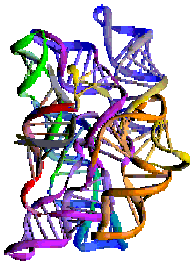
Johann Wolfgang Goethe Universität
Frankfurt am Main

Vorhersage von RNA-Sekundärstrukturen inklusive Pseudoknoten



Natalie Jäger
Natalie.Jaeger@web.de

29. Juni 2005

Inhaltsverzeichnis

1 Einleitung

RNA ist ein so genanntes Polynukleotid. Jedes Nukleotid der RNA besteht aus einer Ribose, d.h. einem C-5-Zucker, sowie einem Phosphatrest und einer von vier möglichen organischen Basen. In der RNA kommen die folgenden vier Basen vor: Adenin, Guanin, Cytosin und Uracil. Die DNA hingegen nutzt anstelle des Uracil die Base Thymin. Der Aufbau von RNA und DNA ist recht ähnlich. Jedoch sind RNA-Moleküle - im Gegensatz zur doppelsträngigen DNA - einsträngige Polynukleotide. Dieser Unterschied ist entscheidend, denn er erhöht die katalytische Funktion der RNA und erlaubt ihr chemische Reaktionen, die der DNA nicht möglich sind. RNA erfüllt eine Vielzahl an Funktionen, wie beispielsweise in Form der messenger-RNA. Diese dient als Mediator zwischen der DNA und der Proteinsynthese. Die Translation der mRNA in ein Protein ist der letzte große Schritt, um die Informationen aus dem Genom aktiv in der Zelle umzusetzen.
RNA-Sekundärstrukturen werden durch Interaktionen zwischen komplementären Nukleotid-Paaren festgelegt (über Wasserstoff-Brücken), die nah oder weit im Molekül voneinander entfernt sind. Diese Interaktionen falten die RNA schließlich in solche Formen wie Stem Loops oder die komplizierteren Pseudoknoten.
Ähnlich den Proteinen, bestimmt die räumliche Struktur der RNA ihre Wirkungsweise. Die Sekundärstruktur hängt also stark mit der Funktion der RNA zusammen, weshalb man versucht diese Strukturen vorherzusagen.
Pseudoknoten wurden generell bei Vorhersage-Methoden zur RNA-Sekundärstruktur außen vor gelassen, da sich deren Modellierung schwierig gestaltet. Zur Vorhersage von Stem Loops existieren dagegen eine Vielzahl an erfolgreichen Methoden.
In dieser Ausarbeitung werden drei verschiedene Methoden zur Vorhersage von RNA-Sekundärstrukturen, wobei diese Ansätze auch Pseudoknoten berücksichtigen, vorgestellt.
Die zuerst vorgestellte Vorhersage-Methode basiert auf stochastischem Modellieren der Sekundärstrukturen durch Grammatiken nach Chomsky, eine weitere Methode verfolgt einen Graph-theoretischen Ansatz, und abschließend wird noch eine Erweiterung des Loop-Matching-Algorithmus nach Nussinov zur Strukturvorhersage vorgestellt.

2 Stochastisches Modellieren von RNA-Pseudoknoten durch Grammatiken

Stem Loops kann man seit kurzem erfolgreich mit stochastisch kontextfreien Grammatiken (SCFG) modellieren. Grammatiken nach Chomsky sind ideal um Modellieren von Interaktionen zwischen Nucleotiden, denn Stems sind komplementär-palindromartig und Palindrome lassen sich durch kontextfreie Grammatiken erzeugen. Pseudoknoten sind jedoch komplexer als Stem Loops und würden formal eine kontextsensitive Grammatik erfordern, was aber die Komplexität stark erhöht.

2.1 Parallel Communicating Grammar Systems

Die hier vorgestellte Methode verfolgt einen neuen Modeling-Ansatz mit Grammatiken für RNA-Pseudoknoten, basierend auf *parallel communicating grammar systems* (PCGS).
So kann eine kontextsensitive Struktur, wie ein Pseudoknoten, durch *eine* kontextfreie Grammatik synchronisiert mit einer Vielzahl an regulären Grammatiken generiert werden. Die stochastische Version der PCGS wird dadurch so einfach wie bei herkömmlichen SCFG. Die (eine) SCFG beinhaltet spezielle Nichtterminale, die sogenannten *query symbols*, aus denen Strukturen ableitbar sind, welche die für Pseudoknoten typischen Crossing Helices formen.
Bevor diese Grammatiken genauer beschrieben werden, folgt eine Definition der RNA-Pseudoknoten, welche die Modellierung basierend auf PCGS erleichtert:
Eine *Base-Region* ist eine Teilsequenz von bestimmter Länge innerhalb der Gesamtsequenz s.
1. In der RNA-Sequenz s beinhaltet die Teilsequenz t_i eine *potential region*, wenn eine Base-Region zu einer Helix in s beiträgt, aber *nicht* zu einer Helix in t_i. Eine *potential region* ist demnach eine Base-Region in einer Teilsequenz t_i, so dass diese Region eine Helix bildet, die jedoch *nicht* in der Teilsequenz t_i liegt, sondern in *s*. In Abb.1 sind c und c´ somit *potential regions*, denn c ist eine potential region bezüglich t_1 und c´ bezüglich t_2. Da sich die RNA-Sequenz s aus t_1 und t_2 zusammensetzt, bildet sich in s eine Helix durch die beiden *potential regions*, wie in Abb.2 dargestellt.

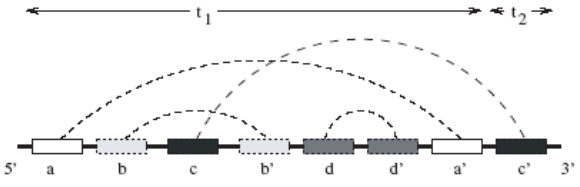


Abb. 1 Eine pseudogeknotete Struktur, wobei c in t1 und c´ in t2 beide potential regions darstellen

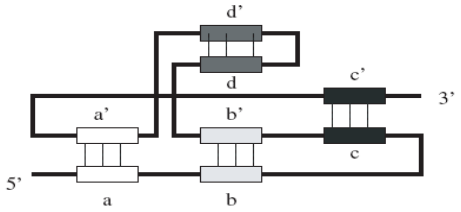


Abb. 2 Die Sekundärstruktur zur in Abb. 1 dargestellten Primärsequenz bildet einen Pseudoknoten durch c und c´

2. Die Teilsequenz t ist eine *P-Structure*, wenn sie eine *potential region* enthält. Zudem ist t eine *nicht-triviale P-Structure***,** wenn die potential region zwischen zwei base-paired regions liegt. In Abb.1 ist t_1 eine nicht-triviale P-Structure, weil die potential region c zwischen der base-pairing region b und b´ liegt.
3. s ist eine RNA-Sequenz. s ist eine *pseudogeknotete Struktur*, wenn sie zwei nicht-überlappende P-Strukturen enthält, wobei *eine* davon *nicht-trivial* ist (hier: t_1), und beide potential regions bilden eine Doppelhelix. In Abb.2 ist zu sehen, dass c und c´ eine solche Doppelhelix bilden und somit die ganze Struktur pseudogeknotet ist.
Durch diese Definition können alle RNA-Pseudoknoten beschrieben werden.

Ein *parallel communicating grammar system* **PCGS** Γ besteht aus einer Anzahl an Chomsky Grammatiken, den sogenannten Components G_i, wobei die Grammatik G_0 *Master* genannt wird. Nur die Master-Grammatik ist kontextfrei, während die Hilfsgrammatiken $G_1,...,G_k$ regulär sind. Alle Grammatiken teilen sich ein Alphabet, wobei das im Falle von RNA die Terminale a, c, g und u beinhaltet, und auch die

Variablen, also Nonterminale werden geteilt. Es gibt zusätzlich spezielle Nonterminale - *Query Symbols* – diese sorgen für die Kommunikation zwischen den Grammatiken. Eine Component kann Sequenzen anfragen, die von anderen Grammatiken erzeugt wurden, also aus diesen Grammatiken ableitbar waren, und mehrere Components können gleichzeitig Sequenzen aus einer bestimmten Component anfragen.
Synchronisierung zwischen den Ableitungen der einzelnen Components erhält man durch die Query Symbols Q_i. Synchronisierung ist wichtig, weil man eine base-pairing region aus zwei verschiedenen regulären Grammatiken ableiten möchte, das heißt, dass eine Grammatik die komplementäre Sequenz zur anderen Grammatik erzeugt. Daher müssen die Ableitungsschritte innerhalb beider Grammatiken parallel erfolgen, um schließlich komplementäre Basen zu erhalten. Ein Beispiel soll dieses Vorgehen verdeutlichen.
Abb.3 zeigt einen Teil des PCGS Γ, nämlich die drei regulären Hilfsgrammatiken G_1, G_2 und G_3. Die Master-Grammatik ist noch nicht spezifiziert. In Abb.4 ist das parallele Ableiten der beiden base-paired regions acg aus G_1 und cgu aus G_2 dargestellt.

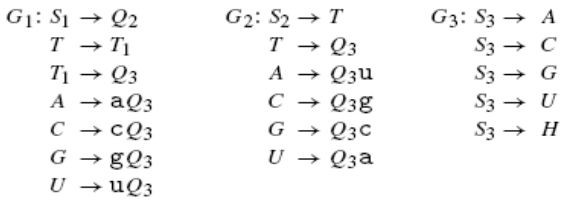


Abb. 3 Die drei regulären Hilfsgrammatiken des PCGS

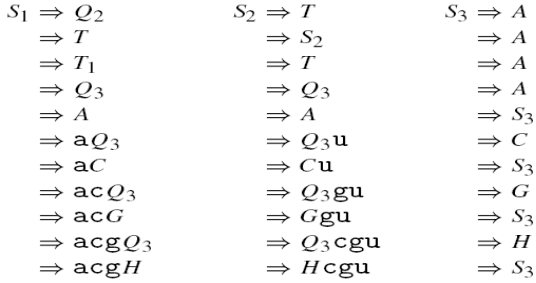


Abb. 4 Paralleles Ableiten der base-pairing regions acg und cgu;
von oben nach unten ist der *zeitliche* Verlauf des Ableitens dargestellt

Synchronisierung zwischen G_1 und G_2 während des parallelen Ableitens erhält man durch die Produktion $S_1 \to Q_2$, weil dadurch G_2 zuerst abgeleitet wird. Das query symbol Q_2 sorgt somit für einen Wechsel in die Grammatik G_2. G_1 muss somit warten bis G_2 mit dem Ableiten des Nichtterminals T beginnt ($S_2 \to T$), und dieses T wird dann in G_1 kopiert (daher $S_1 \to T$ im zweiten Ableitungsschritt in G_1), so dass weitere Ableitungen in G_1 möglich sind. Nachdem eine Grammatik G_i angefragt wurde, kehrt sie immer zum Startsymbol S_i zurück, bevor weiter abgeleitet werden kann. Daher leitet man im zweiten Schritt in G_2 wieder $S_2 \to S_2$ ab, so dass sich Grammatik G_2 wieder am Start der Ableitung befindet. Nach diesem Schritt kann natürlich nur $S_2 \to T$ abgeleitet werden, da sich die Grammatik G_2 schließlich wieder am Startsymbol S_2 befindet. G_1 und G_2 fragen anschließend gleichzeitig Ableitungen in G_3 an, wobei die Grammatik G_3 nach den Anfragen auch wieder zum Startsymbol S_3 zurückkehren muss. Trotzdem erzeugt das in G_1 und G_2 kopierte gleiche Symbol aus G_3 jeweils verschiedene Ableitungen, da eine Base (aus G_1) und ihr Komplement (aus G_2) erzeugt werden soll. Im (zeitlich gesehen) vierten Schritt der Ableitung fragen G_1 und G_2 gleichzeitig in G_3 ein Nichtterminal A an, das dann aber in den beiden Grammatiken G_1 und G_2 verschiedene Terminale erzeugt, nämlich *a* in G_1 und das Terminal *u* in G_2, also komplementäre Basen im Sinne des RNA-Codes.

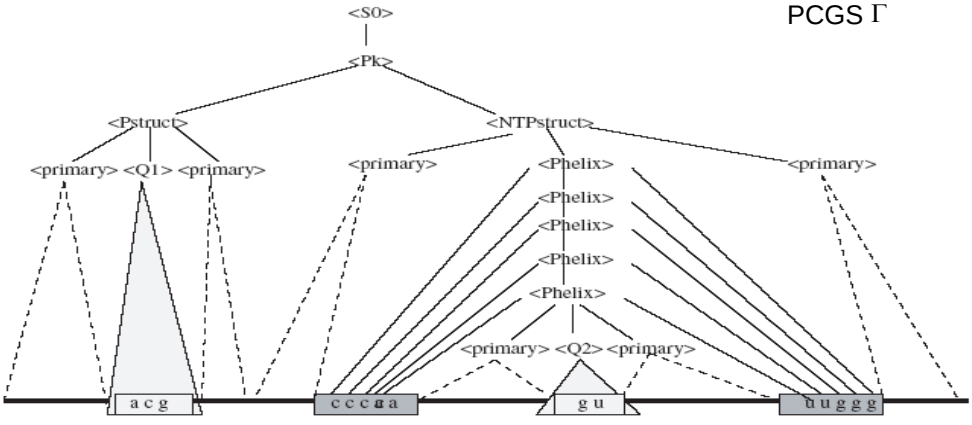
Mit den regulären Hilfsgrammatiken lassen sich, wie oben gezeigt, base-pairing regions, also Helices, erzeugen, aber noch keine pseudogeknoteten Strukturen. Das ist nur durch die kontextfreie Master-Grammatik möglich. Im wesentlichen besteht ein Pseudoknoten nach obiger Definition aus zwei nicht-überlappende P-Strukturen, wovon genau eine nicht-trivial ist, bezüglich einer Zerlegung der Sequenz *s* in t_1 und t_2. Diese beiden P-Strukturen enthalten die base-pairing regions acg und cgu, die in diesem Beispiel durch G_1 und G_2 erzeugt werden. Die entsprechende Master-Grammatik G_0 ist in Abb. 5 angegeben.

G_0: $S_0 \rightarrow Pk$
$Pk \rightarrow P\text{-}struct\ NT\text{-}P\text{-}struct$
$P\text{-}struct \rightarrow Primary\ Q_1\ Primary$
$NT\text{-}P\text{-}struct \rightarrow Primary\ P\text{-}helix\ Primary$
$P\text{-}helix \rightarrow$ a $P\text{-}helix$ u
$P\text{-}helix \rightarrow$ c $P\text{-}helix$ g
$P\text{-}helix \rightarrow$ g $P\text{-}helix$ c
$P\text{-}helix \rightarrow$ u $P\text{-}helix$ a
$P\text{-}helix \rightarrow Q_2$

Abb. 5 Die kontextfreie Master-Grammatik G_0 der PCGS

In den angegebenen Produktionen der Master-Grammatik steht *Pk* für einen Pseudoknoten, und dieser besteht wiederum aus P-Struktur (*P-Struct)* und nicht-trivialer P-Struktur (*NT-P-struct*). Denn es gilt, wie oben erläutert, dass ein Pseudoknoten durch eine P-Struktur und eine nicht-triviale P-Struktur eindeutig charakterisiert ist. Das Nichtterminal *Primary* steht für irgendeine Teilsequenz, während aus *P-helix* eine Helix ableitbar ist, die aus der *potential region* erzeugt durch Q_2 besteht, und paart mit der *potential region* erzeugt durch Q_1. Die *potential region* erzeugt durch Q_2 wird eingebettet in eine weitere base-pairing region und folglich ist dieser Teil nicht-trivial. Somit sind alle Kriterien für einen Pseudoknoten erfüllt. Abb. 6 zeigt den Ableitungsbaum zum PCGS Γ aus obigem Beispiel.

Abb. 6 Der Ableitungsbaum zum PCGS Γ



2.2 Stochastische Version des Modells

Die stochastische Version des PCGS erhält man, indem Wahrscheinlichkeiten mit den Produktionsregeln jeder Component, sprich Grammatik, des PCGS assoziiert werden. Das realisiert man am einfachsten durch die Definition einer Wahrscheinlichkeitsverteilung für jede einzelne Component als unabhängige SCFG. Die Wahrscheinlichkeit für einen parallelen Ableitungsschritt muss aber die bedingten Wahrscheinlichkeiten berücksichtigen, die durch die Kommunikation zwischen den

Grammatiken entstehen. Die einfachen Produktionsregeln der PCGS erlauben den Beweis, dass die stochastische Version des PCGS-Modells für RNA-Pseudoknoten wohldefiniert ist.
Formal bedeutet das folgendes: Sei Σ = {a, u, c, g}, Γ eine PCGS mit m Components, wobei S_0 bis S_{m-1} Startsymbole der m Grammatiken sind. Dann ist L_Γ = {s $\in \Sigma^*$: $(S_0,S_1,\ldots,S_{m-1}) \rightarrow^* (s,y_1,\ldots,y_{m-1})$ } die Menge aller pseudogeknoteten Strukturen die Γ generiert. S_0 ist das Startsymbol der Mastergrammatik G_0, die restlichen S_i sind jeweils Startsymbole der regulären Grammatiken. Dann muss gelten

$$\sum_{s \in L_\Gamma} Pr(S_0 \Rightarrow^* s) = 1$$

Die Wahrscheinlichkeit für einen Pseudoknoten lässt sich somit allein durch die Master-Grammatik G_0 berechnen, wenn die Wahrscheinlichkeiten für *Crossing Helices*, die durch die regulären Hilfsgrammatiken generiert werden, bekannt sind. Zum Beweis sei $s_1r_1s_2r_2s_3$ eine pseudogeknotete Struktur in der r_1 und r_2 beide potential regions sind, die eine Helix formen. Sei $S_0 \rightarrow s_1r_1s_2r_2s_3$ eine Ableitung, dann sieht die dazugehörige Wahrscheinlichkeit wie folgt aus

$$\begin{aligned} &Pr(S_0 \Rightarrow^* s_1 r_1 s_2 r_2 s_3) \\ &= Pr(S_0 \Rightarrow^* s_1 Q_1 s_2 Q_2 s_3) Pr(S_1 \Rightarrow^* r_1 \& S_2 \Rightarrow^* r_2) \\ &= Pr(S_0 \Rightarrow^* s_1 Q_1 s_2 Q_2 s_3) Pr(S_1 \Rightarrow^* r_1) \\ &\quad Pr(S_2 \Rightarrow^* r_2 | S_1 \Rightarrow^* r_1) \end{aligned}$$

weil die Generierung von r_1 und r_2 synchron verläuft. Angenommen $r_1 = x_1 \ldots x_k$ und $r_2 = y_1 \ldots y_k$, wobei $x_j, y_i \ni \Sigma \cup \{\varepsilon\}$ und ε bedeutet leer. Da jeder Buchstabe in r_1 und r_2 unabhängig innerhalb der Sequenz generiert wird, gilt

$$Pr(S_1 \Rightarrow^* r_1) = \prod_{j=1}^{k} Pr(X_j \rightarrow x_j Q_3) Pr(S_3 \rightarrow X_j)$$

$$\text{and } Pr(S_2 \Rightarrow^* r_2 | S_1 \Rightarrow^* r_1) = \prod_{j=1}^{k} Pr(Y_j \rightarrow Q_3 y_j | X_j \rightarrow x_j Q_3)$$

wobei X_j eines der Nonterminale A, C, G und U aus G_1 ist und Y_j ist das entsprechend synchrone, also komplementäre Nonterminal aus G_2. Auf den ausführlichen Beweis sei an dieser Stelle nur verwiesen (Cai, Malmberg, Wu,2003).

Zusammenfassend bedeutet das, dass die generierte Struktur und Wahrscheinlichkeitsberechnung durch das Pseudoknoten-Modell aus PCGS unabhängig von den regulären Hilfsgrammatiken ist. Zum Modellieren der *Crossing Helices* benötigt man somit nur eine 5x5 probabilistische Matrix. Diese Matrix beschreibt die Wahrscheinlichkeits-Verteilung der vier Basen und der gaps, da *bulges* auch erlaubt sind, in den Crossing Helices. Weitere Wahrscheinlichkeitsverteilungen bezüglich der regulären Grammatiken des PCGS sind nicht nötig.

Die stochastische Version des PCGS ist somit *nur* die stochastische Version der kontextfreien Master-Grammatik G_0. Denn die Sprache, die durch das PCGS schließlich erzeugt wird, ist eine Menge von Strings, besser Sequenzen, welche die Master-Grammatik G_0 erzeugt. Einziger Unterschied zu sonstigen SCFG sind die Query Symbols, die als Nonterminale dazu dienen, Pseudoknoten zu generieren.

2.3 Automatisierter Algorithmus zur Pseudoknoten-Vorhersage

Als Resultat aus den Parallel Communicating Grammar Systems erhält man, gleich den SCFG-Modellen, ein System, das automatisch einen „Pseudoknoten-Vorhersage-Algorithmus“ für jede pseudogeknotete Struktur generiert. Zum Modellieren der *Crossing Helices*, repräsentiert durch die Query Symbols, benötigt man eine 5x5 probabilistische Matrix. Diese Matrix beschreibt die Wahrscheinlichkeits-Verteilung der vier Basen und gaps in den Crossing Helices.
Der Algorithmus basiert auf Dynamischem Programmieren, ähnlich dem CYK-Algorithmus, der das Wortproblem für gegebene kontextfreie Sprachen effizient löst.
Die Eingabe besteht aus einer SCFG, nämlich der Master-Component G_0, die in Chomsky Normalform vorliegen muss, gleich der Eingabebedingung für den CYK-Algorithmus.
Für die Eingabe-Sequenz x[1..n] berechnet der Algorithmus für jedes Nichtterminal X die maximale Wahrscheinlichkeit, dass die Teilsequenz x[i..j] daraus abgeleitet wird.
Der Algorithmus unterscheidet dabei drei Kategorien von Teilsequenzen: stem-loops, Pseudoknoten und P-Strukturen. Diese drei möglichen Teilsequenzen werden wie folgt berechnet:
1. Die Berechnung für stem-loops folgt dem gewöhnlichen CYK-Algorithmus.

2. Die Berechnung von Pseudoknoten erfolgt über eine Hilfsfunktion *H,* welche für jedes Paar an Teilsequenzen die maximale Wahrscheinlichkeit angibt, eine Crossing Helix zu bilden. Sei x[h..l] und x[u..v] jeweils eine Teilsequenz, dann steht H(h,l,u,v) für die maximale Wahrscheinlichkeit, dass x[h..l] und x[u..v] eine Crossing Helix bilden. Die Berechnung der Funktion H(h,l,u,v) erfolgt separat durch einen paarweise komplementären Alignment-Algorithmus. Für das Nichtterminal X wird die maximale Wahrscheinlichkeit, aus X einen Pseudoknoten x[i..j] abzuleiten, so berechnet:

$$Pr(X, i, j) = \max_{khluvYZ} \{ H(h, l, u, v)\, Pr(Y, i, k, h, l)\, Pr(Z, k+1, j, u, v)\, Pr(X \rightarrow YZ) \}$$

wobei Y und Z Teilsequenzen sind, welche die potentiellen base-pairing regions x[h..l] und x[u..v] enthalten. Diese Berechnung wird in Abb. 7 veranschaulicht.

3. Die maximale Wahrscheinlichkeit für das Nonterminal X, eine P-Struktur x[k..l] aus der Teilsequenz x[i..j] abzuleiten, ist wie folgt definiert (siehe Abb. 8):

$$Pr(X, i, j, k, l) = Pr(X \rightarrow^* x[i..k-1]\, Q_x\, x[l+1..j])$$

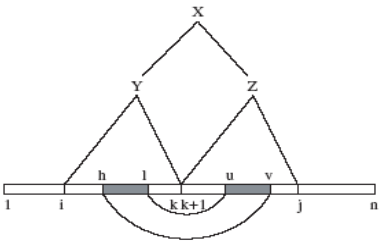


Abb. 7 Die Berechnung der maximalen Wahrscheinlichkeit für die pseudogeknotete Struktur X, aus der maximalen Wahrscheinlichkeit, dass (h,l) und (u,v) eine Crossing Helix bilden

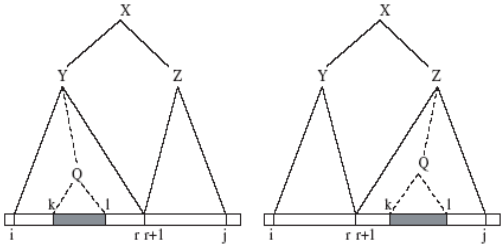


Abb. 8 Die Berechnung der maximalen Wahrscheinlichkeit für eine P-Struktur X, aus der maximalen Wahrscheinlichkeit für zwei Teilstrukturen Y und Z (eine davon ist eine P-Struktur)

Die Laufzeit im worst case beträgt $O(n^6)$ und $O(n^4)$ für den Speicherbedarf (RAM). In Tests konnte jedoch durch bestimmte Implementierungstechniken gezeigt werden, dass die Laufzeit sich signifikant reduzieren lässt.
Verglichen mit bisherigen Pseudoknoten-Vorhersage-Methoden, die auch auf Grammatiken nach Chomsky basieren (Rivas und Eddy, 2000), wird in diesem Modell nur ein neues Symbol – die *Query Symbols* – eingeführt, und nur *eine* kontextfreie Grammatik, synchronisiert mit einer Anzahl an regulären Grammatiken verwendet.

3 Graph-theoretischer Ansatz

Eine weitere neue Methode dient dazu, RNA-Sekundärstrukturen in einer Menge von funktionell oder evolutionär verwandten Sequenzen vorherzusagen, durch Anwendung eines Graph-theoretischen Ansatzes.
Diese Methode basiert auf dem Vergleich von Stem-Loops zwischen verwandten Sequenzen. Der Algorithmus findet eine Menge von stabilen Stem-Loops, die in mehreren Sequenzen konserviert vorliegen – daraus lässt sich die Konsensus-Sekundärstruktur formen.
Das generelle Schema dieser Methode lässt sich in drei Punkten gliedern:
1.Finden aller möglichen stabilen Stems in jeder Sequenz und diese vergleichen mit denen aller anderen Sequenzen.
2.Finden aller potentiell konservierten Stems, die in Teilmengen der Sequenzen gemeinsam vorliegen. Realisiert wird diese Suche durch das Bilden maximaler Cliquen.
3.Zusammenfügen der besten Mengen von konservierten Stems, um eine Konsensus-Sekundärstruktur zu konstruieren.
Die Ausgabe des Algorithmus ist schließlich eine Menge an möglichen Konsensus-Strukturen, geordnet nach höchstem Score. Diese Methode setzt keine globale Sequenzähnlichkeit voraus, kann aber, bei gegebener Ähnlichkeit, diese ausnutzen, was sich sehr positiv auf die Laufzeit auswirkt.

3.1 Finden aller stabilen Stems in jeder Sequenz

Per Definition ist ein *Stem* eine palindromische Helix in einer Sequenz, welche die Basenpaare AU, GC oder das Wobble-Basenpaar GU umfasst, mit einer minimalen

Länge von L Basenpaaren. Es gilt nun, alle Stems in jeder Sequenz zu finden, um jedoch den Suchraum zu reduzieren werden nur stabile Stems betrachtet. Die Stabilität eines Stems wird durch seine Stacking-Energie nach Turner evaluiert, das bedeutet, dass nur Stems mit einer Stacking-Energie niedriger als Wert E als stabil gelten. Dieser Wert E ist benutzerdefinierbar und kleiner Null. Der Default-Wert für E beträgt –5 kcal. Internal Loops oder Bulges sind im Stem nicht erlaubt, bis zur Phase des Structure-Refinements. Das Auflisten aller möglichen stabilen Stems erfolgt durch einen branch-and-bound Algorithmus, realisiert durch das Programm *dotplot*.

3.2 Vergleich von Stems über mehrere Sequenzen

Um konservierte Stems für die Konsensus-Struktur zu finden, vergleicht man jedes Paar an Sequenzen und deren Stems. Globales Alignieren von zwei Sequenzen nach dem Needleman-Wunsch-Algorithmus, hilft große Sequenzähnlichkeit auszunutzen. Im Alignment sucht man dann die *highly conserved regions.* Das sind Regionen, die 10 Nukleotide oder länger sind und mindestens 80% Sequenzidentität aufweisen. Die *highly conserved regions* dienen als Anker für Stem-Vergleiche über mehrere Sequenzen.
Zwei Stems von zwei Sequenzen können nur verglichen werden, wenn die dazu gehörenden 5‘ oder 3‘ half-stems in der gleichen Anker oder Nicht-Anker Region liegen. Solch ein half-stem stellt die eine komplementäre Hälfte des Stems in der Primärsequenz dar. Diese half-stems können nah oder weit voneinander entfernt sein innerhalb einer Sequenz.
Zudem dürfen die 5‘ oder 3‘ half-stems in der Anker Region um maximal 10 Nukleotide versetzt sein, während in der Nicht-Anker Region keine weiteren Einschränkungen gelten. Falls nach dem Alignieren die Ähnlichkeit zwischen zwei Sequenzen nicht groß ist, und somit keine *highly conserved regions* vorhanden sind, gilt die ganze Sequenz als Nicht-Anker Region und somit wird *jeder* Stem der beiden Sequenzen miteinander verglichen. Das hat eine höhere Laufzeit zur Folge. Der Algorithmus kann also globale Sequenzähnlichkeit ausnutzen.
Zum Vergleich der Ähnlichkeit zwischen Stems aus verschiedenen Sequenzen dient die Funktion S. Die Funktion $S(i_x, j_y)$ misst die Ähnlichkeit zwischen zwei *Stems i* und *j* aus den Sequenzen x und y. Die Ähnlichkeit zwischen zwei Stems ist anhand von fünf Eigenschaften messbar: Helix-Länge, Helix-Sequenz, Loop-Sequenz (abgeschlossen durch den Stem), Stem-Stabilität, relative Positionen des Starts und Endes des Stems.
$S(i_x, j_y)$ ist definiert als die gewichtete Summe dieser fünf Ähnlichkeits-Scores, geteilt durch die Summe des Stabilitäts-Scores der beiden Stems. Die Funktion wird noch mit dem *stability adjusting factor f* skaliert, der die Wichtigkeit der Stabilität eines Stems evaluiert.

$$S\left(i_x, j_y\right) = \frac{\sum_{l=l_1 \sim l_5} \left(w_l \cdot s_l\left(i_x, j_y\right)\right)}{r_x\left(i\right) + r_y\left(j\right) + f} \cdot \; f$$

wobei $\mathbf{s_l(i_x, j_y)}$ der Ähnlichkeits-Score zwischen den Stems i_X, j_Y ist, bezogen auf eine der fünf möglichen spezielle Eigenschaft **l**. Die Funktionenen $\mathbf{s_l(i_x, j_y)}$ werden, außer für Helix- und Loop-Sequenz, wie folgt berechnet:

$$s_l(i_X, j_y) = \min \{ s_l(i_X), s_l(j_y)\} / \max\{s_l(i_X), s_l(j_y) \}$$

Für die Helix- und Loop-Sequenz nimmt die Funktion $s_l(i_X, j_y)$ beispielsweise den Wert 0,7 an, wenn die beiden Sequenzen 70% Sequenzähnlichkeit aufweisen; s_l ist in diesem Fall die *fractional identity* zwischen beiden Sequenzen.
In der Funktion $\mathbf{S(i_x, j_y)}$ ist $\mathbf{w_l}$ ist das Gewicht für jede Eigenschaft l und liegt zwischen 0 und 1, wobei alle fünf Gewichte aufsummiert eins ergeben. Die Helix- und Loop-Sequenz, sowie die relative Positionen des Starts und Endes des Stems werden mit 1/4 gewichtet, während die übrigen Eigenschaften mit 1/8 gewichtet werden. Die Werte von **r** liegen zwischen 0 und 1 – je stabiler ein Stem, desto niedriger der r-Wert.Der Wert der Ähnlichkeitsfunktion $\mathbf{S(i_x, j_y)}$ liegt zwischen 0 und 1 – je höher der Wert, um so wahrscheinlicher, dass zwei Stems Instanzen eines konservierten Stems sind.
Nur die Paare an Stems werden als potentiell eingestuft, für die $S(i_x, j_y) >= S$ gilt, für einen Schwellwert S.

3.3 Finden der konservierten Stems

Ziel ist es nun, die konservierten Stems zu finden, die in mindestens k Sequenzen vorkommen, wobei $k = [p * n]$ ist. Dabei ist p das *Signifikanz-Level p* (0< p <=1), welches der minimale prozentuale Anteil aller n Sequenzen ist, die eine gemeinsame Struktur besitzen. Der Default-Wert von p liegt bei 0,5 , basierend auf der generellen Annahme, dass mindestens die Hälfte der Eingabesequenzen eine gemeinsame Struktur besitzen sollte.
Die konservierten Stems, die in mindestens k Sequenzen auftreten, werden mithilfe eines n-partiten ungerichteten gewichteten Graphen gefunden, wie in Abb. 9 dargestellt. Der Graph repräsentiert die Stems und ihre Beziehungen untereinander. Jeder Koten dieses Graphen repräsentiert einen Stem und der Graph ist unterteilt in *n* Teile. Jeder Teil umfasst die Anzahl an Stems einer Sequenz. Nur Knoten der verschiedenen *n* Teilen können verbunden werden, also eine Kante im Graphen erzeugen. Potentiell gleiche Stems aus verschiedenen Sequenzen, die einen Ähnlichkeits-Score größer als Schwellwert S aufweisen, werden verbunden und gewichtet.

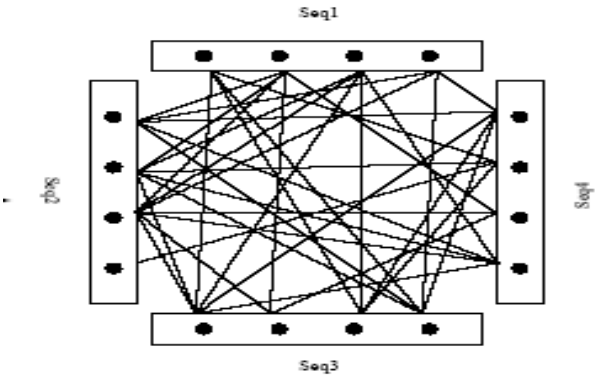


Abb. 9 Der n-partite ungerichtete gewichtete Graph. Die Knoten entsprechen den zu vergleichenden Stems.

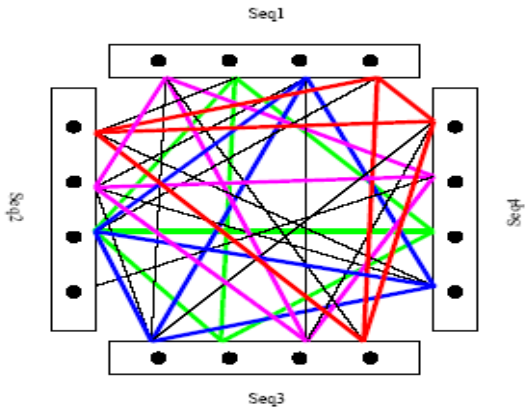


Abb. 10 Alle maximalen Cliquen des Graphen, die >= k sind. Jede maximale Clique repräsentiert einen potentiell konservierten Stem.

Um die Stems zu finden, die über mehrere Sequenzen hinweg konserviert sind, sucht man diese auf Basis eines multi-way Vergleiches. Wenn k Stems aus k verschiedenen Sequenzen alle potentiell gleich sind, ist es sinnvoll anzunehmen, dass diese Instanzen eines konservierten Stems sind. Man realisiert diese all-way Verbindung durch eine *Clique*, wie in Abb. 10 veranschaulicht. In der Graphen-Theorie repräsentiert eine Clique einen vollständigen Teilgraphen, in dem jeder Knoten mit allen anderen verbunden ist. Die Größe einer Clique ergibt sich aus der Anzahl ihrer Knoten. Eine Clique ist maximal, wenn sie nicht in einer größeren Clique enthalten ist. Cliquenbildung stellt ein Maximierungsproblem dar und ist NP-vollständig. Das Finden aller potentiell konservierten Stems in mindestens k Sequenzen, entspricht somit dem Finden aller maximalen Cliquen der Größe >= k im n-partiten Graphen.
Allerdings stellt dies ein NP-hartes Problem dar. Besser ist hier die Anwendung eines enumerativen Algorithmus, basierend auf Depth-First-Suche im Graphen. Die Eingabe für diesen Algorithmus ist der n-partite Graph; Ausgabe ist eine Array mit allen maximalen Cliquen, die >= k sind. Dieser Algorithmus startet eine Clique mit jedem Stem von jeder Sequenz und vergrößert diese durch inkrementelles Einfügen eines Stems aus einer anderen Sequenz, solange danach noch alle Knoten der Clique miteinander verbunden sind. Das Erweitern der Clique wird solange fortgesetzt bis alle Sequenzen durchsucht wurden. Bei diesem Erweiterungsschritt prüft der Algorithmus bereits, ob es überhaupt möglich ist, dass die Clique die Mindestgröße von k erreicht. Ist das nicht möglich, so wird diese Clique verworfen. Jede maximale Clique wird mit einem Score versehen, der sich aus dem durchschnittlichen Kantengewicht multipliziert mit der Anzahl an Knoten der Clique ergibt.
Die Cliquen werden nach absteigendem Score angeordnet – haben zwei Cliquen mehr als 70% an gleichen Stems, wird die Clique mit niedrigerem Score entfernt.
Je größer eine Clique ist, desto ähnlicher sind die Stems zueinander, um so wahrscheinlicher, dass es sich um Instanzen konservierter Stems handelt, die schließlich teil der Konsensusstruktur sein werden.
Die worst case Laufzeit für diesen Algorithmus beträgt $O(m^n)$, wobei *m* die maximale Anzahl an Stems ist, die in einer Sequenz betrachtet wurden. *n* ist die Anzahl aller Sequenzen. Diese Laufzeit lässt den Algorithmus wenig sinnvoll erscheinen, doch in praktischen Anwendungen konnte gezeigt werden, dass die average case Laufzeit weit unter der des worst case liegt. Das hängt damit zusammen, dass die Eingabe-Graphen meist nur dünn besetzt sind, wegen der strikten Definition der Ankerregionen. In Tests mit biologischen Sequenzen von mittlerer Größe wurden durchweg Laufzeiten von nur einigen Sekunden erzielt.

3.4 Zusammenfügen kompatibler konservierter Stems

Jede gefundene maximale Clique entspricht einer Menge ähnlicher Stems aus verschiedenen Sequenzen. Die Größe der Cliquen liegt immer zwischen k und n. Ein *stem block* repräsentiert die Menge an Stems, die der maximalen Clique entspricht. Stem Blocks können unabhängig voneinander sein oder überlappen in verschiedenen Teilmengen von Sequenzen. Es bestehen viele Möglichkeiten diese anzuordnen, um letztendlich die Konsensusstruktur zu formen. Ziel ist es, die bestmögliche Zusammenstellung an *stem blocks* zu finden, welche schließlich die Konsensussekundärstruktur repräsentiert. Um dieses Ziel zu erreichen, wendet man

eine Graph-theoretische Methode an, ähnlich dem topologischen Sortieren. Die Idee dieser Methode ist, möglichste viele kompatible stem blocks zu einer Struktur zusammenzufügen, beginnend beim 5' Ende der Sequenz. Im Anschluss bewertet man die Plausibilität einer solchen Struktur anhand ihres Scores, der sich aus der Summe aller enthaltenen stem blocks ergibt. Das Problem beim Zusammenfügen dieser blocks besteht darin, dass nicht nur die Anordnung der stem blocks innerhalb der Sequenz berücksichtigt werden muss, sondern auch die möglichen Sekundärstrukturmuster, die sich daraus ergeben können.
Zuerst wird ein gerichteter Graph konstruiert, in dem jeder Knoten einem stem block, also einer maximalen Clique entspricht. Alle stem blocks werden miteinander verglichen. Die Beziehung zwischen zwei Stems ist im Folgenden definiert. Innerhalb einer Sequenz liegt stem s_1 *vor* stem s_2 , wenn der Helixanfang von s_1 vor dem von s_2 liegt. stem s_1 und stem s_2 sind *kompatibel,* wenn sie in ihren Helix-Regionen nicht überlappen. In bezug auf die relative Start- und Endposition eines Stems, sind nur drei Anordnungen von zwei kompatiblen Stems möglich: nebeneinander liegend (juxtaposed), eingebettet (embedded) oder überlappend, was dem Pseudoknoten entspricht, wie dargestellt in Abb. 11.

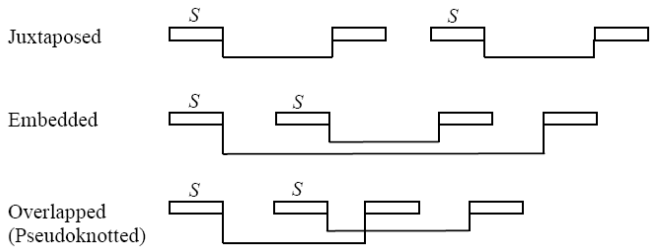


Abb. 11 Die drei möglichen Strukturmuster, die von zwei Stems geformt werden können.

Eine Kante zwischen den stem blocks b_1 und b_2 im gerichteten Graphen ist nur möglich, wenn b_1 *vor* b_2 und *kompatibel* ist, sowie in einer der *drei möglichen Anordnungen* vorliegt. Abb. 12 zeigt dieses Prinzip. Diese Kriterien müssen zudem in mindestens einer *kritischen Anzahl c* von Sequenzen erfüllt sein. *c* ist k oder Hälfte der Anzahl an Sequenzen, die Stems aus einem der blocks besitzen, abhängig davon, ob k oder der andere Wert größer ist.

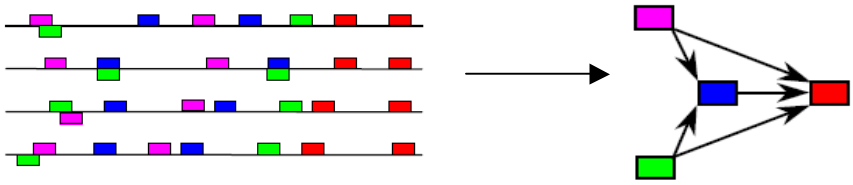


Abb. 12 Die stem blocks, erhalten aus den maximalen Cliquen, sind links in der Primärsequenz dargestellt. Daraus wird der gerichtete Graph von stem blocks konstruiert. Stem blocks b_1 und b_2 können nur dann im Graphen verbunden werden, wenn b_1 vor b_2 ist und kompatibel, sowie eines der drei möglichen Strukturmuster formt.

Eine rekursive Depht-First ähnliche Suche wird auf den entstandenen gerichteten Graphen angewendet, um die maximalen Pfade an stem blocks zu finden, die in mindestens k Sequenzen vorliegen (Abb. 13). Um die stem blocks vom 5`Ende zum 3`Ende in die Sequenz einzufügen, und um dabei sicherzustellen, dass die blocks eines der möglichen Strukturmuster formen in mindestens k Sequenzen, kann ein neuer stem block nur dann in den Pfad eingefügt werden, wenn er mit allen bereits

existierenden stem blocks im Pfad verbunden ist. Somit ist dieser Algorithmus ähnlich dem oben vorgestellten *Finden maximaler Cliquen* – erster stem block wird inkrementell erweitert um einen neuen, stromabwärts liegenden stem block, der mit allen anderen blocks verbunden sein muss. Jeder maximale Pfad bekommt einen Score, der sich aus der Summer aller Scores der stem blocks im Pfad zusammensetzt. Die gefundenen maximalen Pfade repräsentieren mögliche Konsensusstrukturen. Diese potentiellen Konsensusstrukturen werden entsprechend ihres Scores gelistet und die n-besten (Defaultwert n = 10) gelten als Kandidaten für Konsensusstrukturen der RNA-Sequenz.
Dieser Algorithmus hat eine worst case Laufzeit von $O(m^m)$, wobei *m* die Anzahl an stem blocks im gerichteten Graphen ist. Allerdings ergibt sich durch die Einführung der Parameter c und k, dass die Dichte des Graphen meist gering ist, und somit die average case Laufzeit weit unter der des worst case liegt.
Ein schnellerer Ansatz um stem blocks in einem Pfad zusammenzufügen, bietet ein Greedy-Algorithmus. Dieser startet mit dem stem block, der den höchsten Score aufweist und fügt rekursiv den stem block mit dem zweithöchstem Score ein, solange das kompatibel bleibt, mit den bereits eingefügten blocks der Struktur. Dieses Vorgehen wird fortgesetzt bis keine kompatiblen blocks mehr hinzugefügt werden können. Solch ein Greedy-Algorithmus weist eine Zeitkomplexität von O(m) auf, wobei *m* wieder die Gesamtzahl von stem blocks im Graphen ist. Dieser Ansatz ist zwar schnell, aber die Ausgabe besteht aus nur einer Konsensusstruktur und Optimalität wird nicht garantiert. Angenommen ein falscher stem block wurde ausgewählt, so sind mit großer Wahrscheinlichkeit auch die anschließend eingefügten stem blocks falsch. Die Verwendung dieses Greedy-Algorithmus ist somit kritisch, dennoch wurde er im Programm comRNA, das diesen Graph-theoretischen Ansatz zur RNA-Sekundärstruktur-Vorhersage realisiert, implementiert, so dass die Wahl dem Benutzer überlassen bleibt.

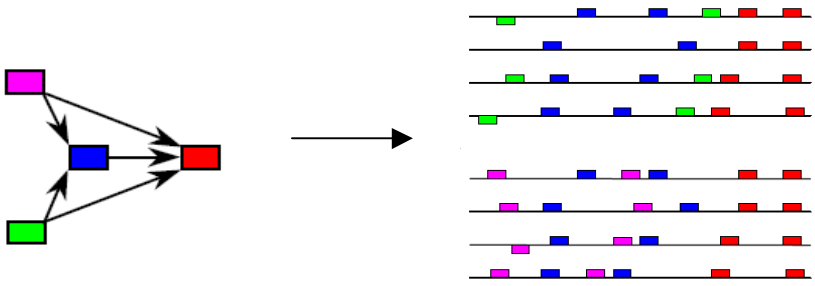


Abb. 13 Aus dem gerichteten Graphen von stem blocks (links) lassen sich durch das Finden maximaler Pfade an stem blocks, mögliche Konsensusstrukturen ableiten. In diesem konkreten Beispiel konnten zwei mögliche Konsensusstrukturen gefunden werden.

Jede dieser Strukturen wird schließlich noch verfeinert, denn aufgrund der Cutoff-Constraints und der angewandten heuristischen Methoden während des gesamten Algorithmus, weisen die so erhaltenen Konsensusstrukturen wahrscheinlich Unvollständigkeiten auf. Um das in jeder möglichen Konsensusstruktur zu beheben, definiert man deren Strukturmuster, und mithilfe des Programms RNAMOT (Laferriere *et al*., 1994) werden die möglicherweise ausgelassenen Stems gesucht und in die Struktur eingefügt. Anschließend erfolgt noch ein Refolding jedes Stems in jeder Sequenz innerhalb von zehn Nukleotiden, ausgehend von den Enden des Stems in beide Richtungen, solange kein anderer Stem erreicht wird. Dadurch werden nun *internal loops*, mit einer Größe von bis zu vier Nukleotiden und *bulges*

der Größe zwei möglich. Erreicht wird dieses Refolding durch einen Algorithmus, basierend auf dynamischem Programmieren, der einen Stem mit minimaler Freier Energie findet, entsprechend Turner´s Energieparametern.

4 Iterated Loop Matching Algorithmus

Der Iterated Loop Matching Algorithmus ist ein dynamic-programming-Algorithmus, der RNA-Sekundärstrukturen inklusive Pseudoknoten vorhersagen kann. Dieser Algorithmus nutzt thermodynamische und vergleichende Information aus und kann jeden Typ von Pseudoknoten vorhersagen, in alignierten *und* einzelnen Sequenzen. Der Iterated Loop Matching Algorithmus basiert auf dem „Loop-Matching-Algorithmus“ nach Nussinov *et al.* (1978), der noch keine Pseudoknoten in die Struktur-Vorhersage mit einbezog.
Eine Sekundärstruktur ist eine Liste von Basenpaaren. Die Basenpaare (i,j) und (k,l) sind *kompatibel,* wenn sie *juxtaposed* (i<j<k<l), also nebeneinander liegend, oder *nested* (i<k<l<j), also eingebettet, sind. Alle anderen Varianten sind *inkompatibel,* so dass i<k<j<l gilt. Eine *inkompatible* Struktur ist eine Pseudoknoten (siehe Abb. 14: C und D).

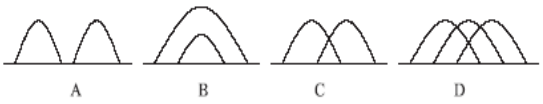


Abb. 14 Hier dargestellt sind die verschiedenen Typen von Beziehungen zwischen Basenpaaren.
(A) 2 Basenpaare nebeneinander liegend
(B) 2 Basenpaare eingebettet
(C) 2 Basenpaare kreuzen sich → Pseudoknoten
(D) 3 Basenpaare kreuzen sich → Pseudoknoten

4.1 Loop Matching Algorithmus nach Nussinov

Der LM-Algorithmus nach Nussinov findet die best-score Sekundärstruktur *ohne* Pseudoknoten. Das bedeutet, dass die Sekundärstruktur hier *kompatibel* sein muss, und somit kann man sie in kleinere Strukturen unterteilen.
Für jede Teilsequenz S[i,j] mit i+1<j gibt es nur drei Möglichkeiten für die Sekundärstruktur: 1) i ist single-stranded 2) i und j sind gepaart 3) i und k sind gepaart, wobei i<k<j gilt.
Somit lässt sich der Score einer optimalen Teilsequenz wie folgt in der NxN-Matrix *Z* rekursiv berechnen:

$$Z(i,j) = \max \left\{ \begin{array}{l} Z(i+1,j); \\ Z(i+1,j-1) + B(i,j); \\ \max_k \{Z(i+1,k-1) + Z(k+1,j) \\ \quad +B(i,k)\}, \quad \forall k,\ i < k < j. \end{array} \right\}$$

Zuvor wurde die Basepair-Score Matrix B berechnet, wobei B(i,j) den Score für die Basenpaarung zwischen der Base i und Base j angibt. Auf die Berechnung dieser Matrix B wird in Abschnitt 4.3 ausführlich eingegangen.

Am Ende des LM-Algorithmus ist Z(1,N) der Score der optimalen Struktur für die Sequenz S[1..N], den man durch traceback in Z erhält. Diese Berechnung und der Traceback ist in Laufzeit $O(n^3)$ möglich.
Im einfachsten Fall ist B(i,j) = 1 wenn Base i mit Base j ein Watson-Crick oder G-U Basenpaar bildet, sonst 0. In diesem Fall findet der LM-Algorithmus die Sekundärstruktur mit der maximalen Anzahl an Basenpaaren.

4.2 Iterated Loop Matching Algorithmus

Der Iterated Loop Matching Algorithmus (ILM) ist eine Erweiterung des LM-Algorithmus nach Nussinov, der auch Pseudoknoten in die Vorhersage der Sekundärstruktur einbezieht. Da ein Pseudoknoten der Interaktion zwischen zwei Loop-Regionen entspricht, ist die Idee des ILM, den klassischen LM–Algorithmus *einfach* zweimal auf die Sequenz anzuwenden. Natürlich sind mehr Iterationen für kompliziertere Pseudoknoten erforderlich. Der LM-Algorithmus läuft zuerst einmal und sagt die Sekundärstruktur wie üblich voraus. Die vorausgesagten Basenpaare werden im Anschluss so behandelt, als wären sie entfernt worden; die nächste Iteration des LM-Algorithmus beginnt. Dieses Vorgehen versagt jedoch oft in der Praxis.
Die Basen, die für Pseudoknoten vorgesehen waren, können im ersten Durchlauf des LM-Algorithmus in falsch-positive Basenpaarungen geraten, wie in Abb. 15 dargestellt. Das lässt sich vermeiden, indem der LM-Algorithmus mehrmals pro Iteration angewendet wird und nur Basenpaare mit höchstem Score akzeptiert werden.

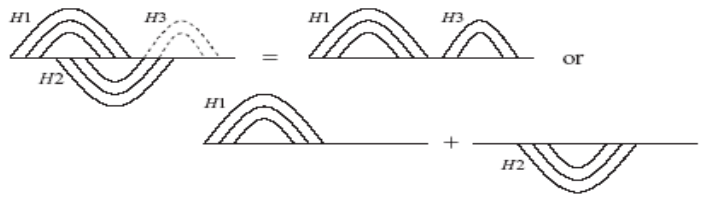


Abb. 15 H1 und H2 formen einen Pseudoknoten. H3 ist eine falsche Helix, die H2 überlappt und somit durch den ILM-Algorithmus der Pseudoknoten eventuell nicht entdeckt werden kann.

Der Iterated Loop Matching Algorithmus lässt sich in sieben grundlegenden Schritten zusammenfassen:
1. Erstellen der Basepair Scorematrix B[1..n][1..n] aus einer Sequenz oder einem Sequenzalignment
2. Aufruf des LM-Algorithmus nach Nussinov; Matrix B wird benutzt um Matrix Z zu erzeugen; Traceback in Z, so dass man die Basepair-Liste L erhält
3. Identifizieren aller Helices in L und Kombinieren der Helices, die durch internal loops oder bulges getrennt sind; existieren keine Helices, geht man direkt zu Schritt 7
4. Jede Helix wird mit einem Score versehen; dieser ergibt sich durch Summieren der Scores von den Basenpaaren in der Helix; *Helix H* mit höchstem Score wird in die Basenpaar-Liste S eingefügt
5. Die Positionen von Helix H werden aus der Initialsequenz genommen; Update von Scorematrix B
6. Schritte 2 - 5 wiederholen bis keine Basen mehr übrig sind
7. Ausgabe ist die Basenpaar-Liste S; Termination

In Schritt 5, dem Update der Score-Matrix B, werden einfach die Zeilen und Spalten entfernt, die zu den Basen korrespondieren, die gepaart wurden in der Sekundärstruktur. Außerdem ist anzumerken, dass nicht alle Elemente der Matrix Z in jeder Iteration neu berechnet werden müssen. Die in Abb. 16 dargestellten drei Dreiecke der Matrix müssen nicht neu berechnet werden: A, B, C und obere und untere Dreiecksmatrix sind ohnehin symmetrisch. Nur Teil D erfordert eine neue Berechnung in jeder Iteration.
Angenommen das Basenpaar (p, q) ist das der letzten Iteration. So muss Z(i, j) nur dann neu berechnet werden, wenn i und j entweder durch p oder q getrennt werden. Liegt aber das Basenpaar (i, j) links oder rechts vom eben vorausgesagten Basenpaar (p, q), oder eingebettet darin, dann ist für (i, j) keine neue Berechnung erforderlich.

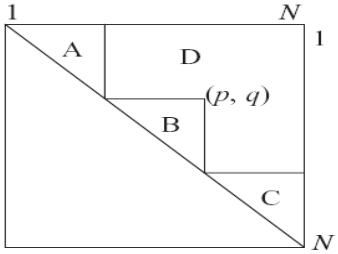


Abb. 16 Die drei dreieckigen Bereiche A, B und C der Matrix erfordern keine neue Berechnung in jeder Iteration. Seien i und j die Zeilen- und Spaltenindices einer Position. (p,q) das Basenpaar aus voriger Iteration.
A: i< j< p B: p< i< j< q C: q< i <j

In einem Array M werden die restlichen ungepaarten Basenpaare gespeichert. M[i] ist die i-te übrige ungepaarte Base, während p und q (mit p<q) die Endpunkte einer Helix aus vorheriger Iteration sind. In der ersten Iteration gilt M[i] = i, zudem sind p und q noch nicht definiert, somit gilt die Rekursion wie im LM-Algorithmus:

$$
\begin{aligned}
&Z'(M[i], M[j]) \\
&= \begin{cases}
Z(M[i], M[j]), \\
\quad \text{if } M[j] < p \ \text{ or } \ M[i] > q \ \text{ or } \ p < M[i] < M[j] < q; \\
0, \quad \text{if } j - i + 1 < VLOOP\_LENGTH; \\
\max \begin{cases}
Z'(M[i+1], M[j]); \\
Z'(M[i+1], M[j-1]) + B(M[i], M[j]); \\
\max_k \{Z'(M[i+1], M[k-1]) + Z'(M[k+1], M[j]) \\
\quad + B(M[i], M[k])\}, \quad \forall k, \ i < k < j.
\end{cases} \\
\quad \text{otherwise.}
\end{cases}
\end{aligned}
$$

Nachdem Teile der Sequenz entfernt wurden, können zwei zuvor getrennte Basen nun nebeneinander liegen. Daher definiert man vorher die virtuelle Distanz zweier Basen, wobei diese die Distanz zwischen den Indices in M darstellt. VLOOP_LENGTH beschreibt die minimale virtuelle Distanz, die zwischen Basenpaaren eingehalten werden muss nach der ersten Iteration. Ihr Default-Wert beträgt 3.

Der gesamte Score einer Struktur kann als Maß für die Wahrscheinlichkeit, unter allen möglichen Strukturen aufzutreten, aufgefasst werden. Deshalb wird der Algorithmus bevorzugt, der die Struktur mit dem höchsten Score berechnet. Der LM-Algorithmus berechnet eine solche Struktur, mit der Bedingung, dass die Basenpaare *kompatibel* zueinander sein müssen. Im ILM-Algorithmus wurde diese Bedingung gelockert, so dass auch *inkompatible* Strukturen – Pseudoknoten – möglich sind. Der

ILM-Algorithmus opfert diese Optimalität des LM-Algorithmus, um lange Helices gegenüber willkürlich gepaarten Basen zu bevorzugen. Obwohl der ILM-Algorithmus keine Optimalität garantiert, so garantiert er doch, dass der Score einer vorausgesagten Struktur nicht niedriger ist, als der Struktur, die der klassische LM-Algorithmus voraussagen würde. Sei S_{ILM} der Score einer Struktur, die durch den ILM-Algorithmus berechnet wurde, und S_{LM} der Score einer Struktur nach Berechnung durch den LM-Algorithmus. Dann gilt $S_{ILM} >= S_{LM}$. Diese Behauptung kann durch Induktion bewiesen werden. Auf den ausführlichen Beweis sei an dieser Stelle nur verwiesen (Ruan, Stormo, Zhang; 2004).
Obgleich diese Methode nicht die theoretisch optimale Struktur berechnet, opfert sie diese Optimalität stabilen Helices.

Die worst case Laufzeit des LM-Algorithmus nach Nussinov braucht $O(n^3)$. Dieser Algorithmus wird *m* mal wiederholt, wobei *m* die Anzahl an Helices ist, die der Algorithmus vorhersagt. Da gilt m <= n/2k, wobei k die minimale Helixlänge ist, kann die worst case Laufzeit im ILM-Algorithmus maximal $O(n^4)$ sein. *m* ist aber typischerweise klein und die Matrix Z muss nur teilweise neu berechnet werden in jeder Iteration, daher erreicht man im average case doch $O(n^3)$.

4.3 Aufbau der Score-Matrix B

Die Score-Matrix wird in dieser Implementierung des ILM-Algorithmus als Summe aus *mutual Information* und *Helix Plot Score* berechnet, was im wesentlichen eine Kombination aus Covarianz und Thermodynamik-Score ist.
Die *mutual information* ist ähnlich der relativen Entropie (Kullback-Leibler-Distanz), und gibt an, wie viel Information eine zufällige Variable, in diesem Fall eine Base, über eine andere enthält. Es handelt sich hier um die relative Entropie zwischen der gemeinsamen Verteilung $f_{ij}(XY)$ und dem Produkt der einzelnen Verteilungen. Für die Berechnung der *mutual Information* muss ein multiples Sequenz-Alignment von N Sequenzen vorliegen. Sei $f_i(X)$ die Häufigkeit der Base X an der alignierten Position i. $f_{ij}(XY)$ gibt an, wie oft man die Base X an der alignierten Position i findet, und Y an Position j. Der mutual-information-Score zwischen Position i und j, also M_{ij} wird wie folgt berechnet:

$$M_{ij} = \sum_{X,Y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X) f_j(Y)}$$

Der *Helix Plot Score* wird wie folgt berechnet*:* Helix Plots sind Mittelwerte von Basepair-Scores – sie kombinieren phylogenetische und thermodynamische Information.
Für *jede* Sequenz im multiplen Alignment wird eine Score-Matrix erstellt, indem Watson-Crick- oder Wobble-Basenpaare good-pair Scores bekommen (=1) und andere Basenpaare bad-pair Scores (=2). Zudem gibt es einen Penalty Score für gaps (=3). Für lange Helices gibt es Bonus Scores, aber auch für zu kurze Helices bad-pair Scores. Nachdem alle einzelnen Score-Matrizen berechnet wurden, addiert man sie zu *einer* gesamten Score-Matrix auf.
Mutual Information und *Helix Plot Scores* werden addiert, um die Score-Matrix B zu generieren, die im ILM schließlich benutzt wird:

$$B_{ij} = \alpha \times 1000 \times M_{ij} + \beta \times 20 \times HP_{ij}/N$$

α und β sind relative Gewichtungen für *mutual Information* und *Helix Plot Scores.* Der *Helix Plot Score* bekommt dann ein höheres Gewicht, wenn die Anzahl an Sequenzen klein ist und umgekehrt, weil der *mutual Information Score* nur bei vielen Sequenzen sinnvoll ist.
HP_{ij} ist der Helix Plot Score eines potentiellen Basenpaares; N ist Anzahl der Sequenzen im Alignment. Die Koeffizienten 1000 und 20 dienen nur der Konvertierung von *mutual information* und *Helix Plot Score* in Integer gleicher Wertebereiche.

Wenn nur eine einzige Sequenz statt eines Alignments vorliegt, und somit keine Covarianz-Information gegeben ist, wird der *Extended Helix Plot Score* verwendet in der Score-Matrix B. Denn in diesem Fall kann es keine *mutual Information,* also wechselseitige Beziehung zwischen Sequenzen geben und der Helix Plot Score allein ist nicht ausreichend. Daher wird beim Vorliegen einer einzigen Sequenz noch die Faltungs-Thermodynamik der RNA miteinbezogen.
Somit hängt der good-pair Score GP_{ij} hier auch vom Typ des Basenpaares ab, was bedeutet, dass G-C einen Score von 80, A-U Score 50 und G-U Score 30 erhält. Zudem ist ein Helix-Bonus $BONUS_{ij}$ definiert, der proportional zur Stacking-Energie der Helix ist. Der *Extended Helix Plot Score* EXT_HP wird schließlich so berechnet:

$$EXT\_HP = GP_{ij} + BONUS_{ij}$$

wobei der Bonus-Score wie folgt definiert ist:

$$BONUS_{ij} = 100 \times \frac{Total\ Stacking\ Energy}{\sqrt{Helix\ Length}}$$

Die Vorhersage der Sekundärstruktur für ein Alignment von Sequenzen, sowie für einzelne Sequenzen ist durch den Iterated Loop Matching Algorithmus möglich, und zudem für alle Typen von Pseudoknoten. Der Algorithmus ändert sich dabei nicht, ob nun die Sekundärstruktur für ein Alignment von Sequenzen oder für einzelne Sequenzen vorausgesagt werden soll. Lediglich die Berechnung der Score-Matrix B verändert sich. Im Falle der Strukturvorhersage für mehrere Sequenzen verwendet man die Summe aus *mutual Information* und *Helix Plot Score*, während man bei einer einzelnen Sequenz den *Extended Helix Plot Score* EXT_HP verwendet.

5 Fazit

Die zuerst vorgestellte Vorhersage-Methode basiert auf stochastischem Modellieren der RNA-Sekundärstrukturen durch Grammatiken nach Chomsky – den Parallel Communicating Grammar Systems. Diese Methode verwendet *eine* kontextfreie Grammatik synchronisiert mit einer Anzahl an regulären Grammatiken, so werden kontextsensitive Regeln vermieden. Die stochastische Version des Modells ist wegen der Verwendung von nur einer kontextfreien Grammatik, so einfach wie bei gewöhnlichen SCFG. Die Vorhersage der Sekundärstruktur ist nur für eine RNA-Sequenz möglich, aber für alle Typen an Pseudoknoten. Die Laufzeit im worst case beträgt $O(n^6)$.

Der hier vorgestellte Graph-theoretische Ansatz für die Vorhersage häufiger RNA Sekundärstrukturmuster – den Konsensusstrukturen - in einer Menge von verwandten Sequenzen, basiert auf der Suchen und dem Zusammenfügen konservierter Stems.
Stems dienen diesem Algorithmus als Vergleichseinheit. Gute Ergebnisse erzielt dieser Ansatz für bis zu 20 RNA-Sequenzen mit einer Länge kleiner 300 nt.
Der Algorithmus kann große Sequenzähnlichkeit ausnutzen, indem im Alignment Anker-Regionen definiert werden, wodurch der Suchraum für Stems kleiner wird und somit auch die Laufzeit geringer.
Ein weiterer Vorteil besteht darin, dass diese Methode eine Menge an möglichen Konsensus-Strukturen findet. Denn die beste Struktur bezogen auf den Score muss nicht unbedingt der realen Struktur entsprechen. Zudem ist es ist hilfreich um mögliche alternative Sekundärstrukturen zu entdecken. Als Nachteil ist die worst case Laufzeit für den maximale Cliquen- bzw. Path-Algorithmus anzumerken, die in NP liegt.

Der Iterated-Loop-Matching Algorithmus, wie in Abschnitt 4 beschrieben, ist eine Erweiterung des LM-Algorithmus nach Nussinov et al. (1978). Er basiert auf folgendem Prinzip: iteratives Vorhersagen einer nicht-pseudogeknoteten Struktur wie es im klassische LM-Algorithmus realisiert ist, daraus Auswählen der wahrscheinlichsten Helix, diese wird dann aus der Sequenz entfernt. Der LM-Algorithmus wird wieder auf diese verkürzte Sequenz angewendet bis keine Basen mehr vorhanden sind oder keine Helices mehr gefunden werden können.
Die Vorhersage der RNA-Sekundärstruktur ist für ein Alignment von Sequenzen ebenso möglich, wie für einzelne Sequenzen, und das gilt für alle Typen von Pseudoknoten. Der Algorithmus ändert sich nicht, ob nun die Sekundärstruktur für ein Sequenzalignment oder für einzelne Sequenzen vorausgesagt werden soll. Der Unterschied liegt lediglich in der Berechnung der Score-Matrix B. Die worst case Laufzeit für den Iterated-Loop-Matching Algorithmus liegt in $O(n^4)$.

6 Literaturverzeichnis

[1] Stochastic modeling of RNA pseudoknotted structures: a grammatical approach; Cai, Malmberg, Wu; 2003

[2] A graph theoretical approach to predict common RNA secondary sructure motifs including pseudoknots in unaligned sequences; Yongmei, Stormo, Xing; 2004

[3] An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots; Ruan, Stormo, Zhang; 2004

[4] The language of RNA: A formal grammar that includes pseudoknots; Rivas, Eddy; 2000

[5] An RNA folding method capable of identifying pseudoknots and base triples; Tabaska, Cary, Gabow, Stormo; 1998

[6] Graph-theoretic approach to RNA modeling using comparative data; Cary, Stormo; 1995

[7] Biological Sequence Analysis; Durbin, Eddy, Krogh, Mitchison; 1998

[8] Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie; Hopcroft, Ullman; 2000