comparative structure prediction of ncRNA molecules

Sebastian Bartschat

01. Februar 2008

Inhaltsverzeichnis

1 ncRNA's - ein Überblick

ncRNA sind kleine RNA-Moleküle, die von der DNA transkribiert wurden, aber anschließend nicht in ein Protein übersetzt werden, d.h. sie erfüllen ihre zellulären Aufgaben in RNA-form. Non coding RNA Moleküle kommen in den verschiedensten Organismen vor und erlebten in der Evolution eine weitgehende Konservierung, sodass sie heute vielfältigste enzymatische und regulatorische Aufgaben im zellulären Stoffwechsel übernehmen:

miRNA *micro RNA* sind einzelsträngige RNA Molekül von ca 21-23 Nukleotiden Länge. Sie erkennen eine bestimte Ziel-mRNA, d.h. sie sind komplementär zu einer Teilsequenz, und regulieren deren Genexpression.

snoRNA *small nucleolar* RNAs sind Moleküle, die chemische Modifikationen an rRNA's (z.B. Methylierung oder Pseudouridylierung) katalysieren.

siRNA *small interfering* RNAs sind ca. 20 - 25 bp lange doppelsträngige RNA Moleküle, die vorallem im RNA interference pathway eine tragende Rolle spielen. Dabei sind sie Teil eines Proteinkomplexes (RISC), der z.B. virale RNA oder bestimmte mRNA erkennt. Anschließend werden diese Targets zerschnitten und somit unbrauchbar gemacht.

Diese Liste ist nur ein Auszug aus den vielfältigsten Funktionen, die ncRNAs übernehmen. Aber viele der Mechanismen sind noch nicht erforscht, da das Gebiet der ncRNAs ein sehr neues ist. So fällt es z.B. auch viel schwerer ncRNAs im Genom zu identifizieren als proteinkodierende Gene. Dies ist dadurch begründet, dass man für proteinkodierende Gene die Positionen im Genom durch Promoter, Enhancer, Silencer und andere bekannte Bindungsmotive sehr gut identifizieren kann. All diese bestimmten Sequenzen sind für ncRNA Moleküle noch nicht bekannt, vielleicht gibt es sie auch gar nicht in dieser Form.

Es gibt aber auch bestimmte Charakteristika für ncRNAs. Man kann z.B. davon ausgehen, dass die vielen Funktionen der ncRNAs durch die Sekundärstruktur begünstigt bzw. erst ermöglicht werden. In diesem Punkt unterscheiden sie sich nicht von proteinbasierenden Katalysatoren. Daher ist es auch von großem Interesse, die Sekundärstruktur der einzelnen RNA Moleküle zu kennen, um somit ihre Funktionen und die ganze Komplexität zu verstehen.

2 Einführung in die Strukturvorhersage

Die Vorhersage von Sekundärstrukturen ist ein breites Feld , bei dem es viele mögliche Ansätze gibt. Einige werden nun näher beschrieben.

2.1 Minimal Free Energy (MFE)

Der Basis-Algorithmus wurde von Nussinov vorgeschlagen und er versuchte einfach diejenige Struktur zu finden, welche die maximale Anzahl von Basenpaaren enthielt. Heutige Implementationen nutzen aber für die Berechnung der Minimal Freien Energie viele Faktoren, wie zum Beispiel *stem energy* (Länge des Stammes und die Art der Wasserstoffbrücken zwischen den Basenpaaren), *Temperatur, Größe von Loops* usw. Dies macht die Berechnung natürlich deutlich schwieriger. Trotz dieser vielen Berücksichtigungen bei der Berechnung der Struktur ist die gefundene, sprich die mit der geringsten freien Energie häufig nicht die korrekte. Programme sind: mfold, RNAfold; RNAsubopt liefert als Output eine Liste von Strukturen, die nach ihrem MFE aufsteigend sortiert werden.

2.2 comparative prediction

Die eben beschriebenen Schwierigkeiten und auch Ungenauigkeiten führten zu der Idee, dass eine bessere Genauigkeit dadurch erzielt werden kann, indem man mehrere ähnliche Sequenzen analysiert. Es ist bekannt, dass die Sekundärstrukturen in den einzelnen ncRNA-Familien stärker konserviert sind als deren Sequenz. Ein weiterer Vorteil besteht in der Möglichkeit eines solchen Vorgehens neue ncRNA Moleküle zu finden, da sich vielleicht neue Moleküle von dem anfänglichen Set an RNAs strukturell doch deutlich unterscheiden. Es gibt einige Programme, die eine einheitlich Struktur bestimmen wenn man ihnen als Input ein Sequenzaligment liefert:

RNAalifold. Andere Programme, z.B. RNAforester, berechnen globale Strukturalignments, wobei man dafür aber die Struktur der Sequenzen kennen muss.

2.3 Sankoff-Ansatz

Sankoff entwickelte als Erster einen *dynamic programming algorithm* für das zeitgleiche alignieren und falten von mehreren RNA Sequenzen. Dabei beachtet der ursprüngliche Algorithmus alle möglichen Faltungen und alle möglichen Alignments für die gegebenen Sequenzen. Dadurch ist er relativ langsam und speicheraufwändig. Für 2 Sequenzen bewegt man sich in $\mathcal{O}(n^6)$ für die Laufzeit und in $\mathcal{O}(n^4)$ für den Speicheraufwand, wobei n die Sequenzlänge beschreibt. Für mehr Sequenzen steigt der Aufwand exponentiell mit der Anzahl der Sequenzen. Es gibt eine Fülle von Programmen die diesen Ansatz verfolgen und dabei versuchen die Ressourcen zu schonen: Stemloc arbeitet dafür mit einer *stochastic context free grammar* und besitzt Vorläufe um jede Sequenz einzeln zu falten bzw. paarweise zu alignieren. Andere Vertreter sind Pmcomp, Dynalign (allerdings nur für 2 Sequenzen)

3 RNAcast - RNA consensus abstract shapes technique

RNAcast beschreibt einen Ansatz, der lediglich versucht die optimale Sekundärstrukturen für eine Inputmenge von Sequenzen zu finden, wobei die Sequenzen nicht aligniert werden. Vielmehr wird einfach versucht eine *consensus structure* zwischen den einzelnen Sequenzen, bzw. deren Strukturen zu finden, die für jede einzelne Sequenz eine suboptimale, aber für die Gesamtheit die optimale Lösung darstellt.

shape: Familie von Sekundärstrukturen die einen gemeinsamen Grundaufbau besitzen, d.h. ähnliches Muster und ähnliche Anordnungen von Helices.

3.1 RNA shape analysis

RNAcast arbeitet mit dem Programm RNAshapes, welches zu einer gegebenen Sequenz verschiedene Faltungen und *shapes* berechnet. Im Folgenden werden die Grundlagen zur Analyse von Faltungen und *shapes* erläutert.

Faltungsraum: jede Sequenz s besitzt einen Faltungsraum $\mathcal{F}(s)$, der alle akzeptablen suboptimalen Strukturen enthält. Diese Faltungen werden durch das RNAshape Programm ermittelt. Zu jeder Faltung $x \in \mathcal{F}(s)$ kann die freie Energie $E(x)$ berechnet werden.

minimal free energy: $mfe(s)$ ist diejenige Struktur $x \in \mathcal{F}(s)$, für die $E(x)$ minimal ist

Darstellung: $\mathcal{S}$ ist eine string-like domain von Strukturen; $\mathcal{P}$ eine string-like domain für *shapes*. Dies bedeutet, dass sowohl Strukturen als auch *shapes* in String Notation geschrieben und verarbeitet werden.

Mapping: π ist eine Mappingfunktion von $\mathcal{S}$ nach $\mathcal{P}$.

$\pi_5(.) = \varepsilon$	$\varrho_5(.) = \varepsilon$
$\pi_5(.s) = \pi_5(s)$	$\varrho_5(.s) = \varrho_5(s)$
$\pi_5(s.) = \pi_5(s)$	$\varrho_5(s.) = \varrho_5(s)$
$\pi_5((s)) = [\varrho_5(s)]$	$\varrho_5((s)) = \varrho_5(s)$
$\pi_5((s)s') = [\varrho_5(s)]\pi_5(s')$	$\varrho_5((s)s') = \pi_5((s)s')$
$\pi_3(.) = \varepsilon$	$\varrho_3((s)) = \varrho_3(s)$
$\pi_3(.s) = \pi_3(s)$	$\varrho_3(s) = \pi_3(s)$ In all other cases
$\pi_3(s.) = \pi_3(s)$	
$\pi_3((s)) = [\varrho_3(s)]$	
$\pi_3((s)s') = [\varrho_3(s)]\pi_3(s')$	

Tabelle 1: Definitionen der Mappingfunktionen für die Abstraktionsniveaus 3 und 5

shapes: Die Menge aller shapes zu einer Sequenz s wird definiert durch:

$$\mathcal{P}(s) \;=\; \{\, \pi(x) \,|\, x \in \mathcal{F}(s) \,\}$$

shape-cluster: Eine Klasse aller *p-shaped*-Strukturen ist beschrieben durch:

$$\mathcal{F}(s \,|\, p) \;=\; \{\, x \,|\, x \in \mathcal{F}(s)\,,\; \pi(x) \;=\; p \,\}$$

shrep: Ein *shrep* $\hat{p}$ ist diejenige Struktur eines *shapes* p mit der kleinsten freien Energie

Das Programm RNAshapes arbeitet nun so, dass es für eine Input-Sequenz s und einen vorher definierten Energiebereich R alle *shapes* und deren jeweilige *shreps* in einer geordneten Liste berechnet. Der Output sieht folgendermaßen aus: $[\,(\,p_1\,,\,\hat{p}_1\,)\,,\,\ldots\,,\,(\,p_k\,,\,\hat{p}_k\,)\,]$. Da die Liste nach minimaler freier Energie geordnet ist, befindet sich an erster Stelle der *shape* p_1, der die Struktur mit dem kleinsten Energieniveau enthält, also den $mfe(s)$.

Die Verwendung von *shapes* hat verschiedene Vorteile. Zum Einen ist der Ansatz nicht heuristisch sondern mathematisch wohl definiert und gibt einen guten Überblick über den Faltungsraum (innerhalb einer Energieschranke R), wodurch eine Reduzierung der zu betrachtenden Strukuren auf alle *shreps* ermöglicht wird. Zum Anderen ist dies auch ein generischer Ansatz, da es durch die Mappingfunktion mehrere Möglichkeiten gibt eine Struktur auf ein *shape* abzubilden, je nach gewünschtem Abstraktionsniveau.

Tabelle 1 beschreibt die Mappingfunktion, die einen Struktur-String (in dot-bracket Schreibweise) in einen String überführt, der einen *shape* charakterisiert. Abgebildet sind allerdings nur Level 3 und Level 5, wobei die Abstraktionsniveaus von 1 bis 5 reichen. Mit steigendem Niveau schwindet der Einfluss den Loops, Bulges oder Helix-Unterbrechungen auf den *shape* haben. D.h. Level 5 verzichtet ganz auf diese Einflüsse.
So wird z.B. die Struktur `'(((...((..(((...)))))).((...))..)))'` im höchsten Abstraktionsniveau auf den *shape* `'[[][]]'` gemappt, wohingegen der *shape* des 3.Levels so aussieht: `'[[[]][]]'`.

3.2 consensus shape prediction

Nun kann man anfangen einen vergleichenden Algorithmus zu entwickeln, der mittels *shape prediction* versucht die natürlichen Strukturen der Eingabesequenzen zu berechnen. Bevor der

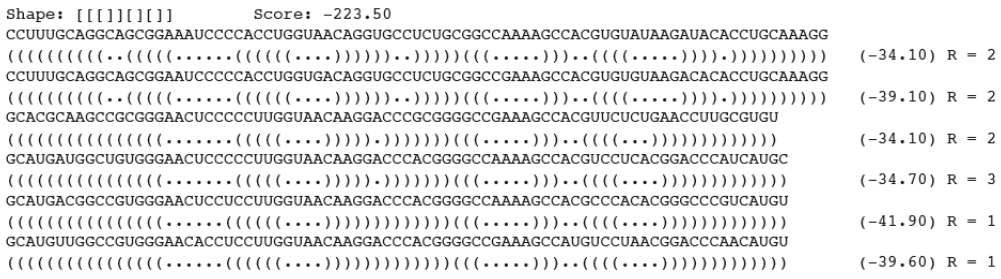

Abbildung 1: Output von RNAcast

Algorithmus beschrieben wird, müssen allerdings noch 2 Definitionen eingeführt werden. Für ein Set von Sequenzen $\{s_1, \ldots, s_k\}$ und deren shape spaces $\mathcal{P}(s_1)$, ... , $\mathcal{P}(s_k)$ gilt :

`DEFINITION 1:` p ist *common shape* $\Leftrightarrow$ $p \in \bigcap_{i=1}^{k} \mathcal{P}(s_i)$

`DEFINITION 2:` *consensus shape* für Sequenzen $\{s_1 \ldots s_k\}$ ist derjenige *common shape* p mit minimalem rank $(\hat{p}_1 \ldots \hat{p}_k)$. Wobei der rank über eine Scorefunktion berechnet wird, welche im 3. Schritt des Algorithmus näher erläutert wird.

Der folgende Algorithmus gliedert sich in 3 Teile:

step1 Der Input wird durch eine Menge von Sequenzen $\{s_1, \ldots, s_k\}$ und einer Energieschranke R beschrieben. Zu jeder einzelnen Sequenz wird das Tool RNAshapes aufgerufen, welches die zugehörigen *shapes* und *shreps* berechnet.

step2 Aus allen resultierenden *shapes* werden nun alle *common shapes* bestimmt, sodass eine Liste mit der Form $[\,(p_1, [\,\hat{p}_1^1, \ldots, \hat{p}_k^1\,])\,, \ldots\,, (p_l, [\,\hat{p}_1^l, \ldots, \hat{p}_k^l\,])\,]$ ensteht. Dabei enthält jeder *shape* eine Liste mit seinen zugehörigen *shreps*.

step3 Auf allen berechneten *common shapes* wird nun die Scoringfunktion angwandt und es entsteht somit eine geordnete Liste, sodass der *consensus shape* an erster Stelle steht. Für die Scorefunktion gibt es verschiedene Ansätze. So ist z.B. denkbar den rank eines *shape* als Summe aus den einzelnen ranks seiner *shreps* zu berechnen . Am Besten erwies sich jedoch die folgende Funktion: $\text{rank}(\,p_i, [\,\hat{p}_1^i, \ldots, \hat{p}_k^i\,]) = E(\hat{p}_1^i) + \ldots + E(\hat{p}_k^i)$, d.h. derjenige *shape* dessen *shreps* die geringste freie Energie besitzen, wird der *consensus shape*. Als Output erhält man neben dem *consensus shape* auch für jede Sequenz den jeweiligen *shrep*.

Abbildung 1 zeigt den möglichen Output, wobei man sieht, dass RNAcast keine Alignments berechnet. Jetzt kann man z.B. diese Ausgaben in das Tool RNAforester geben, welches multiple Strukturalignments berechnet. Die Ausgabe für das Beispiel aus Abbildung 1 sieht man in Abbildung 2.

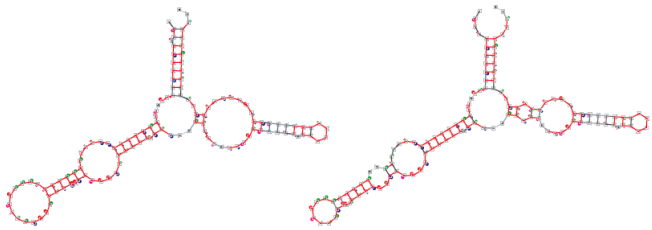

Abbildung 2: Output von RNAforester

4 RNAspa - a shortest path approach

Das Programm RNAspa nutzt einen anderen Ansatz und sucht mittels eines Graphen zwischen allen suboptimalen Strukturen die Ähnlichsten. Zur Vorbereitung wird das Tool RNAsubopt genutzt, welches zu einer gegebenen Sequenz eine benutzerdefinierte Menge von suboptimalen Faltungen aufsteigend nach minimaler freier Energie zurück gibt.

4.1 finding the shortest path

Der Algorithmus gliedert sich in drei Schritte, wobei die ersten beiden zum Aufbau des Graphen genutzt werden und der dritte zum bestimmen des kürzesten Pfades dient. Dieser Ablauf wird anschließend mehrfach wiederholt, wie nachfolgend näher beschrieben wird.

step1 Der Input besteht wie bei RNAcast aus einer Menge von unalignierten Sequenzen $\{s_1, s_2, \ldots, s_n\}$. Im ersten Schritt wird zu jeder Sequenz s_i, aus der Inputmenge, RNAsubopt genutzt um eine Menge von V suboptimalen Faltungen zu berechnen: $\{s_i^1, s_i^2, \ldots, s_i^v\}$.In dieser Menge befindet sich nun die optimale Struktur sowie $v-1$ suboptimale. (Versuche haben ergeben dass zu jeder Sequenz mit sehr großer Wahrscheinlichkeit die optimale Struktur innerhalb der ersten 150 Faltungen von RNAsubopt liegt.)

step2 Im Folgenden wird nun aus den einzelnen Faltungen ein Graph konstruiert. Dabei beschreibt jede Faltung s_i^j einer Sequenz s_i einen Knoten im Graphen. Der Graph besteht aus n Schichten, d.h. jede Eingabesequenz s_i beschreibt eine solche Schicht. Eine Schicht besteht wiederum aus v Knoten, nämlich genau denjenigen Knoten, die RNAsubopt zu der Sequenz s_i als mögliche Faltungen errechnet.

Die Schichten werden nun übereinander gestapelt, sodass der Layer für die Sequenz s_1 an oberster Stelle liegt. Darunter folgen die Schichten der Sequenzen s_2, s_3 usw. . Des Weiteren ist jeder Knoten der Schicht s_i mit einer gerichteten und gewichteten Kante mit allen Knoten der Schicht s_{i+1} verbunden. Die Gewichte berechnen sich aus den Ähnlichkeiten der beiden Strukturen, die sich hinter dem jeweiligen Knoten verbergen, sodass das Gewicht die Editierdistanz (ED) beider Faltungen wiederspiegelt. Der Editierabstand wird durch den Needleman-Wunsch Algorithmus für globale Alignments berechnet.

step3 Im dritten Schritt wird der kürzeste Pfad im Graph von der ersten bis zur n-ten Schicht gesucht, sodass jede Schicht genau einmal besucht wird. Dies geschieht mit einem

modifzierten Ansatz zur Breitensuche und läuft in linearer Zeit gemessen zur Anzahl der verschiedenen Sequenzen. Die Kosten eines Pfades ergeben sich dabei aus der Summe der Kosten der enthaltenen Kanten. Je kürzer so eine Pfad ist, desto ähnlicher sind sich die enthaltenen Strukturen, denn ähnliche Faltungen ergeben auch die kleinsten scores. Im Endeffekt enthält der Pfad für jede Sequenz eine Faltung und beschreibt den besten Kompromiss zwischen den verschiedenen Strukturen.

Im Folgenden werden die Schritte 2 und 3 mehrmals wiederholt, allerdings immer mit einer anderen zufälligen Reihenfolge der eingegebenen Sequenzen. Dies ist notwendig, weil der Vergleich auf Ähnlichkeit zwischen Strukturen lediglich zwischen den Knoten der Schicht s_i und s_{i+1} stattfindet. Daher ist es möglich, dass die Input-Reihenfolge die Strukturvorhersage in eine negative Richtung beeinflusst. Natürlich ist es nicht möglich alle $n!$ Anordnungen durchlaufen zu lassen, da sonst die Laufzeit exponentiell zur Anzahl der Samples steigt. In der Studie wurden mit einem Samplespace von 4 relativ gute Ergebnisse erzielt. Dieser Ansatz hat zur Folge dass RNAspa in den seltensten Fällen die optimale Struktur vorhersagt, da man dafür alle Permutationen berechnen müsste. Die Reduzierung der Vergleiche, von einer Sequenz s_i mit allen anderen Sequenzen auf den Vergleich von Nachbarsequenzen, ermöglicht im Endeffekt die fast lineare Skalierung des Algorithmus. Des Weiteren ist die Güte der Strukturen nur minimal schlechter als die optimale Struktur.

Die aus der Iteration entstandenen kürzesten Pfade werden anschließend verglichen und der mit dem kleinsten sum-of-pair score ist derjenige, der den "echten" Strukturen am ähnlichsten ist. Der sum-of-pair score berechnet sich aus den Gewichten aller $\frac{n(n+1)}{2}$ Knotenpaare, die in einem Pfad enthalten sind. In der Ausgabe sind alle Faltungen des "gesamtkürzesten" Pfades enthalten, sodass RNAspa für n Inputsequenzen die ähnlichsten n Faltungen berechnet. Der gesamte Ablauf des Algorithmus ist in Abbildung 3 noch einmal erklärt.

4.2 Laufzeit und Verbesserungen

Die Laufzeit des vorgestellten Algorithmus liegt in:

$$\mathcal{O}(NL^3 + SNV^2L^2 + SN^2L^2)$$

N ... Anzahl von Sequenzen

L ... Sequenzlänge

S ... Anzahl der Iterationen

V ... Anzahl der suboptimalen Strukturen

NL^3 ... Laufzeit von RNAsubopt

SNV^2L^2 ... Zeit um den Graphen aufzubauen

SN^2L^2 ... Zeit zur Berechnung der sum-of-pair scores

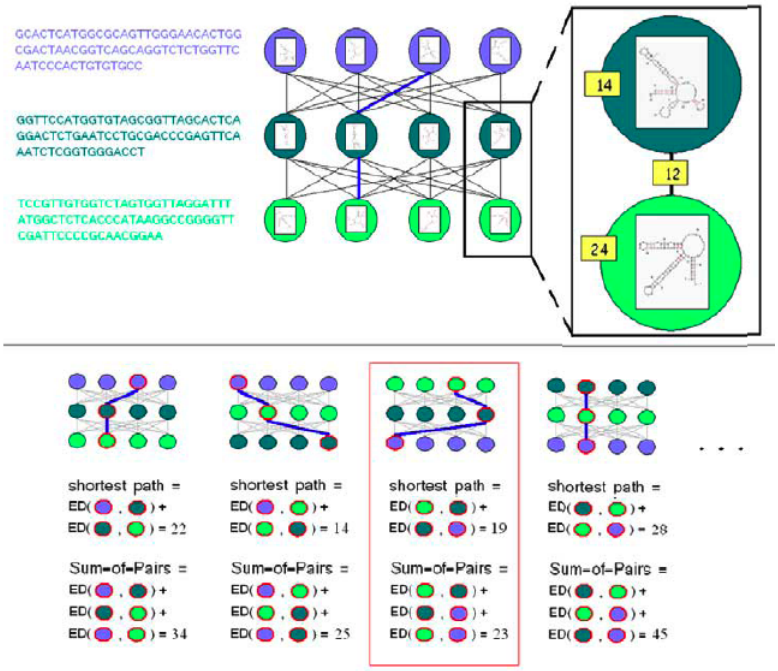

Abbildung 3: RNAspa im Einsatz

Da die Parameter N und S relativ klein sind, wird die Laufzeit haupsächlich durch L^3 und $V^2\ L^2$ bestimmt. Da man aber die Faltungen von RNAsubopt benötigt, liegt der eigentliche Flaschenhals dieses Ansatzes in der Berechnung der Kantengewichte. Denn dort werden zwischen 2 benachbarten Schichten jeweils alle Knoten miteinander verglichen um die V^2 Kanten zu berechnen. Der Vergleich folgt dem Ansatz zur Berechnung der Editierdistanz zwischen 2 Strukturen die als string in *bracket*-Notation dargestellt werden, also dem Needleman-Wunsch Algorithmus für ein globales Alignment, welcher in $\mathcal{O}(L^2)$ liegt.

Um diesen Aufwand ein wenig zu minimieren gibt es folgenden Ansatz:
Da jeder Layer nur unter Voraussetzung des jeweiligen direkt darüber liegenden berechnet wird ist es möglich einige zahlreiche Berechnungen zu verkürzen bzw. zu streichen. Zum Einen wäre es ein Fortschritt für den Vergleich von 2 Strukturen auf Ähnlichkeit nur eine bestimmte Anzahl von k Mismatches, Insertionen und Deletionen zuzulassen. Dies ist relativ einfach zu implementieren indem man in der Alignmentmatrix nur die $2k$ Diagonalen zulässt. Wenn in der ganzen oberen Schicht kein Knoten diese Bedingung erfüllt verdoppelt man einfach den k-Wert und durchläuft diese Berechnung noch einmal. Dieser Ansatz führt dazu, dass die Laufzeit für den Vergleich von $\mathcal{O}(L^2)$ zu $\mathcal{O}(KL)$ sinkt. Zum Anderen ermöglicht eine zweite Verbesserung sogar, dass man auf bestimmte Berechnungen ganz und gar verzichten kann: Wenn man zu einem Knoten s_i^j den ED-Wert berechnen will, kann man die Berechnung der Editierdistanz zwischen diesem Knoten und dem Knoten s_{i-1}^h streichen, falls die aktuelle ED in Knoten s_{i-1}^h schon größer ist, als die ED in s_i^j bzw. die ED von $s_{i-1}^h + k$ größer der ED von s_i^j ist. In diesem Fall kann man gleich mit der Berechnung der Distanz zwischen s_i^j und s_{i-1}^{h+1} fortfahren. Diese Modifizierungen ergeben eine Ersparnis von ca. 25% der Laufzeit. Im Endeffekt skaliert der

Algorithmus fast linear mit der Anzahl der Sequenzen, da der aufwendige sum-of-pair score lediglich im letzten Schritt nötig ist.

5 Vergleich von RNAcast und RNAspa

Um vorhergesagte Strukturen vergleichen zu können, muss man zuerst ein Maß für die Güte einer Prediction entwickeln.
Solch ein Wert ist *Matthews Correlation Coefficiant (MCC)*, der sich wie folgt berechnet:

$$\text{MCC} = \frac{TP * TN - FP * FN}{\sqrt{(TP+FP) * (TP+FN) * (TN+FP) * (TN+FN)}}$$

Diese Formel wird genutzt um eine Strukturvorhersage eines beliebigen Algorithmus mit der "echten" Struktur zu vergleichen. Dies geschieht Base für Base zwischen beiden Faltungen. Man zählt die Falsch Positiven (FP), Falsch Negativen (FN), Richtig Positiven(TP) und Richtig Negativen (TN) und setzt sie in die Formel ein. Somit ensteht ein Wert, der in dem Intervall $[-1, 1]$ liegt. Dabei gilt: Je näher der Koeffizient an der 1 liegt, um so genauer ist die vorhergesagte Struktur, d.h. wenn der MCC = 1 ist, sind beide Faltungen identisch.

Der Datensatz, der zum Vergleich beider Algorithmen benutzt wird, setzt sich aus mehreren verschiedenen ncRNA Familien zusammen.

lin4 miRNA Kleine RNA, die teilweise an target-RNAs binden kann um somit deren Translation zu unterbinden.

tRNA Transfer RNA, welche bei der Translation die passende Aminosäure zum jeweiligen Codon liefert.

5S rRNA Ribosomaler RNA Bestandteil der 50S- (Prokaryoten) und 60S-Untereinheit (Eukaryoten).

SRP RNA Diese RNA ist Bestandteil des *signal recognition particles*, welches sich an Proteine anlagert (während deren Translation) und diese unter GTP-Verbrauch an das Endoplasmatische Retikulum bindet, wo später eine Modifizierung des Proteins stattfindet.

IRES Virale internal ribosom entry site element, welches eine bestimmte Faltung annehmen kann, wodurch die Translation umgehend gestartet werden kann.

purine RS Eine RNA, die die Proteinbiosynthese reguliert und dabei speziell Guanin erkennt.

SAM RS Dieses Riboswitch ist bei der Methionin- und/oder Cysteinbiosynthese in grampositiven Bakterien involviert, wobei es vorallem die Termination der Tranksription reguliert.

snRNA U1, U2, U12 sind kleine RNA Moleküle, die im Zellkern von Eukaryoten vorkommen. Sie dienen vielen Regulationsmechanismen, wie dem RNA splicen, der Regulation von Tranksriptionsfaktoren oder der RNA Polymerase II. Dabei sind sie immer mit Proteinen assoziiert, mit denen sie sogenannte snRNPs bilden: small nuclaer ribonucleoproteins oder auch snurps.

Family	Average run-time per sequence in seconds				
	RNAspa	StemLoc	pmMulti	FoldAlignM	RNAcast
tRNA	**13**	47	21	549	<1
Tymo	**8**	70	12	93	<1
SAM	**13**	3	18	248	<1
Lysine	**20**	11	69	1076	<1
Purine	**84**	339	135	1518	<1
Cobalamin	**133**	105	no data	no data	<1
FMN	**56**	6	141	5042	<1
glmS	**58**	60	no data	no data	no data

Abbildung 4: Vergleich der Laufzeiten von RNAspa, StemLoc, pmMulti, FoldAlignM und RNAcast

Family	Avg. sequence identity	MCC				
		RNAspa	RNAcast			
			-t 5 –c 10 (default)	-t 5 –c 20	-t 3 –c 10	-t 3 –c 20
lin4	66%	**0.86**	0.73	0.73	no data	0.72
tRNA	50%	**0.81**	0.42	0.42	no data	0.65
5S RNA	59%	**0.58**	0.62	0.62	no data	0.84
U2	78%	**0.74**	0.72	0.72	0.69	0.69
S box riboswitch	67%	**0.75**	0.69	0.69	0.78	0.78
SRP RNA	52%	**0.95**	0.99	0.99	0.94	0.94
IRES	66%	**0.97**	0.93	0.93	0.90	0.90
Purine riboswitch	56%	**0.93**	0.78	0.78	0.81	0.81
U1	65%	**0.67**	0.72	0.72	no data	0.76
U12	83%	**0.76**	0.50	0.50	no data	0.68

Abbildung 5: Vergleich der Genauigkeit bei den Strukturvorhersagen zwischen RNAspa und RNAcast

Im Folgenden werden verschiedene Aspekte für die Güte eines Algorithmus erläutert. So wird in Abbildung 4 die Laufzeit verschiedener Ansätze mit diversen ncRNA Familien untersucht. Dabei stellte man fest, dass RNAcast signifikant schneller ist als alle anderen Algorithmen, aber auch zu einem Datensatz gar keine Ausgaben lieferte. RNAspa ist langsamer, aber im Vergleich zu den restlichen Algorithmen etwa gleich schnell (StemLoc) oder schneller (pmMulti und FoldAlignM), wobei alle Programme mit deren default - Einstellungen getestet wurden.

Abbildung 5 gibt einen großen Überblick über die verschiedenen Einstellungen von RNAcast im Vergleich zu RNAspa. Dabei beschreibt t das Abstraktionsniveau für das Mappen der Struktur auf den *shape* und c ist die Energieschranke, die RNAshapes für die *shape*-Analyse benötigt. An dieser Auswertung sieht man, dass die Genauigkeit der Vorhersagen von Familie zu Familie teilweise recht deutlich variiert, unabhängig wie ähnlich die Sequenzen sind. RNAspa lieferte in 6 von 10 Familien bessere Resultate, in den anderen 4 Familien lief RNAcast besser, wobei nicht von vornherein zu sagen ist, welche Einstellungen die besten Ergebnisse erreicht. Dies dürfte aber kein Problem sein, da RNAcast sehr schnell Resultate liefert, sodass man auch mehrere Einstellungen laufen lassen kann.

Family	MCC	
	RNAspa	RNAcast
lin4	**0.86**	no data
tRNA	**0.80**	0.42
5S RNA	**0.51**	no data
U2	**0.72**	no data
S box riboswitch	**0.77**	0.82
SRP RNA	**0.93**	0.78
IRES	**1.00**	no data
Purine riboswitch	**0.90**	0.78
U1	**0.61**	no data
U12	**0.83**	no data

Abbildung 6: Vergleich der Robustheit der Algorithmen von RNAspa und RNAcast

Abbildung 6 zeigt die verschiedenen Ergebnisse, wenn man beide Programme auf einem kontaminierten Datensatz laufen lässt, d.h. dass sich in dem Datensatz einer bestimmten ncRNA Familie eine Sequenz einer anderen Familie eingeschleust hat. Hier ist deutlich die Schwäche von RNAcast zu erkennen, denn in 6 von 10 ncRNA Familien konnte RNAcast keinen *common shape* ,also somit auch keinen *consensus shape*, finden. RNAspa hingegen konnte zu jeder Familie einen Output generieren, der von der Qualität der Strukturen sogar im Bereich der nicht kontaminierten Datensätze liegt.

FAZIT:
Zusammenfassend kann man sagen, dass RNAspa die besseren Vorhersagen liefert, besonders wenn der Datansatz womöglich verunreinigt ist. RNAcast ist dagegen sehr schnell und nur minimal ungenauer, hat aber Probleme mit relativ inhomogenen Datensätzen. Egal für welches Tool man sich zur Vorhersage von Sekundärstrukturen von ncRNAs entscheidet, muss man sich darüber im Klaren sein, dass es noch kein Programm gibt, welches zu jedem Datensatz absolut zufriedenstellende Ergebnisse liefert.

Literatur

[1] Yair Horesh, Tirza Doniger, Shulamit Michaeli, Ron Unger *RNAspa: a shortest path approach for comparative prediction of the secondary structure of ncRNA molecules*, Bioinformatics 2007 Jun 7; 8(366)

[2] Robert Griegerich, Jens Reeder *Consensus Shapes: an alternative to the Sankoff algorithm for RNA consensus structure prediction*, Bioinformatics: Vol.21 no. 17 2005, pages 3519-3523

[3] Stefan Wuchty, Walter Fontana, Ivo L. Hofacker, Peter Schuster *Complete Suboptimal Folding of RNA and the Stability of Secondary Structure* Biopolymers: Vol 49, 145-165 (1999)