oder Distanz von Zeichenketten, in Zusammenhang dieser Arbeit speziell RNA Sequenzen, zu berechnen. Zur Erstellung eines Alignments stehen im wesentlichen drei Edit-Operationen zur Verfügung: Einfügen, Löschen und die Substitution. Mit Hilfe dieser möglichen Operationen wird eine einzelne Position der Sequenz auf eine Position der anderen aligniert, wobei nur die Substitution direkt zwei Zeichen aus den Sequenzen aligniert. Um ein Alignment korrekt durchzuführen wird ein Symbol benötigt, mit dessen Hilfe sich die Einfüge- und Löschoperationen darstellen lassen. Zu diesem Zweck existiert das so genannte „Gap-Symbol“, welches durch ein – an der entsprechenden Stelle ausgedrückt wird. Das Einfügen lässt sich nun durch das Hinzufügen eines Gap-Symbols in die eine Sequenz, und das Löschen durch Einfügen des Symbols in die andere Sequenz beschreiben. Das Ergebnis eines Alignments sind am Ende also exakt gleich lange Sequenzen, in denen sich neben den ursprünglichen Zeichen noch eine beliebig große Anzahl an Gap-Symbolen befinden. Ein Beispiel eines solchen Alignments ist in Abbildung 3.1 dargestellt.

```
Seq1   AGUGGUCUAAGGCGCCAGACUCAAGUUCUGGUCUUCGAGAGA
Seq2   AA-U-GGUAGAACGAGAGCUUCCCAAGCUC-U-A----U-AC
Seq3   AA-U-GGUAGACUACUUAGCUACCACCUAA-G-A----U-GU
Seq4   AA-UGGAUAGGACAUAGGUCUUCUAAACCU-U-U----G-GU
Seq5   AGCUCGGUAGAGCGCUCGGCUCAUAACCGA-G-U----G-GU
```

Abbildung 3.1: Ein multiples Alignment bestehend aus 5 Sequenzen

Einfache Algorithmen zur Berechnung eines Alignments von zwei Sequenzen sind unter anderem der Needleman-Wunsch- und der Smith-Waterman-Algorithmus [NW70, SW81].

In Zusammenhang mit Alignments werden oft Schlüsselwörter wie Ähnlichkeit, Distanz, paarweises oder multiples Alignment erwähnt. Aus diesem Grund werden die Unterschiede und Bedeutungen dieser Begriffe nun kurz erläutert.

3.1.1 Ähnlichkeit und Distanz

Für die Bewertung eines Alignments stehen zwei mögliche Kriterien zur Verfügung, auf deren Basis die Bildung aufgebaut werden kann. Dabei handelt es sich zum Einen um die Distanz und zum Anderen um die Ähnlichkeit zwischen den Sequenzen.

Zur Bildung eines Alignments aufgrund der Distanz werden den einzelnen Operationen Kosten zugewiesen, die sie bei ihrer Anwendung verursachen. So kann man sich vorstellen, dass das Einfügen eines Gaps teurer ist als ein A auf ein C zu alignieren, oder gar ein A auf ein A. Bei der Berechnung wird also versucht die Gesamtkosten, welche sich durch die Summierung der Kosten aller nötigen Operationen ergeben, minimal zu halten.

Auf andere Weise lässt sich ein Alignment mittels Ähnlichkeiten bestimmen. Hierbei erhalten Paare von Zeichen einen Score, der die Ähnlichkeit (oder auch Mutationswahrscheinlichkeit) der beiden Zeichen zueinander ausdrücken soll. Das Paar „AA“ bekommt beispielsweise einen hohen Wert zugesprochen wohingegen „A-“ einen sehr niedrigen Wert erhält. Ziel eines Alignments auf Ähnlichkeitsbasis liegt demnach zufolge darin die maximale Ähnlichkeit für das gesamte Alignment zu erreichen.

3.1.2 Lokales und Globales Alignment

Algorithmen zur Bestimmung eines Alignments lassen sich aufteilen in die Bereiche des lokalen und globalen Alignments. Während das globale Alignment die Sequenzen von Anfang bis zum Ende hin aligniert, berechnen die lokalen Algorithmen ein Alignment, das innerhalb der Sequenzen Abschnitte findet, die sich besonders gut alignieren lassen. Während das lokale Alignment versucht weitere Abschnitte mit möglichst guten Alignments innerhalb der Sequenzen zu finden, sobald sich ein solcher Abschnitt nicht weiter vergrößern lässt, aligniert ein globaler Algorithmus weiter und nimmt damit eine Reduzierung des bisher erreichten Scores in Kauf.

Eine Beobachtung, die sich durch den Vergleich von lokalen und globalen Alignments ergibt zeigt, dass bei ausreichend hoher Sequenzidentität kein Unterschied zwischen den erreichten Alignments durch beide Verfahren mehr besteht, da das lokale Alignment den Gesamtscore durch Verlängerung der Teilsequenzen immer weiter erhöhen kann, bis das Ende beider Sequenzen erreicht ist.

3.1.3 Paarweise und multiple Alignments

Eine weitere Unterscheidung zwischen Alignments kann in der Anzahl der alignierten Sequenzen erfolgen. Für den Fall von nur zwei zu alignierenden Sequenzen spricht man von paarweisem Alignment. Sobald ein Alignment über mehrere Sequenzen erstellt wird, handelt es sich um ein multiples Alignment. Ein multiples Alignment hat dabei den Vorteil auf mehr nützliche Informationen zurückgreifen zu können aufgrund der größeren Anzahl zu betrachtender Sequenzen. Deshalb erreichen Algorithmen mit multiplen Alignments oft bessere Ergebnisse, als diejenigen die nur zwei Sequenzen bearbeiten. Ein Nachteil den multiple Alignments mit sich bringen liegt in dem Mehraufwand der benötigt wird

um ein solches Alignment zu erstellen. Die Komplexität des Problems steigt natürlich mit jeder weiteren Sequenz die aligniert werden muss, weshalb die Suche nach effizienten Algorithmen für das multiple Alignment große Aufmerksamkeit genießt. Von großer Bedeutung für die Qualität des Ergebnisses eines multiplen Alignments ist auch die Methode mit der die Zusammenführung der Sequenzen erfolgt. Hier stehen mehrere Methoden zur Verfügung, wie der bereits erwähnte progressive Ansatz. Andere Ansätze wären beispielsweise die Verwendung eines iterativen Algorithmus [HTHI95, Got96], der nach einzelnen Alignmentschritten das bisherige Ergebnis erneut betrachtet und verbessert, oder die Bildung des Alignments mit Hidden-Markov-Models [Edd95]. Tree-Coffee wird sich dabei auf einen progressiven Ansatz festlegen, weshalb die anderen Möglichkeiten hier keine größere Beachtung erhalten werden.

3.2 Welchen Nutzen haben Alignments

Anhand eines Alignments lässt sich also die Ähnlichkeit oder Distanz zwischen Sequenzen ermitteln. Mit den gefunden Homologien lassen sich so Rückschlüsse auf die Evolution der analysierten Sequenzen ziehen. Es liegt nahe, dass sehr ähnliche Sequenzen einen gemeinsamen Vorfahren besitzen, was wiederum ein Indiz für die Funktion einer neu entdeckten RNA sein kann. Alignments können also dazu verwendet werden für neu entdeckte RNAs möglichst nahe Verwandte zu finden, die bereits erforscht wurden, um einen ersten Anhaltspunkt für die Funktion dieser neuen RNA zu bekommen. Alignments sind daher von großer Bedeutung für die biologische Forschung speziell im Bereich der Gen- und Zellforschung.

4 Algorithmische und formale Grundlagen

Bevor nun eine genaue Beschreibung des Tree-Coffee Verfahrens erfolgt, ist es notwendig einige grundlegende formale Definitionen zu treffen, welche im Verlauf des Algorithmus eine Rolle spielen werden. Aus diesem Grund wird in diesem Kapitel das Fundament für die erfolgreiche formale Beschreibung von Tree-Coffee gelegt werden. Zusätzlich wird ein Modell vorgestellt, das dazu dient, die Ähnlichkeit zwischen Sequenz-Struktur-Paaren zu bestimmen, gefolgt von einem Einblick in den T-Coffee Algorithmus, aus dem die Ableitung von Tree-Coffee erfolgte.

4.1 Allgemeine Definitionen

Im Verlauf der letzten Kapitel wurden bereits einige Begriffe verwendet, welche nun eine formale Definition erhalten werden, die im weiteren Verlauf dieser Arbeit für diese Begriffe gelten. Zuerst wird konkretisiert was gemeint ist wenn im folgenden von einer Sequenz gesprochen wird.

Definition 4.1. Eine Sequenz S ist ein Wort über Σ^*, wobei gilt $\Sigma = \{A, C, G, U\}$. Dabei bezeichnet $S[i]$ das Zeichen an Position i der Sequenz.

Nachdem nun genau feststeht was unter einer Sequenz zu verstehen ist, kann die formale Beschreibung eines Alignments erfolgen.

Definition 4.2. Seien a und $b \in \Sigma^*$ dann bezeichnet das Paar $a^0, b^0 \in (\Sigma \cup \{-\})^*$ das Alignment von a, b wenn gilt:

1. $|a^0| = |b^0|$
2. $\nexists i \in \{1, ..., |a^0|\} : a_i^0 = b_i^0 = -$
3. a^0 beschränkt auf Σ ergibt genau a, gilt analog für b^0 und b

Nach der Definition des paarweisen Alignments, wird noch das multiple Alignment benötigt.

Definition 4.3. Ein multiples Alignment der Sequenzen $a^1, ..., a^N \in \Sigma^*$ kann durch eine Matrix $(A_{ij}) : 1 \leq i \leq N, 1 \leq j \leq K$ dargestellt werden, für die folgende Eigenschaften erfüllt sein müssen:

1. $A_{ij} \in (\Sigma \cup \{-\})$
2. $A_{i1} \cdots A_{iK}$ beschränkt auf $\Sigma = a^i$
3. $\nexists j : \forall i A_{ij} = -$

Die Definition eines multiplen Alignments zeigt, dass ein paarweises Alignment im formalen Sinne nur ein Spezialfall davon ist.

Definition 4.4. Betrachtet man die alignierten Sequenzen a^0, b^0 des Alignments $\mathcal{A}$, dann gilt $(i, j) \in \mathcal{A}$ (i und j realisieren eine Kante im Alignment) falls gilt $(a^0[i'] \neq - \wedge b^0[j'] \neq -)$, wobei i und j die ursprünglichen Positionen in den Eingabesequenzen darstellen.

Da es sich bei Tree-Coffee um einen Sequenz-Struktur-Alignment Ansatz handelt müssen noch einige Definitionen zur Struktur erfolgen.

Definition 4.5. Ein Arc a ist ein Paar $(i, j) \in \mathbb{N} \times \mathbb{N}$ für das gilt: $i < j$. i und j werden dabei als Enden eines Arcs bezeichnet, welche auch durch a^l und a^r symbolisiert werden können. Ein Arc stellt dabei die Verbindung zweier Basen einer Sequenz dar.

Mit dem Wissen über die Arcs kann nun die Sekundärstruktur P einer Sequenz S definiert werden.

Definition 4.6. Die Struktur P ist eine Menge von Arcs mit der Eigenschaft, dass keine der Positionen, die einen Arc bilden, in mehr als einem Arc vorkommen. Weiter wird vorausgesetzt, dass für 2 beliebig gewählte Arcs $(i_1, i_2) \in P$ und $(j_1, j_2) \in P$ eine der folgenden Bedingungen gilt:

1. $i_2 < j_1$ oder $j_2 < i_1$ (unabhängige Basenpaare)
2. $i_1 < j_1 < j_2 < i_2$ oder $j_1 < i_1 < i_2 < j_2$ (geschachtelte Basenpaare)

Aufgrund dieser Definition einer Struktur ist sichergestellt, dass es sich dabei um eine so genannte „Nested“ Struktur handelt, in der keine Überschneidung von Arc-Kanten vorkommen dürfen (Abbildung 4.1).

Definition 4.7. $S[i]$ wird als freie Base bezeichnet, wenn gilt: $(i, i') \notin P$ und $(i', i) \notin P$.

Die Verbindung zwischen einer Sequenz und ihrer Struktur ist dadurch wie folgt darstellbar:

Definition 4.8. Ein Tupel $\mathcal{S} = (S, P)$ bezeichnet ein Sequenz-Struktur-Paar

Nachdem nun diese grundlegenden Definitionen getroffen wurden ist es möglich einige spezifischere Dinge zu betrachten, die als Grundlage für Tree-Coffee dienen.

4.2 Die Edit Distanz

Zur erfolgreichen Berechnung eines Sequenz-Struktur-Alignments ist ein Ansatz notwendig, der die Ähnlichkeit zwischen zwei Strukturen messen kann. Ein solcher Ansatz, der dabei, sowohl auf den Basen der Sequenzen, als auch auf deren Strukturen arbeitet, wurde von Jiang et al. in dem Artikel: „A General Edit Distance between RNA Structures“ [JLMZ02], welcher im „Journal of computational Biology“ erschienen ist, vorgestellt. Um das genaue Verfahren für eines der dort vorgestellten Probleme geht es nun in diesem Teil.

Zur Darstellung einer Struktur stehen verschiedene Möglichkeiten zur Verfügung und die entsprechenden Methoden, um auf dieser Basis die Ähnlichkeit zu bestimmen. Da mehrere Algorithmen existieren, um die Distanz zwischen zwei Bäumen zu berechnen, zum Beispiel ein Ansatz von Shapiro und Zhang Anfang der 90er Jahre [SZ90], liegt es nahe, Strukturen mit Hilfe von Bäumen zu repräsentieren, was durch die *nested* Eigenschaft von Strukturen ermöglicht wird. Eine weitere Möglichkeit ist die Verwendung von stochastischen kontext-freien Grammatiken (Sakakibara et al. [SBH+94]). Anders als bei diesen beiden Verfahren wird in dem Ansatz von Jiang die Distanz nicht nur aufgrund der Strukturen berechnet, sondern auch die Nukleotidebene in die Bewertung mit aufgenommen.

4.2.1 Unterteilung der Strukturen

Benötigt werden drei Unterscheidungen zwischen den möglichen Strukturen: *plain, nested* und *crossing*. Beispiele für *nested* und *crossing* Strukturen sind bereits in Abbildung 4.1 zu sehen. Bei *plain* Strukturen handelt es sich um Strukturen, in denen keine Arcs existieren, also alle Basen der Sequenz frei sind.

Die Schwierigkeit Sequenz-Struktur-Paare zu vergleichen, nimmt mit der Komplexität der Struktur zu, wodurch sich sechs differenzierbare Problemstellungen der Tabelle 4.1 definieren lassen.

Der Fall *plain* $\Leftrightarrow$ *plain* ist dabei am einfachsten zu lösen und entspricht durch das Fehlen jeglicher Arcs einem einfachen Sequenzalignment. Bei Tree-Coffee werden aufgrund der Strukturdefinition maximal *nested* $\Leftrightarrow$ *nested* Fälle betrachtet werden, weshalb dies auch der einzige Fall ist, der zum Thema Edit Distanz behandelt wird.

4.2.2 Die möglichen Edit Operationen

Betrachtet werden müssen die möglichen Operationen, die dabei helfen sollen, die Distanz zweier Sequenz-Struktur-Paare zu bestimmen. Dabei muss darauf geachtet werden,

Abbildung 4.1: Vergleich Nested - Crossing Struktur

crossing	$\Leftrightarrow$	*crossing*
crossing	$\Leftrightarrow$	*nested*
crossing	$\Leftrightarrow$	*plain*
nested	$\Leftrightarrow$	*nested*
nested	$\Leftrightarrow$	*plain*
plain	$\Leftrightarrow$	*plain*

Tabelle 4.1: Komplexitätsklassen der Berechnung von Sequenz-Struktur-Paaren

sowohl die Operationen auf freien Basen, als auch die Operationen auf Basen, welche Teil eines Arcs sind, genau zu formulieren.

Für Basen die Teil einer Struktur sind, existieren folgende Operationen: *arc-match, arc-mismatch, arc-breaking, arc-altering* und *arc-removing*. Anschaulich können diese Operationen in Abbildung 4.2 betrachtet werden. Formal ergeben sich daraus die kommenden Definitionen.

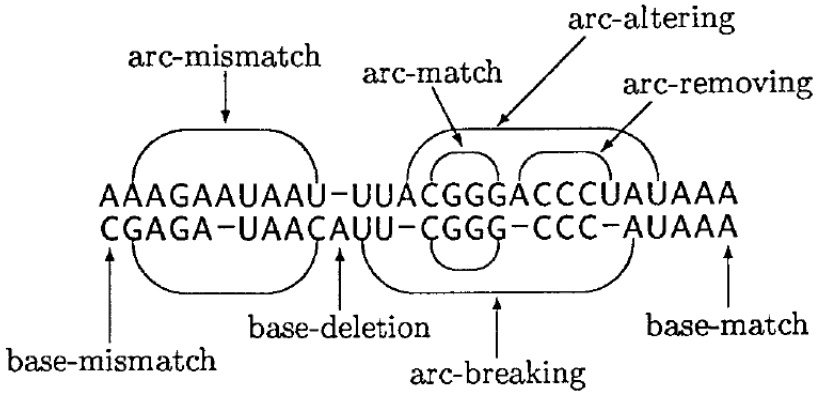

Abbildung 4.2: Mögliche Edit Operationen auf Sequenz-Struktur-Paaren
Quelle: [JLMZ02]

Definition 4.9. Gegeben sind die Arcs $(i_1, i_2) \in P_1$ und $(j_1, j_2) \in P_2$ und bei den Kanten (i_1, j_1) und (i_2, j_2) handelt es sich um realisierte Kanten, dann wird dies als *arc-match* bezeichnet, falls $S_1[i_1] = S_2[j_1] \wedge S_1[i_2] = S_2[j_2]$ gilt. Einen *arc-mismatch* erhält man, wenn mindestens eine der beiden Anforderungen nicht zutrifft.

Definition 4.10. Betrachtet man die realisierten Kanten (i_1, j_1) und (i_2, j_2) dann bezeichnet man dies als *arc-breaking* falls gilt:

$$((i_1, i_2) \in P_1 \wedge (j_1, j_2) \notin P_2) \vee ((i_1, i_2) \notin P_1 \wedge (j_1, j_2) \in P_2)$$

Biologisch kann die arc-breaking Operation dadurch erklärt werden, dass die Mutationen von Basen die Kräfte innerhalb der Struktur beeinflussen und deshalb die Kraft nicht mehr ausreicht, um die Bindung aufrecht zu erhalten.

Definition 4.11. Betrachtet man einen Arc $(i_1, i_2) \in P_1$, und wird $S_1[i_1]$ mit einer Base aus S_2 zum Beispiel $S_2[j_1]$, und $S_1[i_2]$ mit einem Gap aligniert, dann bezeichnet das eine *arc-altering* Operation, die die Base $S_1[i_2]$ löscht und dadurch den Arc (i_1, i_2) bricht. Die Folge daraus ist, dass die Base $S_1[i_1]$ ab sofort als freie Base betrachtet wird.

In der Evolution spiegelt diese Operation das Verschwinden einer Base, die Teil einer Bindung war, wider. Natürlich muss nicht zwangsweise nur eine der gebunden Basen verschwinden, was dazu führt, dass auch diese Option einer der möglichen Edit Operationen darstellt.

Definition 4.12. Falls beide Positionen eines Arcs $(i_1, i_2) \in P_1$ mit Gaps aligniert werden spricht man also von *arc-removing*.

Nachdem nun alle Operationen mit Arc-Involvierung definiert wurden, müssen noch die betrachtet werden, die auf Basen stattfinden können. Dabei gibt es vier mögliche Fälle die beobachtet werden können *base-match, base-mismatch, base-deletion* und *base-insertion*. In Abbildung 4.2 sind auch diese Operationen anschaulich dargestellt.

Definition 4.13. Aligniert man die freien Basen $S_1[i]$ mit $S_2[j]$ und es gilt $S_1[i] = S_2[j]$ spricht man von einem *base-match*. Sollte diese Bedingung nicht erfüllt sein handelt es sich um einen *base-mismatch*.

Definition 4.14. Wird eine freie Base $S_1[i]$ mit einem Gap aligniert ist dies eine *base-deletion*. Analog bezeichnet man die Alignierung von $S_2[j]$ mit einem Gap als *base-insertion*.

4.2.3 Kosten der Edit Operationen

Für die vorangegangenen Operationen werden natürlich noch Kosten benötigt, um mit ihrer Hilfe rechnen zu können. Da hier von Distanzen ausgegangen wird, lässt sich direkt festlegen, dass ein *base-* respektive *arc-match* keine Kosten verursacht. Anders verhält es sich bei den verbliebenen Operationen welche die folgenden Kosten erhalten:

- *base-mismatch*: w_m
- *base-deletion/insertion*: w_d
- *arc-mismatch*: $\frac{w_{am}}{2}$ oder w_{am} in Abhängigkeit der beiden beteiligten Arcs. Gilt entweder $S_1[i_1] \neq S_2[j_1]$ oder $S_1[i_2] \neq S_2[j_2]$ betragen die Kosten $\frac{w_{am}}{2}$. Gelten beide Bedingungen sind die Kosten $2 * \frac{w_{am}}{2} = w_{am}$.
- *arc-breaking*: w_b
- *arc-altering*: w_a

- *arc-removing*: w_r

Zusätzlich wird angenommen, dass $w_r \geq w_d$. Durch Kenntnis über die entstehenden Kosten und möglichen Edit Operationen kann jetzt also der Ansatz zur Berechnung erstellt werden.

4.2.4 Distanzberechnung für den Fall Nested ⇔ Nested

Bevor ein effizienter Algorithmus für diese Berechnung aufgestellt werden kann, bedarf es einer kleinen Änderung an der Anzahl der möglichen Kosten, welche durch folgende Annahme erreicht werden kann:

$$2w_a = w_b + w_r$$

Dadurch wird es ermöglicht, die Fälle *arc-altering* und *arc-removing* nur mit Hilfe von *arc-breaking* und *base-deletion* auszudrücken, wodurch diese beiden Fälle „vernachlässigt“ werden können.

- Eine *arc-altering* Operation ist äquivalent zu einer *arc-breaking* Operation in Addition mit einer *base-deletion*. Dies liefert die Kosten: $\frac{w_r - w_b}{2}$
- Jede *arc-removing* Operation besteht aus einer *arc-breaking* und zwei *base-deletion* Operationen, was auch wieder zu den Kosten von $\frac{w_r - w_b}{2}$ führt.

Wird nun eine Base gelöscht, welche Teil eines Arcs ist, und nur in diesem Fall, können anstelle der eigentlichen *base-deletion* Kosten, die soeben eingeführten Kosten verwendet werden. Aufgrund dieser Maßnahme, müssen in der Rekursion nur noch die Fälle *arc-match, arc-mismatch* und *arc-breaking* in Zusammenhang mit Arcs betrachtet werden. Um nun eine effiziente Berechnung zu ermöglichen, muss für Basen unterschieden werden, ob es sich dabei um eine freie Base, oder das Ende eines Arcs, handelt. Zu diesem Zweck wird der Begriff der Inzidenz definiert.

Definition 4.15. Die Base $S[i]$ ist inzident genau dann wenn $i = a^l \vee i = a^r$ erfüllt ist.

Damit lässt sich eine Funktion $\psi_1(i)$ wie folgt definieren (und analog $\psi_2(j)$):

Definition 4.16. $\psi_1(i) = \begin{cases} 1 & \text{falls } i \text{ inzident zu einem Arc} \in P_1 \\ 0 & \text{sonst} \end{cases}$

Hinzukommend wird eine Funktion benötigt, welche unterscheidet ob zwei Basen identisch sind oder nicht.

Definition 4.17. $\chi(i,j) = \begin{cases} 1 & \text{falls } S_1[i] \neq S_2[j] \textit{ (mismatch)} \\ 0 & \text{sonst } \textit{(match)} \end{cases}$

Dies schafft die Möglichkeit, die *arc-breaking* Kosten zu gleichen Teilen unter den beiden inzidenten Basen eines Arcs aufzuteilen, welche sich auf alle vorhandenen Operationen anwenden lässt, und damit diese Kosten für die einzelnen Fälle verursachen:

- *base-deletion* von $S_1[i]$: $w_d + \psi_1(i)(\frac{w_r}{2} - w_d)$
- *base-match* oder *base-mismatch*: $\chi(i,j)w_m + (\psi_1(i) + \psi_2(j))\frac{w_b}{2}$

Es ist durch diese Kostendefinitionen nicht weiter nötig die Fälle *arc-altering, arc-removing* und *arc-breaking* explizit zu betrachten, da ihre Kosten in die Basenoperationen integriert wurden. Daraus lässt sich nun die Rekursionsgleichung aufstellen:

Für alle $1 \leq i \leq i' \leq |S_1|$ und $1 \leq j \leq j' \leq |S_2|$,

$$DP(i,i';j,j') = min \begin{cases} DP(i,i'-1;j,j') + w_d + \psi_1(i')(\frac{w_r}{2} - w_d), \\ DP(i,i';j,j'-1) + w_d + \psi_2(j')(\frac{w_r}{2} - w_d), \\ DP(i,i'-1;j,j'-1) + \chi(i',j')w_m + (\psi_1(i') + \psi_2(j'))\frac{w_b}{2}, \\ DP(i,i_1-1;j,j_1-1) + DP(i_1+1,i'-1;j_1+1,j'-1) \\ \quad +(\chi(i_1,j_1) + \chi(i',j'))\frac{w_{am}}{2}, \\ \quad\quad \text{falls } i \leq i_1, j \leq j_1, (i_1,i') \in P_1 \text{ und } (j_1,j') \in P_2 \end{cases}$$

Durch diese Rekursion kann die Berechnung der Distanz für zwei Sequenz-Struktur-Paare $\mathcal{S}_1, \mathcal{S}_2$ in einer Laufzeit von $O(n^2m^2)$, wobei $n = |S_1|$ und $m = |S_2|$, absolviert werden.

Die genaue Initialisierung der Berechnungsmatrix wird hier vorerst ausser Acht gelassen, da diese in Zusammenhang mit dem Tree-Coffee Algorithmus noch ausführlich behandelt werden wird.

Jiang et al. haben hier ein effizientes Verfahren zur Distanzberechnung vorgestellt, das mit einer leichten Modifikation problemlos von der Minimierung der Distanz zur Maximierung der Ähnlichkeit adaptiert werden kann. Dieser Aspekt wird bei Tree-Coffee zur Anwendung kommen, da dort mit Ähnlichkeiten gearbeitet wird.

4.3 T-Coffee

Einen großen Einfluss auf die Beschaffenheit von Tree-Coffee hatte der von Cedric Notredame, Desmond G. Higgins und Jaap Heringa entwickelte Algorithmus, genannt „T-Coffee“, um ein multiples Sequenzalignment zu erstellen [NHH00]. Angestrebt wurde bei T-Coffee die Verbesserung der Ergebnisse durch Vermeidung von frühen Fehlern während der Bildung des multiplen Alignments, unter der Prämisse, dies nicht mittels eines erheblichen Mehraufwandes bezüglich der Laufzeit zu erreichen. Um Tree-Coffee zu verstehen, ist es also nötig, die Mechanismen von T-Coffee zu kennen, weshalb eine Betrachtung des T-Coffee Algorithmus in dieser Arbeit nicht fehlen darf.

Sieht man den T-Coffee Algorithmus, so lässt sich dieser in zwei größere Bearbeitungsschritte unterteilen. Zuerst findet eine Informationssammlung und Weiterverarbeitung dieser Informationen statt, um die Eigenschaften aller paarweisen Alignments zu erfassen. Diese Daten werden dann in mehreren Schritten erweitert und zusammengeführt, um anhand derer letztendlich ein progressives Alignment durchzuführen.

4.3.1 Sammlung und Weiterverarbeitung der paarweisen Daten

Um die benötigten Daten zu erhalten, werden für alle $M(M-1)/2$ möglichen Sequenzpaare Alignments berechnet:

- Ein globales Alignment mit Hilfe des ClustalW Programmes [THG94].
- Die zehn besten lokalen Alignments, die sich nicht überschneiden. Diese Berechnung wird durch „Lalign“ übernommen, welches Teil des FASTA Softwarepaketes ist [PL88].

Das liefert für jedes Alignment eine Liste mit realisierten Kanten, die in einer primären Bibliothek gesammelt werden, von denen jedoch einige wichtiger sind als andere. Indem nun für alle Kanten der Bibliothek ein Gewicht berechnet wird, das sich in Verbindung mit ihrem Einfluss auf das Alignment ausdrückt, kann diese Information in die Bewertung der realisierten Kanten eingearbeitet werden. Um ein Gewicht für die Kanten des Alignments zu erhalten wird eine durchschnittliche Identität, der alignierten Positionen des gesamten Alignments berechnet. Der Vorteil besteht in der Einfachheit dieser Methode bei dennoch ausreichender Effektivität. Jede Kante des Alignments erhält also ein Gewicht entsprechend dieser berechneten Identität nach dieser Methode:
$\forall : 1 \leq i \leq |A|, 1 \leq j \leq |B| :$

$$\text{Gewicht}(i,j) = \begin{cases} \text{Identität}(\mathcal{A}) & \text{falls } (i,j) \in \mathcal{A} \end{cases}$$

Es stehen nun also die gewichteten Datensätze für jede Sequenzpaarung zur Verfügung, bestehend aus den Daten für das globale und denjenigen für das lokale Alignment, welche es zunächst gilt zusammen zu führen.

Die Kombinierung der Daten erfolgt dabei durch einfache Addition der Gewichte.
$\forall : 1 \leq i \leq |A|, 1 \leq j \leq |B| :$

$$\text{Gewicht}_{Kombiniert} = \begin{cases} \text{Gewicht}_{lokal}(i,j) + \text{Gewicht}_{global}(i,j) & \text{falls } (i,j) \in \mathcal{A}_{lokal} \\ & \text{und } (i,j) \in \mathcal{A}_{global} \\ \text{Gewicht}_{lokal}(i,j) & \text{falls } (i,j) \in \mathcal{A}_{lokal} \\ & \text{und } (i,j) \notin \mathcal{A}_{global} \\ \text{Gewicht}_{global}(i,j) & \text{falls } (i,j) \in \mathcal{A}_{global} \\ & \text{und } (i,j) \notin \mathcal{A}_{lokal} \\ 0 & \text{sonst} \end{cases}$$

Mit diesen paarweisen Bibliotheken ließe sich jetzt theoretisch schon das multiple Alignment erstellen, jedoch kann mit Hilfe einer Erweiterung der Bibliotheken die Qualität dieser Daten noch weiter verbessert werden.

Um diese Erweiterung durchzuführen werden nun Triplets von Sequenzen analysiert. Angenommen es soll eine Transformation der Gewichte für die Kanten des Alignments von *Sequenz A* und *Sequenz B* über die *Sequenz C* durchgeführt werden. Hierzu werden die Kanten

$$\begin{aligned} &(i,j) : 1 \leq i \leq |A| \wedge 1 \leq j \leq |B| \\ &(i,k) : 1 \leq i \leq |A| \wedge 1 \leq k \leq |C| \\ &(k,j) : 1 \leq k \leq |C| \wedge 1 \leq j \leq |B| \end{aligned}$$

betrachtet. Für die Kante (i,j) beträgt das Gewicht W_1 und die Kanten (i,k) und (k,j) sind gewichtet mit W_2 und W_3. Zu W_1 wird nun der Wert $min(W_2, W_3)$ addiert womit die Kante (i,j) über die Sequenz C transformiert wurde. Siehe auch Abbildung 4.3 zur Veranschaulichung dieses Triplet-Ansatzes. Um die Transformation abzuschließen, müssen nun alle Kanten in A,B über alle verbleibenden Sequenzen transformiert werden, und dieser Vorgang für alle weiteren möglichen Sequenzkombinationen wiederholt werden.

Auf diese Weise spiegeln die Gewichte der Kanten nicht nur die Informationen der beiden alignierten Sequenzen wieder, sondern beinhalten auch die Kombinationen zu den verbleibenden Sequenzen des multiplen Alignments. Mit diesen transformierten Werten, in der erweiterten Bibliothek, kann dann begonnen werden das multiple Alignment zu erstellen.

4.3.2 Erstellung des multiplen Alignments

Progressives Alignieren bedeutet, dass, entsprechend einer gegebenen Reihenfolge, die Sequenzen oder bereits berechnete Alignments sukzessive miteinander aligniert werden, bis ein multiples Alignment produziert wurde, in dem alle Sequenzen vertreten sind. Um

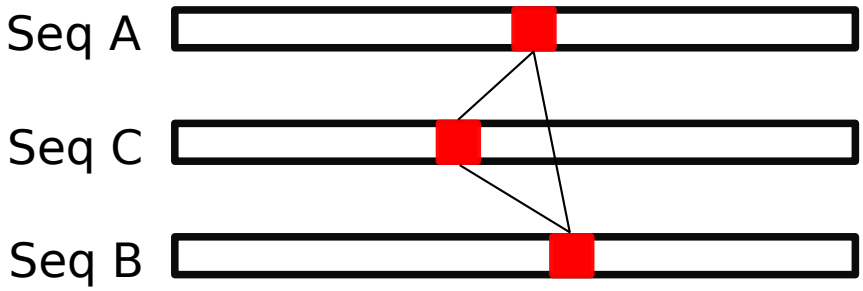

Abbildung 4.3: Transformation einer Kante aus Sequenzen A und B über Sequenz C

diese Reihenfolge der Alignierungen festzulegen, wird anhand der Distanzen zwischen allen Sequenzen ein phylogenetischer Baum berechnet, in dem die Sequenzen entsprechend dieser Distanzen aufsteigend angeordnet sind. Die Ableitung eines solchen Baumes anhand der Distanzen wird von einem „Neighbour-Joining“ Algorithmus [SN87] übernommen und liefert einen Baum (Abbildung 4.4), welcher besagt, dass zuerst die Sequenzen A und B aligniert werden, bevor zu diesem Alignment dann Sequenz C hinzualigniert wird. Im Verlauf eines progressiven Alignments können einmal eingeführte Gaps in den

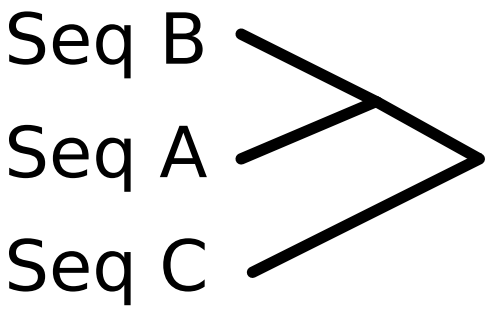

Abbildung 4.4: phylogenetischer Baum für 3 Sequenzen

Sequenzen nicht mehr bewegt oder entfernt werden, weshalb die Analyse der paarweisen Alignments und die Konsistenztransformation essentiell wichtig sind, um dadurch verursachte Fehler, welche sich durch den ganzen Prozess hindurchziehen würden, schon in den ersten Schritten zu vermeiden. Ein weiterer Vorteil der Vorverarbeitung besteht darin, dass für die Bildung des multiplen Alignments keine weiteren Parameter wie Gapkosten mehr nötig sind, da diese bereits in die Kantengewichte eingearbeitet wurden.

Während der einzelnen Teilschritte muss darauf geachtet werden, ob es sich in dem aktuellen Schritt um ein Alignment zweier einzelner Sequenzen handelt, oder bereits ein schon berechnetes Alignment mit mehreren Sequenzen involviert ist, da in den Bibliotheken nur die Gewichte der paarweisen Alignments gespeichert sind. Damit Alignments, welche bereits mehrere Sequenzen enthalten, aligniert werden können, müssen die durchschnittlichen Gewichte der einzelnen Alignmentspalten verwendet werden, womit der endgültigen Erstellung des multiplen Alignments nichts weiter im Wege steht.

5 Der Tree-Coffee Algorithmus

Durch die Betrachtung der vorangegangenen Grundlagen, zu denen einige biologische Grundkenntnisse, ein Verständnis, über die Eigenschaften von Alignments, und elementare formale Dinge gehören, erfolgt nun die detaillierte Einführung in das von Tree-Coffee verwendete Verfahren zur Berechnung eines multiplen Sequenz-Struktur-Alignments. Es sei anzumerken, dass es sich hier um eine erste voll funktionale Version dieses Algorithmus handelt, welcher sich aktuell noch in der Entwicklung befindet. Ziel ist es anhand dieser Arbeit die generelle Funktionalität dieses Ansatzes zu validieren und gegebenenfalls einige erste Schwächen aufzuzeigen, die bis zu einer Finalität des Programms noch auszubessern sind. In diesem Kapitel wird nun diese erste Version eingeführt werden.

Bei Tree-Coffee handelt es sich um einen konsistenzbasierten Algorithmus, welcher sich fester Strukturen, die vom Benutzer bereitgestellt werden müssen, bedient und auf Basis dieser bereitgestellten Sequenz-Struktur-Paare ein multiples Alignment erstellt. Zu diesem Zweck wird ein ähnlicher Programmablauf verwendet, wie es bei T-Coffee eingeführt wurde. Es wird eine paarweise Analyse aller möglichen Kombinationen der Eingabedaten durchgeführt werden und zum Zwecke der konsistenzbasierten Idee, eine Gewichtung der Kanten erfolgen. Nach diesem Prozess wird durch einen verwandten Triplet-Ansatz, wie ihn auch T-Coffee verwendet, eine Transformation der bis dahin gesammelten Daten stattfinden, bevor diese Informationen dann dazu verwendet werden ein progressives Alignment durchzuführen. Eine einfache schematische Darstellung dieses Ablaufs kann der Abbildung 5.1 entnommen werden. Dieser Plan spiegelt auch in groben Zügen den Aufbau dieses Kapitels wieder, in dem entlang dieses Pfades die einzelnen Schritte ausführlich präsentiert werden.

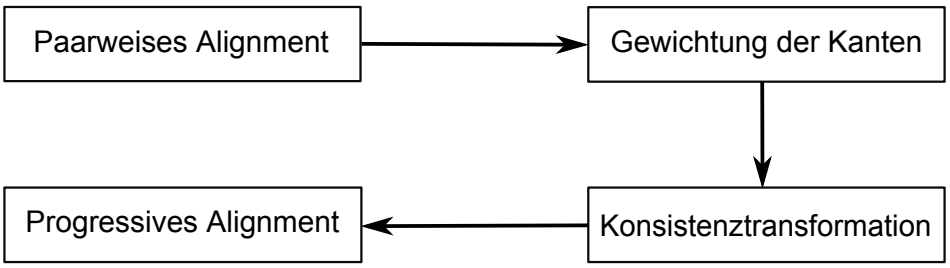

Abbildung 5.1: Vereinfachter Prozessablaufplan von Tree-Coffee

Abbildung 5.2: Beispiel der Arcindizierung

5.1 Eine effiziente Indizierung der Arcs

Für den weiteren Verlauf wird es benötigt, den vorhandenen Arcs einer Struktur P auf geschickte Weise einen Index zuzuordnen, mit dem die weitere Arbeit erleichtert wird. Zur Erinnerung - Es handelt sich um *nested* Strukturen, aufgrund deren diese Art der Indizierung überhaupt erst ermöglicht wird. Gewünscht ist eine Indizierung die abhängig von dem rechten Ende eines Arcs ist. Hierfür wird der Ausdruck a_h^1 definiert, der den Arc mit dem Index h aus der Struktur P_1 symbolisiert.

Definition 5.1. a_h^1 bezeichnet den h-ten Arc $(i_1, i_2) \in P_1$ für $h \in \mathbb{N}$. Für alle Paare von Arcs $a_h^1 = (i_1, i_2) \in P_1$ und $a_g^1 = (j_1, j_2) \in P_1$ muss eine, und nur eine, der folgenden Bedingungen zutreffen:

1. $h > g : i_2 > j_2$
2. $h < g : i_2 < j_2$
3. $h = g : i_2 = j_2 \wedge i_1 = j_1$

Da es sich hier um *nested* Strukturen handelt, müssen die linken Enden der Arcs in dieser Definition nicht explizit betrachtet werden, da per Definition von *nested* die notwendigen Anforderungen an die öffnende Position eines Arcs, in Abhängigkeit der Schließenden, abgedeckt sind. Aufgrund der Tatsache, dass zu einem Zeitpunkt maximal die beiden Strukturen P_1 und P_2 betrachtet werden, ist es möglich die Darstellung eines Arcs zu vereinfachen: $a_h = a_h^1 \in P_1$ und $b_h = b_h^2 \in P_2$. So lassen sich die öffnenden und schließenden Positionen eines Arcs h mit (a_h^l, a_h^r), respektive (b_h^l, b_h^r), umschreiben.

Definition 5.2. $|P_1|$ drückt die Anzahl der Arcs $\in P_1$ aus. Für den Index h eines Arcs gilt demzufolge: $1 \leq h \leq |P_1|$.

Zur Veranschaulichung, dieses Verfahrens der Indizierung, kann in Abbildung 5.2 ein Beispiel dazu eingesehen werden.

Die Erstellung dieser Strukturinformationen bedarf nur linearer Laufzeit und ist somit für die Gesamtlaufzeit des Algorithmus unbedeutend, liefert aber eine einfache Möglichkeit im weiteren Verlauf, Bezug auf einen bestimmten Arc der Struktur P zu nehmen, ohne zwangsläufig die Enden des Arcs mitzuführen.

5.2 Paarweise Sequenzalignments

Zu Beginn des Algorithmus steht die Sammlung aller benötigter Informationen, der paarweisen Alignments, im Vordergrund. Es werden für alle $\frac{M(M-1)}{2}$ möglichen Paarungen aus $\mathcal{S} = \{\mathcal{S}_1, ..., \mathcal{S}_M\}$ Alignments erstellt. Zu diesem Zweck werden zwei Algorithmen eingeführt, mit denen sich globale Alignments von Teilsequenzen (bis hin zu den vollständigen Sequenzen), der aktuell betrachteten Paarung, berechnen lassen. Dabei wird das Ziel sein, eine maximale Ähnlichkeit dieser Segmente zu realisieren, weshalb eine leicht veränderte Fassung der von Jiang et al. vorgestellten Rekursion verwendet wird [JLMZ02]. Die Beiden, im folgenden vorgestellten, Algorithmen werden für diesen Schritt verwendet:

- Der Insidealgorithmus
- Der Outsidealgorithmus

Als Basis dieser Berechnungen werden die von Klein und Eddy eingeführten „RIBOSUM“ Scores [KE03] verwendet. Dabei handelt es sich um Substitutionsmatrizen, die sowohl für die Substitution zweier Basen i, j, als auch zweier Basenpaare i_1, i_2 und j_1, j_2, Werte bereit stellt, mit denen in diesem Fall gearbeitet werden kann.

Auf diese Weise kann eine abstrakte Definition, der Ähnlichkeit eines Alignments A erfolgen:

Definition 5.3. $SIM_{\mathcal{S}_1}^{\mathcal{S}_2}(A)$ beschreibt die Ähnlichkeit des Alignments A, die sich aus der Summe, der Ähnlichkeiten der möglichen Edit Operationen, welche zur Bildung des Alignments verwendet wurden, zusammen setzt.

Ziel der Beiden hier vorgestellten Algorithmen wird es sein, die nötigen Verfahren zu definieren, die benötigt werden, um die maximale Ähnlichkeit eines Sequenz-Struktur-Alignments zu berechnen. Die Funktion dieser beiden Algorithmen, wird dazu dienen Bereiche innerhalb und ausserhalb von Arcs zu alignieren. Diese Alignments werden dazu benötigt die Gewichtung der Kanten durchzuführen, und somit die konsistenzbasierte Idee umzusetzen.

Mit diesem Wissen kann nun der Erste, dieser beiden Algorithmen, behandelt werden.

5.2.1 Der Insidealgorithmus

Betrachtet werden die Tupel $\mathcal{S}_1 = (S_1, P_1)$ und $\mathcal{S}_2 = (S_2, P_2)$, unter der Prämisse eine Menge von $|P_1| * |P_2|$ Teilalignments zu berechnen. Ziel ist es dabei, für jedes Arcpaar $(a_h, b_g) : 1 \leq h \leq |P_1|$ und $1 \leq g \leq |P_2|$, die maximale Ähnlichkeit zu bestimmen, mit der Bedingung, dass (a_h^l, b_g^l) und (a_h^r, b_g^r) als Kanten realisiert werden. Anders ausgedrückt, die beiden Arcs bilden strukturelle Kanten im Alignment. Aus diesem Umstand lässt sich erkennen, dass zwischen zwei Arten von realisierten Kanten im Alignment unterschieden werden kann.

- strukturelle Kante: $\{(i, i', j, j') \,|(i, i', j, j')$ ist ein *arc-(mis)match* $\}$

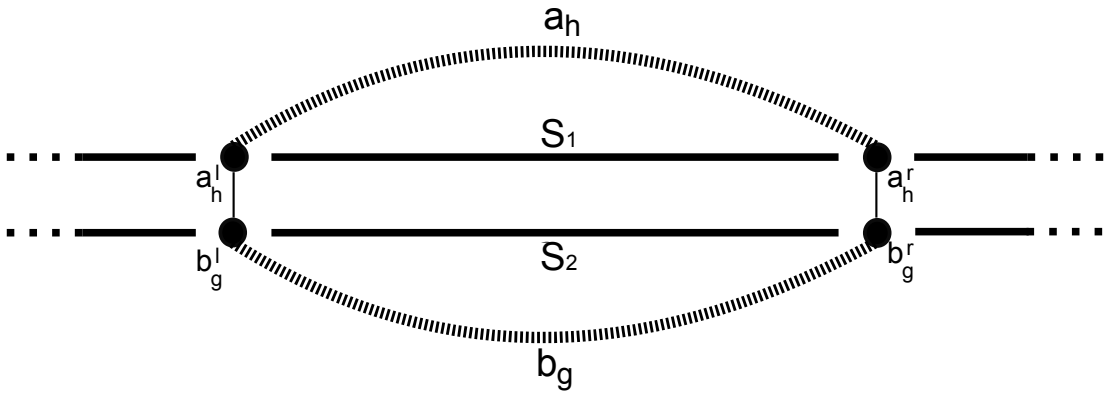

Abbildung 5.3: Segment eines Alignmentschrittes begrenzt durch zwei Arcs

- sequentielle Kante: $\{(i,j)\,|i$ und j beteiligen sich an keiner strukturellen Kante$\}$

Aus der Eigenschaft mit diesem Algorithmus die Bereiche, welche sich innerhalb einer Struktur befinden, zu alignieren resultiert, auch der Name: Insidealgorithmus.

Formale Beschreibung aller nötigen Teilalignments

Betrachtet werden die Strukturen P_1,P_2 und die Anzahl der jeweils darin enthaltenen Arcs, so werden $|P_1| * |P_2|$ Alignments benötigt, um jede mögliche Paarung von (a_h, b_g) zu erreichen. Ein Teilalignment, welches von zwei Arcs begrenzt ist, lässt sich wie folgt definieren:

Definition 5.4. Ein Teilalignment $\mathcal{A}_{teil}$ begrenzt durch die Arcs $a_h \in P_1$ und $b_g \in P_2$, vereinfacht durch $\mathcal{A}_{teil}(a_h, b_g)$ ausgedrückt, ist ein Alignment der Teilsequenzen $[a_h^l..a_h^r]$ und $[b_g^l..b_g^r]$, wobei die beiden Arcs strukturelle Kanten realisieren.

Wichtig für diese partiellen Alignments ist die Reihenfolge, in der sie berechnet werden. Die verwendete Rekursion zur Erstellung eines Alignments benötigt die Werte solcher Teilalignments, falls innerhalb des betrachteten Intervalls Strukturen enthalten sind. Um dies verständlich zu machen betrachten wir ein Beispiel.

Beispiel 5.1. Gegeben sind die Arcs: $\{1,..,3\} \in P_1$ und $\{1,2\} \in P_2$. Für die Berechnung des Alignments begrenzt durch a_2 und b_2, muss der Wert für das Alignment von $[a_1^l..a_1^r], [b_1^l..b_1^r]$ bereits bekannt sein.

Definition 5.5. $\mathcal{A}_{teil}(a_h, b_g)$ kann aligniert werden, genau dann wenn gilt:
$\nexists \mathcal{A}_{teil}(a'_h, b'_g) : (1 \leq h' < h \wedge 1 \leq g' < g)$, das nicht bereits berechnet wurde.

Nach Festlegung der Reihenfolge, in der die einzelnen Teilalignments berechnet werden, können nun die notwendigen Datenstrukturen, zur Speicherung der berechneten Daten, betrachtet werden, gefolgt von einigen, für die Rekursion wichtigen, Parameterdefinitionen, bevor dann endgültig die Rekursion aufgestellt werden kann.

Benötigte Speicherstrukturen

Zur Speicherung der Ergebnisse werden zwei Matrizen benötigt. Eine dieser Matrizen wird für die Berechnung der Teilalignments, und die Andere für die maximalen Ähnlichkeiten dieser, verwendet werden. Aus der Anzahl der Arcs beider Strukturen und der Längen $n_1 = |S_1|, n_2 = |S_2|$ der Sequenzen ergeben sich für die benötigten Matrizen die Größen $|P_1| \times |P_2|$ und $n_1 + 1 \times n_2 + 1$ (kann unabhängig der Größe der Teilalignments verwendet werden).

Definition 5.6. Zur Speicherung der Ähnlichkeiten der Teilalignments wird folgende Matrix $L : |P_1| \times |P_2|$ definiert:
$L_{h,g} = max\left\{SIM_{S_1}^{S_2}(\mathcal{A}_{teil}(a_h, b_g))\right\}$

Definition 5.7. Die für die Berechnung der Teilalignments benötigte $n_1 + 1 \times n_2 + 1$ Matrix M ist definiert durch:
$$M_{i,j}^{a_h,b_g} = max\left\{ SIM_{S_1}^{S_2}(A) \;\middle|\; \begin{array}{c} \text{A ist Teilalignment von}[a_h^l + 1..i] \\ \text{und } [b_g^l + 1..j] : i < a_h^r \text{ und } j < b_g^r \end{array} \right\}$$

Da maximal die Ähnlichkeiten gesamter Teilalignments während der Rekursion verwendet werden, ist nur eine konsistente Speicherung der L-Matrix nötig. Somit wird für die Berechnung von Alignments nur quadratischer Speicherplatz benötigt, was in Anbetracht der möglichen Sequenzlängen einen großen Vorteil darstellt.

Neue Kostendefinitionen und Matchscores

In dem von Jiang et al. vorgestellten Verfahren wurden bereits alle nötigen Kosten der Edit Operationen aufgelistet, da hier jedoch dieser Ansatz auf ein ähnlichkeitsbasiertes Modell umgestellt wird, wird die Gelegenheit genutzt einige dieser Kosten, zur vereinfachten Darstellung neu festzulegen. Konkret wird es sich hierbei um die Kosten für *arc-removing, arc-breaking* und *base-deletion* handeln. Die Änderungen belaufen sich also wie folgt:

1. $\frac{w_r}{2} = \beta_r$. β_r entspricht also den halben *arc-removing* Kosten.
2. $\frac{w_b}{2} = \beta_b$. β_b entspricht den halben *arc-breaking* Kosten.
3. $w_d = \gamma$. γ entspricht den Kosten für eine *base-deletion/insertion*.

Die Kosten sind hier nicht im Sinne von positiven Werten zu verstehen, sondern stellen negative Ähnlichkeiten dar.

Da nun Ähnlichkeiten berechnet werden, ist es nötig die Kosten für *arc-match/mismatch* und *base-match/mismatch* anders zu definieren. Zu diesem Zweck werden die Funktionen $\sigma(i, j)$ und $\tau(i, i'j, j')$ eingeführt.

Definition 5.8. Die Funktion $\sigma(i, j)$ liefert einen Wert für die Ähnlichkeit der Basen $S_1[i]$ und $S_2[j]$, welcher dem entsprechenden Eintrag dieser Basenkombination in der „RIBOSUM“ Matrix entspricht.

Definition 5.9. $\tau(i, i', j, j')$ liefert den Wert für die Ähnlichkeit der Basenpaare $S_1[i]S_1[i'] \times S_2[j]S_2[j']$, entsprechend dem Wert aus der „RIBOSUM“ Matrix.

Mit diesen abschließenden Definitionen ist es nun möglich die Rekursion zur Berechnung eines Alignments, basierend auf der Idee des Insidealgorithmus, zu formulieren.

Initialisierung der Berechnungsmatrix

Berechnet werden soll $\mathcal{A}_{teil}(a_h, b_g)$, demzufolge sind gegeben: Die Positionen (a_h^l, a_h^r) und (b_g^l, b_g^r), als Grenzen des Teilalignments, die Segmente der beiden Sequenzen $S_1[a_h^l + 1..a_h^r - 1]$ und $S_2[b_g^l + 1..b_g^r - 1]$, die Parameter β_r und γ und letztlich die Matrix M, welche zur Berechnung benötigt wird. (für die Definition von $\psi_1(i)$ siehe Definition 4.16 auf Seite 22.)

Durchgeführt werden soll nun die Initialisierung der Matrix M, für das aktuelle Teilalignment, als Voraussetzung der Berechnung durch die Rekursionsgleichung. Für die ersten Einträge in M ergeben sich folgende Werte:

$$M_{a_h^l, b_g^l} = 0$$

$$\forall i : a_h^l < i < a_h^r \Rightarrow M_{i, b_g^l} = M_{i-1, b_g^l} + \gamma + \psi_1(i)\beta_r$$

$$\forall j : b_g^l < j < b_g^r \Rightarrow M_{a_h^l, j} = M_{a_h^l, j-1} + \gamma + \psi_2(j)\beta_r$$

Nach dieser Initialisierung kann damit begonnen werden, die maximale Ähnlichkeit für dieses Teilalignment zu berechnen, anhand der folgenden Rekursionsgleichung.

Die Rekursion

Betrachtet wird weiterhin $\mathcal{A}_{teil}(a_h, b_g)$, mit den dazugehörigen Grenzen und Parametern. Zusätzlich werden für die Rekursion der Parameter β_b und die Funktionen $\sigma(i, j)$ und $\tau(i, i', j, j')$ benötigt, welche bereits definiert wurden. Die rekursive Berechnung folgt dieser Rekursionsgleichung:

$\forall i', j' : a_h^l < i' < a_h^r \wedge b_g^l < j' < b_g^r :$

$$M_{i',j'} = max \begin{cases} M_{i'-1,j'} + \gamma + \psi_1(i')\beta_r & \\ M_{i',j'-1} + \gamma + \psi_2(j')\beta_r & \\ M_{i'-1,j'-1} + \begin{cases} (\psi_1(i') + \psi_2(j'))\beta_b & \text{falls min. eine der Basen inzident} \\ \sigma(i', j') & \text{sonst} \end{cases} & \\ M_{i_1-1,j_1-1} + \underbrace{L_{a,b}}_{\text{wo } a=(i_1,i') \text{ und } b=(j_1,j')} & \begin{array}{l} \text{falls } (i_1, i') \in P_1, (j_1, j') \in P_2 \\ \text{und für } i_1, j_1 \text{ gilt:} \\ i_1 > a_h^l \wedge j_1 > b_g^l \end{array} \end{cases}$$

Aus dieser Rekursion wird auch ersichtlich, weshalb die Reihenfolge der betrachteten Teilalignments von Bedeutung ist. Für den 4. Fall werden die Ähnlichkeiten von möglicherweise im aktuellen Teilalignment eingeschlossenen Strukturen benötigt.

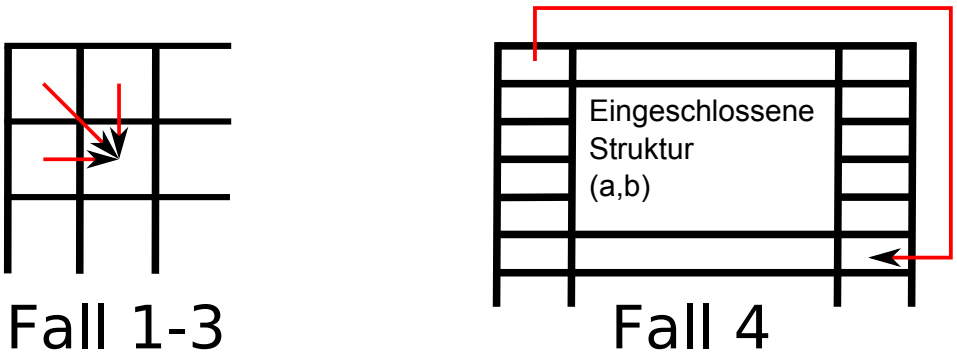

Abbildung 5.4: Ursprung der Scores der 4 Rekursionsfälle

Nachdem mit dieser Rekursion die Ähnlichkeit für die Teilsegmente $[a_h^l + 1..a_h^r - 1]$ aligniert mit $[b_g^l + 1..b_g^r - 1]$ berechnet wurde, ergibt sich für $\mathcal{A}_{teil}(a_h, b_g)$ folgende Ähnlichkeit, welche in L_{a_h,b_g} gespeichert wird:

$$L_{a_h,b_g} = M_{a_h^r-1,b_g^r-1} + \tau(a_h^l, a_h^r, b_g^l, b_g^r)$$

In Abbildung 5.4 sind die Zugriffe auf bereits berechnete Werte der Matrix M anschaulich dargestellt, um ein besseres Verständnis der Rekursionsgleichung zu gewährleisten.

Ähnlichkeit der gesamten Sequenzen

Der Insidealgorithmus ermöglicht auch die Berechnung der maximalen Ähnlichkeit, der uneingeschränkten Tupel $\mathcal{S}_1$ und $\mathcal{S}_2$. Da bereits alle möglichen Teilalignments berechnet wurden, kann die Rekursion nun auf die volle Länge der Sequenzen angewendet werden. Diese Ähnlichkeiten werden für alle Sequenzpaarungen gespeichert, da auf deren Basis im späteren Verlauf der phylogenetische Baum, welcher für das progressive Alignment benötigt wird, berechnet werden kann.

Komplexitäten des Insidealgorithmus

Für die komplette Berechnung des Algorithmus werden $|P_1| * |P_2| + 1$ Alignments berechnet, welche sich aus der Anzahl aller möglichen Teilalignments und eines Gesamtalignments zusammensetzen. Gegeben die Sequenzlängen $n_1 = |S_1|$ und $n_2 = |S_2|$ kann für die Laufzeit eines Teilalignments (respektive eines Gesamtalignments) die folgende Laufzeit angenommen werden:

- Es müssen pro Alignment $O(n_1 * n_2)$ Einträge mit Hilfe der Rekursion berechnet werden. Werden die Längen der Sequenzen auf $N = \frac{n_1+n_2}{2}$ vereinfacht, um ein durchschnittliches Maß dafür zu erhalten, ergibt sich somit eine Laufzeit von $O(N^2)$ pro partiellem Alignment.

Insgesamt werden von diesen Teilalignments $|P_1|*|P_2|$, und ein Gesamtalignment, benötigt. Die Anzahl der Arcs einer Struktur lassen sich mit $|P_1| \leq \frac{n_1}{2}$ und $|P_2| \leq \frac{n_2}{2}$ nach oben hin

begrenzen, da pro Arc zwei Positionen einer Sequenz benötigt werden, und per Definition keine Position Teil mehrerer Arcs sein kann. Hieraus ergibt sich:

- Die Gesamtlaufzeit des Insidealgorithmus ergibt sich aus maximal: $(\frac{n_1}{2} * \frac{n_2}{2}) * O(N^2) + O(N^2)$. Wird wieder ein allgemeiner Wert für die Anzahl der Arcs einer durchschnittlichen Sequenzlänge verwendet, lässt sich dies vereinfachen zu: $\frac{N^2}{4} * O(N^2) + O(N^2)$, woraus eine Gesamtkomplexität von $O(N^4)$ entsteht.

Die Speicherkomplexität des Insidealgorithmus verhält sich, wie bereits bei der Vorstellung, der benötigten Berechnungsmatrizen erwähnt, quadratisch im Verhältnis zur Länge der Eingabedaten. Dies wird im Speziellen durch die Berechnungen der unterschiedlichen Teilalignments ermöglicht, da aufgrund der damit bereits errechneten und gespeicherten Werte für die Berechnung weiterer Alignments nur quadratischer Speicher benötigt wird. Somit kann der Insidealgorithmus ein Sequenz-Struktur-Alignment mit einem Speicherplatzbedarf in $O(N^2)$ bewerkstelligen.

5.2.2 Der Outsidealgorithmus

Nachdem der Insidealgorithmus die Bereiche unterhalb von Strukturen aligniert hat, liegt nahe, was mit Hilfe des Outsidealgorithmus erreicht werden soll. Diese beiden Verfahren werden im späteren Verlauf in Kombination zur Kantengewichtung verwendet werden.

In Abbildung 5.5 können innerhalb der blauen Markierung, die Teile der Sequenzen erkannt werden, welche mit Hilfe des Outsidealgorithmus berechnet werden sollen. Es wird sich diesmal, anders als beim Insidealgorithmus, um zwei Teilalignments pro in Frage kommender Arcpaarung (a_h, b_g) handeln. Dies wird dazu führen, dass unterschieden werden muss, ob das zu alignierende Segment sich vor oder hinter den Arcgrenzen befindet. Aufgrund dieser Eigenschaft werden zwei Rekursionen nötig sein, Eine für das vordere, und die Andere für das hintere Segment. Zuerst muss jedoch formal betrachtet werden, wie genau diese Teilalignments, in Abhängigkeit der beteiligten Strukturen, begrenzt werden.

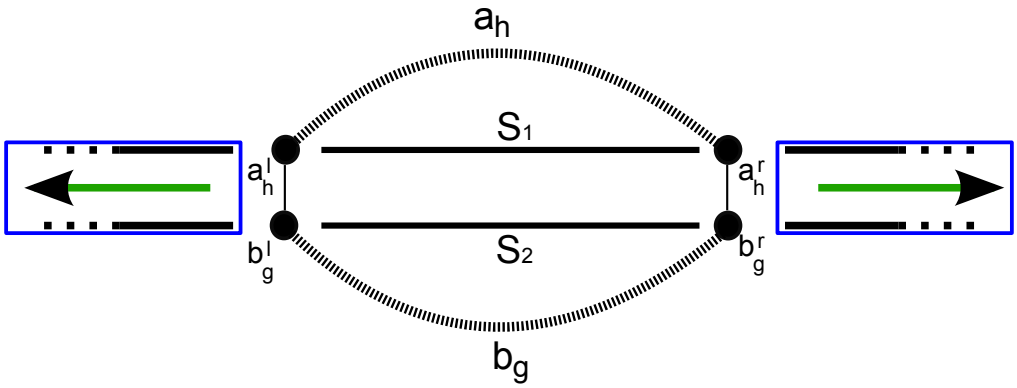

Abbildung 5.5: Zu alignierende Bereiche des Outsidealgorithmus

Benötigte Teilsegmente

Exemplarisch werden wieder die Paarungen der Arcs (a_h, b_g) und deren Grenzen (a_h^l, a_h^r), (b_g^l, b_g^r) betrachtet, um die benötigten Teilalignments auf dieser Basis zu definieren. Berechnet werden müssen aufgrund dessen die Teilalignments $\mathcal{A}^l_{outside}(a_h, b_g)$ und $\mathcal{A}^r_{outside}(a_h, b_g)$, welche Alignments der folgenden Teilsequenzen symbolisieren:

Definition 5.10. $\mathcal{A}^l_{outside}(a_h, b_g)$ ist das Alignment der Segmente $[1..a_h^l - 1] \in S_1$ und $[1..b_g^l - 1] \in S_2$.

Definition 5.11. $\mathcal{A}^r_{outside}(a_h, b_g)$ ist das Alignment der Segmente $[a_h^r + 1..|S_1|] \in S_1$ und $[b_g^r + 1..|S_2|] \in S_2$.

Nachdem nun diese beiden Arten definiert wurden, kann schon ein kleiner Einblick erfolgen, auf welche Art und Weise diese Alignments berechnet werden.

Präfix- und Suffixalignment

Der Outsidealgorithmus unterscheidet zwischen diesen beiden Arten des Alignments, was aufgrund der späteren Zusammenführung, der beiden partiellen Alignments, zu einer maximalen Gesamtähnlichkeit, der Bereiche ausserhalb der betrachteten Arcs, notwendig ist. Die Rekursionen für diese beiden Typen werden sich nur geringfügig unterscheiden, da Präfix- und Suffixalignment lediglich besagen, ob von einer Position ausgehend in Richtung des Endes oder des Anfangs der Sequenzen aligniert wird. In diesem Fall wird wie in Abbildung 5.5 dargestellt ein Suffixalignment, von den linken Enden der Arcs ausgehend bis zum Beginn der Sequenzen, und ein Präfixalignment ausgehend von den rechten Ende der Arcs bis zum Sequenzende erstellt werden.

Nötige Matrizen für den Outsidealgorithmus

Für diesen Algorithmus werden drei Matrizen benötigt, um die Berechnung der Alignments durchzuführen und die summierten Ähnlichkeiten aller Paarungen zu speichern. (In einer praktischen Implementation werden nur zwei benötigt, da es in den Berechnungsmatrizen zu keinen Überschneidungen kommt.) Auch hier werden die Matrizen unabhängig der Länge der Segmente, welche aligniert werden, definiert und erhalten die Größe, die notwendig wäre, um ein Alignment über beide kompletten Sequenzen durchzuführen.

Definition 5.12. Die $n_1 + 1 \times n_2 + 1$-Matrix $D^{a_h,b_g}_{\leq i,j}$ ist wie folgt beschrieben:

$$D^{a_h,b_g}_{\leq i,j} = max \left\{ SIM^{S_2}_{S_1}(A) \;\middle|\; \begin{array}{l} \text{A ist Teilalignment von} [a_h^r + 1..i] \\ \text{und } [b_g^r + 1..j] : i \leq |S_1| \text{ und } j \leq |S_2| \end{array} \right\}$$

Definition 5.13. Die $n_1 + 1 \times n_2 + 1$-Matrix $D^{a_h,b_g}_{\geq i,j}$ ist wie folgt beschrieben:

$$D^{a_h,b_g}_{\geq i,j} = max \left\{ SIM^{S_2}_{S_1}(A) \;\middle|\; \begin{array}{l} \text{A ist Teilalignment von} [i..a_h^l - 1] \\ \text{und } [j..b_g^l - 1] : i \geq 1 \text{ und } j \geq 1 \end{array} \right\}$$

Diese beiden Matrizen dienen den Berechnungen der Teilalignments. Fehlt also noch die Matrix für die Speicherung der Gesamtähnlichkeiten.

Definition 5.14. Für die $|P_1| \times |P_2|$ Matrix U_{a_h,b_g} gilt:
$$U_{a_h,b_g} = max \left\{ SIM_{S_1}^{S_2}(A) \;\middle|\; \begin{array}{l} \text{A sind die Teilalignments von} [1..a_h^l - 1], [1..b_g^l - 1] \\ \text{und } [a_h^r + 1..|S_1|], [b_g^r + 1..|S_2|] \end{array} \right\}$$

Identisch zum Insidealgorithmus, müssen die beiden Hilfsmatrizen zur Berechnung der Alignments nicht dauerhaft gespeichert werden. Sie werden nach der Berechnung für den Bereich ausserhalb eines Arcpaares, gelöscht. Dies impliziert wiederum einen benötigten Speicherplatz in $O(n^2)$, durch die konsistente Speicherung der U-Matrix.

Initialisierung der beiden Berechnungsmatrizen

Zur Berechnung der Teilalignments $\mathcal{A}^r_{outside}(a_h, b_g)$ und $\mathcal{A}^l_{outside}(a_h, b_g)$ müssen die Matrizen $D_{\leq|S_1|,|S_2|}$ und $D_{\geq 1,1}$ auf folgende Weisen initialisiert werden:
Für $D_{\leq|S_1|,|S_2|}$ berechnet sich die Initialisierung durch:

$$D_{a_h^r,b_g^r} = 0$$

$$\forall i : a_h^r < i \leq |S_1| \Rightarrow D_{i,b_g^r} = D_{i-1,b_g^r} + \gamma + \psi_1(i)\beta_r$$
$$\forall j : b_g^r < j \leq |S_2| \Rightarrow D_{a_h^r,j} = D_{a_h^r,j-1} + \gamma + \psi_2(j)\beta_r$$

Dies war die Initialisierung zur Berechnung eines Präfixalignments, ausgehend von den rechten Strukturenden. Diese ist äquivalent zur entsprechenden Initialisierung der M-Matrix des Insidealgorithmus, mit Ausnahme der Start- und der Endpunkte.

Betrachtet wird nun die Initialisierung der Matrix zur Berechnung des Suffixalignments von $[1..a_h^l - 1]$ und $[1..b_g^l - 1]$.
$D_{\geq 1,1}$ wird mit Hilfe dieser Gleichungen initialisiert:

$$D_{a_h^l,b_g^l} = 0$$

$$\forall i : a_h^l > i \geq 1 \Rightarrow D_{i,b_g^l} = D_{i+1,b_g^l} + \gamma + \psi_1(i)\beta_r$$
$$\forall j : b_g^l > j \geq 1 \Rightarrow D_{a_h^l,j} = D_{a_h^l,j+1} + \gamma + \psi_2(j)\beta_r$$

Berechnung der beiden Teilalignments

Zuerst soll $\mathcal{A}^r_{outside}(a_h, b_g)$ berechnet werden. Das bedeutet es wird die Matrix $D_{\leq|S_1|,|S_2|}$ für die folgenden Berechnungen verwendet. In Bezug zu den Positionen a_h^r und b_g^r wird die folgende Rekursionsgleichung verwendet.

$\forall i', j' : a_h^r < i' \leq |S_1| \wedge b_g^r < j' \leq |S_2| :$

$$D_{\leq i',j'} = max \begin{cases} D_{\leq i'-1,j'} + \gamma + \psi_1(i')\beta_r & \\ D_{\leq i',j'-1} + \gamma + \psi_2(j')\beta_r & \\ D_{\leq i'-1,j'-1} + \begin{cases} (\psi_1(i') + \psi_2(j'))\beta_b & \text{falls min. eine der Basen inzident} \\ \sigma(i', j') & \text{sonst} \end{cases} & \\ D_{\leq i_1-1,j_1-1} + \underbrace{L_{a,b}}_{\text{wo } a=(i_1,i') \text{ und } b=(j_1,j')} & \begin{array}{l} \text{falls } (i_1, i') \in P_1, (j_1, j') \in P_2 \\ \text{und für } i_1, j_1 \text{ gilt:} \\ i_1 > a_h^r \wedge j_1 > b_g^r \end{array} \end{cases}$$

Hier gilt, ähnlich der Initialisierung, dass es sich bei dieser Rekursion um die beinahe Identische, wie sie für den Insidealgorithmus verwendet wird, handelt. Neben den veränderten Wertebereichen für i' und j', haben sich die Anforderungen an die Positionen i_1 und j_1 im 4. Fall der Rekursion geändert. Es werden nur diejenigen Arcs betrachtet, welche komplett innerhalb von $\mathcal{A}^r_{outside}$ liegen. Arcs die innerhalb von diesem Intervall enden, aber bereits davor geöffnet wurden, werden in dieser Rekursion ausgeschlossen und später noch bei der Kombination der beiden Teilalignments betrachtet. Weiterhin lässt sich am 4. Fall der Rekursion noch feststellen, dass auf die Werte $L_{a,b}$ zurückgegriffen wird. Diese Tatsache impliziert, dass eine Berechnung des Outsidealgorithmus in der Regel erst dann durchgeführt werden kann, wenn bereits die Teilalignments aller Arcpaarungen mit Hilfe des Insidealgorithmus erstellt wurden.

Bis jetzt wurde nur die maximale Ähnlichkeit der Sequenzteile, welche sich rechts von a_h^r und b_g^r befinden, berechnet. Fehlt also noch der Bereich links von a_h^l und b_g^l. Zur Bestimmung von $\mathcal{A}^l_{outside}$ wird die nächste gezeigte Rekursion in Verbindung mit der Hilfsmatrix $D_{\geq 1,1}$ verwendet:
$\forall i', j' : a_h^l > i' \geq 1 \wedge b_g^l > j' \geq 1 :$

$$D_{\geq i',j'} = max \begin{cases} D_{\geq i'+1,j'} + \gamma + \psi_1(i')\beta_r & \\ D_{\geq i',j'+1} + \gamma + \psi_2(j')\beta_r & \\ D_{\geq i'+1,j'+1} + \begin{cases} (\psi_1(i') + \psi_2(j'))\beta_b & \text{falls min. eine der Basen inzident} \\ \sigma(i', j') & \text{sonst} \end{cases} & \\ D_{\geq i_1+1,j_1+1} + \underbrace{L_{a,b}}_{\text{wo } a=(i',i_1) \text{ und } b=(j',j_1)} & \begin{array}{l} \text{falls } (i', i_1) \in P_1, (j', j_1) \in P_2 \\ \text{und für } i_1, j_1 \text{ gilt:} \\ i_1 < a_h^l \wedge j_1 < b_g^l \end{array} \end{cases}$$

Dies ist nun die Rekursion für das Suffixalignment der Segmente $[1..a_h^l - 1]$ und $[1..b_g^l - 1]$. Dementsprechend dieser umgekehrten Rekursion bezeichnen die Positionen i_1 und j_1, nicht wie bisher bei den Präfix-basierten Rekursionen, die Enden eines Arcs, sondern in diesem Fall die öffnenden Positionen. Auch hier werden nur diejenigen Strukturen

betrachtet, welche sich komplett innerhalb der Segmente befinden, wie bereits bei der vorherigen Präfixrekursion des Outsidealgorithmus.

Aufgrund der Ausnahme von Strukturen, welche nicht komplett innerhalb dieser zweier Segmente liegen, müssen nun bei der Kombination der beiden Teilalignments $\mathcal{A}^r_{outside}$, $\mathcal{A}^l_{outside}$ zu einer maximalen Gesamtähnlichkeit der Bereiche ausserhalb von a_h,b_g, diese Arcs noch berücksichtigt werden. Diese Kombination der Teilalignments soll nun erläutert werden.

Bestimmung der Gesamtähnlichkeit und Maximierung über Arcs

Nach der Berechnung der beiden Teilalignments muss noch der Wert für U_{a_h,b_g} bestimmt werden, welcher die maximale Ähnlichkeit der Bereiche ausserhalb dieser beiden Arcs darstellt. Aufgrund der fehlenden Betrachtung von Strukturen, die nicht vollständig in einem der Teilbereiche enthalten sind, muss auf diese nun Rücksicht genommen werden, um einen korrekten Wert zu bestimmen. Zu diesem Zweck wird der Vergleichsoperator $\prec$ eingeführt, mit dem sich eine partielle Ordnung der Arcs einer Struktur darstellen lässt.

Definition 5.15. Für zwei Arcs $\in P$ gilt $(a^l_h, a^r_h) \prec (a^l_{h'}, a^r_{h'})$ falls, und nur falls $a^l_{h'} < a^l_h < a^r_h < a^r_{h'}$ erfüllt ist.

Mit diesem Operator lässt sich für U_{a_h,b_g} folgende Gleichung aufstellen:

$$U_{a_h,b_g} = max \begin{cases} \underbrace{max}_{a_h \prec a_{h'}, b_g \prec b_{g'}} \left\{ U_{a_{h'},b_{g'}} + \tau(a_{h'}, b_{g'}) + D_{\geq a^l_{h'}+1, b^l_{g'}+1} + D_{\leq a^r_{h'}-1, b^r_{g'}-1} \right\} \\ D_{\geq 1,1} + D_{\leq |S_1|,|S_2|} \end{cases}$$

Das Maximum für den Outsidescore von a_h, b_g ergibt sich also anhand dieser Gleichung, entweder durch die beiden Teilalignments bis zum Anfang, respektive Ende, der Sequenzen, oder aus der Summe von Segmenten dieser Teilalignments in Verbindung mit dem Outsidescore, von oberhalb verlaufenden Strukturen, und dem Beitrag die Arcs $a_{h'}, b_{g'}$ zu alignieren. Eine genaue Aufteilung dieser Summe kann in Abbildung 5.6 verfolgt werden.

Durch diese Berechnung, des finalen Outsidescores einer Paarung a_h, b_g, erfolgt auch die verpflichtende Reihenfolge, in der die Paarungen betrachtet werden müssen. Durch den benötigten Zugriff auf die Werte von oberhalb verlaufenden Arcs, welche per Definition einen größeren Index besitzen, als die unter ihnen Verlaufenden, lässt sich festlegen, dass mit der Paarung $a_{|P_1|}, b_{|P_2|}$ begonnen werden muss. Gefolgt von der Betrachtung aller Paare mit niedrigerem Index bis hin zu a_1, b_1.

Komplexitätsanalyse

Auch für den Outsidealgorithmus gilt, dass insgesamt $O(N^2)$ mögliche Kombinationen von Arcpaarungen betrachtet werden müssen. Dabei werden für jedes Paar folgende Berechnungen nötig:

- Ein Präfixalignment, das $O(N^2)$ Zeit benötigt.

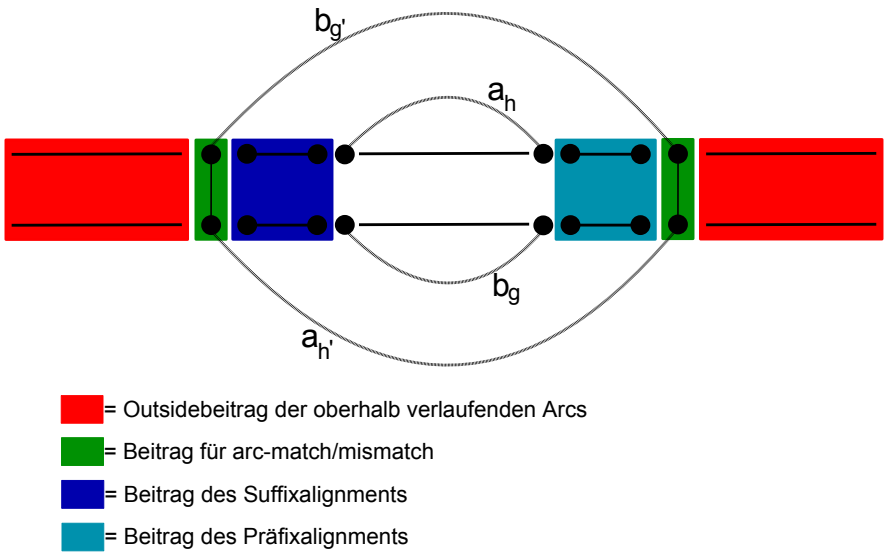

Abbildung 5.6: Betrachtete Bereiche während der Arc-Maximierung

- Ein Suffixalignment, ebenso in $O(N^2)$.
- Die Maximierung über oberhalb verlaufende Strukturen. Das sind wiederum $O(N^2)$ Arckombinationen.

Diese Schritte für ein Arcpaar summieren sich zu einer Gesamtlaufzeit von $O(3 * N^2)$. Insgesamt ergibt sich also für die Laufzeit $O(N^2 * 3N^2) = O(N^4)$.
Auch der Outsidealgorithmus kann, aufgrund der verwendeten Aufteilung in Teilalignments, sehr platzeffizient arbeiten. Durch Berechnung der Teilalignments und der Speicherung des Gesamtwerts für eine Arckombination wird für einen Berechnungsschritt wieder nur quadratischer Speicher benötigt. Das bedeutet der Speicherbedarf des Outsidealgorithmus bewegt sich, wie der des Insidealgorithmus, nur in $O(N^2)$.

5.3 Gewichtung der Kanten

Die vorgestellten Algorithmen, zur Berechnung von paarweisen Alignments, sollen nun dazu verwendet werden, für alle Sequenzkombinationen Kantengewichte aller realisierbaren Kanten zu bestimmen. Die Haupteigenschaft eines solchen Kantengewichts besteht darin, einer Kante ein Maß der Qualität zuzuordnen, welche sie in einem vollständigen Alignment besitzt. Diese Gewichtung ist Hauptbestandteil eines Konsistenz-basierten Alignments und hat demzufolge großen Einfluss auf die Ergebnisse eines Algorithmus. Bei T-Coffee bedeutete dies, jeder realisierten Kante des Alignments ein Gewicht entsprechend der Sequenzidentität der alignierten Sequenzen zuzuweisen. Jede nicht realisierte Kante erhielt demnach kein Gewicht.

Dieser Ansatz wird von Tree-Coffee, in dieser Form, nicht verfolgt werden. Der Grund dafür liegt in der Eigenschaft des Sequenz-Struktur-Alignments. Für eine Sequenz sind

mehrere Strukturen wichtig, weshalb es bei der Gewichtung von Nachteil wäre nur denjenigen Strukturkanten ein Gewicht zuzuweisen, die im bestmöglichen paarweisen Alignment realisiert wurden. Daraus resultierend wird hier die Idee verfolgt **jeder** möglichen Kante ein Gewicht, entsprechend der maximalen Ähnlichkeit des Alignments, in dem sie realisiert wird, zu geben.

Ziel dieser Kantengewichtung wird sein für alle Sequenzpaare $\mathcal{S}_1, \mathcal{S}_2$ eine Bibliothek von Gewichten zu erstellen, die sowohl sequentielle, als auch strukturelle Kanten, umfassen. Zu diesem Zweck werden die folgenden Notationen eingeführt:

Definition 5.16. $WL^{seq}_{S_1,S_2}$ ist die Bibliothek sequentieller Kantengewichte für die Sequenzen (S_1, S_2), wobei $WL^{seq}_{S_1,S_2}$ eine Funktion von $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ darstellt.

Definition 5.17. $WL^{str}_{P_1,P_2}$ ist die Bibliothek struktureller Kantengewichte für die Strukturen (P_1, P_2), wobei $WL^{str}_{P_1,P_2}$ eine Funktion von $\mathbb{N}^4 \rightarrow \mathbb{R}$ darstellt.

Definition 5.18. $WL_{\mathcal{S}_1,\mathcal{S}_2}$ ist ein Paar bestehend aus: $WL^{seq}_{S_1,S_2}$ und $WL^{str}_{P_1,P_2}$

Anhand dieser Definitionen lässt sich erkennen, dass es ein Verfahren zum Gewichten von strukturellen, und eines für das Gewichten der sequentiellen Kanten, geben muss. Im folgenden wird die Berechnung der strukturellen Kantengewichte für ein Tupel $\mathcal{S}_1, \mathcal{S}_2$ vorgestellt.

Berechnung struktureller Kantengewichte

Es sollen also Gewichte für alle möglichen strukturellen Kanten vergeben werden. Zur Erinnerung: Eine strukturelle Kante bezeichnet ein Quadrupel (i, i', j, j') für das gilt $S_1[i]$ wird aligniert mit $S_2[j]$ und entsprechend $S_1[i']$ mit $S_2[j']$. Zusätzlich muss die Bedingung erfüllt sein $(i, i') \in P_1 \wedge (j, j') \in P_2$. Demnach werden die Gewichte der strukturellen Kanten während des progressiven Alignments als Wert der Ähnlichkeit für einen *arc-match/mismatch* verwendet.

Bei $WL^{str}_{P_1,P_2}$ wird es sich deshalb um eine Matrix mit $|P_1| \times |P_2|$ Einträgen handeln, um für jedes Arcpaar (a_h, b_g) das benötigte Gewicht zu beinhalten. Die Berechnung der Werte für diese Einträge gestaltet sich aufgrund der gesammelten Informationen durch den Inside- und Outsidealgorithmus sehr einfach.

- Der Insidescore für das Arcpaar (a_h, b_g) beschreibt die maximale Ähnlichkeit für den Bereich innerhalb dieser beiden Strukturen in Addition mit dem Wert um diese Strukturen zu alignieren.

- Der Outsidescore für das entsprechende Paar beschreibt den bestmöglichen Wert für das Alignment der Bereiche, die sich links und rechts ausserhalb der Arcs befinden.

Da es sich bei Tree-Coffee um ein Sequenz-Struktur-Alignment handelt, möchte man noch die Möglichkeit haben, Einfluss auf die Bevorzugung von strukturellen Kanten zu nehmen. In der Theorie ist es möglich, dass die Summe zweier sequentieller Kanten das

Alignment minimal verbessert, als wenn diese Beiden als strukturelle Kanten realisiert werden, obwohl ein Alignment mit diesen strukturellen Kanten biologisch gerechtfertigter wäre. Zu diesem Zweck wird der Parameter ω eingeführt:

Definition 5.19. ω bezeichnet ein Maß zur Verstärkung struktureller Kanten, um Einfluss auf die Konserviertheit der Eingabestrukturen zu nehmen.

Um nun das Gewicht für (a_h, b_g) zu erhalten, den Wert des besten Alignments, in dem diese Kante realisiert wird, unter Berücksichtigung des Wertes ω zur Stärkung/Schwächung struktureller Kanten, müssen nun die beiden Werte addiert und mit dem Faktor ω multipliziert werden, woraus sich für $(a_h^l, a_h^r, b_g^l, b_g^r)$ folgendes Gewicht ergibt:

$$wa(a_h^l, a_h^r, b_g^l, b_g^r) = \omega * (U_{a_h,b_g} + L_{a_h,b_g})$$

Laufzeit

Die aufgebrachte Zeit für die Berechnung der strukturellen Kantengewichte ist vergleichsweise, zur Erstellung eines Alignments, gering. Es müssen hierbei maximal $\frac{n_1}{2} * \frac{n_2}{2}$ Einträge für $WL^{str}_{P_1,P_2}$ betrachtet werden. Jeder dieser Einträge kann in konstanter Zeit berechnet werden, da alle nötigen Werte bereits vorliegen. Aufgrund dessen wird hier nur eine quadratische Laufzeit von $O(N^2)$ benötigt.

Berechnung sequentieller Kantengewichte

Die benötigte Gewichtung sequentieller Kanten gestaltet sich ungleich schwerer, als die der strukturellen Kanten. Prinzipiell müssen für $n_1 = |S_1| \times n_2 = |S_2|$ Kanten, die bestmöglichen Alignments bestimmt werden, in denen diese realisiert werden. Würde man hierfür nur die Alignments bis zur Position $(i-1, j-1)$ und ab Position $(i+1, j+1)$ in Addition mit dem *base-match/mismatch* der Kante (i, j) heranziehen, ließe sich sich die gesamte Berechnung in quadratischer Zeit durchführen. Jedoch würde hierbei die mögliche Alignierung von Strukturen, die oberhalb dieser Positionen verlaufen vernachlässigt werden. Aus diesem Grund muss, ähnlich des Outsidealgorithmus, noch eine Maximierung über diese Strukturen erfolgen, um das beste Alignment zu bewerten. Diese Tatsache wird die Berechnung weiterer Teilalignments mit sich ziehen, welche die Laufzeit der Kantengewichtung verändern wird.

Zur Reduzierung dieses Aufwandes wird die Anzahl, der zu betrachtenden Kanten, mit Hilfe einer biologisch gerechtfertigten Einschränkung reduziert, die im folgenden erläutert wird.

Die maximale Kantenschrägheit

Es wird der Begriff der Kantenschrägheit eingeführt. Diese Kantenschrägheit bezeichnet die Distanz zwischen $S_1[i]$ und $S_2[j]$. Der Wert $\Delta(i, j)$ dafür ergibt sich durch:

Definition 5.20. $\Delta(i, j) = |i - j|$

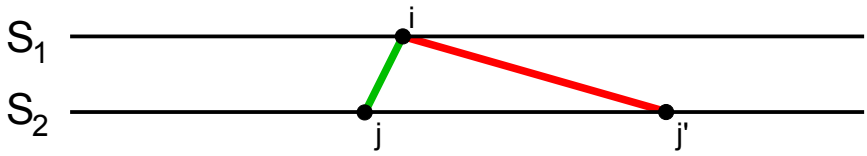

Abbildung 5.7: Illustration der maximalen Kantenschrägheit

Mit Hilfe dieser Definition kann der Parameter δ_{max} festgelegt werden, womit sich die Funktion $\theta(i,j)$ definieren lässt:

Definition 5.21. $\theta(i,j) = \begin{cases} 1 & \text{falls } \Delta(i,j) \leq \delta_{max} \\ 0 & \text{sonst} \end{cases}$

Diese Funktion wird später anzeigen, ob eine Kante gewichtet wird, oder den Wert $-\infty$ erhält. (Aufgrund der benötigten Zeit für die Gewichtung.) Abbildung 5.7 verdeutlicht diese Entscheidung und illustriert den Begriff der Kantenschrägheit.

Wird der Parameter δ_{max} in ausreichender Größe, welche von den Längen der Sequenzen abhängt, festgelegt ergeben sich daraus keine negativen Einflüsse auf die Korrektheit des Alignments. Angenommen es werden zwei Sequenzen der Länge 100 betrachtet, so ist es biologisch sehr unwahrscheinlich, dass $S_1[10]$ mit $S_2[80]$ eine Kante im Alignment bildet. Dieser Fall könnte nur dann eintreten, wenn sehr viele Positionen eingefügt oder gelöscht werden würden, was auf einen sehr großen Evolutionszeitraum verweisen würde, aber dem Sinn des Vergleichs von verwandten Sequenzen widerspricht. Aus diesem Grund werden für die Bildung des Alignments nicht alle möglichen Kanten nötig sein, und die Berechnung eines Kantengewichts kann für diese Kanten ausgesetzt werden.

Berechnung des Kantengewichts ohne oberhalb verlaufende Strukturen

Um die Gewichte für alle nötigen Kanten zu erhalten, ohne dabei Rücksicht auf Strukturen zu nehmen in denen sich die Kante befindet, werden nur zwei Alignments benötigt.

- Ein Präfixalignment der Segmente $[1..|S_1|], [1..|S_2|]$.
- ein Suffixalignment der Segmente $[1..|S_1|], [1..|S_2|]$.

Für diesen Zweck können die im Outsidealgorithmus vorgestellten Rekursionen und Berechnungsmatrizen (hier DG genannt) verwendet werden. Das Gewicht für die Kante (i,j) ergibt sich dann aus folgender Summe:

$$wa_1(i,j) = DG_{\leq i-1,j-1} + \sigma(i,j) + DG_{\geq i+1,j+1}$$

Aufgrund der Verwendung dieser beiden Alignments müssen die Matrizen nur einmal berechnet werden, um für alle Kanten ein Gewicht zu erhalten, welches die möglicherweise vorhandenen Arcs oberhalb der Positionen ignoriert. Durch die Vernachlässigung dieser Strukturen ist jedoch nicht sichergestellt, dass es sich dabei um das bestmögliche Sequenz-Struktur-Alignment handelt, weshalb für jede Kante noch weitere Berechnungen unternommen werden müssen.

Berechnung eines Gewichts mit oberhalb verlaufenden Strukturen

Zur Berechnung eines Gewichts, das auch die Strukturen widerspiegelt, die sich nicht in den Intervallen $[1..i-1], [1..j-1]$ und $[i+1..|S_1|], [j+1..|S_2|]$ befinden, werden für jede Kante (i, j) noch zwei weitere Alignments benötigt, falls solche Strukturen existieren. Hier verhält es sich wie beim Outsidealgorithmus, wo ein Alignment des Bereichs links der Positionen und eines rechts der Positionen benötigt wird.

Die nötigen Bereiche, die hierfür aligniert werden müssen sind ein Suffixalignment der Intervalle $[1..i-1], [1..j-1]$ und ein Präfixalignment der Intervalle $[i+1..|S_1|], [j+1..|S_2|]$. Entsprechend der gleichen Beschaffenheit, der Anforderungen an die Teilalignments, können auch hier die beiden Rekursionen des Outsidealgorithmus verwendet werden. Das Verfahren der Maximierung der Ähnlichkeit, über die möglichen Strukturkombinationen verhält sich, bis auf eine Ausnahme, ähnlich. Auf dieser Basis kann für das Gewicht $wa_2(i, j)$ folgende Formel verwendet werden
$\forall a_h \in P_1, b_g \in P_2$ für die gilt: $a_h^l < i < a_h^r \wedge b_g^l < j < b_g^r$

$$wa_2(i,j) = max\left\{U_{a_h,b_g} + \tau(a_h, b_g) + D_{\geq a_h^l+1, b_g^l+1} + D_{\leq a_h^r-1, b_g^r-1} + \sigma(i,j)\right\}$$

Anders als die *DG*-Matrizen können diese Hilfsmatrizen hier nach der Betrachtung der Kante (i, j) gelöscht werden, da sie für jede weitere Kante neu berechnet werden müssen.

Nun müssen die beiden berechneten Gewichte wa_1 und wa_2 für eine Kante noch verglichen werden, um das endgültige Gewicht zu erhalten.

Das Gewicht der Kante (i, j)

Für einen Eintrag in die Bibliothek $WL^{seq}_{S_1,S_2}$ ergibt sich unter Voraussetzung der Gewichte wa_1, wa_2 und der Funktion $\theta(i, j)$ das maximale Kantengewicht durch:

$$wa(i,j) = \begin{cases} -\infty & \text{falls } \theta(i,j) = 0 \\ max(wa_1, wa_2) & \text{falls } \theta(i,j) = 1 \end{cases}$$

Laufzeit

Betrachtet werden muss eine Anzahl von Kanten, die sich in $O(N^2)$ bewegt. Ähnlich des Outsidealgorithmus müssen für jede Kante zwei Alignments gebildet werden, die je $O(N^2)$ Zeit benötigen. Hinzu kommt noch die Maximierung über oberhalb verlaufende Strukturen, die von der Anzahl der möglichen Arckombinationen abhängen. Diese bewegt sich auch hier wieder in $O(N^2)$, wie bereits beim Outsidealgorithmus. Demzufolge ergibt sich für die Gewichtung sequentieller Kanten eine Gesamtlaufzeit von $O(N^2 * 3N^2) = O(N^4)$.

Zieht man die eingeführte Heuristik, der maximalen Kantenschrägheit, hinzu verringert sich die Laufzeit, da nicht weiterhin $O(N^2)$ Kanten betrachtet werden, sondern nur noch $O(N * \delta_{max})$. Dies ändert zwar nichts an der Komplexität, jedoch kann damit die tatsächliche Berechnungsdauer stark reduziert werden.

5.4 Konsistenztransformation

Die Verbesserung der gesammelten paarweisen Gewichte lässt sich mit dem Ansatz einer Konsistenztransformation erreichen. Ähnlich zu T-Coffee, wird hierzu ein Triplet-Verfahren genutzt (Abbildung 4.3, Seite 26), um in den berechneten Gewichten nicht nur Informationen der zwei beteiligten Sequenzen zu verarbeiten, sondern diese auch in Relation zu allen anderen vorhandenen Sequenzen, der Eingabe, zu setzen.

Diese Transformation muss hier jedoch auf eine etwas andere Weise erfolgen, als dies bei T-Coffee der Fall war. Dort existierte pro Position einer Sequenz maximal ein Kantengewicht für die Kombination mit einer weiteren Sequenz, dies ist hier nicht der Fall. Das bedeutet um eine Kante $(i, j) \in S_1, S_2$ über die Sequenz S_3 zu transformieren, müssen die Kanten $(i, k), (k, j)$ für alle $k = 1 \leq k \leq |S_3|$ betrachtet werden.

Durch die Trennung sequentieller und struktureller Kanten werden für dieses Verfahren folgende Bibliotheken mit transformierten Gewichten benötigt:

Definition 5.22. $EL^{S_3}_{S_1,S_2}(i, j)$ für die Speicherung der sequentiellen Gewichte von S_1, S_2 transformiert über S_3.

Definition 5.23. $EL^{P_3}_{P_1,P_2}(i, i', j, j')$ für die Speicherung der strukturellen Gewichte von P_1, P_2 transformiert über P_3.

Für die folgenden Transformationen wird noch festgelegt, dass $WL_{\mathcal{S}_1,\mathcal{S}_2} = WL_{\mathcal{S}_2,\mathcal{S}_1}$.

Transformation struktureller Gewichte

Die Durchführung der Transformation um $EL^{P_3}_{P_1,P_2}$ zu erhalten, bedarf der Betrachtung aller möglichen Strukturkombinationen zwischen P_1, P_3 und P_3, P_2. Daraus ergeben sich die transformierten Werte durch:
$\forall (i, i') \in P_1, (j, j') \in P_2 :$

$$EL^{P_3}_{P_1,P_2}(i, i', j, j') = \underbrace{max}_{\forall (k,k') \in P_3} \left\{ min(WL^{str}_{P_1,P_3}(i, i'k, k'), WL^{str}_{P_3,P_2}(k, k', j, j')) \right\}$$

Nachdem alle strukturellen Kanten transformiert wurden folgt die Transformation der sequentiellen Kanten.

Transformation sequentieller Kanten

Gesucht werden die transformierten Kantengewichte für $EL^{S_3}_{S_1,S_2}$. Diese lassen sich ähnlich einfach wie die strukturellen Gewichte berechnen.
$\forall i : 1 \leq i \leq |S_1|, j : 1 \leq j \leq |S_2| :$

$$EL^{S_3}_{S_1,S_2}(i, j) = \underbrace{max}_{1 \leq k \leq |S_3|} \left\{ min(WL^{seq}_{S_1,S_3}(i, k), WL^{seq}_{S_3,S_2}(k, j)) \right\}$$

Diese Formel ist allgemeingültig für die Kanten (i, j), jedoch müssen in der praktischen Anwendung nur diejenigen Kanten transformiert werden, für die bei der ursprünglichen Berechnung der Kantengewichte, $\theta(i, j) = 1$ erfüllt war.

Nachdem nun die Transformation von S_1, S_2 über S_3, für beide Sorten von Kanten, erstellt werden kann, lassen sich diese auf alle möglichen Sequenzen S_3 anwenden und somit die, über alle Sequenzen transformierten, Gewichte erstellen.

Zusammenführung der Transformationen

Die Transformationen über alle möglichen Sequenzen können nun zusammengefasst werden, um die neuen Gewichte für ein Paar $\mathcal{S}_1, \mathcal{S}_2$ zu erhalten. Daraus ergeben sich die folgenden zwei Bibliotheken für diese Paarung:

Definition 5.24. $EL^{seq}_{S_1,S_2}$, für die finalen sequentiellen Kantengewichte. Hierbei handelt es sich um eine Funktion $\mathbb{N} \times \mathbb{N} \to \mathbb{R}$

Definition 5.25. $EL^{str}_{P_1,P_2}$, für die finalen strukturellen Kantengewichte, unter Abbildung von $\mathbb{N}^4 \to \mathbb{R}$

Die Berechnung der Gewichte erfolgt durch diese beiden Gleichungen:

$$EL^{seq}_{S_1,S_2}(i,j) = \sum_{S_3} EL^{S_3}_{S_1,S_2}(i,j)$$

$$EL^{str}_{P_1,P_2}(i,i',j,j') = \sum_{S_3} EL^{P_3}_{P_1,P_2}(i,i',j,j')$$

In diesen Gleichungen ist S_3 nicht eingeschränkt, was bedeutet das auch die Transformation von S_1, S_2 über S_1 oder S_2 erlaubt ist. Dieser Schritt sorgt dafür, dass den direkten Kanten zwischen beiden Sequenzen eine höhere Qualität zugesprochen wird, als den über dritte Sequenzen Transformierten.

Werden diese beiden Bibliotheken zu $EL_{\mathcal{S}_1,\mathcal{S}_2}$ zusammengefasst, erhält man wieder eine Sammlung von Gewichten für $\mathcal{S}_1, \mathcal{S}_2$, die sowohl sequentielle als auch strukturelle Kanten beinhaltet. Mit diesen gesammelten und weiterverarbeiteten Informationen zu allen Sequenzpaarungen kann später das multiple Alignment gebildet werden. Zuvor jedoch noch eine kurze Betrachtung der Laufzeit dieser Konsistenztransformation.

Laufzeit

Die Laufzeit der Transformation hängt von der Anzahl der Eingabesequenzen ab. M bezeichnet die Anzahl der Sequenzen. Somit wird eine Komplexität von $O(M^2)$ benötigt alle Sequenzpaare zu betrachten. Für alle dieser Paare müssen $O(N^2)$ Kanten zwischen beiden Sequenzen betrachtet werden. Zur Transformation dieser Kanten müssen Linear viele „dritte“ Sequenzen herangezogen werden, demnach $O(M)$, und für diese dritte Sequenz jede Position einmal betrachtet werden. Dies benötigt $O(N)$ Zeit. Dadurch ergibt sich eine Gesamtlaufzeit von $O(M^3N^3)$ für die Transformation einer Kantensorte. Die Komplexität von strukturellen und sequentiellen Kanten sind dabei identisch, weshalb die Gesamtkomplexität dadurch nicht beeinflusst wird.

5.5 Bildung des multiplen Alignments

Mit der Berechnung des multiplen Alignments ist der letzte Schritt des Tree-Coffee Algorithmus erreicht. Die Informationen der Eingabesequenzen und Strukturen wurden gesammelt und in ihrer Qualität, mittels der beschriebenen Verfahren, verbessert. Anhand dieser Vorverarbeitung kann nun damit begonnen werden das gewünschte multiple Sequenz-Struktur-Alignment zu bilden. Dies wird mit Hilfe eines progressiven Ansatzes realisiert werden, ähnlich des von Feng und Doolittle eingeführten Verfahrens [FD87], wie er auch in T-Coffee Verwendung fand. Dieser Abschnitt der Arbeit wird sich damit auseinandersetzen und alle Schritte, die während der Berechnung nötig sind, erläutern. Bevor jedoch mit der Berechnung von Alignments begonnen werden kann muss der Guide-Tree, der durch den gesamten progressiven Prozess führt erstellt werden, um eine möglichst geschickte Reihenfolge der zu bildenden Alignments zu erhalten.

5.5.1 Berechnung des phylogenetischen Baums

Um den benötigten Baum zu erhalten, implementiert Tree-Coffee das „WPGMA[1]“ Verfahren (unter anderem beschrieben in [CB00]). Dieses berechnet anhand von paarweisen Distanzen, die mittels eines Clustering-Verfahrens zusammengeführt werden, einen phylogenetischen Baum. Eine wichtige Bedingung für diesen Algorithmus besteht darin, dass die berechneten Distanzen $dist(x, y)$ die Eigenschaft einer Metrik aufweisen müssen, um einen korrekten Guide-Tree zu erstellen.

Definition 5.26. Die Abbildung $dist : x \times y \to \mathbb{R}^+$ ist eine Metrik falls gilt:

$$dist(x, x) = 0 \tag{5.1}$$
$$dist(x, y) = 0 \Rightarrow x = y \tag{5.2}$$
$$dist(x, y) = dist(y, x) \tag{5.3}$$
$$dist(x, y) \leq dist(x, z) + dist(z, y) \tag{5.4}$$

Bisher jedoch waren alle Berechnungen bezüglich der gegebenen Sequenzpaare auf die Bestimmung der Ähnlichkeit ausgelegt, weshalb ein kleiner Trick angewandt werden muss, um ein Maß der Distanz zu erhalten. Während der paarweisen Sequenzanalyse durch den Insidealgorithmus, wurden die maximalen Ähnlichkeiten aller Sequenzkombinationen mitberechnet und gespeichert. Anhand dieser Werte kann eine Ordnung erstellt werden, die im weitesten Sinne die Distanzen widerspiegelt. Zu diesem Zweck wird aus der Menge dieser Ähnlichkeiten das Maximum als

$$SIM_{max} = \underbrace{max}_{\forall S_1, S_2} \left\{ SIM_{S_1}^{S_2}(A) \right\}$$

[1] engl. weighted pair grouping with arithmetic mean

ermittelt. Mit diesem Wert lässt sich die Distanz zwischen den Ähnlichkeiten, und somit eine ausreichende Distanzmatrix für den Algorithmus, bestimmen:
$\forall S_1, S_2 : dist(S_1, S_2) = SIM_{max} - SIM_{S_1}^{S_2}(A)$
Betrachtet man diese Bearbeitung nun genauer, kann man feststellen, dass die zweite Bedingung einer Metrik verletzt wird, da die Distanz zwischen einem Paar $S_m, S_{m'} = 0$ erhalten wird. Wird die zweite Bedingung einer Metrik vernachlässigt, bezeichnet man dies als Pseudometrik. Auswirkungen auf die Erstellung des Baums hat diese Einschränkung jedoch nicht, da dies nur bedeuten wird, dass im ersten Schritt des progressiven Alignments die beiden Sequenzen mit der höchsten Ähnlichkeit zuerst aligniert werden, was auch der Fall wäre falls dieses Paar die niedrigste Distanz $\neq 0$ hätte.

Mit dieser erstellten Distanzmatrix kann nun der Baum für die Menge $Sequences = \{S_1, ..., S_M\}$ in $M - 1$ Rekursionsschritten erstellt werden. Fürs weitere gilt ein Cluster C_f beinhaltet eine Menge von Sequenzen ≥ 1.

1. Suche Paar von Clustern für das gilt: $dist(C_a, C_b) = d_{min}$
2. Bestimme neuen Cluster $C_c = C_a \cup C_b$.
3. Erstelle neuen Knoten C_c mit den Söhnen C_a, C_b und der Distanz zu Blättern $= \frac{d_{min}}{2}$
4. Berechne $dist(C_c, C_d) : \forall C_d \neq C_c : dist(C_c, C_d) \frac{dist(C_a, C_d) + dist(C_b, C_d)}{2}$

Entlang des Baumes, der hierbei erstellt wurde, können nun die Schritte des multiplen Alignments durchgeführt werden.

5.5.2 Bildung eines Alignments

Im folgenden Alignmentprozess müssen zwei Fälle unterschieden werden:

1. Alignierung von zwei Sequenzen.
2. Alignierung von einer Sequenz mit einem bereits berechneten Alignment, oder zweier Alignments miteinander.

Die Unterscheidung betrifft im speziellen die verwendeten Ähnlichkeitsmatrizen, auf die die Rekursion während der Berechnung zurückgreift.

Nach jedem berechneten Alignment muss ein Traceback erstellt werden, der die Ergebnisse der Rekursion auf die alignierten Sequenzen überträgt, und so die neuen Sequenzen erstellt. Weiter wird anhand dieses Tracebacks eine Konsensusstruktur für das gebildete Alignment aus den beteiligten Strukturen errechnet.

Definition 5.27. Eine Konsensusstruktur P_K eines Alignments A, enthält die Elemente der alignierten Strukturen, die strukturelle Kanten realisieren. Demzufolge gilt für eine Konsensusstruktur: $|P_K| \leq max(|P_1|, |P_2|)$, da maximal so viele strukturelle Kanten gebildet werden können wie Arcs in den beiden alignierten Strukturen vorhanden sind.

Dies impliziert, dass sich jedes gebildete Alignment A_x aus folgenden Teilen zusammen setzt:

Definition 5.28. Ein Alignment A_x beinhaltet $T : T > 1$ Wörter $\in \Sigma^*$, wobei $\Sigma = \{A, C, G, U, -\}$, und eine Konsensusstruktur P_x^K. $A_x[t]$ bezeichnet dabei das Wort an der Stelle t. Aufgrund der gleichen Länge aller Wörter innerhalb eines Alignments, kann die Länge der in A_x enthaltenen Wörter, ähnlich zu Sequenzen durch $n = |A_x|$ ausgedrückt werden. Zusätzlich bezeichnet die Funktion $\Gamma(A_x)$ die Anzahl der enthaltenen Wörter: $\Gamma(A_x) = T$.

Diese Informationen zu einem Alignment, werden später benötigt werden wenn zwei Alignments oder ein Alignment und eine Sequenz aligniert werden sollen.

Mit diesem Wissen über die Anforderungen an die einzelnen Schritte des progressiven Alignments kann nun die verwendete Rekursion betrachtet werden.

Die Alignmentrekursion

Vorgestellt wird hier eine Rekursion, die für beide benötigten Fälle die Berechnung durchführen kann, und sich nur in den verwendeten Ähnlichkeitsmatrizen unterscheiden wird. Diese Rekursion wird sich ähnlich darstellen, wie die im Insidealgorithmus verwendete, jedoch einen Unterschied beinhalten, der später erläutert wird. Um auch in diesen Schritten den Speicherplatzbedarf auf ein quadratisches Niveau zu reduzieren, werden zuerst wieder alle möglichen Teilalignments, die durch die vorhandenen Strukturen begrenzt sind, betrachtet, bevor die kompletten Sequenzen aligniert werden. Dies führt zur Verwendung der identischen Speichermatrizen, wie sie schon bei der Einführung des Insidealgorithmus vorgestellt wurden. Zur Erinnerung:

- Eine Matrix $L_{h,g}$, der Größe $|P_1| \times |P_2|$ zur Speicherung der Ähnlichkeiten des Teilalignments $\mathcal{A}_{teil}(a_h, b_g)$.
- Eine Matrix M^{a_h,b_g}, der Größe $n_1 + 1 \times n_2 + 1$, zur Berechnung eines Teil- oder des Gesamtalignments. $M^{0,0}$ bezeichnet dabei die Matrix für das Gesamtalignment.

Diese Aufteilung impliziert auch, dass die Betrachtungsreihenfolge der Teilalignments entsprechend der Definition 5.5 (Seite 30) folgen muss. Nach der Berechnung aller Teilalignments kann dann das Gesamtalignment berechnet werden.

Die Initialisierung der Berechnungsmatrix für für ein Teilalignment von $\mathcal{A}_{teil}(a_h, b_g)$ wird folgendermaßen durchgeführt:

$$M^{a_h,b_g}_{a_h^l,b_g^l} = 0$$

$$\forall i : a_h^l < i < a_h^r \Rightarrow M^{a_h,b_g}_{i,b_g^l} = 0$$

$$\forall j : b_g^l < j < b_g^r \Rightarrow M^{a_h,b_g}_{a_h^l,j} = 0$$

Zur Berechnung des Teilalignments wird die folgende Rekursion verwendet:
$\forall i', j' : a_h^l < i' < a_h^r \wedge b_g^l < j' < b_g^r$:

$$M_{i',j'}^{a_h,b_g} = max \begin{cases} M_{i'-1,j'}^{a_h,b_g} & \\ M_{i',j'-1}^{a_h,b_g} & \\ M_{i'-1,j'-1}^{a_h,b_g} + LIB(i,j) & \\ & \text{falls } (i_1,i') \in P_1, (j_1,j') \in P_2 \\ M_{i_1-1,j_1-1}^{a_h,b_g} + \underbrace{L_{a,b}}_{\text{wo } a=(i_1,i') \text{ und } b=(j_1,j')} & \text{und für } i_1, j_1 \text{ gilt:} \\ & i_1 > a_h^l \wedge j_1 > b_g^l \end{cases}$$

Für die maximale Ähnlichkeit von $\mathcal{A}_{teil}(a_h, b_g)$ ergibt sich dann:

$$L_{h,g} = M_{a_h^r-1,b_g^r-1}^{a_h,b_g} + LIB(a_h^l, a_h^r, b_g^l, b_g^r)$$

Um mit dieser Rekursion das Gesamtalignment zu berechnen werden als Grenzen einfach die Positionen $(0, n_1 + 1)$ und $(0, n_2 + 1)$ verwendet, wobei n_1 und n_2 der Länge einer Sequenz oder eines Alignments entsprechen.

Auffallend an Initialisierung und Rekursion, sind das Fehlen jeglicher Kosten für die Operationen *base-deletion/insertion, arc-breaking* und *arc-removing*. Dies resultiert daraus, dass diese Kosten bereits in die Kantengewichte eingearbeitet wurden und nicht weiter nötig sind.

Die Funktionen $LIB(i,j) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ und $LIB(i,i',j,j') : \mathbb{N}^4 \to \mathbb{R}$ realisieren die Ähnlichkeiten. Die Definition dieser Funktionen ist abhängig davon, ob nur zwei Sequenzen oder eine andere Kombination aligniert wird. Die Beschaffenheit dieser Bibliotheken werden im Folgenden erläutert.

Die Ähnlichkeitsfunktionen für das alignieren zweier Sequenzen

Sollen nun zwei Sequenzen S_1, S_2 und ihre dazugehörigen Strukturen P_1, P_2 aligniert werden, so lassen sich die Funktionen sehr leicht bereitstellen.

- $LIB(i,j) = EL_{S_1,S_2}^{seq}$ für die sequentiellen Kanten.
- $LIB(i,i',j,j') = EL_{P_1,P_2}^{str}$ für die strukturellen Kanten.

Diese paarweisen Bibliotheken wurden bereits berechnet und können in ihrer einfachen Form verwendet werden, um das Alignment dieser beiden Sequenzen zu bestimmen.

Komplizierter wird es für den zweiten Fall, wo diese Bibliotheken in der benötigten Form noch nicht existieren und erst berechnet werden müssen.

Die Ähnlichkeitsfunktionen für das alignieren zweier Alignments

Im folgenden wird die Berechnung der Ähnlichkeitsfunktionen für das alignieren von zwei Alignments betrachtet werden. Der Fall ein Alignment mit einer Sequenz, oder anders

herum, ist Teil dieser Menge, da ein Tupel $\mathcal{S}_1 = (S_1, P_1)$ als Sonderfall eines Alignments A_1 mit der Eigenschaft $\Gamma(A_1) = 1$ angesehen werden kann, und somit eine getrennte Betrachtung dieses Falls nicht notwendig ist. Ziel wird es nun sein durchschnittliche Ähnlichkeitswerte für das Alignieren von Spalten des einen Alignments mit Spalten des anderen zu berechnen. Im folgenden werden die Alignments A_1 und A_2, mit den Längen n_1, n_2, und den Strukturen P_1^K und P_2^K betrachtet.

Zuerst werden die Werte für $LIB(i, i', j, j')$ erstellt, die $|P_1^K| \times |P_2^K|$ Einträge umfassen wird.
Um dies formal ausdrücken zu können wird eine Funktion $A_x(t)$ benötigt, die für ein Element $t : 1 \leq t \leq \Gamma(A_x)$ aus A_x den ursprünglichen Index $l : 1 \leq l \leq M$ des Tupels $\mathcal{S}_l$ liefert, für das gilt $A_1[t] = S_l \cup \{-\}$.
Weiter wird eine Funktion $\epsilon_1(a_h)$ benötigt, die für einen Arc $a_h \in P_1^K$ (respektive $b_g \in P_2^K$) die ursprünglichen Positionen $(a^l_{h'}, a^r_{h'}) \in P_{l_1}$ (analog für $b_g \in P_{l_2}$) berechnet.
Mit diesen Funktionen lässt sich nun eine Formel aufstellen, mit der die Einträge in $LIB(i, i', j, j')$ berechnet werden können:
$\forall h : 1 \leq h \leq |P_1^K|, g : 1 \leq g \leq |P_2^K| :$

$$LIB(a^l_h, a^r_h, b^l_g, b^r_g) = \frac{\sum_{t_1=1}^{\Gamma(A_1)} \sum_{t_2=1}^{\Gamma(A_2)} EL^{str}_{P_{l_1}, P_{l_2}}(\epsilon_1(a_h), \epsilon_1(b_g))}{\Gamma(A_1) * \Gamma(A_2)}, \begin{array}{l}\text{wobei } l_1 = A_1(t_1) \\ \text{und } l_2 = A_2(t_2)\end{array}$$

Informal ausgedrückt werden für alle Sequenzpaarungen, der beiden Alignments, berechnet welches Arcs der ursprünglichen Strukturen, den aktuell betrachteten der Konsensusstrukturen entspricht. Die Ähnlichkeitswerte für diese ursprünglichen Arcs aller Sequenzkombinationen werden aufsummiert und durch die Anzahl der Sequenzkombinationen normalisiert.

Als nächstes muss die Bibliothek für sequentielle Kanten erstellt werden. Hierfür wird eine Funktion $\epsilon_2(i)$ benötigt, die für $A_1[t[i]]$ den ursprünglichen Index dieser Position i' in S_{l_1} (analog für j und S_{l_2}) liefert. Dies ergibt für einen Eintrag in die Bibliothek die Formel:
$\forall i : 1 \leq i \leq |A_1|, j : 1 \leq j \leq |A_2| :$

$$LIB(i,j) = \frac{\sum_{\underbrace{t_1 = 1}_{t_1[i] \neq -}}^{\Gamma(A_1)} \sum_{\underbrace{t_2 = 1}_{t_2[j] \neq -}}^{\Gamma(A_2)} EL^{seq}_{S_{l_1}, S_{l_2}}(\epsilon_2(i), \epsilon_2(j)), \text{falls } EL^{seq}_{S_{l_1}, S_{l_2}}(\epsilon_2(i), \epsilon_2(j)) \neq -\infty}{\sum_{\underbrace{t_1 = 1}_{t_1[i] \neq -}}^{\Gamma(A_1)} \sum_{\underbrace{t_2 = 1}_{t_2[j] \neq -}}^{\Gamma(A_2)} 1}$$

l_1, l_2 verhalten sich dabei analog zur Berechnung der strukturellen Ähnlichkeiten.
Nachdem nun die Einträge für beide Ähnlichkeitsfunktionen erstellt wurden, kann auch das Alignment zweier Alignments anhand der gegebenen Rekursion erfolgen. Mit den berechneten Teil- und dem Gesamtalignment, kann das Traceback durchgeführt werden, um die alignierten Sequenzen und die Konsensusstruktur zu erstellen. Dieses Traceback wird nun erklärt.

5.5.3 Erstellen des Tracebacks

Sinn eines Tracebacks ist herauszufinden, durch welche Reihenfolge und Art der Operationen die maximale Ähnlichkeit eines Alignments erreicht wurde. Dieses Traceback wird dazu benötigt die alignierten Sequenzen zu erstellen und somit das fertige Alignment darzustellen. Die Erstellung eines Tracebacks kann entweder parallel zur Berechnung des Alignments, oder im Anschluss, anhand einer Rekursion welche vom finalen Eintrag der Berechnungsmatrizen bis zur ersten Position diese Matrizen rückwärts durchläuft, erfolgen. Hier wird die letzte Methode zu sehen sein. Während dieses Tracebacks wird ein Tracebackstring erstellt, der definiert ist durch:

Definition 5.29. Der Tracebackstring TBs ist ein Wort über Σ^*, wobei Σ das Alphabet $\{\leftarrow, \uparrow, \nwarrow, \nwarrow_b\}$ bezeichnet.

Mit dieser Definition kann die verwendete Rekursion für die Bestimmung des Tracebackstrings aufgestellt werden.

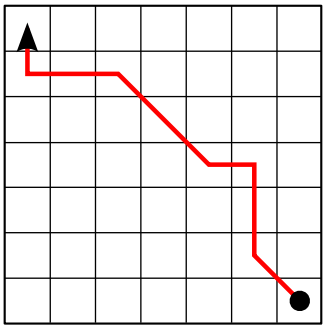

Abbildung 5.8: Beispiel eines Tracebackpfades durch eine Matrix

Die Tracebackrekursion

Mit Hilfe dieser Rekursion soll der Pfad durch die Matrix $M^{0,0}$ berechnet werden, der die verwendeten Operationen zur Maximierung der Ähnlichkeit enthält. Aufgrund der Aufteilung, der gesamten Alignmentberechnung in die möglichen Teilalignments, wird eine rekursive Betrachtung nötig sein, um den korrekten Pfad zu erhalten. Damit ergibt sich folgende Rekursion mit dem Aufruf $TB(i_l, i_r, j_l, j_r)$, wobei der erste Aufruf durch $TB(0, |S_1| + 1, 0, |S_2| + 1)$ (oder $|A_1| + 1, |A_2| + 1$) gegeben ist, die für die übergebenen Segmente $[i_l..i_r], [j_l..j_r]$ den Tracebackstring liefert.

$\forall i' : i_l < i' < i_r, j' : j_l < j' < j_r :$

$$TBs = \begin{cases} \uparrow +TBs & \text{für } M_{i',j'} = M_{i'-1,j'} \\ \leftarrow +TBs & \text{für } M_{i',j'} = M_{i',j'-1} \\ \nwarrow +TBs & \text{falls } M_{i',j'} = M_{i'-1,j'-1} + LIB(i',j') \\ \nwarrow_b +TB(a_h^l, a_h^r, b_g^l, b_g^r) + \nwarrow_b +TBs & \text{falls } M_{i',j'} = M_{a_h^l-1,b_g^l-1} \\ & +LIB(a_h^l, a_h^r, b_g^l, b_g^r), \\ & \text{dann berechne TBs für dieses} \\ & \text{Teilalignment und setze} \\ & i' = a_h^l - 1 \text{ und } j' = b_g^l - 1 \end{cases}$$

Die verwendete Matrix M ist immer abhängig davon, welches Teilalignment (bzw. Gesamtalignment) betrachtet wird.
Diese Rekursion liefert, vollständig berechnet, den kompletten Tracebackstring für das Alignment. Anhand dieses Strings können die überarbeiteten Sequenzen erstellt, und die Konsensusstruktur berechnet werden.

Das Traceback auf die Sequenzen (Alignments) anwenden

Um einen Tracebackstring auf die Sequenzen S_1, S_2 anzuwenden wird auf diese Weise vorgegangen:

- Für S_1^0 ersetze das i-te Vorkommen von $\{\nwarrow_b, \nwarrow, \uparrow\}$ durch $S_1[i]$, und $\{\leftarrow\}$ durch $-$.
- Für S_2^0 ersetze das j-te Vorkommen von $\{\nwarrow_b, \nwarrow, \leftarrow\}$ durch $S_2[j]$, und $\{\uparrow\}$ durch $-$.

Wendet man den Tracebackstring auf die Alignments A_1, A_2 an müssen alle vorhandenen Sequenzen aus A_1 mit der ersten Regel, und alle aus A_2 mit der zweiten bearbeitet werden.

Berechnung der Konsensusstruktur

Um die Konsensusstruktur zu erhalten, dürfen nur die Arcs beibehalten werden, die im Alignment eine strukturelle Kante realisiert haben. Alle anderen Positionen der Sequenzen werden zu freien Positionen. Zur Erinnerung Strukturen können in Dot-Bracket Notation angegeben werden, wodurch sich eine Position der Struktur durch $P[i]$ ansprechen lässt.

- Für P^K ersetze das i-te Vorkommen von $\{\nwarrow_b\}$ durch $P_1[i]$, die i-ten Vorkommen von $\{\uparrow, \leftarrow\}$ durch $\bullet$ und jedes Vorkommen von $\{\nwarrow\}$ durch $\bullet$

Das so berechnete und erstellte Alignment, liefert eine Menge von Sequenzen und eine Konsensusstruktur und kann dazu verwendet werden es durch weitere Alignments entlang des Guide-Trees zu vergrößern.

5.5.4 Die Schritte entlang des phylogenetischen Baumes

Nachdem erläutert wurde, wie ein einzelner Alignmentschritt funktioniert, kann für jeden Knoten des Guide-Trees das Alignment der Blätter berechnet werden. Sobald das letzte Alignment an der Wurzel des Baumes erstellt wurde erhält man ein multiples Alignment, dass alle Eingabesequenzen und eine Konsensusstruktur für dieses Alignment beinhaltet. Dieses Alignment ist das gesuchte multiple Sequenz-Struktur-Alignment der Eingabedaten.

Laufzeit des progressiven Alignments

Nach der Erstellung des Guide-Trees müssen $M - 1$ Alignments gebildet werden, die jeweils in einer Komplexität von $O(N^4)$ bearbeitet werden können. Demzufolge ergibt sich für die Laufzeit des multiplen Alignments eine Komplexität von $O(MN^4)$.

5.6 Gesamtlaufzeit und Speicher

Abschließend zur Beschreibung des Tree-Coffee Algorithmus, wird noch einmal die Gesamtlaufzeit, anhand der einzelnen Schritte, betrachtet. Zudem wird noch einmal die Speicherkomplexität erwähnt, die für ein Sequenz-Struktur-Alignment auch von großer Wichtigkeit ist.

Laufzeit

Die beiden verwendeten Algorithmen, zur Analyse aller paarweisen Sequenzpaarungen benötigen jeweils ein Laufzeit in $O(N^4)$ zur Berechnung der maximalen Ähnlichkeit, für die Bereiche die damit berechnet werden sollen. Aus diesen Daten werden in $O(N^4)$ für jedes Paar die gewichteten Kanten (sequentiell, strukturell) erstellt, um die Hauptanforderung für einen konsistenzbasierten Ansatz zu erfüllen. Am Ende dieser Prozesse für die paarweisen Alignments, können die berechneten Gewichte transformiert werden, um

Prozess	**Komplexität**
Insidealgorithmus	$O(M^2N^4)$
Outsidealgorithmus	$O(M^2N^4)$
Kantengewichtung	$O(M^2N^4)$
Konsistenztransformation	$O(M^3N^3)$
Progressives Alignment	$O(MN^4)$

Tabelle 5.1: Laufzeitkomplexitäten der einzelnen Teilschritte

frühe Fehler während des progressiven Alignments zu vermeiden, indem Informationen aller anderen vorhandenen Sequenzen in die paarweisen Gewichte eingearbeitet werden. Dieser Vorgang benötigt eine maximale Laufzeit $O(M^3N^3)$. Zuletzt muss das progressive Alignment mitsamt Erstellung des phylogenetischen Baums und teilweise einer Neuberechnung der Ähnlichkeitsfunktionen, und den dazugehörigen Alignments durchgeführt werden. Während des progressiven Prozesses benötigt die Bildung des Alignments die meiste Zeit, die sich entsprechend des gleichen Aufbaus, in Vergleich zum Insidealgorithmus, in $O(N^4)$ befindet. Die Anzahl der benötigten Alignments steigt dabei nur linear in Abhängigkeit der Anzahl aller Sequenzen und liegt am Ende in $O(MN^4)$.

Zu sehen ist hierbei, dass alle verwendeten Prozesse und Unterprozesse mit einer maximalen Komplexität von $O(N^4)$ zu Buche schlagen, und sich dadurch die Gesamtkomplexität des Algorithmus auch in $O(N^4)$ befindet.

Speicherplatz

Fast wichtiger als die Laufzeit eines Algorithmus ist der benötigte Speicherplatz, während der Berechnung. Während eine hohe Komplexität der Laufzeit zwar ein Ärgernis ist, aber im Grunde die Berechnung nicht verhindern kann, wenn nur ausreichend Zeit aufgebracht wird, so kann eine zu hohe Speicherkomplexität jedoch eine Berechnung völlig unmöglich machen (immer in Anbetracht der verfügbaren technischen Möglichkeiten). Berechnungen können nur solange durchgeführt werden, wie auch die Möglichkeit besteht die Ergebnisse davon zu speichern.

Im Grunde genommen müssten alle Sequenz-Struktur-Alignments die von Tree-Coffee erstellt werden eine Speicherkomplexität von $O(N^4)$ besitzen. Dies wird jedoch durch die Aufteilung der Alignments in verschiedene Subalignments verhindert. Durch die Betrachtung abgeschlossener Strukturen als Einheit, kann im späteren Verlauf auf solch eine abgeschlossene Struktur zurückgegriffen werden, in diesem Fall die maximale Ähnlichkeit eines gesamten Teilalignments. Da nun jedoch nur noch ein Wert für eine solche Teilstruktur gespeichert werden muss, kann die Berechnung eines Alignments mit einem Speicherbedarf von $O(N^2)$ realisiert werden.

Bei allen weiteren gespeicherten Daten handelt es sich auch maximal um eine Speicherstruktur, die mit einer quadratischen Matrix beschrieben werden kann. Daraus ergibt sich eine Gesamtkomplexität des benötigten Speicherplatzes von $O(N^2)$ während des gesamten Algorithmus.

6 Testergebnisse von Tree-Coffee

In diesem Kapitel werden die praktischen Tests des Tree-Coffee Programms vorgestellt. Im speziellen wird die Qualität der erreichten Ergebnisse überprüft werden. Dies geschieht anhand eines Benchmarks, der speziell dafür entwickelt wurde diese Qualität zu berechnen. In diesem Zusammenhang wird ein Vergleich zwischen Tree-Coffee, MARNA und dem ClustalW Algorithmus betrachtet.

Weiter werden einige Laufzeitanalysen durchgeführt werden, die zeigen sollen, welche Schritte des Algorithmus wie viel Zeit beanspruchen. Hierfür werden unter anderem paarweise Alignments berechnet, um zu zeigen wie sich die benötigten Zeiten von Insidealgorithmus, Outsidealgorithmus und Kantengewichtung im Verhältnis zur Sequenzlänge entwickeln. Danach wird ein Test durchgeführt werden, der Gesamtlaufzeiten von paarweisen bis multiplen Alignments mit bis zu 15 Sequenzen betrachtet. Zuletzt wird mit Hilfe eines paarweisen Alignments, mit längeren Sequenzen gezeigt wie effizient die gewählte Heuristik, zur Einschränkung der Anzahl gewichteter Kanten, arbeitet.

Zuerst jedoch soll ein kleiner Überblick über die Implementation des Programms erfolgen und erläutert werden, welche Daten bereitgestellt werden müssen, um eine erfolgreiche Durchführung des Algorithmus zu gewährleisten.

6.1 Einige Details zur Implementation

Für die Implementation von Tree-Coffee wurde die Programmiersprache C++ gewählt, da eine Objekt-Orientierte Sprache für die benötigten Zwecke alle Anforderungen erfüllt. Es ist lauffähig unter UNIX und Windows Systemen, die auch dazu verwendet wurden die praktischen Tests durchzuführen. Neben einer Datei im „FASTA“ Format, die alle Sequenz-Struktur-Paare beinhaltet, und einer „RIBOSUM“ Datei, welche für die paarweisen Alignments die Scores bereitstellt, werden die Parameter aus Tabelle 6.1 benötigt.

Mit diesen Daten beginnt das Programm die Berechnungen und stellt am Ende eine Ausgabe des multiplen Alignments in Form einer Datei zur Verfügung, die alle Sequenzen und die gebildete Konsensusstruktur beinhaltet (Abbildung 6.1). Zusätzlich wird eine Log-Datei angelegt, die den Verlauf des Programms mitprotokolliert und die benötigten Zeiten einzelner Abschnitte angibt.

Parameter	Format
Gapkosten γ	Bsp: -0.75
Arc-Removing Kosten β_r	Bsp: -1.00
Arc-Breaking Kosten β_b	Bsp: -0.66
Faktor ω für die strukturellen Kantengewichte	Bsp: 2.00
Maximale Kantenschrägheit δ_{max}	Bsp: 0 (um keine Einschränkung zu treffen)

Tabelle 6.1: Benötigte Parameter von Tree-Coffee

```
Tree-Cofee (1.00) multiple sequence structure alignment

Seq1          UACGUG--GACGAUA
Seq2          UACGUGA-GACGAUA
Seq3          UACGUGAAGACGAUA
Seq4          -ACGU---GACG-UA
              .(((......)).).
```

Abbildung 6.1: Beispiel der Ausgabe eines multiplen Alignments

6.2 Benchmark auf der BraliBase

Zunächst wird die Qualität der erzielten Ergebnisse von Tree-Coffee betrachtet. Zu diesem Zweck wurde ein Benchmark anhand der BraliBase (v2.1) [WMS06], bei der es sich um eine Weiterentwicklung des ein Jahr zuvor vorgestellten Benchmarks [GWW05] handelt, erstellt und mit den Ergebnissen von MARNA und ClustalW verglichen. Bevor die Ergebnisse betrachtet werden aber noch einige Informationen zur BraliBase.

6.2.1 BraliBase

Die BraliBase ist eine Datenbank von RNA Sequenzen, mit deren Hilfe die Qualität von Alignmentalgorithmen getestet werden können. Die dabei betrachteten Sequenzen stammen aus 32 unterschiedlichen RNA-Familien, um zu gewährleisten, dass die Ergebnisse nicht durch eine eingeschränkte Anzahl von betrachteten Familien verfälscht werden.

Für jede Familie existieren mehrere Datensätze, von paarweisen bis hin zu multiplen Alignments mit 15 Sequenzen. Diese Datensätze können in die Anzahl der zu alignierenden Sequenzen unterteilt werden. Es stehen 6 unterschiedliche Sets mit folgenden Größen $k \in \{2, 3, 5, 7, 10, 15\}$ zur Verfügung. Jedes dieser Sets beinhaltet eine Menge von zu bildenden Alignments, die sich in der paarweisen Sequenzidentität (APSI) und dem strukturellen Konservationsindex (SCI) unterscheiden.

Der APSI rangiert dabei in Bereichen von 20% bis hin zu 95%. Mit diesen Werte kann untersucht werden, wie die Qualität der erreichten Ergebnisse von der Sequenzidentität abhängt, da ein großes Interesse darin besteht effiziente Algorithmen zu finden, die auch

auf niedrigen Sequenzidentitäten gute Ergebnisse liefern.

Für jeden Satz von Daten der BraliBase existieren Referenzalignments, deren Anzahl und Familien der Tabelle A.1 entnommen werden können, die von Hand bearbeitet wurden, um die bestmöglichen Alignments zu erreichen. Anhand dieser Referenzalignments und den Ergebnissen eines Algorithmus kann die Qualität der Ergebnisse berechnet werden. Hier wird als Maß der Qualität der „Sum-of-Pairs“ Score (im weiteren Verlauf SPS genannt) betrachtet werden, der die Ähnlichkeit zwischen Referenzalignment und Testalignment angibt.

Dieser SPS wird im folgenden verwendet, um Aussagen über die Qualität des Tree-Coffee Algorithmus treffen zu können.

6.2.2 Ergebnisse

Bei diesem Benchmark wurden nicht alle möglichen Alignments der BraliBase berechnet, da das Hauptinteresse bei Sequenz-Struktur-Alignments darin besteht gute Ergebnisse für niedrige Identitäten zu liefern. Aus diesem Grund wurden nur die Alignments betrachtet, deren APSI maximal 50% beträgt.

Da die BraliBase keine Sekundärstrukturen für die Sequenzen beinhaltet, mussten diese für jede Sequenz berechnet werden. Dies wurde mit Hilfe von „RNAFold“ aus dem Vienna RNA Package gelöst [uPS98], das die *mfe-Struktur*[1] einer Sequenz, anhand eines thermodynamischen Modells, berechnet.

Die nötigen Parameter für Tree-Coffee wurden wie folgt gewählt:

Parameter	**Wert**
Scoringwerte für paarweise Alignments	RIBOSUM85_60.txt
γ	-0.75
β_r	-1.00
β_b	-0.66
ω	2.00
δ_{max}	40

Tabelle 6.2: Gewählte Parameter für den BraliBase Benchmark

Aus den erreichten Ergebnissen wurden für die verschiedenen Sets Diagramme angefertigt, die die Ergebnisse von Tree-Coffee, MARNA und ClustalW darstellen. Diese Diagramme können auf den folgenden drei Seiten betrachtet werden.

[1] engl. minimal free energy structure

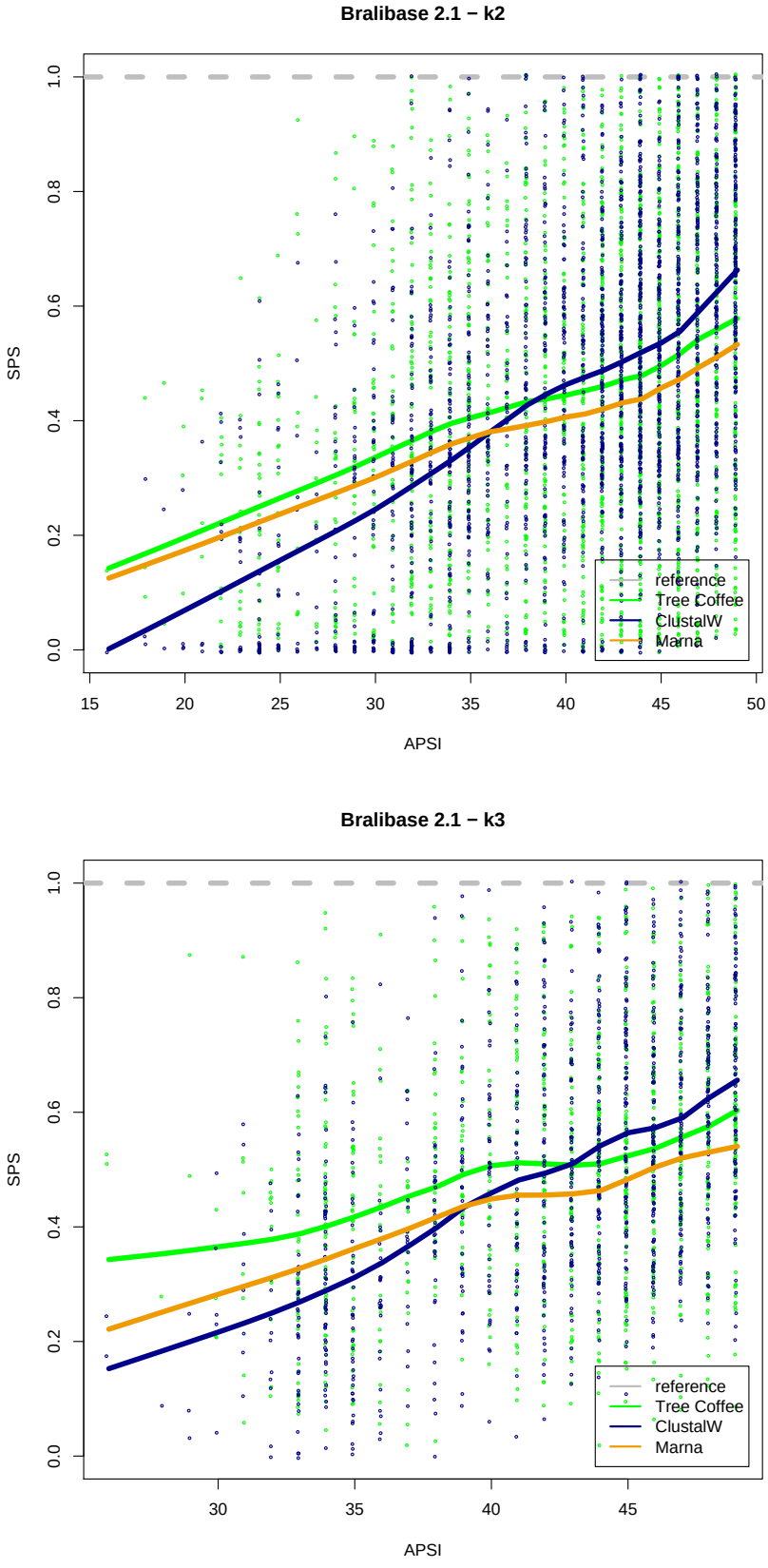

Abbildung 6.2: Benchmarkergebnisse für die Sets $k2$ und $k3$

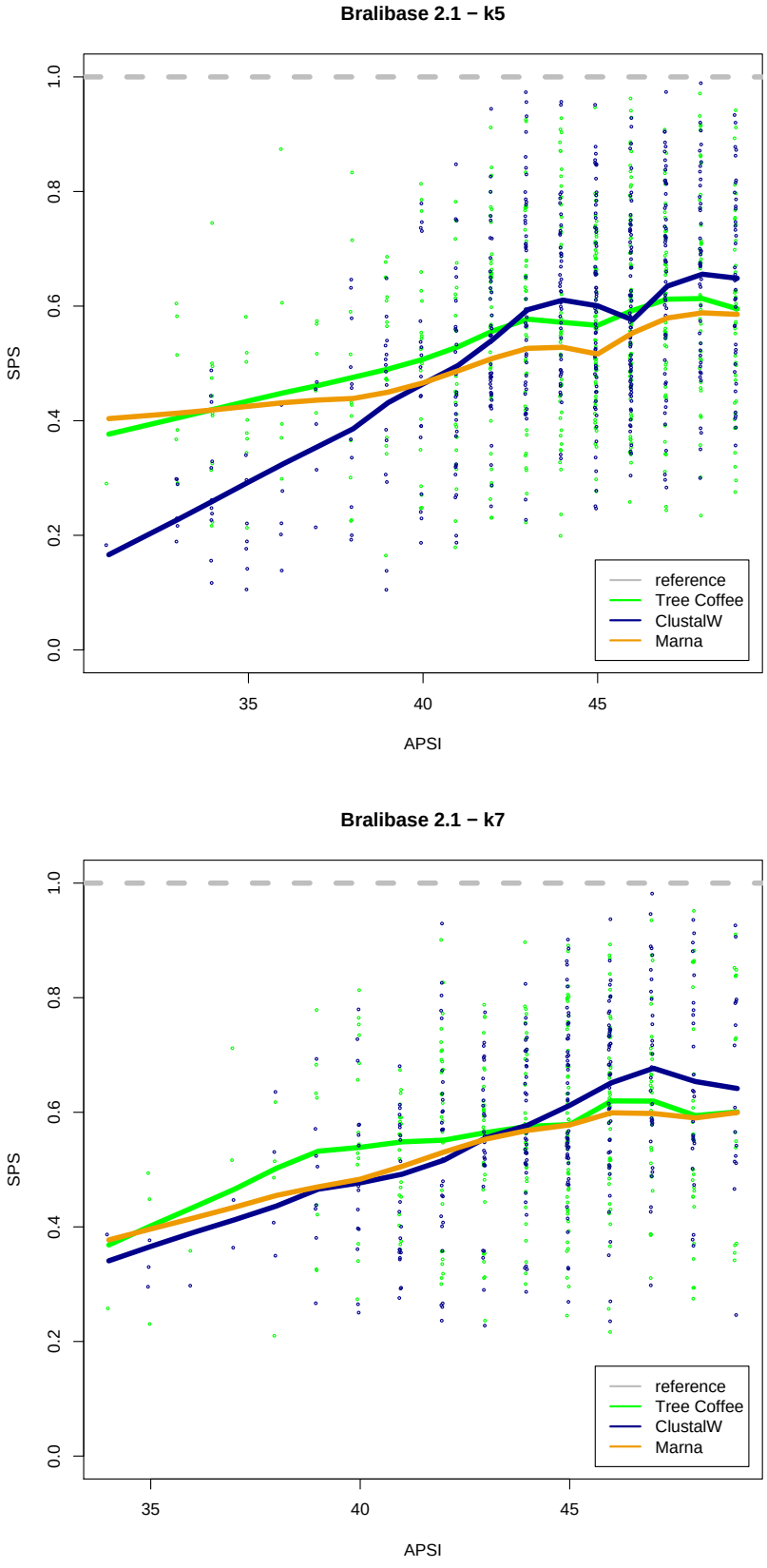

Abbildung 6.3: Benchmarkergebnisse für die Sets $k5$ und $k7$

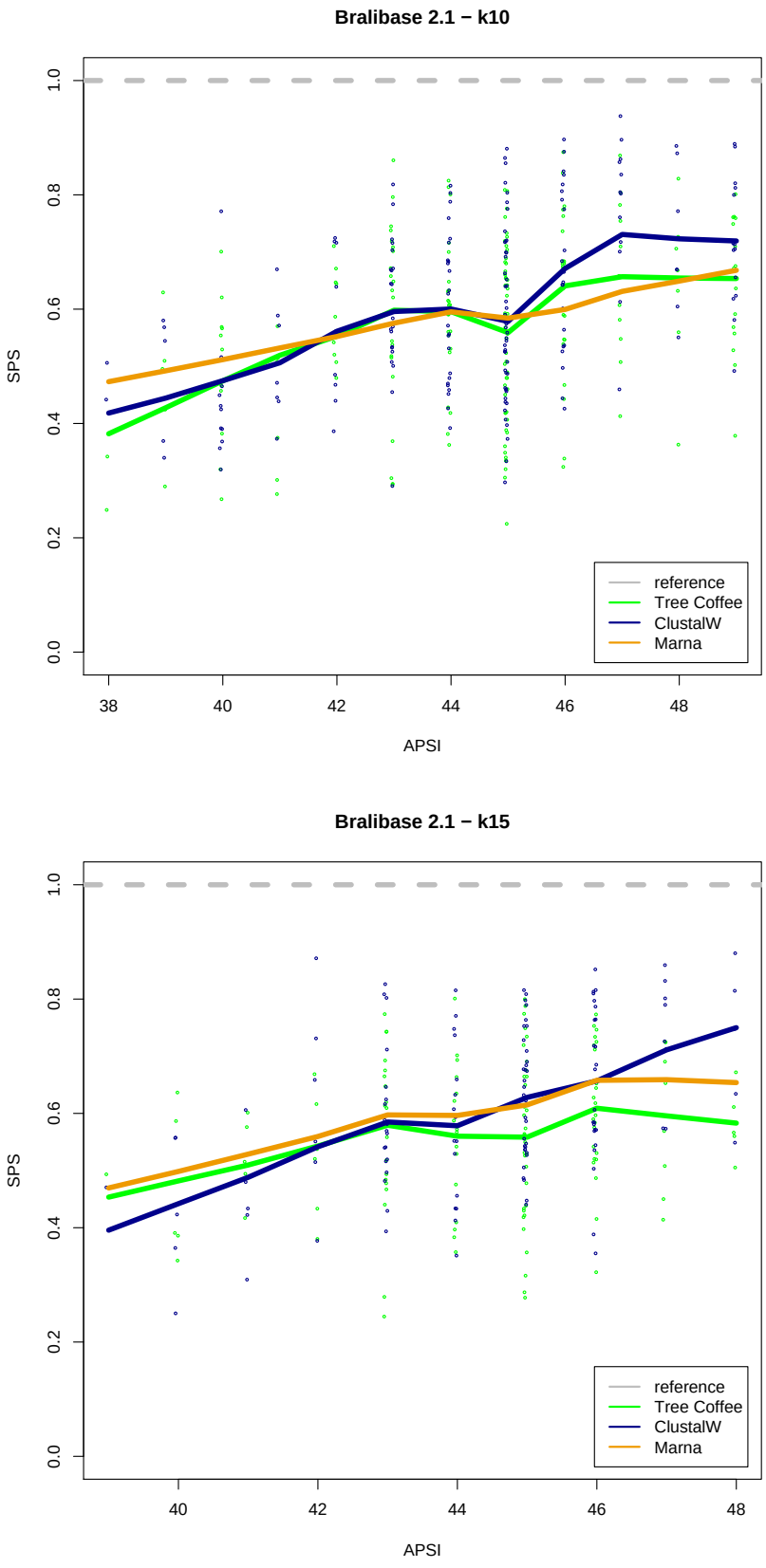

Abbildung 6.4: Benchmarkergebnisse für die Sets $k10$ und $k15$

6.3 Laufzeitvergleiche

In diesen Bereich werden die Testergebnisse zu verschiedenen Laufzeitmessungen von Tree-Coffee angegeben. Zuerst werden die Laufzeiten der paarweisen Sequenzanalyse gezeigt, bevor Gesamtlaufzeiten des Algorithmus für eine variierende Sequenzanzahl des multiplen Alignments erstellt werden. Daraufhin wird an einem paarweisen Alignment mit längeren Sequenzen gezeigt, wie sich die Laufzeit der Kantengewichtung bei unterschiedlichen maximalen Kantenschrägheiten verändert.

Sofern nicht anderweitig angegeben wurden für die Messungen folgende Parameter gewählt:

Parameter	Wert
Scoringwerte für paarweise Alignments	RIBOSUM85_60.txt
γ	-0.75
β_r	-1.00
β_b	-0.66
ω	2.00
δ_{max}	0 (keine Einschränkung)

Tabelle 6.3: Gewählte Parameter für die Laufzeiten

Verwendetes Testsystem

Als Testsystem, für die Laufzeitmessungen, wurde ein System mit folgenden Spezifikationen verwendet:

- **CPU:** Intel Core2Duo E6750 mit $2 * 2,66$GHz
- **RAM:** 2 GB
- **HDD:** 500 GB
- **Betriebssystem:** Windows XP mit ServicePack 2

6.3.1 Laufzeiten der paarweisen Sequenzanalyse

Die Testdaten für diese Messungen können der Tabelle B.1 in Anhang B entnommen werden. Diese Daten wurden aus der BraliBase entnommen und beinhalten eine kleine Auswahl an Sequenzen unterschiedlicher Längen. Die Größenordnung bewegt sich dabei zwischen 70 und 200 Nukleotiden, da in der BraliBase maximal Sequenzen mit einer Länge von ca. 300 Nukleotiden vorkommen. Die interessanten Laufzeiten der Kantengewichtung für eine Sequenz dieser Länge werden später noch detaillierter betrachtet werden und wurden aus diesem Grund hier ausgenommen. Die gewählten Sequenzen stammen hier aus dem $k2$-Set der BraliBase. Es wurden aus fünf Familien jeweils die

zwei Alignments herausgesucht, die den niedrigsten beziehungsweise höchsten APSI-Wert besitzen. Dabei wurden die folgenden Zeiten in Tabelle 6.4 gemessen.

Nr.	Insidealgorithmus	Outsidealgorithmus	Kantengewichtung
1	2 sek	3 sek	56 sek
2	2 sek	3 sek	50 sek
3	13 sek	24 sek	453 sek
4	10 sek	18 sek	301 sek
5	2 sek	4 sek	65 sek
6	1 sek	1 sek	6 sek
7	1 sek	1 sek	6 sek
8	1 sek	1 sek	7 sek
9	4 sek	4 sek	107 sek
10	2 sek	3 sek	49 sek

Tabelle 6.4: Gemessene Laufzeiten der paarweisen Sequenzanalyse

Im dargestellten Diagramm 6.5 lässt sich erkennen, wie sich die benötigte Zeit der einzelnen Analyseschritte in Abhängigkeit der durchschnittlichen Längen, der zu alignierenden Sequenzen, verhält.

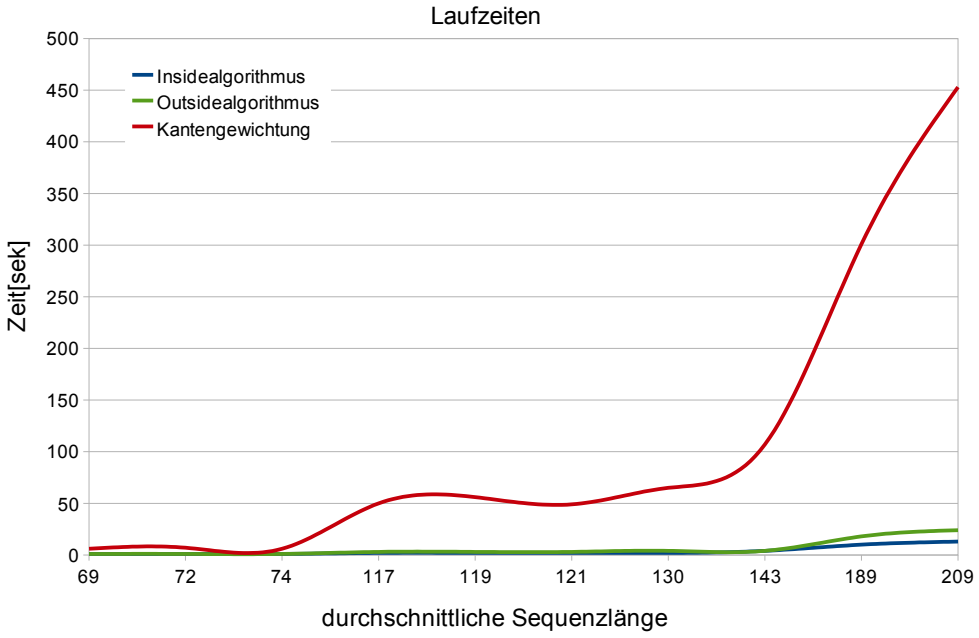

Abbildung 6.5: Verhältnis der benötigten Zeit zur Sequenzlänge

6.3.2 Laufzeiten multipler Alignments

Um die Laufzeiten des Algorithmus in Abhängigkeit der Sequenzanzahl zu bestimmen, wurde eine RNA-Familie aus der BraliBase ausgewählt, und aus jedem Set ein Datensatz mit einem APSI-Wert von ca. 50% gewählt. Die genaue Auswahl und durchschnittliche Länge der enthaltenen Sequenzen ist in Tabelle B.2 in Anhang B aufgelistet.

Zur Darstellung der benötigten Laufzeiten wurden diese in 5 Abschnitte unterteilt: Für alle paarweisen Sequenzalignments wurden die Zeiten, die für den Insidealgorithmus, den Outsidealgorithmus und die Kantengewichtung benötigt wurden aufsummiert, um darzulegen, wie hoch die Anwendungen dieser Schritte ausfallen. Weiter wurde die Zeit gemessen, die für die Konsistenztransformation nötig war. Zuletzt wurde festgestellt, wie lange der Algorithmus benötigt hat, um das finale multiple Alignment zu erstellen.

Mit diesen Zeiten wurde die Gesamtdauer errechnet, die für die Erstellung des multiplen Alignments für die gegebenen Sequenzen nötigt war.

Diese Ergebnisse wurden zu einem Diagramm verarbeitet, welches in Abbildung 6.6 zu sehen ist.

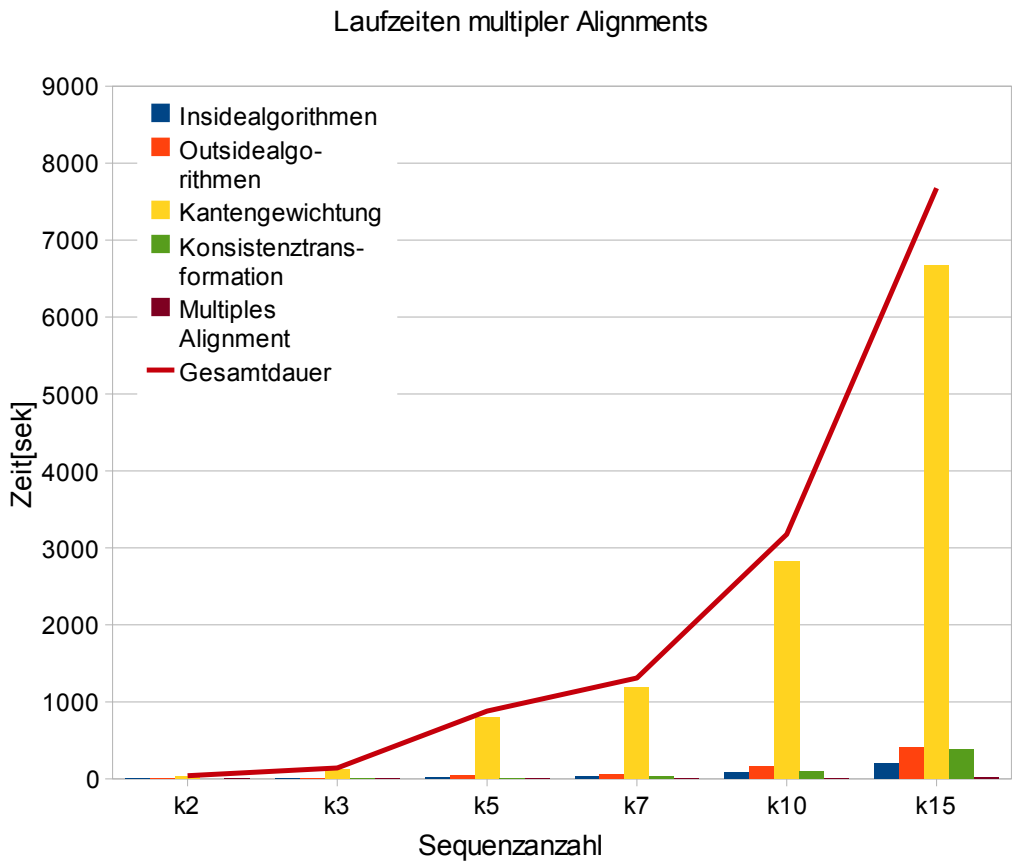

Abbildung 6.6: Laufzeiten multipler Alignments

6.3.3 Laufzeit bei Reduzierung der gewichteten Kanten

Die Effektivität der eingeführten Heuristik, um die Anzahl der zu gewichtenden sequentiellen Kanten zu reduzieren, wurde in einem separaten Test festgestellt. Zu diesem Zweck wurde aus der BraliBase Datenbank eine RNA-Familie ausgewählt, die von Natur aus vergleichsweise lange Sequenzen beinhaltet. Die Wahl der Familie fiel aus diesem Grund auf *SRP_euk_arch*, da ihre Sequenzen mit Längen jenseits der 300 Nukleotide an der oberen Grenze, der in der BraliBase vorhandenen Sequenzen, liegen. Für die Analyse wurde ein Datensatz aus dem $k2$-Set dieser Familie entnommen, da ein paarweises Alignment ausreichend ist, um die Heuristik zu testen. Die vollständigen Sequenzen und dazugehörigen Strukturen können der Tabelle B.3 in Anhang B entnommen werden.

Insgesamt wurden 9 Messungen durchgeführt, mit Werten von

$$\delta_{max} = \{20, 40, 60, 80, 100, 150, 200, 250\}$$

und schließlich mit $\delta_{max} = 0$, um die benötigte Zeit für die Kantengewichtung ohne jegliche Einschränkung zu erhalten.

Aufgrund dieser Messungen ergab sich die in Abbildung 6.7 grafisch dargestellte Laufzeitentwicklung, in Abhängigkeit der gegebenen Parameter.

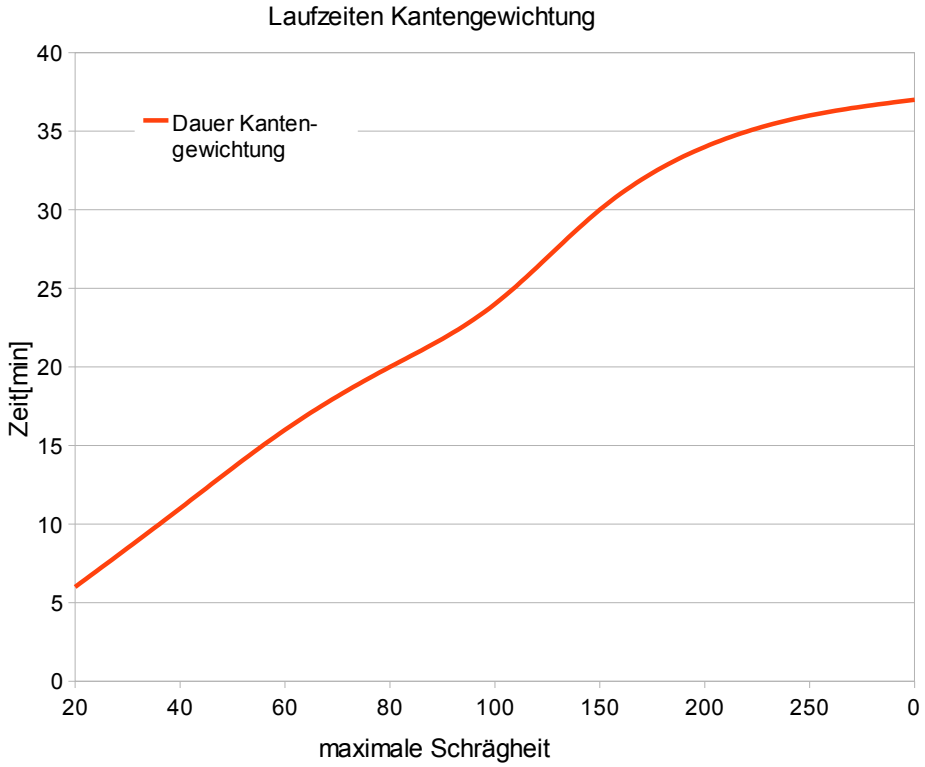

Abbildung 6.7: Gemessene Laufzeiten bei Reduzierung der gewichteten Kanten

7 Diskussion

Zuletzt wurden die Ergebnisse der praktischen Tests, des Tree-Coffee Programms, vorgestellt. Im nächsten Schritt werden diese Tests analysiert, um einen Eindruck über Effizienz und Qualität des Algorithmus zu gewinnen. Es wird erläutert welche Rückschlüsse sich aus diesen Tests ziehen lassen und diskutiert, welche Möglichkeiten bestehen, um Tree-Coffee noch weiter zu verbessern.

7.1 Diskussion der Benchmarkergebnisse

Bevor die Ergebnisse der einzelnen Sets betrachtet werden muss noch eine kurze Information bezüglich der gewählten Parameter erfolgen. Bei diesen verwendeten Parametern handelt es sich um das zweite von zwei getesteten Parametersets und sind demzufolge noch vergleichsweise ad hoc gewählt. Diese Parameter reichen aus, um einen Trend des Benchmarks zu erhalten, sind aber noch weit entfernt von optimalen Parametern, wie sie von ClustalW oder MARNA verwendet werden. Mit weiterem Parametertuning dürfte noch eine messbare Verbesserung der Ergebnisse zu erreichen sein. Um dies zu erreichen, sind jedoch noch eine Vielzahl von Testläufen auf der BraliBase notwendig.

Die Ergebnisse der Sets $k2$ und $k3$ (Abbildung 6.2, Seite 58) liefern bereits einen guten Eindruck über die Leistungsfähigkeit des Tree-Coffee Algorithmus. Es ist zu erkennen, dass trotz der gewählten Parameter der Vergleich mit MARNA sehr positiv ausfällt. Die Qualität der erreichten Ergebnisse ist durchweg besser als die, der durch MARNA gebildeten Alignments. Bis zu einer Sequenzidentität von ca 37% für das $k2$-Set und sogar bis zu 43% auf dem $k3$-Set liefert Tree-Coffee bessere Ergebnisse als ClustalW. Damit lässt sich schon einmal feststellen, dass Tree-Coffee auf diesen beiden Sets die Erwartungen, die an diesen Algorithmus speziell im Bereich niedriger Sequenzidentitäten gestellt wurden, erfüllen kann.

Auf den Sets $k5$ und $k7$ (Abbildung 6.3, Seite 59) nimmt der Abstand zwischen Tree-Coffee und MARNA bereits ab, jedoch noch immer behält der Ansatz von Tree-Coffee insgesamt einen leichten Vorsprung. Der Unterschied zu ClustalW verhält sich im $k5$-Set ähnlich zu den beiden vorherigen Sets, aber bei den multiplen Alignments mit sieben Sequenzen wird speziell der Vorsprung im niedrigen Bereich der Sequenzidentitäten deutlich geringer. Der Einfluss der gewählten Methode zur Bildung des multiplen Alignments beginnt sich negativ auf die Ergebnisse auszuwirken. Noch deutlicher lässt sich dies bei der weiteren Erhöhung der Sequenzanzahl eines Alignments zeigen.

Die letzten beiden Sets (Abbildung 6.4, Seite 60) des Benchmarks bestätigen die Entwicklung, welche sich bei der Betrachtung der vorherigen Sets angedeutet hatte. Die Ergebnisse brechen soweit ein, dass sie nur noch in seltenen Fällen, die von MARNA er-

reichten Werte übertreffen, oder im Fall von 15 Sequenzen sogar überhaupt nicht mehr. Es kann jedoch nicht behauptet werden, dass Tree-Coffee auf diesen Sets nicht funktioniert, denn die Ergebnisse bewegen sich zwar inzwischen unterhalb von MARNA und auch ClustalW in einem großen Bereich, liefern jedoch insgesamt weiterhin akzeptable Resultate. Betrachtet man nun die Ursache dieser Entwicklung zeigt sich auch, dass es eine Lösung für dieses Problem gibt.

Ursache der Entwicklung

Die Erklärung für diese Entwicklung liegt speziell im Verfahren, dass verwendet wird um das multiple Alignment zu bilden. Aktuell wird nach jedem Alignmentschritt entlang des Guide-Trees eine Konsensusstruktur für das erstellte Alignment gebildet. Daraus resultiert eine fortschreitende Reduzierung der Anzahl der Arcs, die in diesen Strukturen existieren. Das bedeutet für jeden Alignmentschritt stehen sukzessive weniger Strukturinformationen zur Verfügung, da die Informationen über nicht alignierte Strukturen komplett entfernt werden. Dieser Nachteil wirkt sich vor allem bei Alignments mit niedriger Sequenzidentität negativ aus, denn sobald die Strukturen komplett entfernt wurden handelt es sich eigentlich nur noch um ein Sequenzalignment, das in diesen Identitätsbereichen nicht sonderlich gut funktioniert.

Hier zeigt sich eine erste deutliche Schwachstelle des Tree-Coffee Ansatzes, die es gilt zu beseitigen.

Behebung dieses Problems

Es muss erreicht werden, dass während des gesamten Alignmentprozesses alle Strukturinformationen erhalten bleiben. Aus diesem Grund muss eine andere Methode, als die Bildung einer Konsensusstruktur gewählt werden. Dies ist allerdings nicht so einfach wie es auf den ersten Blick vielleicht erscheinen mag. Jede realisierte Strukturelle Kante eines Alignments hat Auswirkungen auf die Möglichkeiten, die für die folgenden Alignments zur Verfügung stehen. Das hat zur Folge, dass bei der Bildung eines Alignments nicht nur die Ähnlichkeit maximiert werden muss, sondern auch Rücksicht auf die vorherigen Alignments genommen werden muss.

Um hierfür eine effektive und praktikable Lösung zu finden müssen noch einige Überlegungen für diesen Schritt des Algorithmus angestellt werden.

Eine weitere Beobachtung

Bei der Betrachtung der Benchmarkresultate konnte beobachtet werden, wie sehr die gewählten Strukturen Einfluss auf das Ergebnis nehmen können. So war es möglich aufgrund der Strukturen selbst für Sequenzidentitäten $\leq 30\%$ teilweise eine mehr als 80%ige Übereinstimmung zum Referenzalignment zu erreichen. Leider war auch das Gegenteil zu beobachten, wo aufgrund der verwendeten Strukturen selbst sehr ähnliche Sequenzen nur sehr schlecht aligniert werden konnten. Damit diese Fälle verhindert werden können wäre ein Eingriff in die gewählten Strukturen nötigt, was allerdings

dem Ansatz, mit festen Eingabestrukturen zu arbeiten, widersprechen würde. Dennoch kann darüber nachgedacht werden, wie ein solcher Eingriff aussehen könnte, sollte die Entscheidung getroffen werden in diesen Fällen die festen Strukturen zu vernachlässigen.

Ein möglicher Eingriff in die Eingabestrukturen

Die Entwicklung eines Verfahrens zur Analyse der gegeben Strukturen und Sequenzen, würde die Möglichkeit bereitstellen, zu erkennen, ob der Einsatz der Strukturen in dieser Kombination vorteilhaft wäre. Sofern dies erfüllt ist wäre keine Änderung an den Strukturen nötig. In allen anderen Fällen jedoch könnte darüber nachgedacht werden weitere Strukturen vorherzusagen, die einen weniger negativen Einfluss auf das Alignment haben werden. Es müsste dabei allerdings eine Möglichkeit eingearbeitet werden, dieses Verfahren zu untersagen und unter allen Umständen die Eingabestrukturen zu verwenden. Mit diesem Kompromiss könnte an der Idee der festen Eingabestrukturen weiterhin festgehalten und zusätzlich eine gewisse Flexibilität, in der Wahl der Strukturen, gegeben werden.

Speziell bei multiplen Alignments mit einer geringen Sequenzanzahl wäre dieses Vorgehen von Vorteil, da der Einfluss eines paarweisen Alignments, das aufgrund schlechter Strukturen fehlerhaft ist, deutlich größer ausfällt als bei umfangreicheren multiplen Alignments.

Schlussfolgerung

Diese erste Implementation des Tree-Coffee Ansatzes konnte zeigen, dass die Idee, welche in diesem Algorithmus angewandt wird, funktioniert. Die Qualität der erreichten Ergebnisse waren für die erste Version des Programms zufriedenstellend und haben gezeigt wo Schwachstellen vorhanden sind, die einen negativen Einfluss auf die Ergebnisqualität haben. Dies liefert auch die nötigen Erkenntnisse, um diesen Ansatz noch weiter zu verbessern und eine Steigerung der Effektivität zu erreichen. Nach Bearbeitung dieser Punkte dürfte mit Tree-Coffee ein guter Algorithmus für multiples Sequenz-Struktur-Alignment zur Verfügung stehen, der für eine große Bandbreite von Sequenzidentitäten gute Ergebnisse liefert.

7.2 Diskussion der gemessenen Laufzeiten

Nach den Tests, bezüglich der Qualität der Ergebnisse, wurde die Effizienz des Algorithmus überprüft. Von großer Bedeutung für den Tree-Coffee Ansatz ist die Laufzeit, welche für die paarweise Sequenzanalyse aufgewendet wird. Dieser Schritt allein ist notwendig für das Erzielen guter Ergebnisse des multiplen Alignments und sollte daher auch möglichst effizient durchgeführt werden können.

Effizienz der paarweisen Analysen

Die Testläufe haben gezeigt, dass die Anwendung des Inside- beziehungsweise Outsidealgorithmus äußerst effizient gelingt. Auch bei stetig steigender Sequenzlänge und der damit verbundenen steigenden Anzahl von Arcs in den Strukturen, ist die Steigerung der Laufzeit beinahe unwesentlich, speziell im Vergleich zur Laufzeit der Kantengewichtung (Abbildung 6.5, Seite 62). Die Gewichtung der strukturellen Kanten benötigt nur eine sehr geringe Laufzeit nach einmaliger Berechnung der Alignments, mit Hilfe der beiden verwendeten Rekursionen. Der größte Kostenfaktor der Kantengewichtung ist also in der Gewichtung der sequentiellen Kanten zu suchen. Die Ursache hierfür ist mit dem zweiten Arbeitsschritt, der nötig ist, um eine sequentielle Kante zu gewichten, schnell gefunden. Die zusätzlichen Alignments, die gebildet werden müssen, um das beste Alignment zu finden in dem diese Kante realisiert, und mit deren Hilfe auch die Strukturen in die Bewertung einbezogen werden können, die oberhalb dieser Kante verlaufen, sind die Ursache für diese dramatisch steigende Laufzeit. In dieser Form ist die Gewichtung der Kanten nur wenig praktikabel einsetzbar, da alleine die Zeit dafür um ein vielfaches größer ausfällt, als die eigentliche Bildung des Alignments.

Verbesserung der Laufzeit für die Kantengewichtung

Eine erste Idee, um den Aufwand der Kantengewichtung zu reduzieren, wurde bereits in dieser Version des Algorithmus eingeführt. Erreicht wurde dies durch eine Reduktion der Anzahl gewichteter Kanten. Zu diesem Zweck wurde die maximale Kantenschrägheit definiert, die besagt ob für eine Kante die Berechnung eines Gewichts notwendig ist, oder vernachlässigt werden kann. Die erreichte Laufzeitreduzierung wurde anhand zwei langer Sequenzen, für unterschiedliche Werte dieser Kantenschrägheit gemessen und in Abbildung 6.7, auf Seite 64, zusammengefasst. Es stellte sich dabei heraus, dass diese Einschränkung besonders dann deutliche Auswirkungen zeigt, falls die Differenz zwischen der durchschnittlichen Länge der Sequenzen und dem gewählten Wert nicht zu gering ausfällt. Besonders bei hohen Sequenzidentitäten kann dieser Wert sehr gering gewählt werden, ohne dabei Einfluss auf das Ergebnis zu nehmen, da bei Ähnlichen Sequenzen nur sehr wenige Gaps eingefügt werden. Aber auch bei niedrigen Identitäten kann diese Einschränkung, sofern passend gewählt, einen deutlichen Geschwindigkeitsvorteil erreichen, ohne dabei das Ergebnis zu verschlechtern. Obwohl mit dieser Kantenschrägheit bereits eine Verbesserung der Laufzeiten erreicht werden kann, wäre es nötig diese Zeit noch weiter zu verkürzen.

Spontan würden sich zwei weitere Veränderungen der Kantengewichtung anbieten, die die Laufzeit noch weiter verbessern können. Eine davon bezieht sich auf die sequentiellen Kanten, die Teil des besten Alignments sind.

Zur Erinnerung, bei der ersten Anwendung des Insidealgorithmus, werden bereits alle Teilalignments und das komplette Alignment berechnet, um die maximale Ähnlichkeit zu erhalten. Es wäre sehr effizient möglich, durch diese berechneten Werte, bereits ein Traceback durchzuführen. Mit diesem Traceback könnten die sequentiellen Kanten des besten Alignments erkannt werden. Das Gewicht für diese Kanten wäre dann die maximale Ähnlichkeit des kompletten Alignments. Die Durchführung des Tracebacks kann sehr effizient erfolgen und würde damit sofort ein Gewicht für mehrere Kanten liefern, die dann im späteren Verlauf der Kantengewichtung nicht extra betrachtet werden müssten. Bei wie vielen Kanten es sich im besten Alignment um sequentielle Kanten handelt ist variabel. Es hängt von den Strukturen der Sequenzen ab und wie viele davon aligniert werden. Dieses Verfahren würde im besten Falle, die Anzahl von Kanten gewichten können, die der Länge der kürzeren Sequenz entspricht, was nur einen Bruchteil aller möglichen Kanten darstellt. Dennoch würde sich die Verwendung anbieten, da so zumindest ein Teil der Kanten korrekt gewichtet werden kann, ohne für jede Kante neue Alignments zu berechnen.

Eine weitere Möglichkeit bestünde durch die Einführung einer weiteren Heuristik, die während der Gewichtung einer Kante abschätzt, ob noch eine Verbesserung des Scores zu erwarten ist oder nicht. Aktuell werden für jede Kante, die gewichtet werden muss ein vollständiges Präfix- und Suffixalignment, bis zum Ende oder Anfang der Sequenzen, berechnet, um die Maximierung durch die Outsidescores der oberhalb verlaufenden Arcs zu ermöglichen. Es wäre möglich diese beiden Alignments immer nur so weit zu berechnen, bis der nächste Maximierungsschritt möglich ist. Damit könnte dann ein Wert für diese Kante berechnet werden, der dazu verwendet werden kann zu entscheiden, wie wahrscheinlich eine Verbesserung des Wertes zu erwarten ist. Dies kann beispielsweise durch einen Vergleich, des erreichten Wertes mit dem maximalen Wert des Alignments, erreicht werden. Sollte bis dahin ein gesetzter Wert nicht überschritten sein, können die Alignments fortgeführt werden bis der nächste Maximierungsschritt ermöglicht wird, und erneut der Vergleich durchgeführt werden. Auf diese Weise würde sich für einen Großteil der Kanten die benötigte Zeit für die Berechnungen der Alignments verkürzen und somit die Gesamtdauer der Kantengewichtung reduziert werden.

Diese beiden Methoden können zusätzlich zur bereits eingeführten Kantenschrägheit verwendet werden. Mit der Kombination dieser Möglichkeiten sollte sich die Dauer, der Kantengewichtung, auf ein akzeptables Niveau senken lassen, ohne dabei die Qualität der Kantengewichte zu sehr zu beeinflussen.

Effizienz der Bildung des multiplen Alignments

Der Test, um die Entwicklung der Gesamtlaufzeit des multiplen Alignments zu testen, hat gezeigt, dass die verbleibenden Schritte nach der paarweisen Analyse sehr effizient arbeiten (Abbildung 6.6, Seite 63). Die Dauer der Konsistenzerweiterung bewegt sich selbst bei einer Sequenzanzahl von 15 Sequenzen in einem sehr akzeptablen Rahmen

und hat bei der aktuell benötigten Laufzeit nur einen sehr geringen Einfluss auf die Gesamtdauer des Algorithmus. Auch die Bildung des multiplen Alignments ist durch die Verwendung des Insidealgorithmus äußerst effizient, und bedarf in dieser Form keiner weiteren Verbesserungen. Der überproportional größte Bedarf an Laufzeit besteht also für den Algorithmus in der Gewichtung der Kanten. Dies war nicht anders zu erwarten, da genau diese Gewichtung der wichtigste Punkt eines konsistenzbasierten Ansatzes darstellt. Zum aktuellen Zeitpunkt jedoch ist das Zeitverhältnis zwischen den einzelnen Schritten von Tree-Coffee noch nicht ausreichend.

Schlussfolgerung

Tree-Coffee stellt einen, in größten Teilen, sehr effizienten Ansatz für das multiple Sequenz-Struktur-Alignment zur Verfügung. Die Tests haben gezeigt, dass einzig die Kantengewichtung noch weiterer Verbesserungen bedarf, um Tree-Coffee eine gute Position zwischen den Algorithmen für dieses Problem einnehmen zu lassen.

Für eine erste funktionsfähige Version von Tree-Coffee, waren sowohl die Qualität der Ergebnisse, als auch die Effizienz, mit der diese erreicht wurden, zufriedenstellend. Es ist abzusehen, dass mit der weiteren Entwicklung dieses Algorithmus ein sehr gutes Programm entstehen kann, das den Vergleich mit diversen Konkurrenten auf diesem Gebiet nicht zu scheuen brauch.

8 Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde ein von Rolf Backofen und Sebastian Will entwickeltes Verfahren, genannt Tree-Coffee, für multiples Sequenz-Struktur-Alignment vorgestellt und implementiert. Die Arbeit sollte in erster Linie einen Anhaltspunkt für die Qualität, des von Tree-Coffee verwendeten Verfahrens ermitteln, und aufzeigen, welche Punkte des Algorithmus noch weiterer Aufmerksamkeit benötigen.

Aufgrund des frühen Stadiums der Entwicklung, mussten einige Aspekte des Algorithmus noch ausführlicher betrachtet und definiert werden. Es ist dabei gelungen diesen Ansatz soweit zu konkretisieren, um einen Punkt zu erreichen, an dem erste Untersuchungen durchgeführt werden konnten, die zeigen sollten, wie gut das entwickelte Verfahren funktioniert.

Diese Evaluation hat gezeigt, welches Potential Tree-Coffee für das vorhandene Problem bietet. Die Qualität der Ergebnisse geben Aufschlüsse, aus denen sich ableiten lassen, dass die Ideen, auf deren Basis Tree-Coffee entwickelt wurde, die gewünschten Effekte entwickeln und somit eine gute Basis darstellen, die für die weitere Entwicklung notwendig ist.

Bei den unternommenen Tests kristallisierten sich Schwachstellen heraus, die es noch zu beseitigen gilt. Diese aufgezeigten Probleme erstreckten sich jedoch nur auf explizite Designentscheidungen und nicht auf generell falsche Annahme des Verfahrens. Das Fazit dieser Arbeit lässt sich demnach in folgender Weise ausdrücken.

Mit Tree-Coffee entsteht ein sehr gutes Verfahren für das Problem der multiplen Sequenz-Struktur-Alignments, das bereits jetzt viele Erwartungen erfüllen konnte. Die festgestellten Ansätze, zur Verbesserung des Algorithmus, werden in der weiteren Entwicklung dafür sorgen, dass der Algorithmus an den richtigen Stellen überarbeitet wird und sich somit zu einem sehr effektiven Werkzeug weiterentwickeln kann. Diese Effektivität wird dabei nicht durch große Opfer, im Bereich der Laufzeiten, erreicht werden, sondern durch geschickte Kombination diverser etablierter Verfahren, im Bereich der Bioinformatik.

A BraliBase Referenzalignments

RNA family	k2	k3	k5	k7	k10	k15	∑
5S_rRNA	1162	568	288	150	90	50	2308
5_8S_rRNA	76	45	17	5	3	0	146
Cobalamin	188	61	15	4	0	0	268
Entero_5_CRE	48	32	19	10	8	5	122
Entero_CRE	65	38	20	13	8	4	148
Entero_OriR	49	31	17	11	8	4	120
gcvT	167	67	22	12	3	1	272
Hammerhead_1	53	32	9	1	0	0	95
Hammerhead_3	126	99	52	32	17	12	338
HCV_SLIV	98	63	36	26	16	10	249
HCV_SLVII	51	33	19	13	10	7	133
HepC_CRE	45	29	18	11	7	3	113
Histone3	84	59	27	11	7	6	194
HIV_FE	733	408	227	147	98	56	1669
HIV_GSL3	786	464	246	151	95	61	1803
HIV_PBS	188	124	76	55	38	25	506
Intron_gpII	181	82	35	22	11	4	335
IRES_HCV	764	403	205	146	83	47	1648
IRES_Picorna	181	117	75	53	35	25	486
K_chan_RES	124	40	2	0	0	0	166
Lysine	80	48	30	17	7	3	185
Retroviral_psi	89	57	34	24	17	11	232
SECIS	114	67	33	16	11	6	247
sno_14q I_II	44	14	1	0	0	0	59
SRP_bact	114	76	39	19	12	7	267
SRP_euk_arch	122	94	42	21	12	6	297
S_box	91	51	25	12	7	2	188
T-box	18	8	0	0	0	0	26
TAR	286	165	92	62	42	28	675
THI	321	144	69	32	17	5	588
tRNA	2039	1012	461	267	143	100	4022
U1	82	65	26	16	6	0	195
U2	112	83	38	22	14	7	276
U6	30	21	14	7	1	0	73
UnaL2	138	71	43	20	7	0	279
yybP-ykoY	127	64	33	18	12	8	262
∑	8976	4835	2405	1426	845	503	18990

Tabelle A.1: Anzahl und Familien der Referenzalignments [WMS06]

B Testdaten für Laufzeiten

Nr.	Familie	Sequenzen	Länge	APSI
1	5S_rRNA	M19950.1_1-120 X52300.1_5-122	120 118	37%
2		K03160.1_1-118 X00073.1_1-117	118 117	95%
3	Cobalamin	M34485.1_360-150 AP003013.2_206912-207118	211 207	36%
4		AE017037.1_59439-59627 AE017277.1_42695-42883	189 189	95%
5	Intron_gpII	X04465.1_117436-117320 AF143425.1_760-901	117 142	28%
6		AJ277126.1_2879-2953 AF029891.1_4522-4594	75 73	84%
7	tRNA	Z74797.1_2502-2432 L07095.1_4950-5016	71 67	16%
8		X04465.1_90332-90261 D13107.1_530-459	72 72	92%
9	yybP-ykoY	AE004877.1_3967-4112 AE008848.1_13783-13922	146 140	27%
10		AP005044.1_134132-134252 AL939112.1_94442-94322	121 121	90%

Tabelle B.1: Verwendete Daten für Laufzeitmessungen der paarweisen Sequenzanalyse

Familie: yybP-ykoY			
Nr.	**Set**	**Dateiname**	**ØLänge**
1	$k2$	yybP-ykoY.apsi-50.sci-95.no-1.raw.fa	114
2	$k3$	yybP-ykoY.apsi-50.sci-87.no-1.raw.fa	118
3	$k5$	yybP-ykoY.apsi-52.sci-81.no-1.raw.fa	133
4	$k7$	yybP-ykoY.apsi-44.sci-104.no-1.raw.fa	124
5	$k10$	yybP-ykoY.apsi-45.sci-69.no-1.raw.fa	127
6	$k15$	yybP-ykoY.apsi-45.sci-69.no-2.raw.fa	127

Tabelle B.2: Verwendete Testdateien zur Laufzeitmessung multipler Alignments

Familie: SRP_euk_arch			
Name	**Sequenz/Struktur**	**Länge**	**APSI**
Z29106.1_1-301	GCCGAGCUUAGUAAUGUGGGCUUGUAACCCAAAUGGGGGCAUUAAUGUGGUGGAAUGUUGGGCUG UACCAGAUGGUUGGGCUUGGUGGGCUAAUUUCUGACUUGCCCUAUCCAAGCUUAGAGUUGGAUCA UGUGGCCCAAUUGAAGAAAUGGACUACUUGAUUCGUAAAGUGGGGCGGAAUGCGUGAGGCUGGUU UCACAGAGCAACGAUAAGUUUCGCUCUAAGCGGUGGAAGGAUAACGAGUCGGUGUUCUCUGAGUC CAGUAACGCCUGAUGGGUUGCUCCAAUUAAACCACCAUUUU	301	61%
	((((((((((.....))))))).....(((....)))))).....(((((((...(((((((.. ..(((...(((((((((..(.(((.....(((((((.(((.((((.((((((....((((((.((((.(((.((....)).))).)))).))))))).....(((((((.(((....(((... ...)))....)))....))))))))))))..)))))))....)))))))).))))..)..)))))).....)))...)))...)).)))))....)))))))...		
X65985.1_1-302	GGCCUUAGCAACGUGGGCCUGUAACCCAAGUGGGGGCAUGUGGGAAAUGGAACAUUGGGUCAGCC CAGUGGAUCGGGUCCAGUGUUAGCUGCUUACUGGUCUGCCCAUUCCAAGCCGGGAGUUGGGCUGA GUGACCUGGGCGAAGGCCUGGGUUGCGCAGCUCCUAGAGUGGAGGGCAAUGCGUGAGGCUGGUUU CACAGAGCAGCGACUACCUCCCGCUCCCGGCAGUGGAAGGAUAACGGGCCGGUGCUACUUGGUCC ACCAUGUUCACUGGUCCUGACUCUUAAUCGGACCAUUUCCUU	302	
	.(((((....((.((((.......)))).)).)))))....(((((((((..((((((....)))))))..(((((.((((((.((((.....(((((((((.((.(((((.(((((((((.((((((.(. .(((..(((....)))..)))..).))))))).......((((....(((.((...(((......)))..)))))....)))))).)))))))))..)))))))....))))))))))..(((....))) ....))))))..)))))...........)))))))).		

Tabelle B.3: Verwendete Sequenzen zur Laufzeitanalyse bei unterschiedlichen maximalen Kantenschrägheiten

Literaturverzeichnis

[CB00] Clote, Peter und Rolf Backofen: *Computational Molecular Biology: An Introduction*. Mathematical and Computational Biology. Jon Wiley & Sons, Chichester, August 2000. series editor S. Levin. 290 pages.

[Cou02] Couzin, Jennifer: *Breakthrough of the year. Small RNAs make big splash.* Science, 298(5602):2296–7, 2002.

[Edd95] Eddy, SR: *Multiple alignment using hidden Markov models.* Band 3, Seiten 114–20, 1995.

[Edd01] Eddy, S. R.: *Non-coding RNA genes and the modern RNA world.* Nat Rev Genet, 2(12):919–29, 2001.

[FD87] Feng, D. F. und R. F. Doolittle: *Progressive sequence alignment as a prerequisite to correct phylogenetic trees.* J Mol Evol, 25(4):351–60, 1987.

[Got96] Gotoh, O.: *Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments.* 264(4):823–38, 1996.

[GVR04] Giegerich, Robert, Björn Voss und Marc Rehmsmeier: *Abstract shapes of RNA.* 32(16):4843–51, 2004.

[GWW05] Gardner, Paul P., Andreas Wilm und Stefan Washietl: *A benchmark of multiple sequence alignment programs upon structural RNAs.* 33(8):2433–9, 2005.

[HBS04] Hofacker, I. L., S. H. Bernhart und P. F. Stadler: *Alignment of RNA base pairing probability matrices.* Bioinformatics, 20(14):2222–7, 2004.

[HLSG05] Havgaard, Jakob Hull, Rune B. Lyngso, Gary D. Stormo und Jan Gorodkin: *Pairwise local structural alignment of RNA sequences with sequence similarity less than 40Bioinformatics, 21(9):1815–24, 2005.*

[HTGK03] Höchsmann, Matthias, Thomas Töller, Robert Giegerich *und* Stefan Kurtz*:* Local Similarity in RNA Secondary Structures*. In:* Proceedings of Computational Systems Bioinformatics (CSB 2003)*, Seite 159. IEEE Computer Society, 2003.*

[HTHI95] Hirosawa, M., Y. Totoki, M. Hoshida *und* M. Ishikawa*:* Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput Appl Biosci, 11(1):13–8, 1995.*

[JLMZ02] Jiang, Tao, Guohui Lin, Bin Ma *und* Kaizhong Zhang*:* A General Edit Distance between RNA Structures. *9(2):371–88, 2002.*

[JWZ95] Jiang, T., J. Wang *und* K. Zhang*:* Alignment of trees - an alternative to tree edit. *Theoretical Computer Science, 143(1):137–148, 1995.*

[KE03] Klein, Robert J. *und* Sean R. Eddy*:* RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics, 4(1):44, 2003.*

[NHH00] Notredame, C., D. G. Higgins *und* J. Heringa*:* T-Coffee: A novel method for fast and accurate multiple sequence alignment. *302(1):205–17, 2000.*

[NW70] Needleman, S. B. *und* C. D. Wunsch*:* A general method applicable to the search for similarities in the amino acid sequence of two proteins. *48(3):443–53, 1970.*

[PL88] Pearson, W. R. *und* D. J. Lipman*:* Improved tools for biological sequence comparison. *85(8):2444–8, 1988.*

[San85] Sankoff, David*:* Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math., 45(5):810–825, 1985.*

[SB03] Siebert, Sven *und* Rolf Backofen*:* MARNA: A Server for Multiple Alignment of RNAs. *Seiten 135–140, October 2003.*

[SB05] Siebert, Sven *und* Rolf Backofen*:* MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics, 21(16):3352–9, 2005.*

[SBH+94] Sakakibara, Yasubumi, Michael Brown, Richard Hughey, I. Saira Mian, Kimmen Sjolander, Rebecca C. Underwood *und* David Haussler*:* Recent Methods for RNA Modeling Using Stochastic Context-Free Grammars. *1994.*

[SN87] Saitou, N. *und* M. Nei*:* The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol, 4(4):406–25, 1987.*

[SW81] Smith, T.F. *und* M.S. Waterman*:* Comparison of Biosequences. *Adv. appl. Math., 2:482–489, 1981.*

[SZ90] Shapiro, B. A. *und* K. Z. Zhang*:* Comparing multiple RNA secondary structures using tree comparisons. *Comput Appl Biosci, 6(4):309–18, 1990.*

[THG94] Thompson, J. D., D. G. Higgins *und* T. J. Gibson*:* CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *22(22):4673–80, 1994.*

[uPS98] Peter Stadler, Ivo Hofacker und*:* Vienna RNA Package. *Paper as Print Copy, 1998.*

[WMS06] Wilm, Andreas, Indra Mainz *und* Gerhard Steger*:* An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol Biol, 1:19, 2006.*

[ZS81] Zuker, M. *und* P. Stiegler*:* Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *9(1):133–48, 1981.*

Glossar

Algorithmus	Ein Algorithmus bezeichnet eine genaue Vorschrift, mit deren Hilfe sich ein bestimmtes Problem lösen lässt.
Alignment	Ein Alignment wird in der Bioinformatik dazu verwendet die Homologie zwischen Nukleotid- oder Proteinsequenzen zu berechnen.
Benchmark	Ein Benchmark dient dem Vergleich der Leistung von unterschiedlichen Programmen. Zu diesem Zweck werden vorgegebene Datensätze berechnet und die Ergebnisse der verschiedenen Programme dann verglichen.
Bioinformatik	Die Bioinformatik verbindet die Bereiche Informatik und Biologie, um mit den Möglichkeiten der Informatik Verfahren zu entwickeln, die bei der Lösung von biologischen Problemen helfen sollen.
DNA	Speichert die Erbinformationen eines Organismus in einer Kette bestehend aus Nukleotiden.
dynamisches Programmieren	Kann man ein Problem in mehrere kleinere Teilprobleme zerlegen, so ist es möglich die Lösungen der Teilprobleme zu berechnen, um mit ihrer Hilfe das Gesamtproblem zu lösen. Diesen Vorgang bezeichnet man in der Informatik als dynamische Programmierung.
Homologie	In der Biologie beschreibt die Homologie evolutionäre Ähnlichkeiten z.B. zwischen zwei RNA Molekülen.
Konsensusstruktur	Die Konsensusstruktur von zwei oder mehr Sequenzen beschreibt die Bindungen, die in allen Sequenzen vorhanden sind.
Nukleotid	Ein einzelnes Nukleotid ist ein Baustein der DNA/RNA.

Polymer Ein Polymer bezeichnet ein großes Molekül, bestehend aus einer langen Kette von einzelnen Molekülen.

Rekursion Mit Rekursion ist eine Funktion gemeint, die sich immer wieder selbst aufruft und auf diese Weise durch Verkleinerung des Problem eine Lösung findet.

RNA Eine RNA ist eine Kette von Nukleotiden, die anders als DNA keine Erbinformationen speichert, sondern Funktionen in Zellen reguliert, oder die Translation der genetischen Informationen in Proteine übernimmt.

Sequenz Mit einer Sequenz ist im Kontext dieser Arbeit die Primärstruktur eines RNA Moleküls gemeint.

Struktur Struktur (hier im speziellen Sekundärstruktur) beschreibt die Bindungen die innerhalb eines RNA Moleküls gebildet werden.

Strukturvorhersage Nimmt man eine RNA Sequenz, so kann eine Strukturvorhersage dazu verwendet werden, die Menge von Bindungen zwischen den einzelnen Nukleotiden vorherzusagen. Dies muss jedoch nicht zwangsläufig die biologisch korrekte Struktur sein.

Traceback Ein Traceback liefert für eine gefundene Lösung den Weg, wie diese Lösung erreicht wurde.

Danksagung

Abschließend möchte ich noch die Gelegenheit nutzen mich bei einigen Personen, die mich während des Studiums unterstützt, oder mir dieses erst ermöglicht haben, bedanken.

Der größte Dank gebührt dabei in erster Linie meiner Familie und meinen Freunden, ohne deren Unterstützung während meiner gesamten Studienzeit, dies alles überhaupt nicht möglich gewesen wäre. Speziell in Zeiten des Zweifels, oder während der anstrengenden Klausurzeiträume, haben sie mich immer wieder angespornt nicht aufzugeben, was eine große Stütze darstellte.

Speziell im Rahmen dieser Diplomarbeit gilt mein Dank natürlich auch Prof. Dr. Rolf Backofen und Dr. Sebastian Will, die mir die Möglichkeit gegeben haben mich mit diesem interessanten Thema zu beschäftigen und die Bewertung dieser Arbeit übernehmen werden. Die Bioinformatik hat mich während meines gesamten Hauptstudiums begleitet und mein Interesse für dieses Gebiet geweckt.

Dr. Sebastian Will möchte ich auch in seiner Funktion als Betreuer dieser Diplomarbeit danken, der mich bei Fragen immer unterstützt, oder durch seine Anregungen inspiriert hat, um ein möglichst gutes Ergebnis zu erzielen.

Ich möchte mich auch bei all denjenigen bedanken, die ihre Zeit geopfert haben, um diese Arbeit zu lesen und mich auf Fehler jeglicher Art aufmerksam zu machen.

Zuletzt möchte ich meinen Kommilitonen Daniel Schüssele und Jörg Bruder danken, die mich das ganze Studium begleitet haben und auf diese Weise auch zu guten Freunden geworden sind. Danke für die gemeinsame Arbeit während des Studiums und die Möglichkeit sich in einer netten Umgebung mit dem Lernstoff gemeinsam auseinanderzusetzen. Auch danke für die vielen Unternehmungen ausserhalb der Uni, die immer für großen Spaß gesorgt haben. Bleibt nur zu sagen, dass diese Freundschaft hoffentlich auch weit über die Zeit an der Uni hinaus erhalten bleiben wird und irgendwann der Moment erreicht ist, an dem man sich gemeinsam an diese Zeit erinnert.