

Ausarbeitung im Rahmen des Seminars

**- Aktuelle Themen der Bioinformatik -**

Johann-Wolfgang-Goethe-Universität Frankfurt am Main

Juniorprof. Dr. Dirk Metzler

**Evolution und Vorhersage**

**von**

**RNA-Sekundärstruktur**

von Matthias Wirth

# Gliederung

1. Einleitung	- 03 -
2. Evolution von RNA-Sekundärstruktur	
2.1. Einleitung	- 04 -
2.2. Das TKF91-Modell	- 04 -
2.3. Der TKF91-Structure Tree	- 09 -
2.4. Analyse des Modells	- 13 -
2.5. Auswertung	- 15 -
3. Vorhersage von RNA-Sekundärstruktur	
3.1. Einleitung	- 17 -
3.2. Der Akutsu-Algorithmus	- 19 -
3.3. Nearest-Neighbour Thermodynamik Regeln	- 23 -
3.4. Berechnung minimaler Energien von RNA-Substrukturen	- 24 -
3.5. Optimale Energie eines Pseudoknoten	- 27 -
3.6. Auswertung	- 34 -
4. Zusammenfassung und Resumée	- 35 -
5. Quellen	- 36 -

## 1. Einleitung

Diese Seminararbeit beschäftigt sich mit zwei wissenschaftlichen Arbeiten zum Thema RNA-Sekundärstrukturevolution und RNA-Sekundärstrukturvorhersage.

In den letzten Jahren hat das Interesse an der RNA extrem zugenommen. Der Grund dafür ist, dass die RNA eine wichtige Rolle innerhalb von genetischen und katalytischen Regulationsprozessen zu spielen scheint, und scheinbar nicht nur in der Zelle vorhanden ist, um Proteininformationen aus dem Nukleus in das Cytosol zu transportieren. In den ersten Tagen nach der vollständigen Analyse des menschlichen Genoms sind diejenigen Abschnitte der DNA, die nicht für Proteine kodieren, als sog. *junk-DNA* in den Hintergrund gestellt worden. Man hat ihnen keine besondere Funktion zugemessen, da man der Annahme gewesen ist, dass Teile der DNA, die nicht für Proteine kodieren, keine Funktion haben können. Grund für diese Annahme ist unter anderem gewesen, dass frühere Studien an dem Bakterium *E. coli* gezeigt haben, dass Proteine im Wesentlichen die Hauptrolle bei allen katalytischen Prozessen wie auch bei deren Regulation spielen. Nur pflegt dies wohl nicht bei allen Organismen so zu sein. Experimentelle Analysen und Versuche haben gezeigt, dass es eine Vielzahl von regulatorischen und katalytischen Prozessen gibt, an denen RNA-Moleküle beteiligt sind. Als Beispiel sind hier unter anderem die *snRNA*, die beteiligt am Splicing von Intron- und Exon-Teilen der DNA ist, die *Ribonukleasen*, die eine Rolle in der Regulation der tRNA-Synthese spielen, die *rRNA*, die einen wesentlichen Bestandteil der Ribosomen ausmacht oder die *miRNA*, die eine Vielzahl von regulatorischen Funktionen besitzt, zu nennen.

Allen diesen RNA-Molekülen ist gemein, dass sie nicht über die Suche nach *open reading frames* (ORF) gefunden werden können. Als sog. *non-coding RNA* (ncRNA) befinden sich diese RNA-Teile nicht in den Regionen, die für Proteine kodieren und können demnach auch nicht durch die Suche nach der typischen Startstelle für die Proteinsynthese gefunden werden. Da die *ncRNA* aber wohl essentiell wichtig erscheint, muss also ein Weg gefunden werden, um diese identifizieren zu können.

## 2. Evolution von RNA-Sekundärstruktur

### 2.1. Einleitung

In dem Paper von Ian Holmes [5] wird ein probabilistisches Modell eingeführt, das in der Lage ist evolutionäre Ereignisse auf RNA-Sequenzen unter Berücksichtigung der Sekundärstruktur zu modellieren. Da die Sekundärstruktur eine wichtige Rolle für die funktionellen Eigenschaften von RNA spielt, ist es nötig, diese in den Modellen auch zu berücksichtigen. Frühere evolutionäre Modelle ziehen Änderungen in der RNA-Sekundärstruktur nicht in Betracht. Da aber in der Biologie Funktion mit Struktur stark gekoppelt ist, kann es sein, dass diese früheren Modelle deshalb erhebliche Schwächen aufweisen. Der in dem Paper beschriebene TKF91 Structure Tree ist aufgrund dessen in der Lage, nicht nur kovariante Punktsubstitutionen von Nukleotiden, sondern auch kovariante Insertionen und Deletionen von Basen, Basenpaaren, ganzen Stems und Multi-Stem Strukturen zu behandeln. Weiterhin ist es möglich mithilfe des Modells Alignments von Sequenzen, unter der Berücksichtigung der Sekundärstruktur, durchzuführen.

### 2.2. Das TKF91 Modell

Das TKF91 Modell von Thorne, Kishino und Felsenstein [7] stellt ein probabilistisches Modell dar, das die Evolution einer einzelnen Sequenz unter dem Einfluss von zwei Mutationsereignissen modelliert. Diese Ereignisse sind Punktsubstitutionen von einzelnen Positionen der Sequenz, bei denen eine Base durch eine andere oder sich selbst ersetzt wird, und Insertionen und Deletionen, bei denen eine neue Base in die Sequenz eingeführt wird oder eine bestehende gelöscht wird. Das Modell folgt fest gegebenen Raten für die verschiedenen Ereignisse und jedes dieser Ereignisse ist unabhängig von Ereignissen an benachbarten Positionen. Außerdem ist es zeit-reversibel, was zur Folge hat, dass man bei Vergleich zweier Sequenzen mithilfe des Modells davon ausgehen kann, dass o.B.d.A. eine der beiden Sequenzen die Ursequenz der anderen ist, und diese aus der Ursequenz über eine gegebene evolutionäre Distanz entstanden ist. Darstellbar ist dies durch die folgende Formel, wobei  $X$  und  $Y$  die beiden Sequenzen und  $t$  die evolutionäre Distanz darstellen:

$$P_{\infty}(X)P_t(Y|X) = P_{\infty}(Y)P_t(X|Y)$$

für  $\forall X, Y$  und  $t > 0$

Für die Punktsubstitutionen gilt, dass alle Positionen unabhängig voneinander evolvieren. Weiterhin nimmt man ein zugrunde liegendes Substitutionsmodell an, das in Form einer Ratenmatrix gegeben ist, in der gespeichert ist, mit welcher Wahrscheinlichkeit eine Base  $i$  durch eine Base  $j$  ersetzt wird. Die Punktsubstitution erfolgt nach dieser Formel:

$$f_{ij}(t) = \begin{cases} e^{-st} + \pi_j(1 - e^{-st}) & i = j \\ \pi_j(1 - e^{-st}) & i \neq j \end{cases}$$

$s = \text{Substitutionsrate}$

$\pi_j = \text{relative Häufigkeit für Base } j$

Für die Wahrscheinlichkeit eines Übergangs von Base  $i$  nach Base  $j$  existieren zwei Fälle. Der erste Fall beschreibt die Wahrscheinlichkeit für den Übergang von Base  $i$  in Base  $i$ . Diese setzt sich aus zwei Summanden zusammen, wobei der erste die Wahrscheinlichkeit beschreibt, dass Base  $i$  nicht substituiert wird, und der zweite die Wahrscheinlichkeit beschreibt, dass die Base  $i$  durch Base  $i$  ersetzt wird. Der zweite Fall beschreibt die Wahrscheinlichkeit des Austauschs von Base  $i$  durch Base  $j$ .

Bei der Insertion oder der Deletion von Basen wird die Sequenz als eine Abfolge von Links betrachtet. Man unterscheidet hierbei zwei verschiedene Arten von Links, dem *immortal link* und den *mortal links*. Die *mortal links* stehen für die Basen der Sequenz und sind mit deren Buchstaben beschriftet. Beide Typen von Links sind in der Lage, beim Auftreten einer Insertion, rechts von sich neue *mortal links* zu erzeugen. Der *immortal link* beschreibt den Link, der sich am linken Ende der Sequenz befindet. Er ist nicht beschriftet und besitzt die Eigenschaft nie das Ziel einer Deletion werden zu können. Dadurch ist gewährleistet, dass mindestens ein Element der Sequenz immer in der Lage ist, neue Basen in die Sequenz einführen zu können. Dies verhindert, dass die Sequenzlänge  $N$  über kurz oder lang gegen null konvergiert. Durch die Einführung des *immortal links* existieren jedoch in der Link-Sequenz  $N+1$  Positionen, an denen eine Insertion stattfinden kann und nur  $N$  Positionen, bei denen eine Deletion stattfinden kann. Damit also, aufgrund der verschiedenen Anzahl von Positionen an denen eingefügt oder gelöscht werden kann, kein Ungleichgewicht entsteht, wird vorausgesetzt, dass die Rate der Insertion, die durch den

Parameter  $\lambda$  ausgedrückt wird, kleiner der Rate für die Deletion ist, die durch den Parameter  $\mu$  ausgedrückt wird.

Anhand der gegebenen Parameter  $\lambda_n$  und  $\mu_n$  sowie der evolutionären Distanz  $t$ , lassen sich die Wahrscheinlichkeiten für bestimmte evolutionäre Prozesse errechnen, die im wesentlichen benötigt werden, um später die Wahrscheinlichkeit für bestimmte Übergänge in einem HMM oder in einer SCFG zu ermitteln. Die Parameter der Prozesse können in bestimmten Fällen unterschiedliche Werte zugewiesen bekommen, weshalb  $n$  als Indizierung für verschiedene Fälle steht. Diese Prozesse sind:

$\alpha_n$  = Wahrscheinlichkeit einer Nicht-Deletion

$\beta_n$  = Wahrscheinlichkeit einer Insertion

$\gamma_n$  = Wahrscheinlichkeit einer Insertion nach einer Deletion

$\kappa_n$  = Wahrscheinlichkeit die Sequenz fortzuführen

Für das Ereignis der Insertion unterscheidet man hierbei zwischen zwei unterschiedlichen Wahrscheinlichkeiten. Zum einen existiert eine eigene Wahrscheinlichkeit dafür, dass eine Insertion stattfindet und der *mortal link*, der diese Insertion auslöst, überlebt. Zum anderen existiert eine eigene Wahrscheinlichkeit dafür, dass der *mortal link*, der die Insertion auslöst, im Laufe der Zeit  $t$  deletiert wird. Da in diesen Fällen die Wahrscheinlichkeit für eine Insertion variiert, müssen diese unterschieden werden.<sup>1</sup>

Die Sequenzlänge im Gleichgewicht ist geometrisch verteilt mit Parameter  $\kappa_n$ . Daraus folgt, dass die Wahrscheinlichkeit für eine Sequenz der Länge  $n$  durch  $(\kappa_n)^n (1 - \kappa_n)$  gegeben ist.

Außerdem beschreibt  $M_n(i, j)$  die Substitutionswahrscheinlichkeit von Base  $i$  durch Base  $j$ .

Mithilfe einer SCFG kann nun ein Alignment von zwei Sequenzen durchgeführt werden. Die gegebene Grammatik in Abbildung 1 beschreibt die Regeln für ein Alignment zwischen zwei gegebenen Sequenzen. Für jede Produktion existiert eine bestimmte Wahrscheinlichkeit, dass diese Produktion eintritt. Diese Wahrscheinlichkeit setzt sich aus dem Produkt der

---

<sup>1</sup> Die genauen Berechnungen der einzelnen Wahrscheinlichkeiten finden sich im Paper von Ian Holmes [5], sowie im Original-Paper des TKF91-Modells [7] und wurden hier der Kürze halber ausgelassen.

Wahrscheinlichkeiten für die Ursequenz und der bedingten Wahrscheinlichkeit für die Nachfolger-Sequenz zusammen.

Betrachten wir zum Beispiel Regel 1 aus Abbildung 1:

Diese Regel besagt, dass, von der Ur- in die Nachfolger-Sequenz, keine Insertion und keine Deletion stattgefunden haben. Es ist aber durchaus möglich, dass eine Punktsubstitution einer einzelnen Base stattgefunden hat. Dementsprechend wird auch die Wahrscheinlichkeit dieser Regel bestimmt.

<i>TKF91 pair rules</i>					
Rule	lhs	→	rhs	$P(a)$	$P(d a)$
1.	$L_1$	→	$X_a Y_d L_1$	$\kappa_0 p_0(X)$	$(1 - \beta_0) \alpha_0 M_0(X, Y)$
2.			$Y_d L_1$	1	$\beta_0 p_0(Y)$
3.			$X_a L_2$	$\kappa_0 p_0(X)$	$(1 - \beta_0)(1 - \alpha_0)$
4.			$\epsilon$	$1 - \kappa_0$	$1 - \beta_0$
5.	$L_2$	→	$X_a Y_d L_1$	$\kappa_0 p_0(X)$	$(1 - \gamma_0) \alpha_0 M_0(X, Y)$
6.			$Y_d L_1$	1	$\gamma_0 p_0(Y)$
7.			$X_a L_2$	$\kappa_0 p_0(X)$	$(1 - \gamma_0)(1 - \alpha_0)$
8.			$\epsilon$	$1 - \kappa_0$	$1 - \gamma_0$

**Abbildung 1**

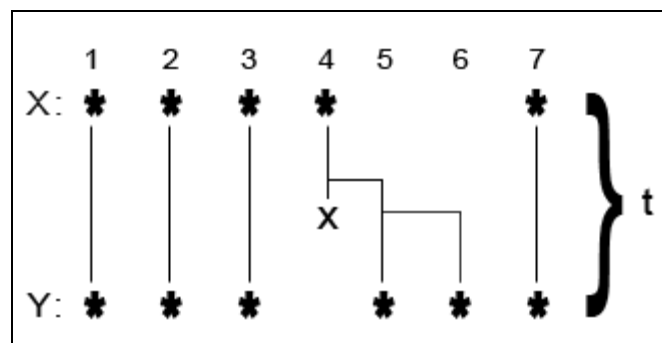
pairSCFG, mit der ein Alignment von zwei Primärsequenzen möglich ist.  $L_1$  und  $L_2$  sind Nicht-Terminale,  $X, Y$  sind Terminale aus dem Alphabet  $\{A, G, U, C\}$ .

Diese setzt sich aus der Wahrscheinlichkeit zusammen, dass die Ursequenz fortgeführt wird ( $\kappa_0$ ) und das an der betrachteten Stelle die Base  $X$  steht ( $p_0(X)$ ), mal der Wahrscheinlichkeit, dass es von der Ur- in die Nachfolger-Sequenz *keine* Insertion gibt ( $1 - \beta_0$ ), dass es weiterhin *keine* Deletion gibt ( $\alpha_0$ ) und das die Base  $X$  durch die Base  $Y$  substituiert wird ( $M_0(X, Y)$ ), wobei  $X$  die gleiche Base wie  $Y$  sein kann.

Regel 2 beschreibt eine Insertion in die Nachfolger-Sequenz, während Regel 3 für eine Deletion einer Position der Ursequenz steht.

Wichtig ist außerdem, den Unterschied von  $L_1$  zu  $L_2$  darzustellen. Wie man anhand der Grammatik erkennt, unterscheiden sich beide Produktionen nur in den Wahrscheinlichkeiten ihrer jeweiligen Regeln. In  $L_1$  wird eine Insertion mit der Wahrscheinlichkeit  $\beta_0$ , in  $L_2$  mit  $\gamma_0$

betrachtet. Nach den oben genannten Definitionen der Parameter, sagt ein Übergang von  $L_1$  nach  $L_2$  (Regel 3) also aus, dass nach einer Deletion in der Ursequenz eine andere Wahrscheinlichkeit für eine Insertion betrachtet werden muss, da die Position aus der die Insertion hervorgeht, im Laufe der vergangenen Zeit deletiert wurde. In Abbildung 2 ist solch ein Vorgang visualisiert. An Position 4 findet irgendwann im Laufe der Zeit  $t$  eine Deletion statt. Im Laufe der Zeit bis zur Deletion geht jedoch eine Insertion aus dieser Position hervor. Wie man jedoch erkennen kann, muss die Wahrscheinlichkeit dafür eine andere sein, als wenn beispielsweise eine Insertion aus Position 3 hervorgegangen wäre. Dort ist es möglich, dass die Insertion im gesamten Zeitraum  $t$  entstanden sein könnte, während dies bei Position 4 nicht möglich ist, da diese im Laufe der Zeit  $t$  verschwindet. In der Grammatik aus Abbildung 1 ist dies dadurch charakterisiert, dass der Parameter für eine Insertion von  $\beta_0$  zu  $\gamma_0$  wechselt, es also einen Übergang von  $L_1$  zu  $L_2$  gibt. Eine weitere Insertion an der nächsten Stelle würde wieder mit dem Parameter  $\beta_0$  bezeichnet, da für diese wieder der gesamte Zeitraum  $t$  zur Verfügung steht.



**Abbildung 2**  
Evolution von Sequenz X zu Sequenz Y innerhalb von Zeit  $t$ . Die einzelnen Positionen sind durchnummeriert.

Betrachten wir, wie sich dies auf die Regeln der Grammatik auswirkt:

Regel 2 bleibt in  $L_1$ , da eine Insertion, die auf eine nicht-deletierte Position folgt, im kompletten Zeitraum  $t$  erfolgen könnte. Folglich ist der Parameter, der die Wahrscheinlichkeit für eine Insertion beschreibt,  $\beta_0$ .

Regel 7 bleibt in  $L_2$ , da nach dieser Deletion eine Insertion erfolgen könnte, die nicht den gesamten Zeitraum  $t$  zur Verfügung hätte. Diese Insertion hätte die Wahrscheinlichkeit  $\gamma_0 p_0(Y)$ , woraus folgt, dass Regel 7 die Wahrscheinlichkeit  $\kappa_0 p_0(X)(1-\gamma_0)(1-\alpha_0)$  trägt.

Regel 3 wechselt zu  $L_2$ , da nach einer Deletion, wie bereits erwähnt, die Möglichkeit einer Insertion besteht, die nicht den gesamten Zeitraum  $t$  zur Verfügung hat.

Regel 6 wechselt wieder zu  $L_1$ , da die nachfolgenden Ereignisse im gesamten Zeitraum  $t$  entstehen können.

Die in Abbildung 1 gezeigte SCFG kann auch als pairHMM beschrieben werden. Mithilfe des Viterbi-Algorithmus [3] kann das beste Alignment, der höchst-wahrscheinlichste Pfad des HMM's, für zwei gegebene Sequenzen bestimmt werden.

### 2.3. Der TKF91 Structure Tree

Der TKF91 Structure Tree stellt eine Erweiterung des TKF91 Modells dar. Er ist in der Lage RNA-Sekundärstrukturen zu betrachten und in diesen evolutionäre Ereignisse zu modellieren. Der Structure Tree stellt einen gewurzelten Baum dar, in dem jeder seiner Knoten einen Grad  $\leq 3$  besitzt. Die Struktur des Baumes beschreibt den Aufbau der RNA-Sekundärstruktur einer gegebenen Sequenz. In dem Baum existieren vier verschiedene Arten von Knoten: singlet-, paired-, loop- und stem-Knoten.

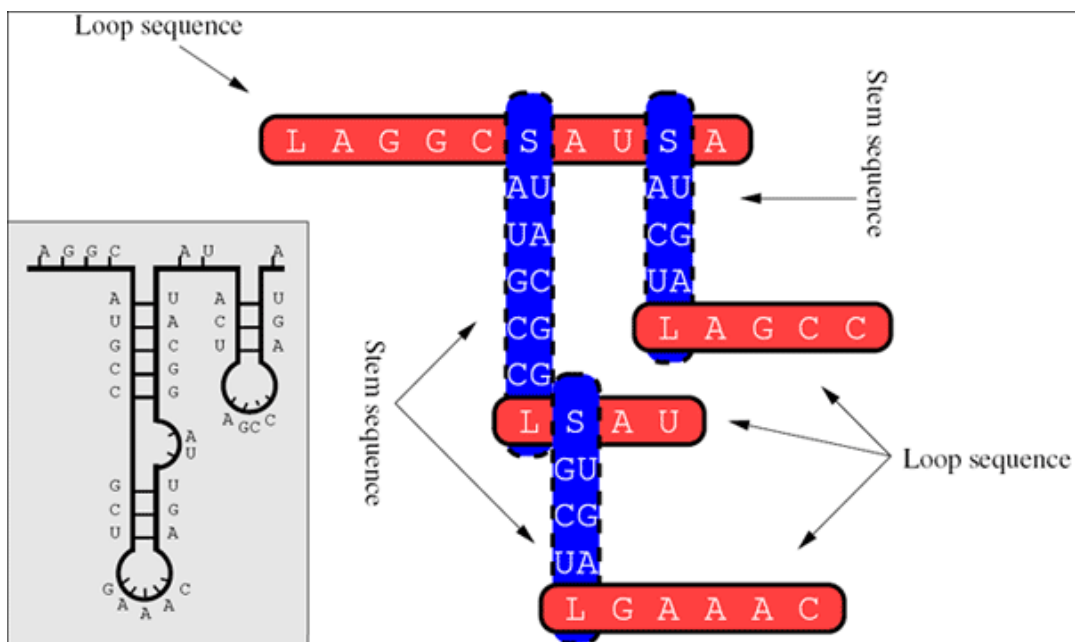
Singlet-Knoten stehen für ungepaarte, unabhängig voneinander evolvierende Nukleotide, die wie im vorher erwähnten TKF91 Modell zu betrachten sind.

Paired-Knoten repräsentieren kovariante Basenpaare, die in den Stem-Bereiche von RNA-Sekundärstrukturen auftreten.

Loop-Knoten (beschriftet mit  $L$ ) leiten eine Loopsequenz ein, die ähnlich zu einer TKF91 Linksequenz ist, zusätzlich aber noch über ein weiteres Symbol verfügt, welches eine Stemsequenz einleitet. Eine Loopsequenz besitzt am linken Ende einen *immortal link* und eine darauf folgende endliche Anzahl von *mortal links*. Diese *mortal links* werden mit Rate  $\lambda_1$  eingefügt und mit Rate  $\mu_1$  gelöscht. Für die Punktsubstitutionen nimmt man eine Markov-Kette mit fünf Zuständen an jeder Position an, die unabhängig von ihren benachbarten Singlet-Knoten-Beschriftungen aus dem Alphabet  $\{A, G, C, U, S\}$  emittiert. Die Substitutionsraten entstammen hierbei einer gegebenen Ratenmatrix  $R_1(i, j)$ . Die

Wahrscheinlichkeit für einen Übergang von  $X \rightarrow S$  oder  $S \rightarrow X$ , wobei  $X \in \{A, G, C, U\}$ , entspricht 0. Aus einer Punktsubstitution eines Nukleotides, kann also kein neuer Stem entstehen und umgekehrt.

Stem-Knoten (beschriftet mit S) leiten eine Stemsequenz ein, die im wesentlichen aus einer Folge von kovarianten Basenpaaren besteht, welche von einer weiteren Loopsequenz terminiert wird. Die Stemsequenz repräsentiert einen unabhängig evolvierenden Stem. Wird der Stem-Knoten gelöscht, führt dies zum Verlust des kompletten Stems. Wird ein Stem-Knoten in die Loopsequenz eingeführt, führt dies zu einem Einbau eines kompletten Subtrees. Innerhalb der Stemsequenz besitzt der Stem-Knoten S die Eigenschaften eines *immortal links*, während die Paired-Knoten die Eigenschaften von *mortal links* besitzen. Er kann also bei Evolutionsereignissen innerhalb des Stems nicht gelöscht werden. Die Löschung einer Sekundärstruktur erfolgt daher nur durch die Deletion des Stem-Knoten S innerhalb der Loopsequenz, innerhalb der er die Eigenschaft eines *mortal links* besitzt. Für die Insertion und Deletion existieren in der Stemsequenz eigene Raten, nämlich  $\lambda_2$  und  $\mu_2$ . Auch hier wird für die Punktsubstitution eine Markov-Kette zugrunde gelegt, deren Zustandswechsel eine Substitution eines kovarianten Basenpaares simuliert. Diese besitzt 16 Zustände, die aus dem Alphabet  $\{AA, AC, AG, AU, CA, CC, CG, CU, GA, GC, GG, GU, UA, UG, UC, UU\}$  emittieren. Die Substitutionsraten entstammen hierbei einer gegebenen Ratenmatrix  $R_2(i, j)$ .



**Abbildung 3**  
Grafische Darstellung eines TKF91 Structure Trees mit der dazugehörigen Sekundärstruktur.

Abbildung 3 zeigt ein Beispiel für einen TKF91 Structure Tree. Die Loopsequenzen werden horizontal geschrieben, die Stemsequenzen vertikal. Links unten in der Abbildung ist die zum Structure Tree zugehörige RNA-Sekundärstruktur zu sehen. Die oben beschriebenen evolutionären Prozesse können den Structure Tree nach den gegebenen Regeln verändern, und ändern dadurch auch die aus diesem resultierende Sekundärstruktur.

Ähnlich wie beim TKF91 Modell, ist es auch hier möglich ein Alignment von zwei Sequenzen zu erstellen. Auch hier dient dabei eine SCFG, deren Produktionen die verschiedenen Evolutionsereignisse beschreiben. Abbildung 4 zeigt die pairSCFG des Structure Trees. Diese ist genauso zu lesen, wie die pairSCFG, die in Abschnitt 2.1. eingeführt wurde. Die verschiedenen Produktionen behandeln die verschiedenen evolutionären Ereignisse, wobei jede Regel eine bestimmte Wahrscheinlichkeit besitzt, die abhängig von den gegebenen Parametern ist. Die Anzahl der Regeln hat sich im Vergleich zu der Grammatik des TKF91 Modells vervierfacht.

Dies ist darauf zurückzuführen, dass bei der Betrachtung der Veränderung von Sekundärstrukturen von einer Ur- in eine Descendantsequenz mehr Fälle auftreten, als beim bloßen Betrachten der Evolution einer Primärsequenz. So finden sich nun in der Grammatik zusätzliche Regeln wieder, die sowohl Veränderungen in den Stems ( $S_1, S_2$ ), als auch die Insertion/Deletion eines ganzen Stems inklusive seiner Substrukturen ( $S_4, L_4 / S_3, L_3$ ) behandeln.

Auch hier unterscheiden sich  $L_1$  und  $L_2$  bzw.  $S_1$  und  $S_2$  dadurch, dass sich die Wahrscheinlichkeit einer Insertion ändert, falls diese Insertion einer Deletion folgt. In diesem Fall trägt der Parameter  $\gamma_n$  statt  $\beta_n$  zur Berechnung der Insertions-Wahrscheinlichkeit bei.

Structure Tree pair rules			$P(a)$	$P(d a)$
Rule	lhs	→ rhs		
1.	$L_1$	→ $X_a Y_d L_1$	$\kappa_1 p_1(X)$	$(1 - \beta_1) \alpha_1 M_1(X, Y)$
2.		$Y_d L_1$	1	$\beta_1 p_1(Y)$
3.		$X_a L_2$	$\kappa_1 p_1(X)$	$(1 - \beta_1)(1 - \alpha_1)$
4.		$S_1 L_1$	$\kappa_1 p_1(S)$	$(1 - \beta_1) \alpha_1$
5.		$S_4 L_1$	1	$\beta_1 p_1(S)$
6.		$S_3 L_2$	$\kappa_1 p_1(S)$	$(1 - \beta_1)(1 - \alpha_1)$
7.		$\epsilon$	$1 - \kappa_1$	$1 - \beta_1$
8.	$S_1$	→ $W_a Y_d S_1 Z_d X_a$	$\kappa_2 p_2(WX)$	$(1 - \beta_2) \alpha_2 M_2(WX, YZ)$
9.		$Y_d S_1 Z_d$	1	$\beta_2 p_2(YZ)$
10.		$W_a S_2 X_a$	$\kappa_2 p_2(WX)$	$(1 - \beta_2)(1 - \alpha_2)$
11.		$L_1$	$1 - \kappa_2$	$1 - \beta_2$
12.	$L_2$	→ $X_a Y_d L_1$	$\kappa_1 p_1(X)$	$(1 - \gamma_1) \alpha_1 M_1(X, Y)$
13.		$Y_d L_1$	1	$\gamma_1 p_1(Y)$
14.		$X_a L_2$	$\kappa_1 p_1(X)$	$(1 - \gamma_1)(1 - \alpha_1)$
15.		$S_1 L_1$	$\kappa_1 p_1(S)$	$(1 - \gamma_1) \alpha_1$
16.		$S_4 L_1$	1	$\gamma_1 p_1(S)$
17.		$S_3 L_2$	$\kappa_1 p_1(S)$	$(1 - \gamma_1)(1 - \alpha_1)$
18.		$\epsilon$	$1 - \kappa_1$	$1 - \gamma_1$
19.	$S_2$	→ $W_a Y_d S_1 Z_d X_a$	$\kappa_2 p_2(WX)$	$(1 - \gamma_2) \alpha_2 M_2(WX, YZ)$
20.		$Y_d S_1 Z_d$	1	$\gamma_2 p_2(YZ)$
21.		$W_a S_2 X_a$	$\kappa_2 p_2(WX)$	$(1 - \gamma_2)(1 - \alpha_2)$
22.		$L_1$	$1 - \kappa_2$	$1 - \gamma_2$
23.	$L_3$	→ $X_a L_3$	$\kappa_1 p_1(X)$	1
24.		$S_3 L_3$	$\kappa_1 p_1(S)$	1
25.		$\epsilon$	$1 - \kappa_1$	1
26.	$S_3$	→ $W_a S_3 X_a$	$\kappa_2 p_2(WX)$	1
27.		$L_3$	$1 - \kappa_2$	1
28.	$L_4$	→ $Y_d L_4$	1	$\kappa_1 p_1(Y)$
29.		$S_4 L_4$	1	$\kappa_1 p_1(S)$
30.		$\epsilon$	1	$1 - \kappa_1$
31.	$S_4$	→ $Y_d S_4 Z_d$	1	$\kappa_2 p_2(YZ)$
32.		$L_4$	1	$1 - \kappa_2$

**Abbildung 4**

pairSCFG, mit der ein Alignment von zwei Sekundärstrukturen möglich ist.  $L_1, L_2, L_3, L_4, S_1, S_2, S_3$  und  $S_4$  sind Nicht-Terminale,  $X, Y, W$  und  $Z$  sind Terminale aus dem Alphabet  $\{A, G, U, C\}$ .

Für die Implementierung eines Grammatik-Parsers, wird die in Abbildung 3 gezeigte Grammatik noch einmal modifiziert. In der gezeigten Grammatik können nämlich unter Umständen ungewünschte Produktionsfolgen entstehen, die zu Strukturen wie Stems der Länge 0 oder sog. *silent bulges*, durch das Auftreten von Null-Zykeln, führen können. Ein Beispiel, woran man sich das Problem deutlich machen kann, wäre die Produktionsfolge  $32 \rightarrow 29 \rightarrow 30$ , die zu einem *silent bulge* führt. Regel 32 terminiert einen Stem durch Aufruf von  $L_4$ , Regel 29 erzeugt nun aber einen neuen Stem, der mit Regel 30 fortgeführt wird. Damit ist ein *silent bulge* gebildet worden, eine Struktur, die per Definition des Structure Trees zwar

einen Bulge besitzt, dieser aber keine Nukleotide enthält. Die erweiterte Grammatik fängt genau diese Fälle ab, enthält aber aufgrund dessen auch noch mal mehr als doppelt so viele Regeln, wie die in Abbildung 4 gezeigte pairSCFG. Dies hat aber im Wesentlichen keinen Einfluss auf die Komplexität der Algorithmen, die die Grammatik verwenden.<sup>2</sup>

Ein dynamischer Algorithmus, für das Alignment von zwei Sequenzen mithilfe des Structure Trees, besitzt die Zeit-Komplexität von  $O(L^3M^3)$  und die Platz-Komplexität von  $O(L^2M^2)$ , wobei  $L$  und  $M$  die Sequenzlängen bezeichnen. Diese Komplexitäten lassen sich auf die Version des CYK-Algorithmus zurückführen, der den wahrscheinlichsten Parse-Baum für beide Sequenzen, gegeben die Grammatik und die Parameter, ausgibt.

Um die Komplexität verstehen zu können, betrachten wir zunächst die Arbeitsweise des Algorithmus. Der CYK-Algorithmus berechnet eine Variable  $\gamma(i, j, v)$ . Diese Variable beschreibt die Wahrscheinlichkeit, wie wahrscheinlich es ist, dass die Subsequenz  $(i \ i+1 \ i+2 \ \dots \ j)$  aus dem Nonterminal  $v$  abgeleitet wurde. Im Falle des CYK-Algorithmus, der ja den optimalen Parsebaum bestimmt, beschreibt die Variable  $\gamma(i, j, v)$  die Subsequenz  $(i \ i+1 \ i+2 \ \dots \ j)$ , die die maximale Wahrscheinlichkeit aller möglichen Subsequenzen besitzt, die vom Nonterminal  $v$  abgeleitet werden können. Für die Berechnung werden die drei Laufvariablen  $i, j$  und  $k$  benötigt, wobei  $i$  den Start,  $j$  das Ende der Subsequenz und  $k$  eine Sequenzposition dazwischen darstellen.<sup>3</sup> Für eine einzelne Sequenz besitzt der CYK-Algorithmus die Komplexität von  $O(L^3NT^3)$ , wobei  $L$  für die Sequenzlänge und  $NT$  für die Anzahl von Nonterminalen steht. Für ein festes  $NT$  beträgt die Laufzeit also  $O(L^3)$ . Bei einem Alignment von zwei Sequenzen über die pairSCFG des Structure Trees beträgt die Laufzeit  $O(L^3M^3)$ , wobei  $L$  und  $M$  für die Länge der jeweiligen Sequenzen, also die Anzahl der Terminale, stehen. Diese Laufzeit kommt dadurch zustande, dass man jede mögliche Kombination von Nukleotiden der einen Sequenz mit denen der anderen betrachten muss.

## 2.4. Analyse des Modells

Um die Fähigkeiten des Modells zu überprüfen, werden einige Tests durchgeführt, die zeigen sollen, in welchem Bereich das Modell seine Stärken und Schwächen hat. Als Testdaten

---

<sup>2</sup> Die genannte Grammatik ist hier nicht als Abbildung aufgeführt. Details zu den erweiterten Regeln sind im Paper von Ian Holmes [5] zu finden.

<sup>3</sup> Genaueres über den CYK-Algorithmus ist in Durbin et al. [3] auf Seite 253 ff. zu finden.

dienen hierbei 8 Sequenzen aus 4 unterschiedlichen RNA-Familien. Die Familien unterscheiden sich stark innerhalb ihrer Variabilität bezüglich ihrer Primärsequenzen und/oder ihrer Sekundärstrukturen. Im Wesentlichen soll nun gezeigt werden, wie gut das Modell in der Lage ist, jeweils zwei Sequenzen einer Familie unter Berücksichtigung ihrer Sekundärstruktur zu alignieren. Um dieses Alignment bewerten zu können, erfolgt gleichzeitig ein Alignment der Sequenzen auf Basis des TKF91-Modells, also ein reines Primärsequenzalignment. Da es mit dem Modell außerdem auch möglich ist, Sekundärstruktur-Bereiche in den Sequenzen hervorsagen zu können, wird auch überprüft, wie zuverlässig diese Vorhersagen sind. Hier wurde zum einen betrachtet, welche Ergebnisse die Vorhersage bei der Betrachtung der einzelnen Sequenz liefert, zum anderen, wie sich die Ergebnisse verhalten, wenn man zusätzlich die Informationen über die zweite Sequenz mit einbezieht.

Um die Ergebnisse interpretieren zu können, sollen hier die Eigenschaften der 4 untersuchten Familien aufgezeigt werden.

Bei der ersten zu untersuchenden Familie handelt es sich um die *Purin Riboswitche*, die ein Bestandteil der post-translationalen Regulation von Purin-Transport und -biosynthese sind. Charakteristisch für die zwei Sequenzen aus dieser Familie ist, dass beide eine identische Sekundärstruktur aufweisen, sich aber in der Primärsequenz voneinander unterscheiden.

Die Vertreter der zweiten zu untersuchenden Familie sind *Nano translational control elements*, bei denen es sich um regulatorische Sequenzen im 3' UTR der *nano*-Gene von *Drosophila* handelt. Betrachtet wurden hier die Regulatorischen Sequenzen zweier *Drosophila* Arten. Diese beiden Sequenzen unterscheiden sich durch eine Deletion des äußeren Stems in einer der beiden Sequenzen. Die Primärsequenz ist ansonsten nahezu identisch.

*U2 splicing factors* ist der Name der dritten zu untersuchenden Familie. Diese markieren die Schnittstelle in der pre-mRNA, an der Exons von Introns beim RNA-Processing getrennt werden. Die beiden Sequenzen besitzen eine sehr ähnliche Primärsequenz, unterscheiden sich jedoch stark in der Sekundärstruktur. Eine der beiden Sequenzen besitzt eine Deletion von drei mittleren Stems im Vergleich zur anderen.

Der letzte Test des Modells wird mit Vertretern der Familie der *RNase P Genes* durchgeführt, bei denen es sich um Endoribonukleasen handelt, die eine katalytische Funktion besitzen und eine wesentliche Rolle bei der Biosynthese von tRNA's spielen. Die Familie der *RNase P Genes* stellt die höchst-variabelste Familie dar, die in der Datenbank *RFAM*, aus der im Übrigen alle getesteten Sequenzen stammen, vorhanden ist. Die beiden verwendeten Sequenzen unterscheiden sich sehr stark in der Sekundärstruktur.

## 2.5. Auswertung

In Tabelle 1 sind die wesentlichen Ergebnisse zusammengefasst, die das Modell bei der Verwendung der oben genannten Vertreter der einzelnen RNA-Familien erzielt. Aus den Ergebnissen kann gefolgert werden, dass das Modell in der Lage ist, aufgrund der Berücksichtigung der Sekundärstruktur, bessere Alignments von Sequenzen zu erstellen, als es das pairHMM des TKF91 Modell in der Lage ist (Spalte 4 und 5). Weiterhin funktioniert die Vorhersage von Sekundärstrukturbereichen in den einzelnen Sequenzen sehr gut, wenn das Modell in der Lage ist, auf die Informationen von einer weiteren Sequenz zugreifen zu können (Spalte 3). Ist dies nicht der Fall (Spalte 2), kann eine Strukturvorhersage nur schlecht erfolgen. Anhand des Tests mit den Vertretern der *RNase P Genes*, sieht man aber auch, dass, ab einem bestimmten Grad der strukturellen Unterschiede zwischen den Sequenzen, das Modell nicht mehr in der Lage ist, vernünftige Ergebnisse zu erzielen.

**Tabelle 1**

In der Tabelle sind die Testergebnisse des Modells aufgelistet.

RNA Sequenzen	Strukturvorhersage StructureTree singlet	Strukturvorhersage StructureTree paired	pairHMM Alignment	pairSCFG Alignment
Purine Riboswitches	schlecht	korrekt	korrekt	Korrekt
Nano translational control element	schlecht	gut	Probleme in den Randbereichen	wesentlich besser
U2 splicing factors	schlecht	korrekt	schlecht	Korrekt
RNase P	schlecht	schlecht	schlecht	Schlecht

Die Autoren machen weiterhin eine Reihe von Vorschlägen, die zu einer Verbesserung des Modells führen könnten. In dem entwickelten Modell sind nur *single-indels*, also die Insertion oder Deletion einer einzelnen Position, möglich. Es sollte auch möglich sein *long-indels* durchzuführen, also Insertionen und Deletionen von mehreren Positionen. Vor allem bei dem Test der *RNase P Genes* fällt auf, dass das Alignment mehrere kleine *gaps* erzeugt, wobei das „wahre“ Alignment dadurch charakterisiert ist, dass es weniger größere *gaps* enthält. Dies ist vor allem auf die lineare *gap penalty* zurückzuführen, die durch die Verwendung von *single-indels* entsteht. Die Einführung von *long-indels* und somit auch von einer affinen *gap penalty*, könnte unter Umständen dazu beiführen, dass das Modell auch in der Lage wäre, die *RNase P Genes* korrekt zu verarbeiten.

Weiterhin schlagen die Autoren vor, eine weitere Möglichkeit zu erschaffen, wie Bulges in Stems eingefügt werden können. Bislang ist dies nur möglich, indem eine Stem-Konstruktion entsteht, bei dem ein Stem durch eine Loopsequenz entsteht und in dieser dann ein neuer Stem gebildet wird.  $S \rightarrow L \rightarrow S$ . Verbessert werden könnte dies, indem man die Möglichkeit schafft, Loop-Knoten in Stem-Sequenzen einzuführen, ohne das dadurch der Stem terminiert wird.

Bestimmte strukturelle Eigenschaften, wie beispielsweise Triloops, Tetraloops, U-Turns und Ähnliches, werden nicht speziell behandelt. Bei solchen Strukturen wurde aber beobachtet, dass diese oftmals evolutionär konserviert sind und scheinbar involviert in intermolekulare Interaktionen sind. Die Einführung der spezifischen Behandlung dieser Strukturen, wäre ohne große Probleme möglich, indem man Spezialklassen von L- oder S-Kanten zulassen würde.

Außerdem sind die Indel-Raten für Stems und Multistems identisch zu den Raten für einen Indel einer ungepaarten Position. Dies widerspricht aber ein wenig der Realität, da der Verlust von ganzen Stems viel langsamer sein sollte, als der Verlust einer einzelnen ungepaarten Position, da dies zu einer Änderung der Struktur führt, die bei dem Verlust einer einzelnen Position nicht beeinträchtigt wird.

Im Bereich der Stems und Multistems gibt es eine weitere Sache, die eventuell verbessert werden sollte. Wenn ein Stem gelöscht wird, werden auch alle Substrukturen innerhalb des Stems mitgelöscht. Empirische Studien von Alignments in der Datenbank *RFAM* haben aber ergeben, dass es oft vorkommt, dass ein äußerer Stem zwar gelöscht worden ist, die inneren

Strukturen aber konserviert sind. Um das Modell zu verbessern, sollte auch diese Möglichkeit in Betracht gezogen werden.

### 3. Vorhersage von RNA-Sekundärstruktur

#### 3.1. Einleitung

Das zweite Paper beschäftigt sich mit der Vorhersage von RNA-Sekundärstruktur. Die Autoren haben einen Algorithmus entwickelt, der es möglich macht, RNA-Sekundärstrukturen mit simplen Pseudoknoten vorherzusagen. Wie schon erwähnt ist die Vorhersage der Sekundärstruktur einer gegebenen RNA-Sequenz wichtig, um etwas über deren Funktion aussagen zu können. Diese Sekundärstrukturen entstehen durch das Ausbilden von Basenpaaren innerhalb der Sequenz. Zunächst sind nur „einfache“ Sekundärstrukturen betrachtet worden, Pseudoknoten sind aufgrund ihrer Komplexität außen vorgelassen worden. Da diese Strukturen aber in der Natur auftreten und auch durchaus wichtige funktionelle Eigenschaften besitzen, ist es nötig diese bei der Vorhersage auch in Betracht zu ziehen. Allerdings stellt die Pseudoknoten-Vorhersage kein triviales Problem dar. Zum einen ist es mit den bekannten Methoden nur möglich simple Pseudoknoten, die später noch genau definiert werden, effizient vorherzusagen. Zum anderen haben auch die Algorithmen, die sich auf die simplen Pseudoknoten beschränken eine hohe Laufzeit. Zum Vergleich beziehen sich die Autoren immer wieder auf den Algorithmus von Rivas und Eddy, der simple Pseudoknoten in  $O(n^6)$  Zeit und mit  $O(n^4)$  Speicherplatz vorhersagt. Aufgrund der hohen Laufzeit ist es bei diesem Algorithmus nur möglich, Sequenzen der Länge  $<100$  zu betrachten. Die Motivation zur Entwicklung eines neuen Algorithmus liegt nun also darin begründet, einen Algorithmus zu finden, der RNA-Sekundärstrukturen mit simplen Pseudoknoten in geringerer Zeit findet. Der in dem Paper präsentierte Algorithmus von Deogun und Komina kommt mit einer Laufzeit von  $O(n^4)$  aus und ist aufgrund dessen in der Lage, wesentlich längere RNA-Sequenzen betrachten zu können. Zur Bewertung der vorhergesagten Strukturen werden die *Nearest-Neighbour Thermodynamic Rules* von Zuker und Turner verwendet.

Der neue Algorithmus zur Vorhersage der Pseudoknoten leitet sich von einem früheren Algorithmus ab, dem Akutsu-Algorithmus. Dieser findet RNA-Sekundärstrukturen mit simplen

Pseudoknoten unter der Verwendung von Maximierung von Basenpaaren. Auch hier kam der Gedanke zu tragen, auf dem auch die *Nearest-Neighbour Thermodynamic Rules* beruhen. Die Ausbildung von Basenpaaren führt zur Minimierung von freien Energien im Molekül. Folgt man nun der Annahme, dass die wirklich vorkommende Struktur die Struktur mit der minimal freien Energie ist, dann besitzt diese Struktur eine maximale Anzahl von Basenpaaren.<sup>4</sup>

Die Erweiterung des Akutsu-Algorithmus, durch Hinzunahme von Thermodynamik-Parametern, zusammen mit dem *mfold*-Algorithmus, der in der Lage ist, RNA-Sekundärstrukturen ohne Pseudoknoten mit minimal freier Energie zu berechnen, berechnet so die optimale RNA-Sekundärstruktur inklusive Pseudoknoten, die eine minimale freie Energie besitzt.

Im Folgenden soll die RNA-Sekundärstruktur, sowie eine Struktur mit und ohne Pseudoknoten noch einmal formal definiert werden.

#### Definition 1:

Eine Sekundärstruktur  $S$  einer RNA-Sequenz  $A = a_1a_2\dots a_n$  ist formal eine Menge von Basenpaaren. Ein Basenpaar zwischen  $a_i$  und  $a_j$  ( $i < j$ ) wird notiert als  $(i - j)$ . Für die Menge von Basenpaaren gilt:

$$M = \{ (i - j) \mid 1 \leq i < j \leq n, (a_i - a_j) \text{ ist Basenpaar und jedes } i \text{ und } j \text{ taucht maximal 1-mal auf} \}$$

#### Definition 2:

Eine Menge von Basenpaaren wird RNA-Sekundärstruktur ohne Pseudoknoten genannt, wenn folgende Bedingung erfüllt ist:

Es existieren keine Basenpaare  $(a_i - a_j), (a_h - a_k) \in M$ , die  $i \leq h \leq j \leq k$  erfüllen. (siehe Abbildung 5)

---

<sup>4</sup> Eine Annahme, die durchaus kontrovers diskutiert wird. Die funktionelle Struktur muss nicht zwangsläufig die Struktur mit der minimal freien Energie sein. Studien belegen, dass auch suboptimale Faltungen in der Natur auftreten. Im Folgenden wird von der „wahren“ Struktur, als die Struktur mit der *minimalen* freien Energie gesprochen.

### Definition 3:

Eine Menge von Basenpaaren wird RNA-Sekundärstruktur mit Pseudoknoten genannt, wenn folgende Bedingung erfüllt ist:

$$M_{i,K} = \{ (i - j) \mid 1 \leq i < j \leq K, (a_i - a_j) \text{ ist Basenpaar und jedes } i \text{ und } j \text{ taucht maximal 1-mal auf} \}$$

Es existieren Positionen  $j'$  und  $j''$  für  $1 < j' < j'' < K$ , so dass für jedes Paar  $(i - j) \in M_{i,K}$  gilt:

$$1 \leq i < j' < j < j'' \leq K \text{ oder } j' < i < j'' \leq j \leq K. \text{ (siehe Abbildung 6)}$$

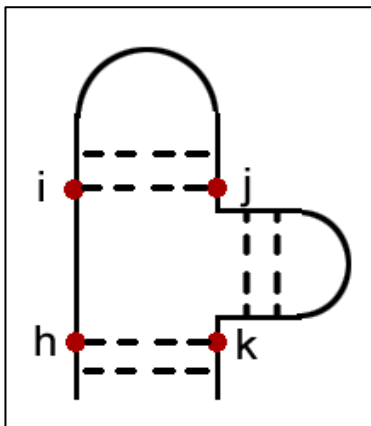


Abbildung 5  
RNA-Sekundärstruktur ohne Pseudoknoten

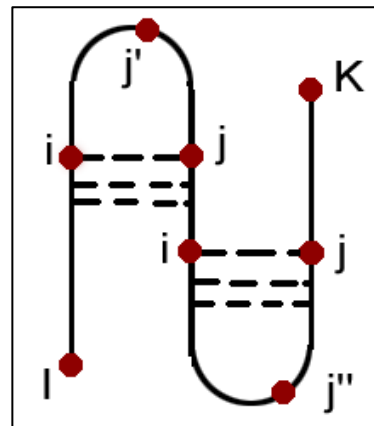


Abbildung 6  
RNA-Sekundärstruktur mit Pseudoknoten

### 3.2. Der Akutsu-Algorithmus

Der Algorithmus von Tatsuya Akutsu ist ein auf dynamischen Programmieren beruhender Algorithmus, der in der Lage ist, RNA-Sekundärstrukturen, unter Berücksichtigung von Pseudoknoten, vorherzusagen. Für die Vorhersage der Sekundärstruktur, wird vorausgesetzt, dass die korrekte Struktur die minimale freie Energie besitzt. Die Bewertung, die der Algorithmus für die verschiedenen möglichen Sekundärstrukturen innerhalb der Sequenz vornimmt, basiert auf deren Anzahl von bildbaren Basenpaaren. Die Struktur mit der höchsten Anzahl wird als optimale Struktur betrachtet und dementsprechend vorhergesagt. Wie bereits in der Einleitung erwähnt, kommt dies dadurch zustande, dass man die Annahme zugrunde legt, dass eine hohe Anzahl von Basenpaaren zu einer erhöhten Ordnung im Molekül führt und dadurch die freie Energie erniedrigt. Die Annahme

ist, dass solche Strukturen in der Natur bevorzugt auftreten, weswegen sie vom Algorithmus besser bewertet werden.

Die Berechnung der optimalen Struktur erfolgt mithilfe einer zwei-dimensionalen Matrix  $S$ . Diese wird wie folgt berechnet:

$$S(i, j) = \max \begin{cases} PS(i, j) \\ \mu(i, j) + S(i+1, j-1) \\ \max_{i \leq m < j} \{ S(i, m) + S(m+1, j) \} \end{cases}$$

$S(i, j)$  beschreibt die Sub-Sekundärstruktur von Position  $i$  bis Position  $j$ , die eine maximale Anzahl von Basenpaaren besitzt. In Position  $S(1, n)$  befindet sich so die maximale Anzahl von Basenpaaren. Durch ein Traceback durch die Matrix, kann berechnet werden, welche Sekundärstruktur zu der Anzahl der Basenpaare in jedem Schritt beigetragen hat. Daraus wird dann die Struktur der RNA-Sequenz errechnet.

Wie man sieht, bezieht  $S(i, j)$  seinen Wert aus dem Maximum aus drei unterschiedlichen Fällen. In Fall 1 wird die Möglichkeit betrachtet, dass die maximale Anzahl von Basenpaaren aus einem Pseudoknoten hervorgeht. Fall 2 enthält die Möglichkeit, dass die Struktur des letzten Schritts zuzüglich eines eventuellen Scores für das Basenpaar  $(i - j)$  die maximale Anzahl von Basenpaaren besitzt.  $\mu(i, j) = 1$ , falls  $i$  mit  $j$  paart, 0 sonst. Und der dritte Fall letztendlich, betrachtet die Möglichkeit, dass die maximale Anzahl von Basenpaaren in schon vorher berechneten Sub-Strukturen enthalten ist.

Die Berechnung von  $PS(i, j)$ , der Score eines Pseudoknoten von Position  $i$  bis Position  $j$  ( $i$  und  $j$  stellen hierbei die Endpunkte  $I$  und  $K$  dar), erfolgt mithilfe von 3  $N \times N \times N$  Matrizen:

1.  $SL(i, j, k)$  enthält den Score des besten Foldings zwischen  $I$  und  $i$ , und  $j$  und  $k$ , vorausgesetzt das  $i$  mit  $j$  paart.
2.  $SR(i, j, k)$  enthält den Score des besten Foldings zwischen  $I$  und  $i$ , und  $j$  und  $k$ , vorausgesetzt das  $j$  mit  $k$  paart.

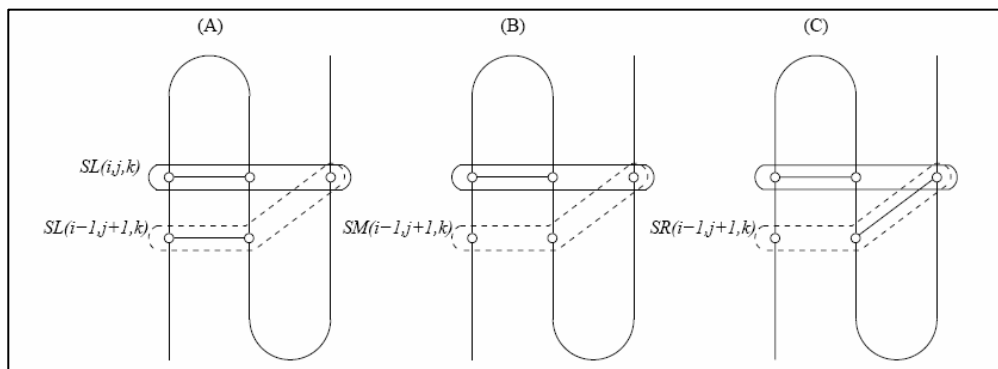
3.  $SM(i, j, k)$  enthält den Score des besten Foldings zwischen  $l$  und  $i$ , und  $j$  und  $k$ , vorausgesetzt dass weder  $i$  mit  $j$ , noch  $j$  mit  $k$  paart.

Für die Berechnung des Pseudoknoten mit Endpunkten  $l$  und  $K$  betrachtet der Algorithmus drei Arten von Triplets  $SL(i, j, k)$ ,  $SR(i, j, k)$  und  $SM(i, j, k)$  für alle  $i, j, k$ , so dass gilt: ( $l \leq i < j < k \leq K$ ).

Die  $SL$ -Matrix wird über folgende Rekursion berechnet. Der Wert in  $SL(i, j, k)$  setzt sich zusammen, aus dem Score den das Paar  $(i - j)$  erzielt plus dem Maximum aus den drei Matrizen, die im Schritt zuvor berechnet worden sind. Grafisch ist dies in Abbildung 7 dargestellt.

$$SL(i, j, k) = \mu(i, j) + \max \begin{cases} SL(i-1, j+1, k) \\ SM(i-1, j+1, k) \\ SR(i-1, j+1, k) \end{cases}$$

$$\mu(i, j) = 1 \text{ falls } i \text{ und } j \text{ paaren, } 0 \text{ sonst}$$

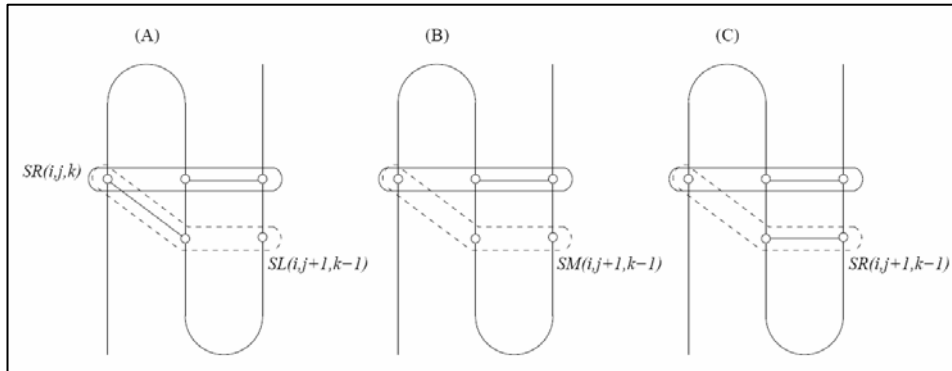


**Abbildung 7**  
Berechnung der  $SL$ -Matrix. (A), (B) und (C) zeigen die drei Möglichkeiten, woraus  $SL(i, j, k)$  berechnet sein könnte.

Die  $SR$ -Matrix wird fast identisch berechnet. Der Wert in  $SR(i, j, k)$  setzt sich zusammen, aus dem Score den das Paar  $(j - k)$  erzielt plus dem Maximum aus den drei Matrizen, die im Schritt zuvor berechnet worden sind. Grafisch ist dies in Abbildung 8 dargestellt.

$$SR(i, j, k) = \mu(j, k) + \max \begin{cases} SL(i, j+1, k-1) \\ SM(i, j+1, k-1) \\ SR(i, j+1, k-1) \end{cases}$$

$$\mu(j, k) = 1 \text{ falls } j \text{ und } k \text{ paaren, } 0 \text{ sonst}$$



**Abbildung 8**

Berechnung der SR-Matrix. (A), (B) und (C) zeigen die drei Möglichkeiten, woraus  $SR(i, j, k)$  berechnet sein könnte.

Die SM-Matrix wird wie folgt berechnet. Der Wert in  $SM(i, j, k)$  kann auf drei Arten zustande kommen. Da weder  $i$  mit  $j$  noch  $j$  mit  $k$  paaren, ist der Wert, den die Matrix SM an der Position  $(i, j, k)$  annehmen kann, in einem der Schritte vorher berechnet worden. Folglich werden alle Möglichkeiten in Betracht gezogen und aus diesen das Maximum gewählt. Ausgedrückt ist dies durch folgende Rekursionsgleichung:

$$SM(i, j, k) = \max \begin{cases} SL(i-1, j, k), SM(i-1, j, k) \\ SL(i, j+1, k-1), SR(i, j+1, k-1), SM(i, j+1, k-1) \\ SR(i, j, k-1), SM(i, j, k-1) \end{cases}$$

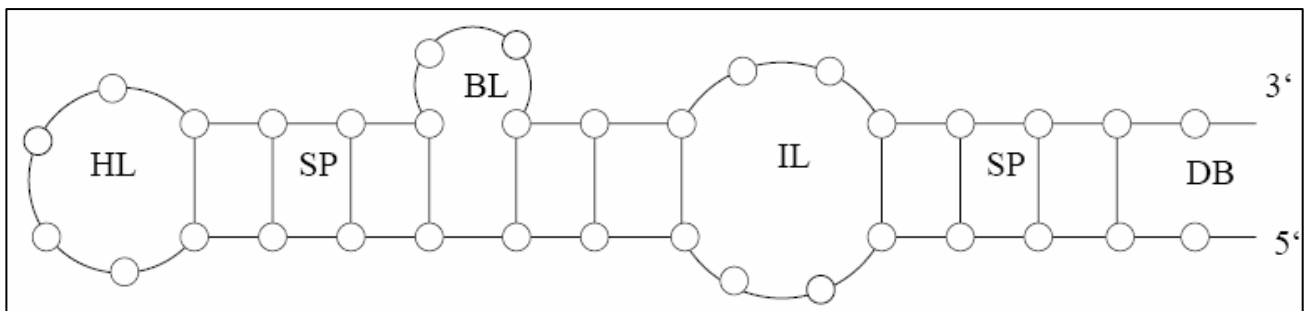
Für jedes Paar  $(I, K)$ , wobei  $I + 6 < K$ , werden die SL, SM und SR Matrizen berechnet. Der Score eines Pseudoknoten mit Endpunkten  $I$  und  $K$  errechnet sich durch:

$$PS(I, K) = \max_{I < i < K} \begin{cases} SL(i, i+1, K) \\ SM(i, i+1, K) \\ SR(i, i+1, K) \end{cases}$$

### 3.3. Nearest-Neighbour Thermodynamik Regeln

In der RNA-Strukturvorhersage sind die Nearest-Neighbour Thermodynamik Regeln weit gebräuchlich. Diese Regeln weisen verschiedenen Typen von RNA-Sekundärstrukturen unterschiedliche Energie-Werte zu. Das Ziel ist, wie schon erwähnt, ein Molekül zu berechnen, das eine minimale freie Energie besitzt.

Im Rahmen der Nearest-Neighbour Thermodynamik Regeln spricht man auch von *loop dependent energy rules*. Der Grund dafür ist, dass die freie Energie den Loops des RNA-Moleküls zugewiesen wird. Die totale freie Energie ist die Summe der freien Energie aller Loops, die in dem Molekül vertreten sind. Loops können null oder mehr Basenpaare und null, eine oder mehrere einzelsträngige Nukleotide besitzen. Abbildung 9 zeigt die wichtigsten RNA-Sekundärstrukturen. Dies sind zum einen der *Hairpin Loop*, ein Folge von ungepaarten Basen, die durch ein Basepaar geschlossen wird, weiterhin das *Stacking Pair*, das zwei direkt aufeinander folgende Basenpaare darstellt, der *Bulge Loop*, eine Art Ausbuchtung in einem Stem-Bereich, der *Internal Loop*, eine von zwei Basenpaaren eingeschlossene Folge von ungepaarten Nukleotiden und zum anderen die *Dangling Bases*, die ungepaarte Nukleotide am 3' oder 5' Ende eines Basenpaares darstellen.



**Abbildung 9**

Zu sehen sind die wichtigsten simplen RNA-Sekundärstrukturen. Nukleotide sind durch Kreise dargestellt. Von links nach rechts: HL zeigt einen Hairpin Loop, SP steht für ein Stacking Pair, BL stellt einen Bulge Loop dar, IL ist ein Internal Loop und DB steht für eine oder mehrere Dangling Bases.

Details zu den einzelnen Thermodynamik-Werten für die verschiedenen Strukturen sind in [8] zu finden.

Das Problem, dass man bei der Berechnung von Pseudoknoten hinsichtlich der Thermodynamik Regeln hat, ist, dass es keine systematische Studie über deren Thermodynamik gibt. Die von Zuker, Mathews und Turner veröffentlichten *Nearest-*

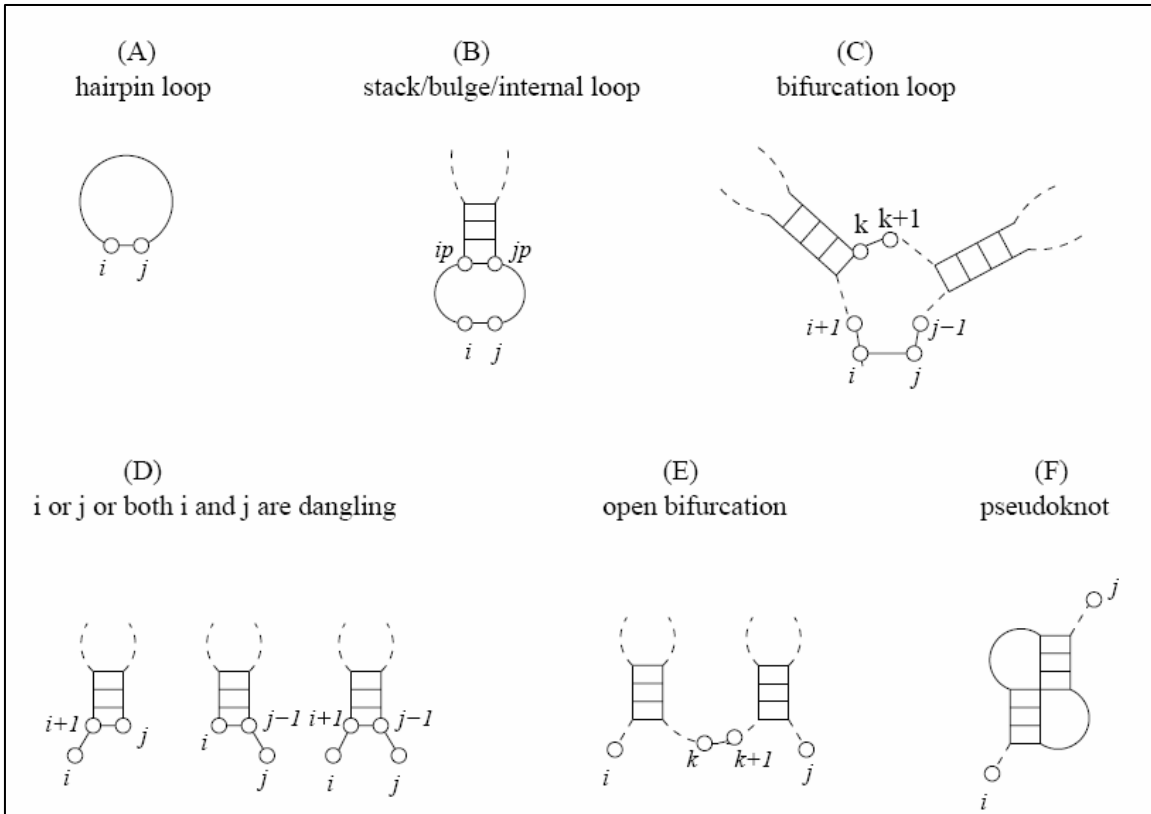
*Neighbour Thermodynamic Rules* beinhalten aufgrund dessen keine Werte für Pseudoknoten. Für die Konstruktion des Algorithmus, der eine Vorhersage von Pseudoknoten ermöglicht, geht man deshalb davon aus, dass sich die freie Energie eines Pseudoknoten aus der Summe der destabilisierenden Energie der Loops und der der stabilisierenden Energie beider Stems zusammensetzt.

### 3.4. Berechnung minimaler Energien von RNA-Substrukturen

Für die Berechnung der minimalen Energie von Substrukturen wird der *mfold*-Algorithmus erweitert. Dieser Algorithmus ist in der Lage, die Substrukturen innerhalb einer RNA-Sequenz vorherzusagen, die einen Beitrag zur minimalen freien Energie der optimalen Gesamtstruktur liefern. Durch die hinzugefügte Erweiterung, ist der Algorithmus nun auch in der Lage als Substruktur simple Pseudoknoten zu betrachten. Zur Berechnung werden 3 NxN Matrizen benötigt.

1.  $V(i, j)$  enthält den Score des besten Foldings zwischen  $i$  und  $j$ , unter der Bedingung das  $i$  und  $j$  paaren.
2.  $W(i, j)$  enthält den Score des besten Foldings zwischen  $i$  und  $j$ , egal ob  $i$  und  $j$  paaren oder nicht.
3.  $PS(i, j)$  enthält den Score der besten Pseudoknoten-Konfiguration zwischen den Positionen  $i$  und  $j$ .

Abbildung 10 zeigt die daraus resultierenden Strukturen, die durch die drei Matrizen abgedeckt werden.



**Abbildung 10**

(A), (B) und (C) werden bei der Berechnung von  $V(i, j)$  betrachtet. (D), (E), (F) und  $V(i, j)$  werden bei der Berechnung von  $W(i, j)$  in Betracht gezogen.

Für die Berechnung von  $V(i, j)$  werden die drei möglichen Fälle betrachtet, bei denen  $i$  mit  $j$  paaren kann. Diese sind in der folgenden Rekursionsgleichung ausgedrückt:

$$V(i, j) = \min \begin{cases} \text{hairpin}(i, j) \\ \min_{i < ip < jp < j} \{V(ip, jp) + \text{loop}(i, j, ip, jp)\} \\ \min_{i+1 < k < j-2} \{W(i+1, k) + W(m+1, j-1)\} \end{cases}$$

$$V(i, j) = \infty, \text{ wenn } i \text{ und } j \text{ nicht paaren können}$$

Da nach der minimalen freien Energie gesucht wird, wird für  $V(i, j)$  das Minimum der drei gegebenen Fälle gewählt. Fall 1 beschreibt den Hairpin Loop, der vom Basenpaar ( $i - j$ ) aufgespannt wird. Die Funktion  $\text{hairpin}()$  liefert hierbei die freie Energie des *Hairpins*. Im zweiten Fall wird die Möglichkeit abgedeckt, dass die minimale freie Energie für  $V(i, j)$  aus

einem *Stacking Pair*, einem *Bulge* oder einem *Internal Loop* stammt.  $V(i, j)$  beschreibt den Wert den eine frühere Struktur erzielt hat.  $(i - j)$  schließt nun mit  $(i - j)$  einen Loop, in dem null oder mehrere ungepaarte Nukleotide enthalten sein können. Die daraus resultierende destabilisierende freie Energie wird durch die Funktion  $loop()$  bestimmt. Der letzte Fall stellt die Möglichkeit eines *Bifurcation Loops* dar. In diesem Fall wird die Möglichkeit abgedeckt, dass die minimale freie Energie für Position  $(i, j)$  in Matrix  $V$  aus verschiedenen Substrukturen zusammengesetzt ist, deren Scores in der Matrix  $W$  gespeichert sind.

Für die Berechnung von  $W(i, j)$  werden alle möglichen Fälle betrachtet, in denen  $(i - j)$  miteinander paaren oder nicht paaren. Das heißt, dass dort auch alle Berechnungen von  $V(i, j)$  mit abgedeckt werden. Folgende Rekursionsgleichung dient zur Berechnung von  $W(i, j)$ :

$$W(i, j) = \min \left\{ \begin{array}{l} V(i, j) \\ W(i + 1, j) + dangle(i + 1, j, i, 2) \\ W(i, j - 1) + dangle(i, j - 1, j, 1) \\ W(i + 1, j + 1) + dangle(i + 1, j - 1, i, 2) \\ \quad + dangle(i + 1, j - 1, j, 1) \\ \min_{i < k < j} \{ W(i, k - 1) + W(k, j) \} \\ PS(i, j) \end{array} \right.$$

Auch hier wird, aufgrund der Minimierung der freien Energie der Gesamtstruktur, immer die Struktur ausgewählt, die eine minimale freie Energie besitzt.  $W(i, j)$  kann entweder die Werte von  $V(i, j)$  enthalten, oder die einer Struktur, bei der  $(i - j)$  nicht Element der Menge von Basenpaaren ist. Dabei werden drei weitere Fälle betrachtet. Zum einen sind dies die *Dangling Bases*, ungepaarten Nukleotiden die adjazent zu einem Basenpaar sind. Hier werden wiederum drei Fälle unterschieden, nämlich, ob sich eine *Dangling Base* am 5'-Ende des Basenpaars befindet, ob sie sich am 3'-Ende des Basenpaars befindet oder ob jeweils am 5'- und am 3'-Ende eine *Dangling Base* existiert. Der Wert, den eine *Dangling Base* zu der totalen freien Energie beiträgt, wird durch die Funktion  $dangle()$  ausgegeben. Ein weiterer Fall ist, dass die minimale Energie aus einer sog. *Open Bifurcation* stammt. In diesem Fall wird die Möglichkeit abgedeckt, dass die minimale freie Energie für Position  $(i, j)$  in Matrix  $W$  aus verschiedenen Substrukturen zusammengesetzt ist, deren Scores in der

Matrix  $W$  bereits gespeichert sind. Der letzte mögliche Fall ist der, dass ein Pseudoknoten mit Endpunkten  $i$  und  $j$  die minimale Energie liefert. Die Funktion  $PS(i, j)$ , die den Score des Pseudoknoten ausgibt wird im nächsten Abschnitt besprochen.

Wie man erkennen kann, nimmt der Algorithmus zur Berechnung der minimalen Energien von RNA-Substrukturen in jedem Schritt ein weiteres Nukleotid hinzu und beobachtet, welche Struktur die beste in jedem Schritt ist. Im letzten Schritt wird  $W(1, n)$  berechnet. Dort findet sich die minimale Energie der gesamten Sequenz. Über ein Traceback durch die Matrizen  $V$  und  $W$  und den Matrizen, die für die Pseudoknoten-Berechnung benötigt werden, können die in der Sequenz enthaltenen Strukturen bestimmt werden.

### 3.5. Optimale Energie eines Pseudoknoten

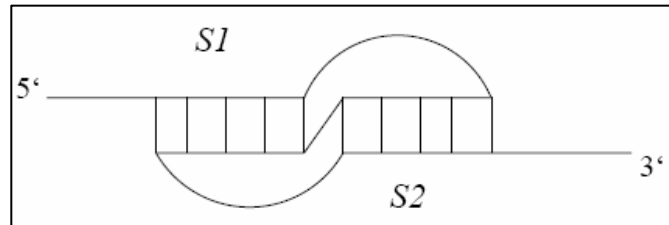
Die Berechnung der optimalen Energie eines Pseudoknoten erfolgt ähnlich wie in dem beschriebenen Algorithmus von Akutsu. Der Unterschied besteht darin, dass auch hier thermodynamische Werte für die Auswertung der Pseudoknoten-Strukturen verwendet werden. Wie bereits in dem Abschnitt über die *Nearest-Neighbour Thermodynamic Rules* erwähnt, geht man davon aus, dass sich die freie Energie eines Pseudoknoten aus der Summe der destabilisierenden Energie der Loops und der der stabilisierenden Energie beider Stems zusammensetzt.

Zur Berechnung werden 3  $N \times N \times N$  Matrizen und 2  $N \times N$  Matrizen benötigt. Dies sind:

1.  $SL(i, j, k)$  Enthält den Score des besten Folding zwischen Positionen  $i$  und  $j$ , und  $j$  und  $k$ . Enthält Energie des Loops, der von  $i$  und  $j$  geschlossen wird. Setzt Paarung von  $i$  und  $j$  voraus.
2.  $SR(i, j, k)$  Enthält den Score des besten Folding zwischen Positionen  $i$  und  $j$ , und  $j$  und  $k$ . Enthält Energie des Loops, der von  $i$  und  $j+1$  geschlossen wird. Setzt Paarung von  $j$  und  $k$  voraus.

3.  $SM(i, j, k)$  Enthält den Score des besten Folding zwischen Positionen  $i$  und  $j$ , und  $j$  und  $k$ . Enthält Energie des Loops, der von  $i$  und  $j+1$  geschlossen wird. Setzt voraus, dass weder  $i$  mit  $j$  paart, noch  $j$  mit  $k$ .
4.  $stem1(i, j)$  Enthält Energie von  $S1$ , die in  $SL(i, j, k)$  gespeichert ist, falls  $i$  mit  $j$  paart und in  $SM(i, j, k)$  falls  $i$  nicht mit  $j$  paart.
5.  $stem2(j, k)$  Enthält Energie von  $S2$ , die in  $SR(i, j, k)$  gespeichert ist, falls  $j$  mit  $k$  paart und in  $SM(i, j, k)$  falls  $j$  nicht mit  $k$  paart.

Die 3  $N \times N \times N$  Matrizen sind direkt aus dem Akutsu-Algorithmus entnommen, mit dem Unterschied, dass diese nun Thermodynamik-Werte speichern. Neu hinzugekommen sind die beiden  $N \times N$  Matrizen  $stem1$  und  $stem2$ . Diese beiden werden zum Berechnen der  $SL$ -,  $SR$ - und  $SM$ -Matrix benötigt. Sie erhalten die Werte, die als minimale Energien für  $SL$ ,  $SR$  oder  $SM$  gewählt werden und speichern daher die minimale Energie von  $S1$  und  $S2$  (siehe Abbildung 11).

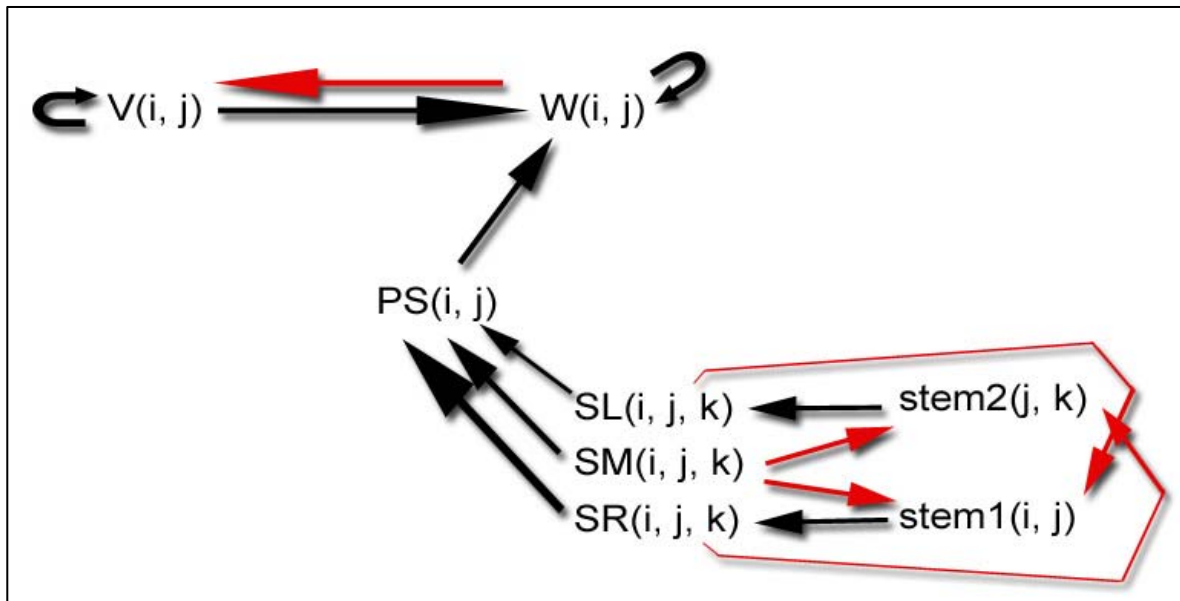


**Abbildung 11**  
Zu sehen ist ein simpler Pseudoknoten.  $S1$  und  $S2$  beschreiben die 2 Stembereiche des Pseudoknoten

$stem1(i, j)$  und  $stem2(j, k)$  enthalten zusammen die Energie einer Struktur  $(i, j, k)$ , so dass gilt:

$$stem1(i, j) + stem2(j, k) = \min \begin{cases} SL(i, j, k) \\ SM(i, j, k) \\ SR(i, j, k) \end{cases}$$

In Abbildung 12 ist zu sehen, wie die Berechnungen der einzelnen Matrizen ineinander übergreifen. Dort ist auch noch mal dargestellt, wie die Berechnung der Strukturen in der Gesamtsequenz erfolgt und welche Teile zur Berechnung der Matrizen benötigt werden.



**Abbildung 12**

Zu sehen ist die Berechnung der Matrizen V, W, PS, SL, SM, SR, stem1 und stem2. Die Pfeile zeigen an welche Matrix zur Berechnung einer anderen benötigt wird.

Wir betrachten nun die einzelnen Schritte in der Berechnung für PS(i, j).

### Initialisierung:

In der Initialisierung werden alle Werte in SL(i, j, k), SM(i, j, k), SR(i, j, k) auf  $\infty$  gesetzt. Eine Ausnahme stellt die Position (i, k-1, k) in SL dar. Wenn i und k-1 paaren können, erhält SL(i, k-1, k) den Wert  $hairpin(i, k-1) + penalty$ . Diese Ausnahme stellt den Fall dar, dass in dem Pseudoknoten nur ein *Hairpin Loop* existiert, was aber der Definition eines Pseudoknoten widerspricht. Da dieser Fall aber aufgrund der Art der Berechnung der Werte auftritt, wird der sog. *penalty*-Wert eingeführt, der verhindern soll, dass der Algorithmus solche, gegen die Pseudoknoten-Definition verstoßende, Strukturen bevorzugt.

Stem1(i, j) erhält den Wert  $hairpin(i, j)$ , falls i und j paaren können und ansonsten den Wert  $\infty$ . Stem2(j, k) erhält auf allen Positionen den Wert  $\infty$ .

### Berechnung der SL-Matrix:

Sind i und j in der Lage ein Basenpaar zu bilden, kann der Wert in SL(i, j, k) auf 3 Arten zustande kommen:

1. Das Paar ( i – j ) schließt einen Hairpin Loop.
2. Das Paar ( i – j ) stackt auf einem Paar ( i-1 – j+1 ).
3. Das Paar ( i – j ) schließt zusammen mit einem Paar ( ip – jp ) einen Bulge oder einen Internal Loop.

SL(i, j, k) enthält also einen der drei möglichen Fälle. Ausgedrückt wird dies durch:

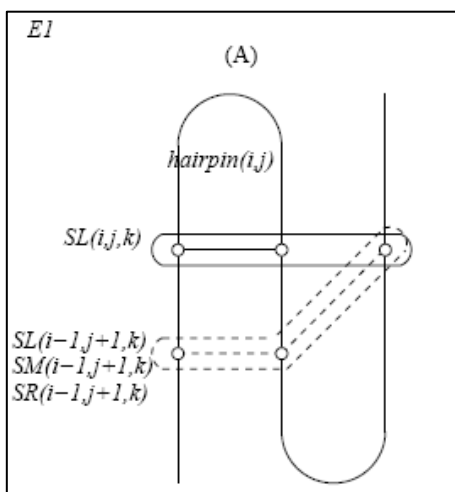
$$SL(i, j, k) = \min \{E_1, E_2\}.$$

$E_1$  steht hierbei für die freie Energie, die der Hairpin Fall liefert. Die Berechnung für  $E_1$  erfolgt durch:

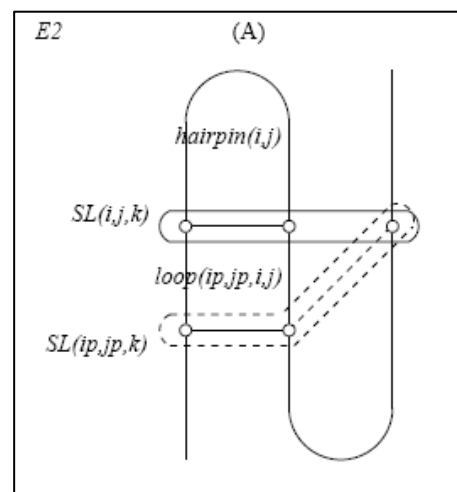
$$E_1 = \text{hairpin}(i, j) + \text{stem2}(j+1, k) \quad (\text{siehe Abbildung 13})$$

$E_2$  steht für die freie Energie, die aus dem Stacking Pair, einem Bulge Loop oder einem Internal Loop zusammen mit dem Basenpaar ( ip – jp ) resultiert. Die Berechnung von  $E_2$  erfolgt durch:

$$E_2 = \min_{|s|, i+4 \leq j < p < k} \{ \text{hairpin}(i, j) - \text{hairpin}(ip, jp) + \text{loop}(ip, jp, i, j) + SL(ip, jp, k) \} \quad (\text{siehe Abbildung 14})$$



**Abbildung 13**  
Berechnung der SL-Matrix für  $E_1$



**Abbildung 14**  
Berechnung der SL-Matrix für  $E_2$

Die Matrix  $stem1(i, j)$  erhält nun den Wert der minimalen Energie, der für  $SL(i, j, k)$  ausgewählt wurde. Dies ist durch folgende Formel charakterisiert:

$$stem1(i, j) = \begin{cases} hairpin(i, j) & , E_1 < E_2 \\ hairpin(i, j) - hairpin(ip, jp) + loop(ip, jp, i, j) + stem1(ip, jp) & , sonst \end{cases}$$

Weiterhin ist es wichtig einen Spezialfall abzudecken. Ist der Wert in  $stem2(j, k)$ , der ja zur Berechnung des Hairpin Falls benötigt worden ist  $= \infty$ , so berechnet sich  $E_1$  durch  $hairpin(i, j) + penalty$ . Dies stellt nämlich wieder den Fall dar, dass nur ein Hairpin-Loop im Pseudoknoten auftritt.

Wenn  $i$  und  $j$  nicht paaren, werden  $SL(i, j, k)$  und  $stem1(i, j)$  wie folgt berechnet:

$$SL(i, j, k) = \min \begin{cases} SL(i-1, j+1, k) \\ SM(i-1, j+1, k) \\ SR(i-1, j+1, k) \end{cases}$$

$$stem1(i, j) = \min \begin{cases} stem1(i-1, j) \\ stem1(i, j+1) \end{cases}$$

### Berechnung der SR-Matrix:

Die Berechnung von  $SR(i, j, k)$  ist sehr ähnlich zu der Berechnung von  $SL(i, j, k)$ . Sind  $j$  und  $k$  in der Lage ein Basenpaar zu bilden, kann der Wert in  $SR(i, j, k)$  auch auf 3 Arten zustande kommen:

1. Das Paar  $(j - k)$  schließt einen Hairpin Loop.
2. Das Paar  $(j - k)$  stackt auf einem Paar  $(j+1 - k-1)$ .
3. Das Paar  $(j - k)$  schließt zusammen mit einem Paar  $(jp - kp)$  einen Bulge oder einen Internal Loop.

SR(i, j, k) enthält also einen der drei möglichen Fälle. Ausgedrückt wird dies durch:

$$SR(i, j, k) = \min \{E_3, E_4\}.$$

$E_3$  steht hierbei für die freie Energie, die der Hairpin Fall liefert. Die Berechnung für  $E_3$  erfolgt durch:

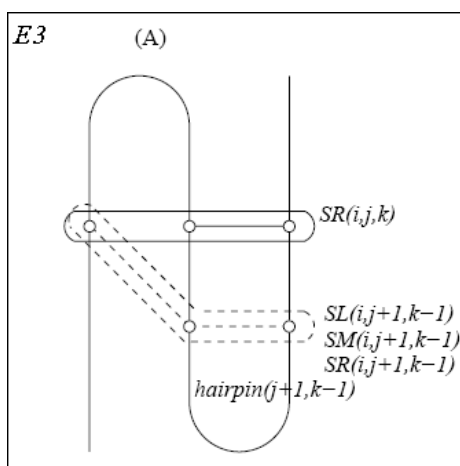
$$E_3 = \text{hairpin}(j, k) + \text{stem1}(i, j+1) \quad (\text{siehe Abbildung 15})$$

$E_4$  steht für die freie Energie, die aus dem Stacking Pair, einem Bulge Loop oder einem Internal Loop zusammen mit dem Basenpaar (  $jp - kp$  ) resultiert. Die Berechnung von  $E_4$  erfolgt durch:

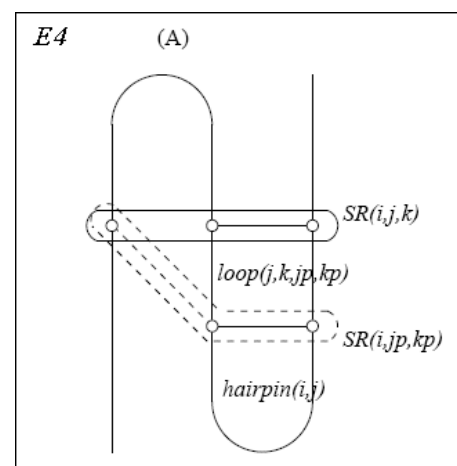
$$E_4 = \min_{j < jp, jp+4 \leq kp < k} \{ \text{loop}(j, k, jp, kp) + SL(i, jp, kp) \} \quad (\text{siehe Abbildung 16})$$

Die Matrix stem2(j, k) erhält nun den Wert der minimalen Energie, der für SR(i, j, k) ausgewählt wurde. Dies ist durch folgende Formel charakterisiert:

$$\text{stem2}(j, k) = \begin{cases} \text{hairpin}(j, k) & , E_3 < E_4 \\ \text{stem2}(jp, kp) + \text{loop}(j, k, jp, kp) & , \text{sonst} \end{cases}$$



**Abbildung 15**  
Berechnung der SR-Matrix für  $E_3$



**Abbildung 16**  
Berechnung der SR-Matrix für  $E_4$

Auch bei der Berechnung von SR kann ein Spezialfall, in der Art, wie der bei der Berechnung von SL, eintreten. Ist hierbei  $stem1(i, j+1)$ , der Wert, der zur Berechnung von  $E_3$  benötigt wird, gleich  $\infty$ , so wird auch hier eine *penalty* dazuaddiert. In diesem Fall ist  $E_3 = hairpin(j, k) + penalty$ .

Wenn  $j$  und  $k$  nicht paaren, werden  $SL(i, j, k)$  und  $stem2(j, k)$  wie folgt berechnet:

$$SR(i, j, k) = \min \begin{cases} SL(i, j+1, k-1) \\ SM(i, j+1, k-1) \\ SR(i, j+1, k-1) \end{cases}$$

$$stem2(j, k) = \min \begin{cases} stem2(j+1, k) \\ stem2(j, k-1) \end{cases}$$

#### Berechnung der SM-Matrix:

Wie schon erwähnt, wird bei der Berechnung von  $SM(i, j, k)$  davon ausgegangen, dass weder  $i$  mit  $j$ , noch  $j$  mit  $k$  paaren, selbst wenn sie dazu in der Lage wären. Daraus resultiert folgende Rekursionsgleichung:

$$SM(i, j, k) = \min \begin{cases} SL(i-1, j, k), SM(i-1, j, k) \\ SR(i, j+1, k), SL(i, j+1, k), SM(i, j+1, k) \\ SR(i, j, k-1), SM(i, j, k-1) \end{cases}$$

$stem1(i, j)$  und  $stem2(j, k)$  werden wie folgt berechnet. Bezieht  $SM(i, j, k)$  sein Minimum aus dem ersten der drei Fälle, so ist  $stem1(i, j) = stem(i-1, j)$ . Wenn das Minimum aus dem zweiten Fall stammt, so ist  $stem1(i, j) = stem1(i, j+1)$  und  $stem2(j, k) = stem2(j+1, k)$ . Und letztendlich, wenn das Minimum aus dem dritten Fall stammt, ist  $stem2(j, k) = stem2(j, k-1)$ .

Um eine simple Pseudoknoten-Struktur mit Endpunkten  $I$  und  $K$  zu finden, betrachtet der Algorithmus, wie auch der Algorithmus von Akutsu, drei Typen von Triplets  $SL(i, j, k)$ ,  $SR(i, j, k)$  und  $SM(i, j, k)$  für alle  $i, j, k$ , für die gilt ( $I \leq i < j < k \leq K$ ). Daraus resultiert auch die Laufzeit des Algorithmus.

Für jedes Paar (I, K) müssen  $O(n^3)$  Triplets berechnet werden. Der Score eines Triplets hängt aber von K nicht ab, da K in keiner Rekursion zur Berechnung benötigt wird. Scores für (i, j, k) hängen nicht von Scores (i', j', k') ab, wobei  $k < k'$ . Somit müssen also für alle I  $O(n^3)$  Triplets berechnet werden, was zu einer Laufzeit von  $O(n^4)$  führt. Der Speicherplatzbedarf beträgt  $O(n^3)$ , was aus der Verwendung der  $N \times N \times N$  Matrizen resultiert.

### 3.6. Auswertung

Für die Analyse des Algorithmus ist eine Menge von Test-Pseudoknoten aus der Datenbank Pseudobase ausgewählt worden. Diese Testmenge besteht aus 169 Sequenzen mit Längen zwischen 19 und 114 Nukleotiden. Der implementierte Algorithmus ist in der Lage von diesen 169 Test-Sequenzen 163 Pseudoknoten vorherzusagen. 6 Strukturen der Testmenge werden als einfache Sekundärstrukturen ohne Pseudoknoten vorhergesagt. Von den 163 vorhergesagten Pseudoknoten entsprechen 131 exakt oder nahezu exakt den Pseudoknoten in der Datenbank. Drei der sechs vorhergesagten einfachen Sekundärstrukturen, können, durch Erhöhung der *penalty*, in Pseudoknoten überführt werden. Eine weitere Struktur aus der Datenbank enthält da unzulässige Basenpaar A-G, weshalb der Algorithmus nicht in der Lage diese richtig vorherzusagen.

Ein Vergleich mit dem Algorithmus von Eddy und Rivas[4] führt zu Ergebnissen, die in Tabelle 2 zusammengefasst sind.

Die Berechnungszeiten für Sequenzen mit einer Länge von 75 Nukleotiden, betragen 55 Sekunden. Für Sequenzen mit der Länge von 114 Nukleotiden benötigt die Berechnung 8 Minuten.

**Tabelle 2**

Vergleich des Algorithmus von Eddy und Rivas mit dem von Deogun und Komina

Eddy/Rivas	Deogun/Komina
50 % der Pseudoknots erkannt	95 % der Pseudoknots erkannt. Davon 78 % mit korrekter oder fast-korrektur Struktur

#### 4. Zusammenfassung und Resümée

Zusammenfassend ist zu sagen, dass die Betrachtung von RNA-Sekundärstrukturen zur Vorhersage der Funktion von RNA-Molekülen essentiell nötig ist. Da es ja das Ziel von vergleichender Genomanalyse ist, funktionelle Signale in Primärsequenzen aufzudecken, stellt die Entwicklung von Modellen, die Sekundärstrukturen in Betracht ziehen, somit eine wichtige Aufgabe der Bioinformatik dar. Der Ansatz von Holmes zeigt, welchen Vorteil es bringt, bei der Bestimmung von Verwandtschaftsgraden zwischen Sequenzen, die Sekundärstruktur in Betracht zu ziehen. Bis zu einem gewissen Punkt, ab dem die strukturellen Unterschiede so groß werden, dass sie von dem Modell nicht mehr verarbeitet werden können, weisen die Alignments von zwei Sequenzen, unter Berücksichtigung ihrer jeweiligen Sekundärstruktur, durchweg bessere Ergebnisse auf, als Alignments, bei denen diese außen vor gelassen worden ist. Das Ziel muss also sein, weiter an diesen Modellen zu feilen. Die vom Autor angegebenen Verbesserungen des Modells, könnten wahrscheinlich zu einem wesentlich effektiveren Modell führen und sollten dementsprechend auch umgesetzt werden.

Auch die Betrachtung von Pseudoknoten-Strukturen, bei dem es im zweiten bearbeiteten Paper geht, ist sehr wichtig, da diese eine funktionelle Rolle ausüben und auch einen Einfluss auf die Tertiärstruktur eines RNA-Moleküls besitzen. Die Verwendung der Thermodynamik-Parameter führt im Wesentlichen zu besseren Ergebnissen, als die Verwendung der Anzahl an Basenpaaren. Die meisten Algorithmen zur Strukturvorhersage, die diese Parameter verwenden, sind darauf getrimmt Strukturen mit minimaler freier Energie zu finden, wozu sie auch zweifelsohne in der Lage sind. Die Frage sollte aber lauten, ob es denn sinnvoll ist, lediglich Strukturen mit minimaler freier Energie zu finden. Sicherlich stellen diese Strukturen die optimalen Faltungen von Molekülen dar. Allerdings ist es auch so, dass in der Natur nicht immer nur optimale Strukturen auftreten. Man sollte also bei der Bestimmung von Strukturen, auch die Strukturen betrachten, die nur suboptimale Faltungen aufweisen, da die Wahrscheinlichkeit sehr hoch ist, dass auch diese in der Natur auftreten. Mit dem in dieser Arbeit gezeigten Algorithmus, würden diese Strukturen unter den Tisch gekehrt, da dort nur die eine minimale Struktur bestimmt wird.

## 5. Quellen

- [1] Akutsu (2000): **Dynamic programming algorithm for RNA secondary structure prediction with pseudoknots**, Discrete Applied Mathematics
- [2] Deogun, Komina et al. (2004): **RNA Secondary Structure Prediction with Simple Pseudoknots**, APBC2004
- [3] Durbin et al. (1998): **Biological Sequence Analysis**, Cambridge University Press
- [4] Eddy, Rivas (1999): **A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots**, JMB 1999
- [5] Holmes (2004): **A probabilistic model for the evolution of RNA structure**, BMC Bioinformatics
- [6] Mattick (2005): **Das verkannte Genom-Programm**, Spektrum der Wissenschaft (März 05)
- [7] Thorne, Kishino, Felsenstein (1991): **An evolutionary model for maximum likelihood alignment of DNA sequences**, J Mol Evol
- [8] Zuker et al.: **Algorithms and thermodynamics for RNA secondary structure prediction: A practical guide**, NATO ASI Series