Albert-Ludwigs-Universität Freiburg

Masterarbeit

Approximative RNA-Kinetiken

Institut für Informatik
Lehrstuhl für Bioinformatik
Prof. Dr. Rolf Backofen

Gregor Entzian

Gutachter:
Prof. Dr. Rolf Backofen
Prof. Dr. Christoph Flamm

Betreuer:
Dr. Martin Mann

19. Mai 2015

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bei allen bedanken, die zum Entstehen dieser Arbeit beigetragen haben.

Ein besonders herzliches Dankeschön gilt Dr. Martin Mann für die hervorragende Betreuung, die regelmäßigen Gespräche und die hilfreichen Ratschläge.

Ebenso danke ich Dr. Rolf Backofen für die Ermöglichung des Projektes.

Ein herzliches Dankeschön gilt auch:

Dr. Björn Voß für die Hinweise zum Mapping mit RNAHeliCes,

Stefan Jankowski für das Bereitstellen der Hardware,

Ronny Lorenz für den Support mit der ViennaRNA-Bibliothek und

Peter Kerpedjiev für die Unterstützung mit hilfreichen Werkzeugen zur Konvertierung von RNA-Strukturen.

Kurzzusammenfassung

Ein wichtiges Molekül in der Bioinformatik ist das Biopolymer RNA (ribonucleic acid). RNA ist viel mehr als ein Transkript der DNA, das in Proteine translatiert wird. Nichtkodierende RNA (ncRNA) übernimmt verschiedene regulatorische Aufgaben in Zellen. Ein RNA Molekül kann mehrere funktionelle Strukturen annehmen, indem sich unterschiedliche Wasserstoffbrücken bilden.

Es gibt Verfahren, mit denen die räumliche Struktur experimentell ermittelt werden kann. Dazu gehören Röntgenkristallographie und Kernspinresonanzspektroskopie (NMR-Spektroskopie). Diese Verfahren sind jedoch teuer und zeitaufwändig. Als Alternative haben sich Algorithmen bewährt, welche die räumliche Struktur mit der kleinsten freien Energie bestimmen.

Um die Wirkungsweise von RNAs besser zu verstehen, ist es jedoch wichtig, den gesamten Faltungsprozess zu betrachten. Ein Grund dafür ist, dass RNA ein kurzlebiges Molekül ist und nach einiger Zeit wieder abgebaut wird. Daher ist es möglich, dass die RNA abgebaut wird, bevor sie die Struktur mit der kleinsten freien Energie angenommen hat und somit in einer anderen funktionellen Struktur vorliegt.

Ein Beispiel für RNAs mit mehreren funktionellen Strukturen sind RNA-switches. Um die Fluktuationen zwischen diesen Strukturen besser untersuchen zu können, verwendet man einen Ansatz mit Energielandschaften. Dadurch werden alle möglichen Faltungstrajektorien betrachtet. Somit lassen sich wichtige alternative Strukturen bestimmen.

Weil die Menge der möglichen Strukturen exponentiell mit der Länge der RNA wächst, ist es erforderlich, die Energielandschaft auf einem groben Level zu betrachten. Ein verbreitetes Verfahren ist das Unterteilen der Landschaft in Bassins.

In dieser Arbeit wird ein Programm entwickelt, mit dem Energielandschaften auf einem groben Level betrachtet werden. Zusätzlich unterstützt es lokale Filterstrategien und nutzt paralleles Fluten der Bassins. Dadurch können auch längere RNA-Sequenzen in kurzer Zeit berechnet werden. Um die Nutzung des Programms komfortabel zu gestalten, kann durch ein Skript eine komplette „Verarbeitungspipeline“ angestoßen werden. Dabei wird eine Kinetik berechnet und grafisch aufbereitet, sodass der Faltungsverlauf von der offenen Kette bis zum Einstellen des Gleichgewichts sichtbar ist. Zusätzlich werden die Strukturen in einer Legende angezeigt. Bei den Eingabeparametern ist es neben verschiedenen Filterfunktionen auch möglich, die Temperatur anzugeben. Somit kann man Kinetiken für verschiedene Lebewesen berechnen.

Erklärung

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

_____ _____

Ort, Datum Unterschrift

Inhaltsverzeichnis

1 Einleitung

Mit der Entdeckung der Nukleinsäuren im Jahr 1869 durch Friedrich Miescher wurde auch der Grundstein für die Forschung mit Ribonukleinsäure (RNA) gelegt. Zunächst wurde die RNA nur als Bote angesehen, der die Information der DNA aus dem Zellkern ins Zytoplasma befördert, wo diese in Proteine translatiert wird. Allerdings ist dies nur die Funktion einer Klasse von RNAs, die man als messenger RNA (mRNA) bezeichnet. Die Erforschung weiterer RNAs (wie transfer RNA, ribosomale RNA und Ribozyme) und deren Eigenschaften führte Walter Gilbert 1986 zu seiner RNA-Welt-Hypothese. Laut dieser sind RNAs die Vorläufer der Organismen mit DNA, weil RNAs sowohl Informationen speichern als auch chemische Reaktionen katalysieren können [12].

Seit den 80er Jahren hat sich die Erforschung der Funktionen von nicht-proteinkodierenden RNAs (ncRNAs) als wichtiges Thema herausgestellt. Diese kodieren keine Proteine wie mRNA, sondern erfüllen katalytische und regulatorische Funktionen. Eine besondere Gruppe von ncRNA sind Ribozyme. Diese RNAs wirken aufgrund ihrer räumlichen Struktur u.a. als Katalysatoren bei der Translation von mRNA in Proteine. Für diese Entdeckung erhielten Sidney Altman und Thomas R. Cech 1989 den Nobelpreis für Chemie. Seitdem wurden Tausende ncRNAs entdeckt, die aufgrund ihrer räumlichen Struktur spezielle Aufgaben in der Zelle übernehmen.

Bioinformatische Methoden spielen eine entscheidende Rolle bei der Analyse von ncRNAs. Durch verschiedene Algorithmen kann man die räumliche Struktur ermitteln, die eine RNA am wahrscheinlichsten annimmt [27,28,38]. Es ist aber nicht nur wichtig, die wahrscheinlichste Struktur, sondern den gesamten Faltungsprozess zu betrachten, denn die RNA-switches können zum Beispiel zwischen verschiedenen Strukturen wechseln. Zudem sind viele Zwischenstrukturen schon so stabil, dass neue Formierungen mit Überwindung hoher Energiebarrieren verbunden sind. Somit ist es wahrscheinlich, dass die RNA im Faltungsverlauf zwischen metastabilen Zuständen fluktuiert und nicht lange genug existiert, um die Struktur mit der kleinsten freien Energie anzunehmen [15] S.81f. Deshalb ist es interessant, den gesamten Faltungsprozess zu betrachten, um wichtige Zwischenzustände oder alternative Strukturen zu untersuchen.

Um den Faltungsverlauf einer RNA zu modellieren eignen sich eher Methoden, die cotranskriptionelle Faltung betrachten, weil sich die räumliche Struktur schon bildet, während die RNA transkribiert wird.

In dieser Arbeit geht es u.a. darum, eine Software zu entwickeln die dabei hilft, RNA-switches und deren wahrscheinlichste Zustände zu detektieren. RNA-switches können zwischen zwei funktionellen Strukturen wechseln, z.B. dadurch, dass sich Metabolite in Zwischenräume der RNA-Struktur einlagern. Der Software liegt ein Ansatz mit Energielandschaften zu Grunde. Energielandschaften sind primär dazu geeignet, wichtige Konformationen zu erkennen [3]. Dadurch kann man die Software als Ergänzung zu einem Programm sehen, das eher auf die Modellierung des Faltungsverlaufs abzielt. So ein Programm ist i.d.R. nicht in der Lage die alternative Struktur eines RNA-switches zu ermitteln, weil es nur einen kleinen Teil der möglichen Strukturen betrachtet. Deshalb benötigt man zusätzlich ein Programm, das wahrscheinlichste alternative Strukturen aus allen möglichen Strukturen sucht, die eine RNA annehmen kann.

1.1 Verwandte Arbeiten

Für Faltungsprozesse gibt es verschiedene stochastische Algorithmen. Es gibt Monte-Carlo-Simulationen wie „rejection-based folding“ und „rejection-free folding“ (letztere auch als Gillespie oder BKL (Bortz-Kalos-Lebowitz-Algorithmus) bekannt). Diese Methoden simulieren jeweils einen möglichen Faltungsprozess eines einzelnen Moleküls. Ein Programm, das „rejection-free folding“ implementiert, ist *Kinfold* [7]. Die ersten beiden Methoden betrachten die einzelnen Strukturen in einer sehr detaillierten Ansicht. Dabei werden schrittweise viele kleine strukturelle Änderungen betrachtet. Dadurch sind Berechnungen für große RNAs sehr zeitaufwändig. Deshalb verwenden manche Algorithmen eine gröbere Simulationsebene, bei der ganze Teilstrukturen auf einmal verändert werden.

Eine Implementierung findet man im Programm *KineFold* [37]. KineFold verwendet mit den helixbasierten Moves auch Pseudoknoten. Diese Strukturen werden von vielen Algorithmen ignoriert, weil effizientere Methoden eingesetzt werden können. Algorithmen mit Pseudoknoten sind näher an der Realität, sind aber i.d.R. rechenintensiver. Deshalb sind sie nur für kurze Sequenzen geeignet. Eine aktuelle Quelle zum Thema Kinetiken mit Pseudoknoten ist [32]. In dieser Arbeit werden u.a. Faltungspfade von KineFold mit Kinetiken verglichen, die Strukturen mit Pseudoknoten enthalten.

Ein Algorithmus, der den Faltungsprozess co-transkriptionell betrachtet, ist im Programm *Kinwalker* implementiert. Co-transkriptionell bedeutet, dass die Faltung während der Transkription von DNA zu RNA erfolgt. Der Algorithmus von Kinwalker verwendet keine Pseudoknoten und basiert auf Heuristiken. Abhängig von der Transkriptionsgeschwindigkeit werden thermodynamisch optimale Teilstrukturen berechnet. Diese können mit dynamischer Programmierung effizient berechnet werden. Strukturen mit einer Länge von ca. 1500 Nukleotiden können damit berechnet werden [11]. Das ist mit allen anderen hier erwähnten Programmen nur schwer möglich, weil sie schon für kurze Sequenzen (bis zu ca. 100 Nukleotiden) Stunden bis mehrere Tage zur Berechnung benötigen. Dadurch, dass Kinwalker nur einen kleinen Teil der möglichen Strukturen betrachtet, verbraucht es weniger Arbeitsspeicher und Rechenzeit.

CoFold [29] ist ein Programm, das die wahrscheinlichste RNA-Struktur bestimmt und dazu die Effekte der co-transkriptionellen Faltung nutzt. Dabei geht es allerdings nur um die Bestimmung der wahrscheinlichsten Struktur und nicht um die Simulation des Faltungsprozesses. Sequenzen mit ca. 2000 Nukleotiden können verwendet werden.

Das Programm *RNAEAPath* [22] simuliert den Faltungsprozess von RNA. Es verwendet einen evolutionären Algorithmus und berechnet Energiebarrieren, wobei es aber ohne thermodynamische Berechnungen auskommt. Die Faltungsschritte werden auf einem groben Level betrachtet. Dadurch wird der Suchraum eingeschränkt und die Wahrscheinlichkeit in lokalen Optima zu enden, wird minimiert.

RNATabuPath [6] simuliert den Faltungsprozess und basiert auf Heuristiken. Diese verhindern, dass die Faltung in lokalen Minima endet.

Das Programm *Barriers* [9] verwendet einen Algorithmus, bei dem die einzelnen Strukturen in Cluster zusammengefasst werden. Diese werden auch Macrostates genannt. Sie sind durch ihre Struktur mit der kleinsten freien Energie identifizierbar. Das Programm berechnet Raten zwischen den Macrostates. Die Ausgabe kann mit dem Programm *Treekin* [36] verwendet werden, um die Faltungsdynamik der Macrostates zu berechnen.

HiKinetics [16] ist ein Programm, das sich für längere Sequenzen mit mehreren hundert Nukleotiden eignet. Es verwendet keine lokalen Minima zur Definition von Macrostates, sondern eine Abstraktion, die mehrere Strukturen mit ähnlichen Formen zu sogenannten Helix-Index-Shapes zusammenfasst. Dabei gibt es mehrere Abstraktionslevel die benutzt werden, um eine bestimmte Anzahl von stark variierenden Macrostates zu erzeugen. Diese Macrostates sind durch eine bestimmte Struktur namens HIShRep (Helix-Index-Shape-Representative) gekennzeichnet. Um die Transitionsraten für die HIShReps zu berechnen, wird für die NxN Strukturpaare der kleinste Pfad (Dijkstra-Algorithmus) ermittelt, über den eine Struktur in die andere überführt werden kann. Es ist hier der Pfad mit dem niedrigsten maximalen Gewicht. Die Struktur mit dem größten Gewicht auf diesem Pfad wird als Barriere bezeichnet und als Gewicht für die Transition verwendet. Diese Art der

Transitionsratenberechnung und das Erzeugen der Macrostates ist (im Vergleich zu anderen Programmen mit Flutalgorithmen) sehr effizient. Deswegen eignet es sich auch für längere Sequenzen, die von genaueren Algorithmen nicht mehr betrachtet werden können.

1.2 Motivation

Eine Standardmethode für die Untersuchung von RNA-Energielandschaften ohne Pseudoknoten, ist die **Lid**-Methode von Sibani et al. [33] oder das Programm *Barriers* [9]. Die dort verwendeten globalen Flutalgorithmen sind hauptsächlich durch den verfügbaren Speicher limitiert, weil alle möglichen RNA-Strukturen im Speicher gehalten werden. Die Anzahl der Strukturen steigt jedoch exponentiell [4]. Somit sind diese Algorithmen auf kurze Sequenzen beschränkt.

Das Problem kann durch exploratives globales Fluten gelöst werden (siehe [26]). In dieser Masterarbeit wird diese Methode detailliert vorgestellt. Dabei werden die Transitionen für jeden Macrostate separat berechnet. Dieser Algorithmus benötigt mehr Rechenzeit, aber weniger Speicher. Zudem lässt er sich leichter parallelisieren und ist offen für verschiedene lokale Filterstrategien, mit denen sich die Berechnung beschleunigen lässt. Da es für diesen Ansatz noch keine veröffentlichte Implementierung gibt, soll im Rahmen dieser Arbeit eine entwickelt werden. Zudem wird die Ausführungsgeschwindigkeit des Programms mit *Barriers* verglichen. Der Vorteil der Parallelisierung wird gemessen und der Effekt von verschiedenen Filtern wird dargestellt.

2 Grundlagen

2.1 RNA-Moleküle

RNAs sind Biopolymere (kettenförmige Moleküle) bestehend aus Nukleotid-Bausteinen, die sich in ihren Basen unterscheiden. Diese sind: Adenin, Cytosin, Guanin und Uracil. Die Basen sind an den Zucker Ribose gebunden. Mit einer weiteren Bindung an einen Phosphatrest entsteht ein Nukleotid. Ein solches markiert der rote Kreis in Abbildung 1. Vereinfacht werden RNA-Sequenzen aus einer Folge der Anfangsbuchstaben der Basen geschrieben, welche in Syntheserichtung (5'- zu 3'-Ende) angegeben wird. Die Bezeichnungen 5'- und 3'-Ende ergeben sich aus der Nummerierung der C-Atome der Ribose. Diese Sequenz wird auch als *Primärstruktur* p bezeichnet.

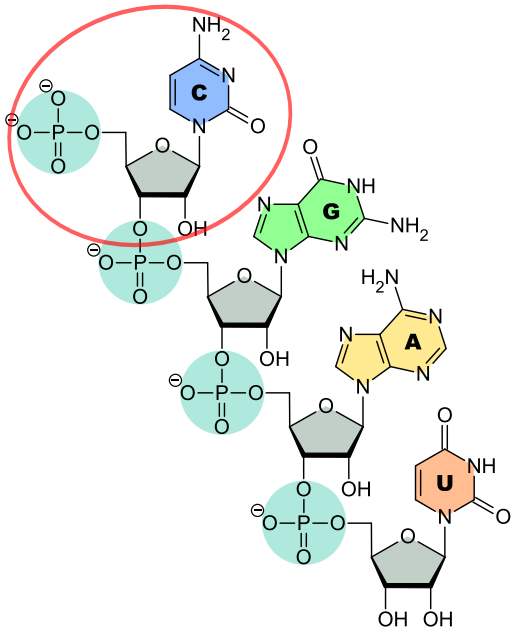


Abbildung 1: Verknüpfung der Basen Cytosin, Guanin, Adenin und Uracil (C,G,A und U) mit Ribose (grau) und dem Phosphatrest (türkis).[1]

Die Sekundärstruktur ist die zweidimensionale Darstellung, die durch Wasserstoffbrücken zwischen komplementären Basen entsteht. Gemäß Watson-Crick-Paarungen sind A-U und G-C komplementär. Es gibt jedoch noch mehr Möglichkeiten, um Wasserstoffbrücken zwischen den Basen zu bilden. In [20] sind 12 verschiedene Basenpaarungen erläutert. Für sogenannte Wobble-Paarungen sind auch G-U komplementär. Im Folgenden werden nur Watson-Crick- und Wobble-Paarungen berücksichtigt, die in Abbildung 2 dargestellt sind. Das sind die meist benutzten Paarungen für viele Algorithmen.

[1] `http://de.wikipedia.org/wiki/Ribonukleinsäure` letzter Besuch: 22.10.2014
[2] `http://en.wikipedia.org/wiki/Base_pair` letzter Besuch: 22.10.2014

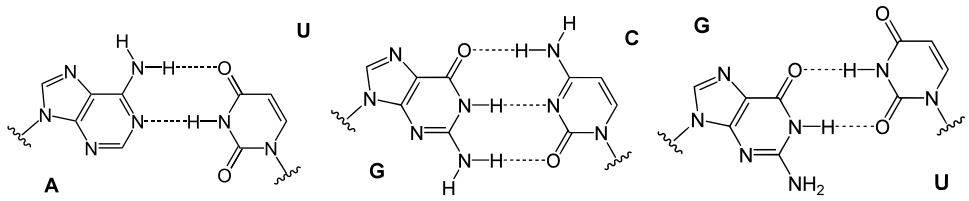


Abbildung 2: Bildung von Wasserstoffbrückenbindungen (gestrichelt) zwischen den verschiedenen Basen. Die Bindung zum Zucker-Phosphat-Rest ist durch eine Schlangenlinie dargestellt.[2]

Definition 1 (Sekundärstruktur s)
Sei p eine gegebene Primärstruktur und Ω die Menge aller erlaubten Basenpaarungen,

$$\Omega = \{(A,U),\ (U,A),\ (G,C),\ (C,G),\ (G,U),\ (U,G)\}.$$

dann ist eine Sekundärstruktur s kompatibel zu p, wenn

$$s = \{(i,j) \mid 1 \leq i < j \leq |p|,\ (p_i, p_j) \in \Omega\}$$

und

$$\forall\ (i,j),(i',j') \in s:\ (i = i') \Leftrightarrow (j = j').$$

d.h., dass eine Base genau eine Paarung eingehen kann.
(vgl. Kochniß 2008 [19])

Damit eine Struktur auch physikalisch geformt werden kann, wird eine minimale Distanz zwischen den Basenpaaren vorausgesetzt. Meistens verwendet man eine Schleifengröße (engl. loop size) von mindestens 3 Nukleotiden.

$$\forall (i,j) \in s : |j - i| > 3$$

Es gibt auch eine Struktur ohne Basenpaare. Diese wird als **offene Kette** oder **open chain** bezeichnet.

Definition 2 (Crossing)
Eine crossing- *oder auch* Pseudoknotenstruktur *entsteht, wenn für mindestens zwei Basenpaarungen $(i,j),(i',j') \in s$ gilt:*

$$i < i' < j < j' \text{oder } i' < i < j' < j.$$

Ansonsten wird s als „non-crossing“, „nested“ *oder* „pseudoknotenfrei“ *bezeichnet.*
(vgl. Kochniß 2008 [19])

Definition 3 (Dot-Bracket-Notation)
Pseudoknotenfreie Sekundärstrukturen können auf verschiedene Arten dargestellt werden. Die gängigsten sind in Abbildung 3 zu sehen. Die Dot-bracket-Notation ist die gängige textuelle Darstellung und wird von vielen Programmen verwendet. Ein „.“ repräsentiert eine ungepaarte Base an dieser Position, „(“ repräsentiert das „Eröffnen“ eines Basenpaars und „)“ repräsentiert das „Schließen“ eines Basenpaars.

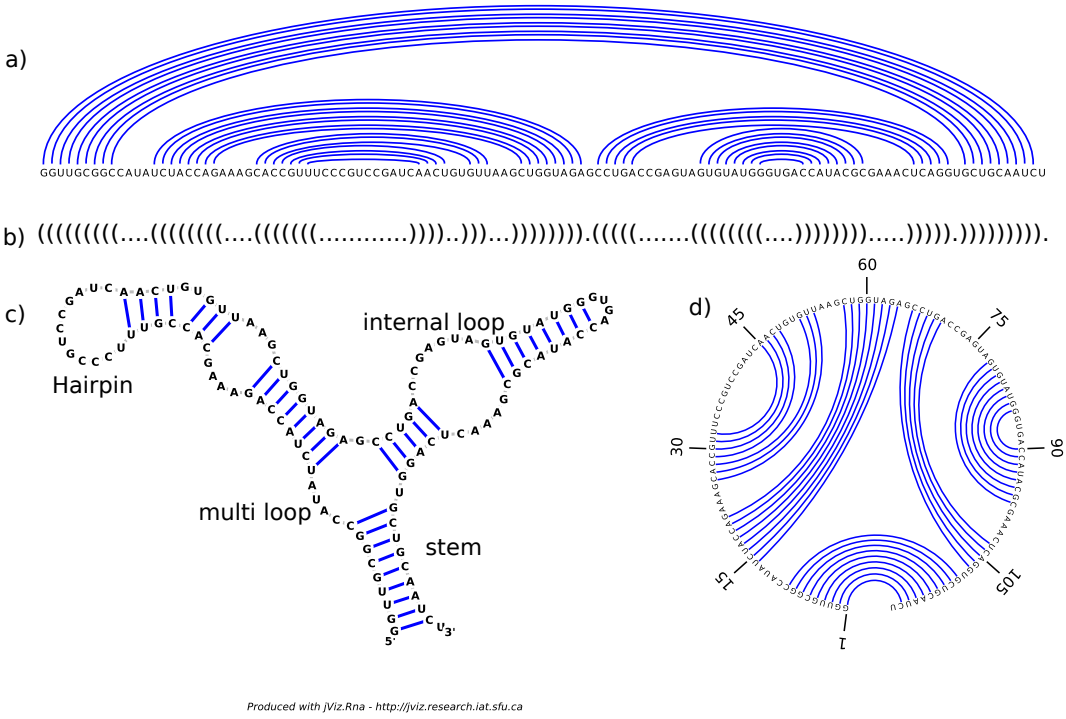


Abbildung 3: Verschiedene Darstellungen der Sekundärstruktur. a) lineares Feynman Diagramm, b) Dot-bracket-Notation, c) Strukturplot mit Strukturelementen, d) Kreisdiagramm. [32]

Definition 4 (Konformationsraum χ_p)
Sei p ein Primärstruktur, so ist der Konformationsraum χ_p die Menge aller Sekundärstrukturen, die mit p kompatibel sind.

$$\chi_p = \{s : s \text{ ist eine Sekundärstruktur und gemäß Def. 1 mit } p \text{ kompatibel. }\}$$

Alle weiteren Erwähnungen von Sekundärstrukturen in dieser Arbeit beziehen sich auf pseudoknotenfreie Strukturen. Algorithmen mit diesen Strukturen lassen sich schneller berechnen als Algorithmen mit Pseudoknoten, die meistens NP-vollständig sind [24].

2.2 Energiemodell

Die freie (ungebundene) Energie einer RNA-Struktur wird durch eine Energiefunktion E bestimmt.

Definition 5 (Energiefunktion E)
Gegeben ein Konformationsraum χ_p, sei E die Energiefunktion für die gilt

$$E : \chi_p \to \mathbb{R}$$

Die freie Energie der Struktur s ist $E(s)$. Ein niedriger Energiewert steht für eine hohe strukturelle Stabilität und umgekehrt.

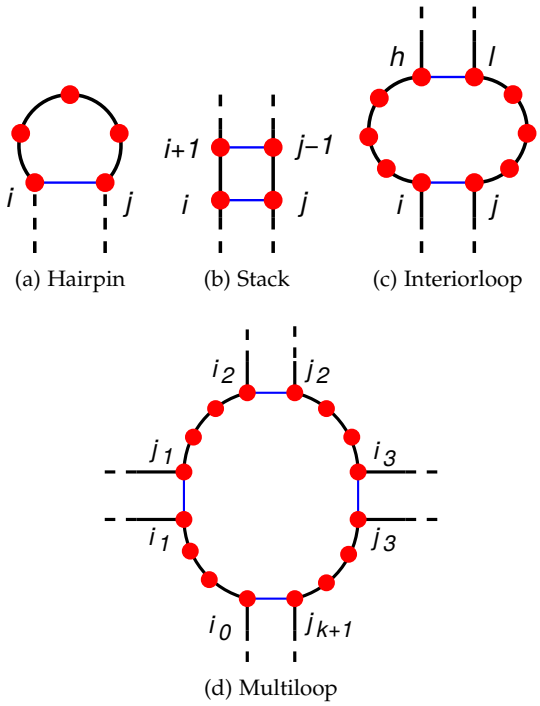
Die freie Energie von Sekundärstrukturen wird meistens mit dem „Nearest-Neighbor-Model“ berechnet. Dabei wird die RNA-Struktur in kleinere Einheiten zerlegt, für welche die freie Energie berechnet wird. Zu diesen RNA-Strukturelementen gehören unter anderem: Hairpin, Stack, Bulge, Interior loop und Multi loop (siehe Abbildung 4).

Definition 6 (Sekundärstrukturelemente)
Die Sekundärstrukturelemente der RNA-Struktur p werden mit einem Basenpaar (i, j) definiert:

- *Hairpin:* $(i,j) \in s$, *wobei* $\nexists (i',j') \in s : i < i' < j' < j$
- *Stack:* $(i,j) \in s$ *und* $(i+1, j-1) \in s$
- *Interior loop:* $(i,j) \in s$ *und* $(h,l) \in s$ *mit* $i+1 < h < l < j-1$ *und die Teilsequenzen* $[i+1, h-1]$ *und* $[l+1, j-1]$ *enthalten keine Basenpaare* ($\nexists (i',j') \in s : i < i' < h$ *und* $l < i' < j$)
- *Spezialfall Bulge: falls* $h = i+1$ *oder* $l = j-1$
- *k-Multi loop: hat ein schließendes Basenpaar* $(i_0, j_{k+1}) \in s$ *und* k *weitere Basenpaare* $(i_1, j_1), \ldots, (i_k, j_k) \in s$, *wobei noch folgende Eigenschaften gelten:*
  - $i_0 < i_1 < j_1 < \ldots < i_k < j_k < j_{k+1}$
  - $\forall (0 \leq l, l' \leq k)$ *gilt* $\nexists (i',j') \in s$ *mit* $i' \in [j_l, i_{l+1}]$ *und* $j' \in [j_l, i_{l+1}]$ *(keine Basenpaare zwischen den einzelnen loops)*

  Die Basenpaare $(i_1, j_1), \ldots, (i_k, j_k)$ *werden auch als* ***Helices*** *der k-Multi loop bezeichnet.*

In der praktischen Anwendung wird für Hairpin Strukturen (i,j) die Bedingung $i < j-3$ hinzugefügt. Die Loop-Sequenz benötigt mindestens 3 ungepaarte Basen, um physikalisch (auf Grund der Bindungswinkel) zu entstehen.

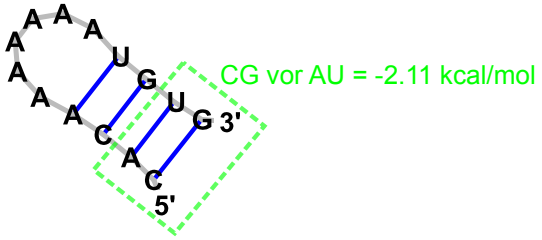


(a) Hairpin (b) Stack (c) Interiorloop

(d) Multiloop

Abbildung 4: Darstellung der Sekundärstrukturelemente: Hairpin, Stack, Interiorloop und Multiloop. [31]

Um die freie Energie der Gesamtstruktur zu erhalten, wird die Summe über die Energien der Strukturelemente gebildet. In [5] wurde schon im Jahr 1962 gezeigt, dass vertikale Stapel von Basenpaaren den größten Beitrag zur Stabilität einer RNA-Helixstruktur liefern. Deshalb ist auch die direkt benachbarte Umgebung eines Basenpaars wichtig, um die Stabilität zu bestimmen [2]. Das „Nearest-Neighbor-Model“ [35] betrachtet, wie der Name vermuten lässt, auch die Energien der unmittelbar benachbarten Basen und Basenpaare. Ein Beispiel ist in Abbildung 5 dargestellt.



$$
\begin{aligned}
\Delta G^\circ_{37} =& \Delta G^\circ_{37}(\text{Watson-Crick-Paare}) + \Delta G^\circ_{37}(\text{Hairpin Loop}) \\
\Delta G^\circ_{37} =& \Delta G^\circ_{37}(\text{Watson-Crick-Paare}) + \Delta G^\circ_{37}(\text{terminal mismatch}) \\
& + \Delta G^\circ_{37}\text{Hairpin Initiation(6 Nukleotide)} \\
\Delta G^\circ_{37} =& \Delta G^\circ_{37}(\text{CG vor AU}) + \Delta G^\circ_{37}(\text{AU vor CG}) + \Delta G^\circ_{37}(\text{CG vor AU}) \\
& + \Delta G^\circ_{37}\text{AU end penalty} + \Delta G^\circ_{37}(\text{AU vor AA}) \\
& + \Delta G^\circ_{37}\text{Hairpin Initiation(6 Nukleotide)} \\
\Delta G^\circ_{37} =& -2.11 kcal/mol - 2.24 kcal/mol - 2.11 kcal/mol + 0.45 kcal/mol \\
& - 0.8 kcal/mol + 5.4 kcal/mol \\
\Delta G^\circ_{37} =& -1.4 kcal/mol
\end{aligned}
$$

Abbildung 5: Beispiel für die Nearest-Neighbor-Energieberechnung einer Hairpin Struktur mit 6 ungepaarten Nukleotiden bei $37°C$ (die Energien sind temperaturabhängig). Dabei sieht man, wie die Energieberechnung des Stacks von den benachbarten Basenpaaren abhängt. $\Delta G°$ ist die Änderung der Gibbsschen freien Energie (auch als freie Enthalpie bekannt). Es wurden die Nearest-Neighbor-Database (NNDB)-Parameter von 2004 verwendet [35].

In dieser Arbeit wird das ViennaRNA-Paket 2.1.7 zur Berechnung der Energien verwendet. Dieses verwendet den „Zuker-Algorithmus“ (siehe [39]), der mit dynamischer Programmierung arbeitet und auf dem „Nearest-Neighbor-Model“ basiert. Seit ViennaRNA-Version 2.0 werden die Turner Parameter von 2004 als Standardparameter verwendet [23]. Deshalb werden sie in dieser Arbeit (sofern nicht anders angegeben) verwendet.

2.3 Energielandschaften

Mit Hilfe des Energielandschaftsmodells kann man Kinetiken studieren, ohne Einzelfaltungssimulationen zu machen [26]. Damit sind stochastische Ansätze gemeint, welche die Mastergleichung direkt auf jede einzelne Struktur anwenden. Diese sind auf kurze Sequenzen beschränkt ([1], [8]). Um Kinetiken für große RNA-Sequenzen berechnen zu können, kann man die komplette Energielandschaft auf einem groben Abstraktionslevel betrachten, indem man sie in Macrostates aufteilt. Als Beschreibung für die Macrostates werden oft lokale Minima der Landschaft verwendet. In der berechneten Kinetik kann man die Fluktuationen zwischen den Macrostates betrachten.

Energielandschaften für RNA-Moleküle sind durch das Tripel Konformationsraum (siehe Def. 4), Nachbarschaftsrelation zwischen Zuständen, Energiefunktion für Zustände (siehe Def. 5) definiert.

Die Nachbarschaft von Struktur s ist $M(s)$. Das „Move-Set“ beschreibt wie $M(s)$ aus s erzeugt werden kann. In dieser Arbeit wird das „Single-Move-Set“ verwendet, bei dem Nachbarn durch Hinzufügen oder Entfernen von Basenpaaren in s erzeugt werden. Ein „Single-Move“ ist das Entfernen oder Einfügen eines Basenpaars. Es gibt auch andere Move-Sets, z.B. Shift-Move-Sets, die das Verschieben von Basenpaaren mit einbeziehen [8]. Ein Move-Set, das die Bildung und Lösung von mehreren Basenpaaren in einem Schritt betrachtet, ist das Helix-Move-Set [17]. Es wird z.B. im Programm Kinefold verwendet [37].

Definition 7 (Nachbarschaft für Single-Moves)
Die Nachbarschaft M ist die Menge aller Nachbarstrukturen einer gegebenen Struktur s. In diesem Fall generiert durch das Entfernen oder Hinzufügen eines einzelnen Basenpaars:

$$M(s) = \{s' \in \chi_p \mid |s' \setminus s| = 1 \vee |s \setminus s'| = 1\}$$

Beispiel 1 (Nachbarschaft)

Sequenz: $p =$ `GGGGGCCCCC`
Struktur: $s =$ `((......))`
$M(s) =$ {`(((....)))`, `(........)`, `.(......).`}

Im Beispiel 1 wurde im ersten Move (rechts) das äußere Basenpaar und im zweiten Move (Mitte) das innere Basenpaar entfernt. Im dritten Move (links) wurde ein Basenpaar hinzugefügt. Weitere Einfügungen sind nicht möglich, weil die innere Schleife der Hairpin weniger als drei freie Basen hätte. Weitere Entfernungen sind auch nicht möglich, weil s nur zwei Basenpaare hat und Single-Moves betrachtet werden.

Eine Energielandschaft umfasst die Menge aller Strukturen, welche durch rekursives Anwenden des Move-Sets erreichbar sind. Jeder Struktur wird durch eine **Energiefunktion** (siehe Def. 5) eine freie Energie zugewiesen. Wenn verschiedene Strukturen den gleichen Energiewert zugewiesen bekommen, wird die Energiefunktion bzw. die resultierende Energielandschaft als **degeneriert** bezeichnet. Da die Energiefunktion basierend auf dem „Nearest-Neighbor-Model“ degeneriert ist, benötigen wir eine **strenge Ordnung** für die Strukturen, damit die Algorithmen, die darauf aufbauen, eindeutig und vergleichbar sind. Bei unserer strengen Ordnung wird:

1. aufsteigend nach Energien geordnet.
2. bei gleichen Energien wird lexikographisch aufsteigend auf Basis der Dot-Bracket-Strukturkodierung 3 geordnet: „(“ < „)“ < „.“

2.4 Kinetik

Eine Kinetik beschreibt das Faltungsverhalten eines RNA-Moleküls. Sie gibt zu jedem Zeitpunkt die Wahrscheinlichkeit für jede Struktur des Moleküls an. Die Berechnung wird durch einen zeitkontinuierlichen Markov-Prozess realisiert [13] S.405ff. Der Algorithmus verwendet einen Vektor mit Anfangswahrscheinlichkeiten und eine Ratenmatrix, die Übergangsraten für alle Zustände enthält. Der Markov-Prozess und die Anfangswahrscheinlichkeiten werden oft als definiert vorausgesetzt, weil das Prinzip bei allen Kinetiken gleich ist. Es gibt jedoch Unterschiede bei der Berechnung der Ratenmatrix. Deshalb reduziert sich der Begriff oft nur auf die Matrix.

Die Matrix muss bestimmte Anforderungen erfüllen, damit der Markov-Prozess gegen eine stationäre Verteilung konvergiert. Eine wichtige Anforderung ist Irreduzibilität (oder auch als Ergodizität bezeichnet). Irreduzibilität bedeutet, dass alle Zustände direkt oder indirekt miteinander verbunden sein müssen. Diese Anforderung ist in Kap. 6 wichtig, wenn es um Filter geht.

2.4.1 Microstate-Kinetik

Eine Microstate-Kinetik betrachtet alle Strukturen (Microstates) einer RNA. Deren Anzahl steigt exponentiell mit der Länge der RNA [4]. Deshalb wird sie nur für kurze Sequenzen verwendet. Sie ist aber die genaueste Kinetik. Die Rate für zwei benachbarte Strukturen $x, y \in \chi_p$ (χ_p ist die Menge aller Strukturen) ist $r_{x \to y}$. Sie wird mittels der Boltzmann-Statistik berechnet.

$$\Delta G_{xy} = E(y) - E(x) \tag{1}$$

$$r_{x \to y} = \Gamma_M \begin{cases} e^{-\frac{\Delta G_{xy}}{RT}} & \text{wenn } \Delta G_{xy} > 0 \\ 1 & \text{wenn } \Delta G_{xy} \leq 0 \end{cases} \tag{2}$$

Das Boltzmann-Gewicht ist

$$w(x) = e^{-E(x)/RT} \tag{3}$$

somit ist

$$e^{-\frac{\Delta G_{xy}}{RT}} = e^{-\frac{E(y)-E(x)}{RT}} = e^{-\frac{E(y)}{RT}} e^{\frac{E(x)}{RT}} = w(y)/w(x) \tag{4}$$

Damit lässt sich das Metropoliskriterium (Formel 2) schreiben als

$$r_{x \to y} = \Gamma_M \min\{1, e^{-\frac{\Delta G_{xy}}{RT}}\} \tag{5}$$

und weiter vereinfachen zu

$$r_{x \to y} = \Gamma_M \min\{1, w(y)/w(x)\} \tag{6}$$

ΔG_{xy} ist die Differenz der freien Energien von x und y. R ist die Gaskonstante und T die Temperatur in Kelvin. Γ_M ist ein Zeitfaktor, der die Zeit eines einzelnen Strukturübergangs widerspiegelt. Dieser Parameter wird jedoch meistens vernachlässigt und auf 1 gesetzt, weil es an experimentellen Daten mangelt. Im Rahmen dieser Arbeit verwenden wir $\Gamma_M = 1/(\max_{x \in \chi_p} |M(x)|)$, um eine Normierung der Raten zu Pseudowahrscheinlichkeiten zu erhalten [25]. Die Fallunterscheidung in Formel 2 stellt das **Metropoliskriterium** dar. Es ist das Standardverfahren, um einen Markov-Prozess **umkehrbar** zu machen, bzw. ein **detailliertes Gleichgewicht** zu erreichen. Ein umkehrbarer Markov-Prozess wird benötigt, weil er die Konvergenz garantiert. Das bedeutet, dass immer eine stationäre Verteilung erreicht wird.

2.4.2 Macrostate-Kinetik

Bei längeren Sequenzen wird eine Macrostate-Kinetik verwendet, weil der Speicher für eine Kinetikberechnung mit allen Strukturen extrem schnell anwächst. Das liegt daran, dass die Anzahl der Sekundärstrukturen exponentiell mit der Länge der RNA-Sequenz zunimmt. Macrostates sind nichtüberlappende Cluster von Microstates. Meistens werden lokale Minima als Clusteranker (ClusterID) verwendet. Ein Cluster wird hier auch als Bassin b bezeichnet. Ein Bassin ist die Menge der Microstates, für die ein Gradient-Walk in einem lokalen Minimum endet.

Definition 8 (Gradient-Walk)
Ein Gradient-Walk g_w startet bei einem Zustand w_0 und „besucht" den kleinsten Nachbarn entsprechend der definierten strengen Ordnung (Def. 2.3). Dieser „Besuch" wird für den Nachbarn und alle folgenden Nachbarn rekursiv ausgeführt, bis ein lokales Minimum w_N gefunden wurde. Dabei ist N die Länge des Walks. Aus der Definition ist ersichtlich, dass der Zustand w_N am Ende eines Walks ein lokales Minimum ist.

$$\forall_{w_i, w_{i+1} \in g_w} : \nexists s \in M(w_i) : s < w_{i+1} \tag{7}$$

Fazit: Es gibt keine Struktur s in der Nachbarschaft $M(w_i)$, die eine kleinere Energie besitzt als w_{i+1}.

Da für jeden Zustand des Gradient-Walks alle Nachbarn samt Energien aufgezählt werden müssen, ist dieser Algorithmus eine zeitlich aufwändige Methode.

Eine Funktion, die alle Microstates allen Bassins zuordnet ist $F : X \to B$. Die inverse Funktion F^{-1} ist die Menge aller Zustände, die zu einem Bassin $b \in B$ zugeordnet ist. Die Raten für Macrostates werden auf Basis der Microstate Raten berechnet:
Für jedes Bassin b wird eine **kanonische Zustandssumme** gebildet.

$$Z_b = \sum_{x \in F^{-1}(b)} w(x) \tag{8}$$

$$\tag{9}$$

Die Wahrscheinlichkeit im Microstate x zu sein, während man im Bassin b ist, sei

$$P_b(x) = \begin{cases} w(x) Z_b^{-1} & \text{wenn } x \in F^{-1}(b) \\ 0 & \text{sonst} \end{cases} \tag{10}$$

Die Übergangsrate $q_{b \to c}$ für zwei Bassins $b, c \in B$ ist dann:

$$\begin{aligned} q_{b \to c} &= \sum_{x \in F^{-1}(b)} \left(P_b(x) \sum_{y \in M(x) \cap F^{-1}(c)} r_{x \to y} \right) \\ &= \sum_{(x,y)} P_b(x)\, r_{x \to y} \\ &= \sum_{(x,y)} \frac{w(x)}{Z_b} \Gamma_M \min\{w(y)/w(x), 1\} \\ &= Z_b^{-1} \sum_{(x,y)} \Gamma_M \min\{w(y), w(x)\} \\ &= \Gamma_M \frac{Z_{\{b,c\}}}{Z_b} \text{ und dadurch} \end{aligned} \tag{11}$$

$$q_{c \to b} = \Gamma_M \frac{Z_{\{b,c\}}}{Z_c} \ . \tag{12}$$

Die Zustandssumme $Z_{\{b,c\}}$ für die Transitionen zwischen den Bassins b und c ist also für beide Richtungen gleich (siehe Formel 12). Zu beachten ist, dass der oben für Γ_M definierte Wert für Microstateraten zu sehr kleinen Werten und numerischen Problemen führen kann. Somit wird hier für lange Sequenzen die maximale Anzahl benachbarter Bassins zur Normierung verwendet. Die Berechnung der Zustandssummen ist zum besseren Verständnis in Abbildung 6 dargestellt.

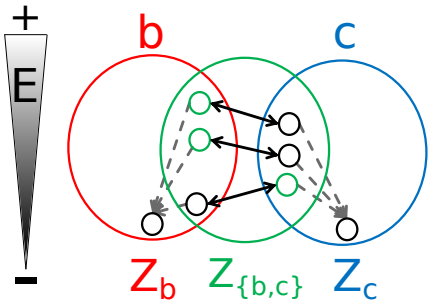


Abbildung 6: Berechnung der Zustandssummen. Die Bassins sind b und c. Die Zustandssummen Z_b und Z_c sind jeweils die Summe über die Boltzmanngewichte der Microstates (Kreise in der roten und blauen Ellipse). Die Zustandssumme für die Transition $Z_{b,c}$ ist die Summe über die Boltzmanngewichte der höher liegenden Microstates aller Microstate Transitionen.

Die Berechnung der Ratenmatrix für eine Macrostate-Kinetik erfolgt über die Zustandssummen der Bassins und Transitionen. Diese können durch verschiedene lokale oder globale Flutstrategien berechnet werden. Sie bestimmen die Art und Weise, in der die einzelnen Zustände ermittelt werden und beeinflussen den Speicherverbrauch und die Laufzeit.

3 Globales Fluten

Das globale Fluten der Energielandschaft ist eine Möglichkeit, die Macrostate Transitionsraten relativ effizient zu berechnen.

Ein naiver Ansatz um die Transitionsraten zu berechnen besteht darin, alle Konformationen $s \in \chi_p$ aufzuzählen. Dabei werden für jedes s die Zuordnung $F(s)$ zu einem Macrostate ermittelt und die Zustandssummen aktualisiert. Weil hauptsächlich die Zuordnungen $F(s)$ rechenintensiv sind, wurden effizientere Algorithmen entwickelt.

Ein Standardalgorithmus ist die **Lid** Methode [33]. Das Lid L ist ein Parameter, der eine obere Energieschranke darstellt. Es werden nur Microstates unterhalb dieser Schranke für die Berechnung der Zustandssummen verwendet. Der Algorithmus startet in einem lokalen Minimum und zählt von dort aus alle weiteren Microstates auf. Es gibt eine Liste D mit bearbeiteten Zuständen und ihrer Zuordnung $F(s)$. Eine weitere Liste T enthält benachbarte Zustände, zu denen auf D, die noch nicht bearbeitet wurden. Die Liste T wird mit dem lokalen Minimum initialisiert. Anschließend wird für jeden aktuellen Zustand $x \in T$ folgende Prozedur ausgeführt:

Algorithm 1 LID Methode

```
1:  procedure BERECHNEZUSTANDSSUMMEN
2:      b ← F(x)
3:      Z_b ← w(x)
4:      for all m ∈ M(x) mit E(m) < L do
5:          if m ∈ D then m wurde schon bearbeitet
6:          end if
7:          if m ∉ T then setze m auf T
8:          end if
9:          if F(x) ≠ F(m) then aktualisiere Z_{F(x),F(m)} mit δ^{-1} min{w(m), w(x)}
10:         end if
11:     end for
12: end procedure
```

Durch das Lid L werden weniger $F(x)$ Operationen durchgeführt. Dieser Algorithmus endet immer in einer zusammenhängenden Energielandschaft. Allerdings wird nur der Teil unterhalb L betrachtet. Wenn der verfügbare Speicher unterhalb L nicht ausreicht um alle Microstates aufzuzählen, entsteht eine unvollständige Energielandschaft, die nicht zusammenhängend sein kann. Damit die Ratenmatrix irreduzibel ist und den Anforderungen des Markovprozesses genügt, müssen die Zustände im Nachhinein verbunden werden.

Ein ähnlicher globaler Flutalgorithmus ist im Programm *Barriers* [9] implementiert. Der *Barriers* Algorithmus beginnt mit einer sortierten Liste aller Strukturen in χ_p. Die Strukturen sind nach einer strengen Ordnung (gemäß Def. 2.3) sortiert. Die Liste wird der Reihe nach abgearbeitet. Dadurch startet der Algorithmus immer mit dem globalen Minimum. Für die aktuelle Struktur s werden alle Nachbarn $M(s)$ generiert und auf einem Stapel gespeichert. Anschließend wird für jeden Nachbarn geprüft, ob er schon einmal betrachtet wurde. Das ist der Fall, wenn der Nachbar n (gemäß der Ordnung) kleiner als s ist. Weil er schon betrachtet wurde, kann seine Macrostate Zuordnung $F(n)$ abgerufen werden. Nachdem alle Nachbarn prozessiert wurden, gibt es drei Möglichkeiten für die aktuelle Struktur:

- Wenn kein kleinerer Nachbar bekannt ist, ist sie ein lokales Minimum und initialisiert ein neues Bassin b. Dabei wird auch die Zustandssumme des Bassins Z_b initialisiert.
- Wenn ein benachbartes Bassin gefunden wurde, gehört die Struktur zu diesem und wird ebenfalls gespeichert (inklusive BassinID). Die Zustandssumme des Bassins Z_b wird aktualisiert.

- Wenn zwei oder mehr lokale Minima Nachbarn der aktuellen Struktur enthalten, werden die Zustandssummen der Transitionen $Z_{b,c}$ aktualisiert.

Nun wurde die aktuelle Struktur bearbeitet und die gleiche Prozedur wird für die nächste Struktur der Liste durchgeführt, bis die ganze Liste abgearbeitet ist.

Der Nachteil beider Algorithmen ist der Speicherverbrauch, weil alle Microstates der Energielandschaft auf einmal in D und T gespeichert werden. Deshalb sind sie nur für kurze Sequenzen geeignet.

Das Problem kann teilweise durch lokales Fluten der Energielandschaft behoben werden. Dieser Lösungsansatz wird im nächsten Kapitel diskutiert.

Bestehende Implementierungen des globalen Flutalgorithmus wurden auf Geschwindigkeit getestet. Das verschafft einen kleinen Einblick in die Leistung der bisherigen Software. Eine Implementierung des globalen Flutalgorithmus ist in der ELL (Energy Landscape Library)[3] enthalten. Diese basiert auf der Bibliothek BIU (Bioinformatik Utilities)[4] und auf der ViennaRNA-Bibliothek[5]. Für das globale Fluten haben Geschwindigkeitstests mit einem ELL-basierten Programm (*RNA_barriers*) und einem rein ViennaRNA-basierten Programm (*Barriers*) gezeigt, dass dieselbe Implementierung auf der ELL ca. viermal langsamer ist (siehe Grafik 36 im Appendix). Der Grund dafür ist vermutlich, dass die ELL eine höhere Abstraktion verwendet, als die ViennaRNA (Vererbung und zusätzliche Methodenaufrufe könnten für den Zeitunterschied verantwortlich sein).

[3] `http://www.bioinf.uni-freiburg.de/Software/Libraries/` letzter Besuch: 31.10.2014
[4] `http://www.bioinf.uni-freiburg.de/Software/Libraries/` letzter Besuch: 31.10.2014
[5] `http://www.tbi.univie.ac.at/RNA/` letzter Besuch: 31.10.2014

4 Lokales Fluten

Wie im letzten Kapitel erwähnt wurde, kann der Speicherverbrauch beim Berechnen der Macrostate-Transitionsraten durch lokales Fluten reduziert werden. Der lokale Flutalgorithmus ist auch der Kern der partiellen Energielandschaftexploration die im Artikel [26] beschrieben ist. Da dieser Algorithmus zentral für diese Arbeit ist, wird er an dieser Stelle detailliert dargestellt.

4.1 Algorithmus

Beim lokalen Fluten wird zunächst nur das Bassin eines einzelnen Minimums x_m geflutet. Das Ziel ist es, die Zustandssumme des Bassins Z_b und die Zustandssummen für die Kontaktflächen $Z_{b,c}$ zu anderen Bassins c zu bestimmen. Dazu wird zunächst Z_b mit $w(x_m)$ initialisiert und x_m wird zur Liste bearbeiteter Zustände D hinzugefügt. Danach werden die Nachbarn $M(x_m)$ generiert und auf eine To-do-Liste T gesetzt. Nun wird folgende Prozedur ausgeführt, bis T leer ist:

Algorithm 2 Lokales Fluten

procedure FLUTEBASSIN
  $x \leftarrow kleinsterMicrostate(T)$ ▷ $\forall_{x' \neq x \in T} : x < x'$ (siehe Def. strenge Ordnung 2.3)
  $g \leftarrow gradientWalk(x)$
  ▷ wg. speichern von Zwischenergebnissen reicht: $g \in M(x)$ mit $\forall_{m \neq g \in M(x)} : g < m$
  if $g \in D$ **then** ▷ ..., dann folgt $F(x) = b$ (siehe Def. 8)
    $D.add(x)$
    $Z_b \leftarrow Z_b + w(x)$
    for all $m \in M(x)$ mit $m > x$ **do**
      $T.add(m)$ ▷ alle Nachbarn werden zu T hinzugefügt
    end for
    ▷ Transitionen, die b von x ausgehend verlassen, werden behandelt:
    for all $m \in M(x)$ mit $m < x$ und $m \notin D$ **do**
      $Z_{b,F(m)} \leftarrow Z_{b,F(m)} + \frac{1}{\delta} w(x)$ ▷ aktualisiere Transitionszustandssumme
    end for
  else ▷ sonst ist $F(x) \neq b$:
    ▷ Transitionen, die von x ausgehend in b enden, werden behandelt:
    for all $m \in M(x)$ mit $m < x$ und $m \in D$ **do**
      $Z_{F(m),b} \leftarrow Z_{F(m),b} + \frac{1}{\delta} w(x)$ ▷ aktualisiere Transitionszustandssumme
    end for
  end if
end procedure

Für die meisten Zustände kann die Zuordnung $F(x)$, aufbauend auf den zuvor berechneten Zuständen, effizient ermittelt werden. Im Gegensatz zum globalen Fluten mit *Barriers* (siehe Kap. 3) muss beim lokalen Fluten ein kompletter Gradient-Walk berechnet werden, um $F(x)$ zu bestimmen. Das gilt für alle Nachbarn, die $\notin b$ sind und für $x \notin b$. Daraus resultiert ein Mehraufwand gegenüber dem globalen Fluten, der bei einem Laufzeitvergleich später in Kap. 5.3 untersucht wird.

4.2 Implementierung

Aufgrund der vorherigen Tests von Softwarebibliotheken mit Flutalgorithmen (siehe Kapitel 3) wurde für diese Arbeit eine ViennaRNA-basierte Implementierung vorgezogen.

Die neue Implementierung des Flutalgorithmus basiert nur auf den Funktionen der ViennaRNA-Bibliothek und ist in C++ geschrieben.

Der Aufbau der Klassen orientiert sich am Aufbau des lokalen „Flooders“ in der ELL in der Hauptklasse *Basin*. Hier ist die zentrale Klasse ein *Flooder*. Der *Flooder* nimmt im Konstruktor die RNA-Sequenz entgegen. Er hat eine Funktion *floodBasin*, welche den minimalen

Zustand für das zu flutende Bassin, einen *StatePairCollector* und einen *StateCollector* entgegen nimmt. Das Berechnen der Zustandssummen für die Kontaktflächen ($Z_{b,c}$) übernimmt indirekt der *StatePairCollector* und das Berechnen der Zustandssumme für das Bassin (Z_b) übernimmt indirekt der *StateCollector*.

Die Klasse *SC_PartitionFunktion* ist vom *StateCollector* abgeleitet und hat eine Funktion *add()*, die einen *State* entgegen nimmt. Daraus wird dann die Energie ermittelt und zur Zustandssumme addiert. Der *StatePairCollector* hat eine Referenz auf eine ganze Liste mit Instanzen der *SC_PartitionFunktion*. Darin werden Zustandssummen für alle Transitionsraten verwaltet. Immer wenn ein benachbarter Microstate x gefunden wird, so wird auch der betreffende Eintrag in der Liste mit den Zustandssummen aktualisiert (bzw. initialisiert, wenn das Bassin $F(x)$ neu ist).

Der *WalkGradientHashed* wird verwendet, um die Zugehörigkeit der Microstates $F(x)$ festzustellen. Das geschieht innerhalb des *StatePairCollector*, bevor die Zustandssumme für eine Transition erhöht wird. Denn bevor das geschieht, muss das benachbarte Bassin ($F(x \notin b)$) identifiziert werden und eine ID bekommen, falls es noch keine hat.

Schritt 1 im Algorithmus wird durch die *PriorityQueue* vereinfacht, welche die To-do-Liste T implementiert. Das ist eine Datenstruktur, die schon beim Einfügen eine Sortierung der Zustände vornimmt. Die Nachbargenerierung im Schritt 2 wird durch die Funktion *browse_Neighbors()* aus der ViennaRNA-Bibliothek realisiert. Diese gibt die Nachbarn inklusive Energieberechnung zurück. Die Interaktionen zwischen den Klassen sind im Sequenzdiagramm 7 dargestellt.

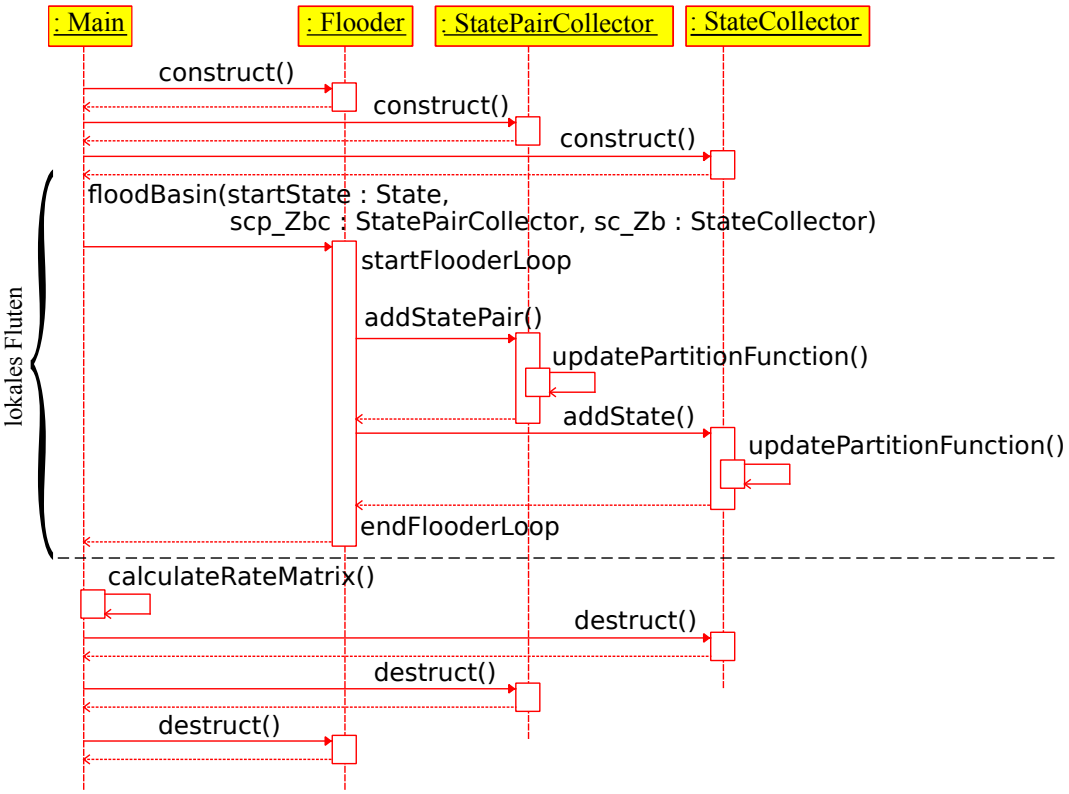


Abbildung 7: Die Implementierung des lokalen Flutalgorithmus ist in diesem Sequenzdiagramm grob skizziert.

4.3 Probleme und Lösungen

Probleme

1. In der ViennaRNA-Bibliothek gibt es eine Funktion *move_gradient*, die einen Gradient-Walk (siehe Def. 8) durchführt. Allerdings verwendet dieser keine lexikographische Ordnung, wenn verschiedene Strukturen gleiche Energien haben. Das führt zu Ergebnissen, die nicht der Definition 2.3 entsprechen.

2. Ein weiteres Problem ist die Neuberechnung schon bekannter Minima und kleinster Nachbarn entlang von Gradient-Walks. Da das lokale Fluten nur einen kleinen Teil der Energielandschaft abdeckt, verlaufen viele Gradient-Walks über die gleichen Zustände. Man könnte deshalb immer, wenn man auf einen schon bekannten Zustand trifft, dessen bekanntes Gradient-Walk Ende nutzen, ohne den Walk abzuschließen. Diese Information wird aber in einer standardmäßigen Implementierung nicht gespeichert. Dadurch wird ein großer Teil der Walks immer wieder neu besucht, woraus längere Laufzeiten resultieren.

Lösungen

1. Der Gradient-Walk wurde neu geschrieben (Klasse WalkGradientHashed, siehe Kap. 4.2) und verwendet die lexikographische Ordnung bei gleichen Energiewerten.

2. Eine mögliche Lösung für das Speichern der lokalen Minima für schon besuchte Zustände ist eine HashMap. Als Hashfunktion kann man den Ternärcode verwenden, der die Zustände in Dot-Bracket-Notation eindeutig darstellt. Allerdings müsste man schon für kurze Sequenzen größere Datentypen erstellen, weil die Integer-Datentypen (32 und 64 Bit) der Standardbibliothek nicht mehr ausreichen würden. Um in dieser Hinsicht flexibel zu bleiben, genügt in diesem Fall eine schnelle Hashfunktion für beliebige Arrays. Diese Implementierung verwendet den SpookyHash[6] von Bob Jenkins, der Hashwerte zwischen 32 und 128 Bit anbietet. Als HashMap wird eine *unordered_map* aus dem c++11 Standard verwendet.

4.4 Optimierungen

Um die Implementierung des lokalen „Flooders“ noch mehr zu beschleunigen, wurde zunächst ein Profiling mit dem Programm **gprof** durchgeführt. Dabei wurde für alle Funktionen die Ausführungszeit, die Anzahl der Aufrufe und der Zeitverbrauch insgesamt in Prozent untersucht.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %
 time        name                          bibliothek
 95.90       get_scaled_parameters         ViennaRNA
  1.15       _mcount_private               gprof
```

Abbildung 8: Ausgabe des Profilings mit gprof. Der „Flooder“ wurde mit der Sequenz „boris1“ verwendet.

[6] `http://burtleburtle.net/bob/hash/spooky.html` letzter Zugriff: 18.11.2014

In der Profilerausgabe (Abbildung 8) sieht man, dass mit 95,9% die meiste Zeit von der Funktion *get_scaled_parameters()* verbraucht wird. Diese Funktion gehört zur ViennaRNA Bibliothek und wird innerhalb der Funktion *browse_neighbors()* aufgerufen, welche die Nachbarschaft einer Struktur generiert. Mit *get_scaled_parameters()* werden neue Parameter für die Energieberechnung geschrieben. Da für die Energieberechnung immer dieselben Energieparameter verwendet werden, wurde eine neue *browse_neighbors()* Funktion erstellt. Diese nimmt Energieparameter direkt entgegen, anstatt sie jedes Mal neu zu generieren.

Inkrementelle Energieberechnung: Die ViennaRNA-Bibliothek verwendet eine inkrementelle Energieberechnung. Dadurch werden beim Hinzufügen oder Entfernen von Basenpaaren nur die Energien für diese Basenpaare zur Gesamtenergie addiert oder subtrahiert. Das kann man auch bei der Berechnung benachbarter Strukturen nutzen. Deshalb wurde die Funktion *browse_neighbors()* so geändert, dass man zur Startstruktur zusätzlich die Energie angibt, um den Vorteil der inkrementellen Berechnung zu nutzen.

4.5 Evaluation

In Kapitel 4.2 wurde erwähnt, dass der ELL-basierte globale Flutalgorithmus langsamer ist, als der ViennaRNA-basierte globale Flutalgorithmus. Deshalb wurde für die Implementierung des lokalen Flutalgorithmus auch die ViennaRNA Bibliothek verwendet.

Zu Testzwecken wurde die neue Implementierung mit einer ELL-basierten Implementierung verglichen. Ein Vergleich mit den RNA-Sequenzen „d33“ und „ire“ (siehe Appendix A.1) hat gezeigt, dass die neue Implementierung des lokalen Flutalgorithmus schneller ist, als die ELL-basierte Implementierung des lokalen Flutalgorithmus. Die neue Implementierung benötigte zum Fluten der offenen Kette jeweils ca. 3 Stunden. Der Test mit der ELL-basierten Version wurde für beide Sequenzen nach 2 Tagen abgebrochen.

5 Exploratives globales Fluten

5.1 Genereller Ansatz

Man kann mit Hilfe des lokalen Flutalgorithmus die globale Ratenmatrix wie folgt erstellen: Das erste lokale Minimum mit entsprechendem Bassin b wird zunächst auf die Liste behandelter Minima D gesetzt. Dann wird der lokale Flutalgorithmus auf b angewendet. Dieser berechnet die Zustandssummen für die Kontaktflächen und das Bassin. Zudem gibt er eine Liste mit benachbarten Minima zurück, zu welchen Transitionen gefunden wurden (Z_{bc}). Diese werden dann auf die To-do-Liste T gesetzt, sofern sie noch nicht in D oder T vorhanden sind. Das lokale Fluten der Bassins wird so oft wiederholt, bis T leer ist.

Die Transitionsraten werden anschließend auf Basis der Zustandssummen berechnet (siehe Macrostate Kinetik Formel 2.4.2).

Der explorative Flutalgorithmus wurde im Programm *ParKin_Explore* umgesetzt. Es berechnet die Transitionsraten und gibt sie auf dem Kommandozeileninterpreter aus.

5.2 Parallelisierung

Der explorative Flutalgorithmus ist parallelisierbar, da das lokale Fluten der Bassins unabhängig von allen anderen geschieht. Deshalb können Bassins in separaten Threads geflutet werden, ohne dass zusätzlicher Kommunikationsaufwand zwischen den Threads betrieben werden muss. Die einfachste Möglichkeit für ein paralleles Fluten ist also die Parallelisierung der Schleife über die To-do-Liste, weil diese immer die aktuellsten, noch nicht gefluteten lokalen Minima, enthält.

Diese Parallelisierung ist auch im Programm *ParKin_Explore* umgesetzt. Dazu wurden zwei verschiedene Ansätze getestet. Eine Implementierung verwendet die c++11 Standardbibliothek **std::thread**, die andere verwendet *OpenMP*. Die Klasse *std::thread* ermöglicht die Parallelisierung dadurch, dass dem Thread beim Anlegen eine Methode mit ihren Parametern gegeben wird. Mit *OpenMP* kann man gezielt Schleifen parallelisieren, indem man compilerspezifische Direktiven angibt. Diese beginnen mit dem Schlüsselwort „#pragma“. Die Anweisung für die Schleife der To-do-Liste des explorativen Algorithmus ist einfach „#pragma omp parallel for num_threads(numThreads)“. Die Variable „numThreads“ gibt dabei die maximale Anzahl der Threads an, die verwendet werden sollen. Zur Kompilierung wird noch das Flag „-fopenmp“ angegeben. Der Code innerhalb der Schleife wird somit parallel ausgeführt. Ein Vorteil der Version mit *OpenMP* ist, dass der Code besser lesbar ist. Zudem wurde auch kein Laufzeitunterschied zur std::thread Implementierung festgestellt. Deshalb wird nur die *OpenMP*-Version weiter entwickelt.

5.2.1 Probleme

1. Die *browse_neighbors()* Funktion der ViennaRNA Bibliothek (2.1.7) ruft für jede einzelne benachbarte Struktur eine benachrichtigende Funktion auf. Diese Funktion muss statisch sein, weil nur solche hierfür zulässig sind. Diese Eigenschaft erschwert die Parallelisierung, weil die Liste der Nachbarn an eine bestimmte Klasse bzw. einen bestimmten Thread gebunden sein sollte.

2. Bei beiden Ansätzen wurde die Laufzeit nicht kürzer, sondern eher länger. Das Betriebssystem für die Entwicklung war ein Windows 8.1. Es wurde mit dem GNU-Compiler (g++ (GCC) 4.8.3[7]) in cygwin (version 1.7.31[8]) kompiliert. Vermutlich werden die parallelen Zugriffe zur Laufzeit vom Speichermanagement des Betriebssystems serialisiert. Dieses Problem wird auch als *Heap Contention* bezeichnet.

[7] https://gcc.gnu.org/ (letzter Zugriff: 20.10.2014)
[8] https://cygwin.com/ (letzter Zugriff: 20.10.2014)

5.2.2 Lösungen

ViennaRNA Bibliothek Änderungen Ein Lösungsversuch des ersten Problems besteht darin, für jeden Thread eine separate statische Liste der Nachbarn zu verwenden und eine statische HashMap. Die HashMap ordnet jeder ThreadID die entsprechende statische Liste zu. Die benachrichtigende Funktion muss für das Einfügen einer Struktur die ThreadID ermitteln und nach der richtigen Liste in der HashMap suchen. Bei der Verwaltung der HashMap muss auch bedacht werden, dass Threads hinzugefügt und gelöscht werden können, während die benachrichtigende Funktion Suchanfragen startet. Das führt innerhalb der HashMap zu Konflikten. Diese können durch das Einfügen von kritischen Regionen um die jeweiligen Zugriffe gelöst werden. Kritische Regionen markieren Abschnitte im Code, die immer nur von genau einem Thread ausgeführt werden können. Die anderen Threads müssen solange warten, bis dieser Thread mit der Region fertig ist. Die praktische Durchführung hat jedoch gezeigt, dass dies die Ausführungsgeschwindigkeit stark verlangsamt.

Ein anderer Lösungsansatz ist das Ändern der *browse_Neighbors()* Funktion in der *ViennaRNA* Bibliothek. Dieser Ansatz wurde so auch für *ParKin_Explore* verwendet. Die neue Funktion gibt eine komplette Liste mit allen benachbarten Strukturen auf einmal zurück, anstatt die Nachbarn an eine benachrichtigende Funktion weiterzureichen.

Heap Contention Das Problem entsteht, wenn mehrere Threads gleichzeitig auf einen Bereich im Speicher zugreifen möchten. Die Bibliothek *OpenMP* benutzt einen Heap Speicher für alle Threads. Je nach *OpenMP* Implementierung von Compilerseite und Zusammenspiel mit der HeapAPI des Betriebssystems werden die Zugriffe auf den Heap parallel oder seriell ausgeführt. Dieses Problem gibt es nicht nur mit dem GNU-Compiler unter Windows, sondern auch mit dem Intelcompiler. Eine Lösung für den Intelcompiler sind separate Befehle für die Speicherallokation mit *OpenMP* für Windows[9]. Die Lösung, einen anderen Compiler zu verwenden kommt aber nicht in Frage, weil der GNU-Compiler weit verbreitet ist und unter Linux ohne Weiteres funktioniert. Andere Befehle zur Speicherallokation sind auch keine gute Lösung, wenn zusätzliche Bibliotheken verwendet werden, die dann ebenfalls angepasst werden müssten.

Das Programm *ParKin_Explore* wurde daher unverändert auf dem Betriebssystem Ubuntu 14.0.4 mit gcc 4.8.2 compiliert. Unter diesen Bedingungen hat sich ein Geschwindigkeitsvorteil der Parallelisierung bemerkbar gemacht. Deswegen wurde diese Entwicklungsumgebung im Folgenden verwendet.

Eine Alternative für die Entwicklung unter Windows wäre die Verwendung von einem anderen Compiler oder einem Heap-Manager (z.B. *HOARD*).

[9] https://software.intel.com/en-us/articles/avoiding-heap-contention-among-threads (letzter Zugriff: 20.10.2014)

5.3 Evaluation

In Abbildung 9 sieht man, dass die Ausführungszeit reduziert wird, wenn mehrere Threads parallel ausgeführt werden. Die Zeit wird bei zwei Threads ca. um 1/3 reduziert. Wenn noch mehr Threads verwendet werden, verändert sich die Ausführungszeit nur noch schwach. Dieser Effekt basiert auf der Art der Parallelisierung. Die benachbarten Bassins werden erst geflutet, wenn das aktuelle Bassin abgearbeitet wurde. Das führt dazu, dass die zwei i.d.R. größten Bassins (das der offenen Kette (OC) und der Struktur mit der kleinsten freien Energie (MFE)) nacheinander geflutet werden, wenn von der offenen Kette ausgehend geflutet wird. Mit einer asynchronen Parallelisierung (wie in Kap. 5.4 diskutiert) könnte man u.a. diese beiden Bassins parallel fluten. Dadurch würde sich die Ausführungszeit schätzungsweise halbieren.

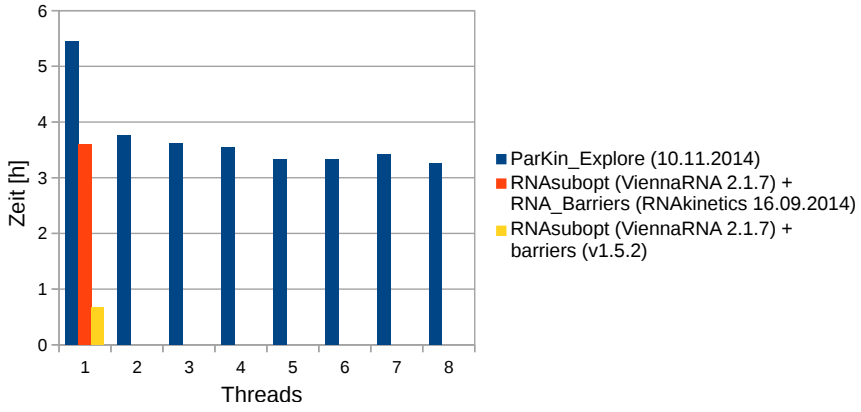


Abbildung 9: Sequenz: „ire“. Die Abbildung zeigt die Ausführungszeit für jedes Programm. Dabei ist zu beachten, dass beide *Barriers* Implementierungen keine eigene Energieberechnung durchführen. Diese wird von RNAsubopt erledigt.

Abbildung 10 zeigt die Ausführungszeit und Größe der To-do-Liste in jeder Iteration. Man könnte denken, dass die Ausführung der dritten Iteration am meisten Zeit benötigt, weil die To-do-Liste dort am größten ist. Wenn man aber die kumulativen Bassingrößen pro Iteration in Abbildung 11 mit der Anzahl der Bassins in Abbildung 10 rechts vergleicht, ist es offensichtlich, dass die Anzahl der Bassins nicht verantwortlich für den Zeitverbrauch ist, sondern die Größe der Bassins bzw. die prozessierten Zustände.

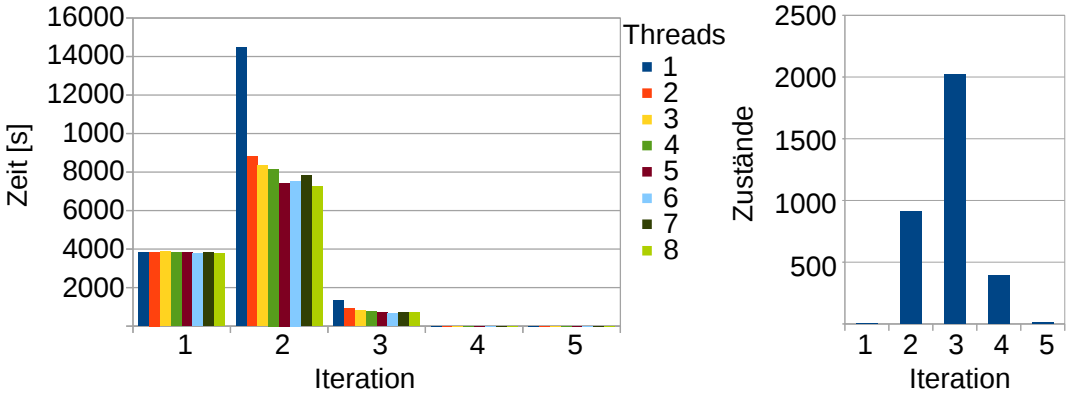


Abbildung 10: Sequenz: „ire“. Die linke Abbildung zeigt den Zeitverbrauch für die To-do-Liste in jeder Iteration von *ParKin_Explore*. Die rechte Abbildung zeigt die Anzahl der Minima in der To-do-Liste in jeder Iteration.

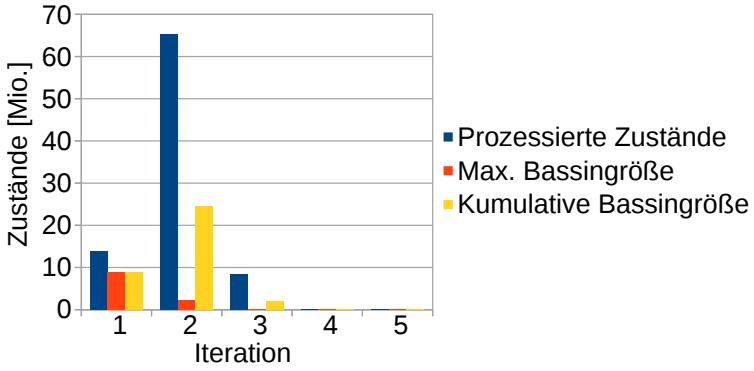


Abbildung 11: Sequenz: „ire“. Die gelben Balken zeigen die kumulativen Bassingrößen in jeder Iteration. Die roten Balken zeigen die Menge der Microstates des größten Bassins in jeder Iteration. Dies repräsentiert die untere Schranke für den Zeitverbrauch von *ParKin_Explore* im parallelen Betrieb. Die blauen Balken zeigen die Gesamtsumme der prozessierten Zustände, die in *ParKin_Explore* betrachtet werden.

Eine untere Grenze für den Zeitverbrauch ist das Bassin mit der maximalen Größe in der To-do-Liste bei jeder Iteration. Die gesamte Laufzeit kann nicht schneller sein als die Summe der Laufzeiten für jedes maximale Bassin in der To-do-Liste.

5.4 Diskussion

In Abbildung 9 fällt auf, dass *ParKin_Explore* im Vergleich zu *Barriers* achtmal soviel Zeit benötigt, um die komplette Energielandschaft zu fluten. Im parallelen Betrieb mit *ParKin_Explore* dauert es viermal so lange. Der entscheidende Grund dafür ist, dass *Barriers* keine Energieberechnungen durchführt. Die Energien aller Strukturen werden von *RNAsubopt* berechnet und vollständig im Speicher gehalten. *ParKin_Explore* berechnet dagegen, um Speicher zu sparen, die Energien immer wieder neu, weil dies durch die *browse_neighbors()* Funktion des ViennaRNA-Pakets automatisch geschieht. Dieser Nachteil wird auch durch das Profiling mit *gprof* deutlich (siehe Abbildung 12). Darin sieht man, dass die Energieberechnung 50% der gesamten Laufzeit einnimmt.

```
Flat profile:

Each sample counts as 0.01 seconds.
  %
 time     name                                                bibliothek
 17.69    loop_energy                                         ViennaRNA
  9.49    get_list.6314                                       ViennaRNA
  9.09    StructureUtils::IsEqual(short*, short*)             ParKin_Explore
  6.53    move_set                                            ViennaRNA
  4.92    update_deepest                                      ViennaRNA
  4.90    energy_of_extLoop_pt                                ViennaRNA
  4.66    energy_of_move_pt                                   ViennaRNA
  3.78    StructureUtils::IsSmaller(short*, short*)           ParKin_Explore
  2.47    energy_of_ml_pt                                     ViennaRNA
```

Abbildung 12: Ausgabe des Profiling mit *gprof*. *ParKin_Explore* vom 07.11.2014 wurde mit der Sequenz „d33“ verwendet.

Ein weiterer Vorteil von *Barriers* ist, dass Zustände der Kontaktflächen nicht mehrfach aufgezählt werden. Das lässt sich bei *ParKin_Explore* nicht vermeiden, ohne den Algorithmus zu verändern. Allerdings wird dieser Nachteil durch die Parallelisierung kompensiert. Durch die zusätzlichen Energieberechnungen ist das in dieser Version jedoch nicht wesentlich bemerkbar. Die Abbildung 13 zeigt den Mehraufwand durch zusätzlich prozessierte Zustände.

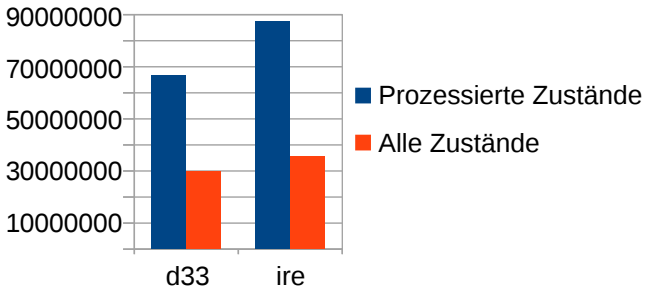


Abbildung 13: Sequenzen: „d33“, „ire“. Die Abbildung zeigt die prozessierten Zustände, die in *ParKin_Explore* betrachtet werden, im Vergleich zu den gesamten Zuständen aller Bassins zusammen.

Durch den Mehraufwand von doppelt so vielen Zuständen sowie der nötigen Energieneuberechnung kann die Laufzeit von *Barriers* mit *ParKin_Explore* nicht erreicht werden, sondern dient nur als untere Schranke.

Die Laufzeit von *ParKin_Explore* wird durch die größten Bassins dominiert. Die zwei größten Bassins sind in der Regel das der offenen Kette (OC) und das der Struktur mit der kleinsten freien Energie (MFE). Für die Sequenz d33 sind das 8.502.796 Zustände für die OC und 4.304.156 Zustände für die MFE bei insgesamt 29.759.371 Zuständen. Somit beanspruchen diese beiden Bassins zusammen 43% der Energielandschaft. Daher wird ab zwei Threads die Laufzeit durch die Strukturen OC und MFE dominiert, da sie nacheinander prozessiert werden (vgl. Abbildung 10). Die restlichen Strukturen verteilen sich auf die Threads. Der Nachteil der aktuellen iterativen Parallelisierung besteht darin, dass gerade die größten Bassins nicht gleichzeitig geflutet werden.

Ein alternativer asynchroner Ansatz für die Parallelisierung könnte für eine zukünftige Implementierung interessant sein. Dabei würde es einen verteilenden Masterthread geben, der die To-do-Liste verwaltet und die Anzahl der Threads. Er teilt den Threads, die nur für das lokale Fluten zuständig sind, die lokalen Minima mit. Die flutenden Threads würden beim Entdecken eines neuen lokalen Minimums sofort eine Nachricht an den Masterthread schicken. Dieser kann dann entscheiden, ob ein neuer flutender Thread gestartet werden soll oder nicht. Weil der Schwerpunkt dieser Arbeit nicht auf der Laufzeitoptimierung liegt, wird der Ansatz in dieser Arbeit nicht weiter verfolgt.

6 Eingeschränktes globales Fluten

Beim eingeschränkten globalen Fluten wird nur ein Teil der Energielandschaft zur Berechnung der Ratenmatrix verwendet. Energielandschaften wachsen exponentiell mit der Sequenzlänge (siehe Kap. 2.4.1). Da die Energielandschaft von der Sequenzlänge direkt abhängig ist, steigt auch der Ressourcenverbrauch der Ratenmatrixberechnung mittels globalem oder explorativem Fluten. Um dies zu kompensieren, kann man den explorativen Flutalgorithmus zur Berechnung der Ratenmatrix mit Filtern verwenden, um für lange Sequenzen in kurzer Zeit einen groben Einblick in die Kinetik zu bekommen. Hierbei kann in Micro- und Macrostate-Filter unterschieden werden. Microstate-Filter bewirken, dass während des lokalen Flutens Microstates mit geringer Bedeutung nicht in die Berechnung miteinbezogen werden. Macrostate-Filter hingegen ermöglichen das Auslassen von ganzen Bassins für die Ratenberechnung. Durch geschickte Wahl der Filterparameter kann man eine relativ aussagekräftige Kinetik in kurzer Zeit berechnen. Verschiedene lokale Macrostate-Filter wurden in [14] untersucht. Im Folgenden werden die wichtigsten Micro- und Macrostate-Filter vorgestellt.

6.1 Microstate-Filter

Die Microstate-Filter arbeiten auf einem sehr detaillierten Level der Energielandschaft. In der Implementierung findet man diese Filter in der Klasse *Flooder* (siehe Kap. 4.2), welche den lokalen Flutalgorithmus implementiert (siehe Kap. 4.1). Dort wird für jeden Microstate entschieden, ob er in die Berechnung eingehen soll oder nicht. In den hier untersuchten Fällen wird dabei geprüft, ob die Energie unterhalb des aktuellen Flutlevels ist. Das Flutlevel ist hierbei die derzeitig erlaubte obere Energieschranke. Beim (im Folgenden dargestellten) Delta-Energie-Filter wird das Flutlevel für jedes lokale Minimum angepasst, beim Maximale-Energie-Filter ist es für die gesamte Exploration ein fester Wert. Ein kurzer Eindruck über die Funktionsweise soll durch Abbildung 14 vermittelt werden.

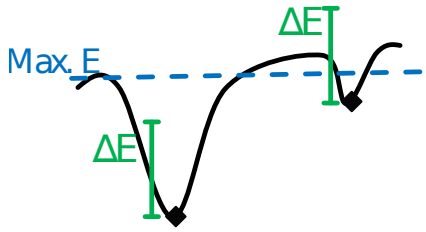


Abbildung 14: Funktion des Delta-Energie und Max. Energie Filters.

Für die Berechnung einer Macrostate-Kinetik gilt es bei den Zustandssummen für Transitionen zu beachten, dass die Summe für beide Richtungen verschieden sein kann, wenn Microstates entfernt werden (d.h. $Z_{bc} \neq Z_{cb}$ entgegen der Theorie in Formel 12). Deshalb sollte man für die Ratenberechnung immer die größere Zustandssumme verwenden, um ein genaueres Ergebnis zu erhalten. Zudem sind auch die Zustandssummen nur untere Schranken, wenn Microstates weggelassen wurden, was auch Einfluss auf die Kinetik haben kann.

Beim Anwenden der Filter werden u.U. weniger Transitionen und weniger Macrostates gefunden. Es sind jedoch auf jeden Fall alle Macrostates miteinander verbunden (ähnlich zur Lid-Methode (siehe Kap. 3)).

6.1.1 Maximale-Energie-Filter

Der Maximale-Energie-Filter ist ein Mikrozustandsfilter, bei dem ein globaler maximaler Energiewert als Flutlevel definiert wird. Für die Berechnung werden somit nur Zustände mit einer Energie verwendet, die unterhalb dieser globalen Schranke liegen.

Dieser Filter ist in der Literatur allgemein bekannt. Er wird sowohl in der *Lid*-Methode [33] als auch von *Barriers* [9] angeboten.

In Abbildung 15 sieht man, dass dieser Filter eine Kinetik erzeugt, die der ungefilterten Kinetik ähnelt.

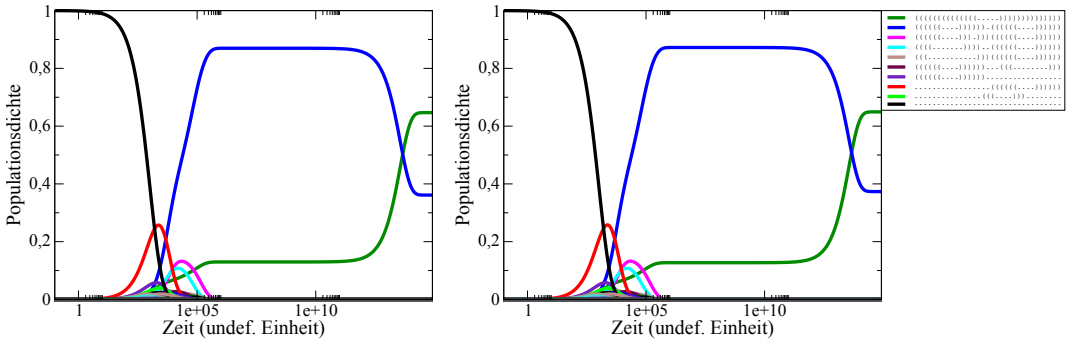


Abbildung 15: Vergleich von zwei Kinetiken der Sequenz d33 mit Maximale-Energie-Filter. Links ungefiltert und rechts mit 10 kcal/mol als Schwellenwert.

Der Maximale-Energie-Filter erzeugt in Verbindung mit dem explorativen Ansatz, wie die *Lid*-Methode, den zusammenhängenden Teil der Energielandschaft bzgl. maximalem Energieniveau und Startpunkt. Das Programm *Barriers* dagegen zählt, je nach maximalem Energieniveau, unzusammenhängende Teile der Energielandschaft auf, die anschließend noch verbunden werden müssen.

6.1.2 Delta-Energie-Filter

Der Delta-Energie-Filter betrachtet nur Zustände in einem bestimmten Energiebereich in Relation zu der Energie des aktuellen Minimums, das geflutet wird. Kleinere Bassins werden dadurch etwas bedeutender in der Kinetik, da die Zustandssummen der großen Bassins kleiner werden, aber die groben Verhältnisse sollten immer noch bestehen. In Abbildung 16 sind die Kinetiken für die Sequenz „d33“ mit dem Delta-Energie-Filter dargestellt. Die Kurven der wahrscheinlichsten Zustände verlaufen in beiden Kinetiken ähnlich, wobei die Endwahrscheinlichkeiten leicht abweichen.

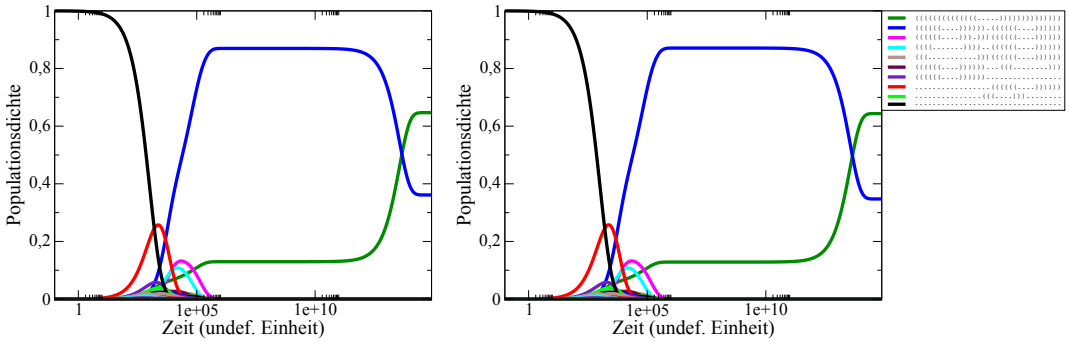


Abbildung 16: Vergleich von zwei Kinetiken der Sequenz d33 mit Delta-Energie-Filter. Links ungefiltert und rechts mit 10 kcal/mol Energiedifferenz.

6.1.3 Vergleich Delta-Energie und Maximale-Energie-Filter

Die beiden Microstate-Filter reduzieren die Ausführungszeit für die Berechnung der Kinetik und erzeugen in kurzer Zeit vergleichbare Kinetiken. In Abbildung 17 sieht man die zeitliche Auswirkung des Delta-Energie-Filters und Maximale-Energie-Filters. Bemerkenswert ist, dass man für die gegebenen Filterparameter in weniger als 5 Minuten eine Kinetik erhält, die mit einer Kinetik vergleichbar ist, die in mehr als 3 Stunden berechnet wurde.

Die Laufzeitreduktion auf ca. 3% korreliert direkt mit der Reduktion der betrachteten Zustände. Durch geeignete Parameter sinkt die Anzahl der Zustände sehr schnell. Dies liegt an der Verteilung der Strukturen im Energiebereich, der Zustandsdichte. Für RNA liegen die meisten Strukturen im hochenergetischen Bereich (> 0).

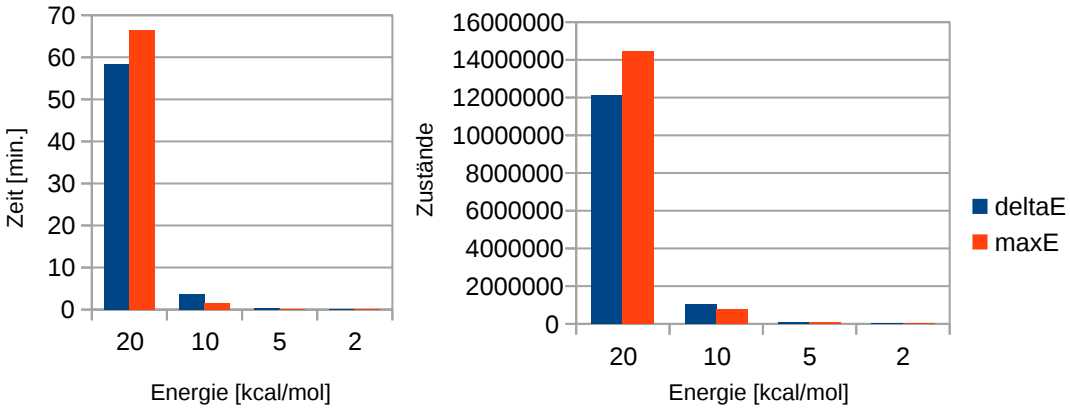


Abbildung 17: Die Abbildung links zeigt die Zeiten für verschiedene Parameter des Delta-Energie Filters und Maximale-Energie-Filters mit 8 Threads und der Sequenz „d33“. Die Abbildung rechts zeigt die Anzahl der Microstates, die für eine bestimmte Delta-Energie noch in der Kinetik betrachtet wurden. Der Maximale-Energie-Filter ist restriktiver, was die Anzahl der Zustände betrifft.

In Abbildung 18 ist die Zustandsdichte innerhalb des Energiebereiches der Energielandschaft für die Sequenz „d33“ dargestellt. Die gestrichelten vertikalen Linien markieren den Schwellenwert des Maximalen-Energie Filters. Dieser Wert entspricht der Differenz des Delta-Energie-Filters. Man sieht in der Grafik, dass der Delta-Energie-Filter mehr Zustände betrachtet, als der Maximale-Energie-Filter. Das lässt sich dadurch erklären, dass das maximale Flutlevel für jedes lokale Minimum separat angepasst wird. Somit arbeitet sich der Delta-Energie-Filter „treppensteigend“ durch die Energielandschaft. Allerdings zählt der Delta-Energie-Filter weniger Zustände für $E < 0$ auf, was durch das beschränkte Aufzählen tiefer Bassins (wie der MFE) zu erklären ist.

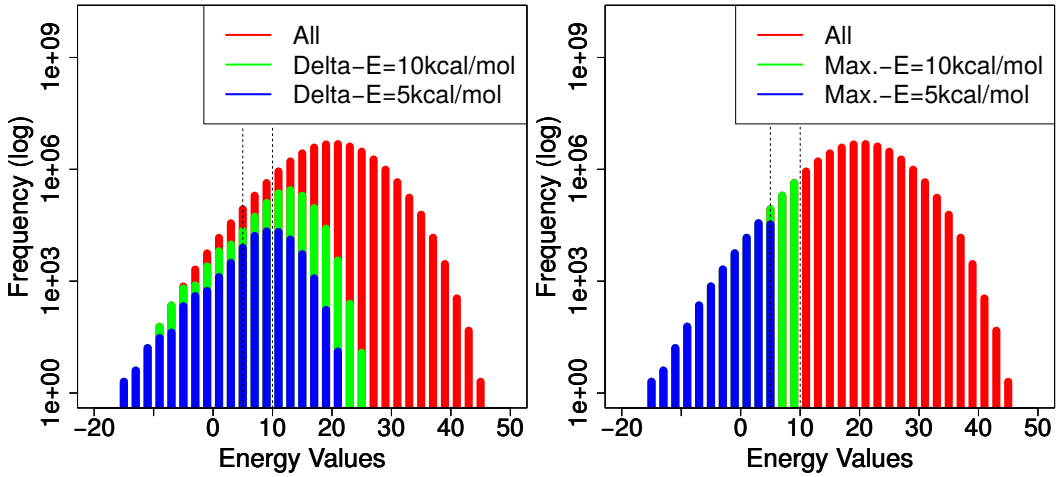


Abbildung 18: Vergleich der Zustandsdichte der Sequenz d33 mit Delta-Energie-Filter und Maximale-Energie-Filter. Die ungefilterte Zustandsdichte ist rot dargestellt, die gefilterte Zustandsdichte mit 10 kcal/mol ist grün und die gefilterte Zustandsdichte mit 5 kcal/mol ist blau. Der prozentuale Anteil an Zuständen ist links jeweils: 100%, 3,49%, 0,31% und rechts: 100%, 2,59%, 0,31%. Zu beachten ist, dass die Anzahl der Zustände (Zustandsdichte) logarithmisch angegeben ist.

6.2 Macrostate-Filter

Nachdem die Microstate-Filter primär die Fluthöhe und damit die Präzision der Raten reduziert haben, kann die Anzahl der für die Ratenmatrix betrachteten Macrostates mittels Macrostate-Filtern reduziert werden. Die Ratenmatrixreduktion beschleunigt die Populationsdichtenberechnung der Kinetik.

Macrostate-Filter können eingesetzt werden, um komplette Bassins aus den energetisch höheren Bereichen der Energielandschaft auszuschließen. Die Kinetik wird somit nur für die wichtigsten Macrostates mit den kleinsten Energien berechnet.

Zwei wichtige Macrostate-Filter sind der K-Best-Filter und der DeltaMinE-Filter. Diese beiden Filter wurden auch in *ParKin_Explore* implementiert. Sie werden in diesem Kapitel zunächst genau erläutert. Anschließend wird die Zeiteinsparung bei der Ratenmatrixberechnung unter Verwendung dieser Filter analysiert.

6.2.1 K-Best-Filter

Der K-Best-Filter betrachtet für jedes Bassin nur die K-benachbarten Bassins mit den größten Transitionsraten. Alle anderen Bassins werden zunächst nicht geflutet.

Der K-Best-Filter setzt, wie alle Macrostate-Filter, beim explorativen globalen Fluten an (siehe Kap. 5). Bevor die To-do-Liste aktualisiert wird, werden die im aktuellen Schritt neu gefundenen Minima gefiltert. Es werden nur die gefilterten Minima auf die To-do-Liste gesetzt (wenn sie nicht schon auf dieser Liste sind). Das Filtern geschieht also immer in Relation zum aktuellen Macrostate. Das bedeutet, dass immer die benachbarten Macrostates des aktuellen Macrostates gefiltert werden.

In [14] wurde gezeigt, dass für ein sehr kleines K immer noch die wichtigsten Zustände (states of interest, kurz SOI) in der Kinetik enthalten sind. SOI sind Macrostates mit einer Populationsdichte, die oberhalb einer gegebenen Schranke liegt (z.B. 10%). Diese Schranke kann willkürlich definiert werden.

Die Wirkungsweise des K-Best-Filters ist in Abbildung 19 dargestellt.

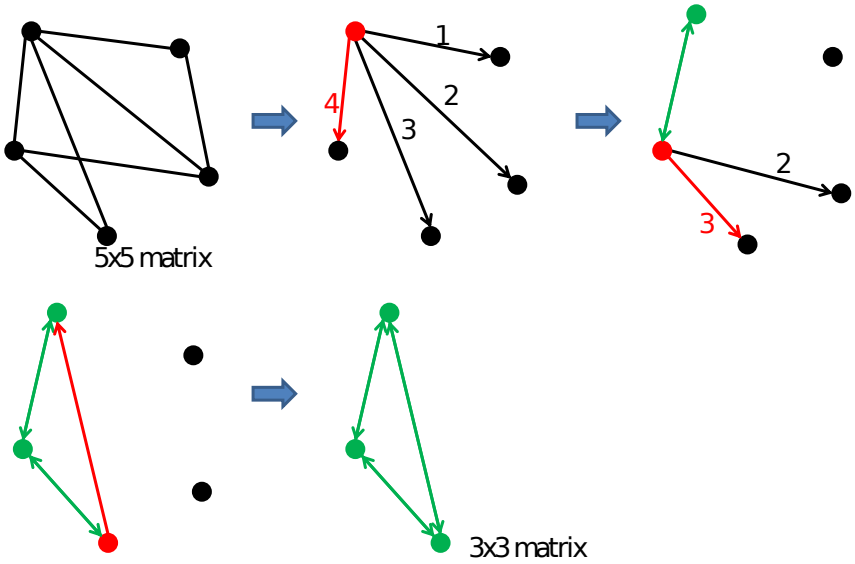


Abbildung 19: Wirkungsweise des K-Best-Filters: Oben links ist ein bidirektionaler Graph mit fünf Knoten dargestellt. Jeder Knoten entspricht einem Macrostate. Der Graph repräsentiert die Verbindungen der Energielandschaft und lässt sich in einer 5x5 Matrix darstellen. In diesem Beispiel ist K=1. Im zweiten Bild wird also nur die rote Kante mit der besten Rate verfolgt. Das Bassin, in dem die rote Kante endet, wird auf die To-do-Liste gesetzt. Die schwarzen Kanten werden ignoriert. Im dritten Bild wird die rote Kante grün dargestellt, weil die Transitionsrate berechnet wurde. Das Bassin am Ende der Transition wird jetzt geflutet. Die beste Transition des aktuellen Zustands (mit Rate 3) ist wieder rot markiert. Im Bild links unten wird auch das Bassin am Ende dieser Transition geflutet, die Rate zurück eingefügt und die beste Transition hinzugefügt. Im letzten Bild sieht man alle erreichbaren Zustände. Dieser zusammenhängende Graph kann in einer 3x3 Matrix gespeichert werden. Durch die Anwendung des Filters mussten also zwei Bassins weniger geflutet werden.

6.2.2 DeltaMinE-Filter

Der DeltaMinE-Filter filtert benachbarte Bassins auf Basis der minimalen Energie im Bassin, das aktuell geflutet wird. In [14] wird dieser Filter „Up-Rate Filter“ genannt. Hier wird er treffender als „DeltaMinE-Filter“ bezeichnet, weil die maximal erlaubte Höhe des Nachbarbassinminimums auch kleiner als die Energie des aktuell betrachteten lokalen Minimums sein kann. In der Implementierung des explorativen lokalen Flutalgorithmus werden nur benachbarte Macrostates zur To-do-Liste hinzugefügt, welche die formale Definition erfüllen:

Definition 9 (DeltaMinE-Filter)
Sei b das aktuelle Bassin und c das benachbarte Bassin und die minimale Energie im Bassin b sei
$E_{min(b)} = \min_{x \in b} E(x)$,
dann gilt:
$E_{min(c)} \leq E_{min(b)} + \Delta_E$

Das Beispiel in Abbildung 20 verdeutlicht die Wirkungsweise des DeltaMinE-Filters.

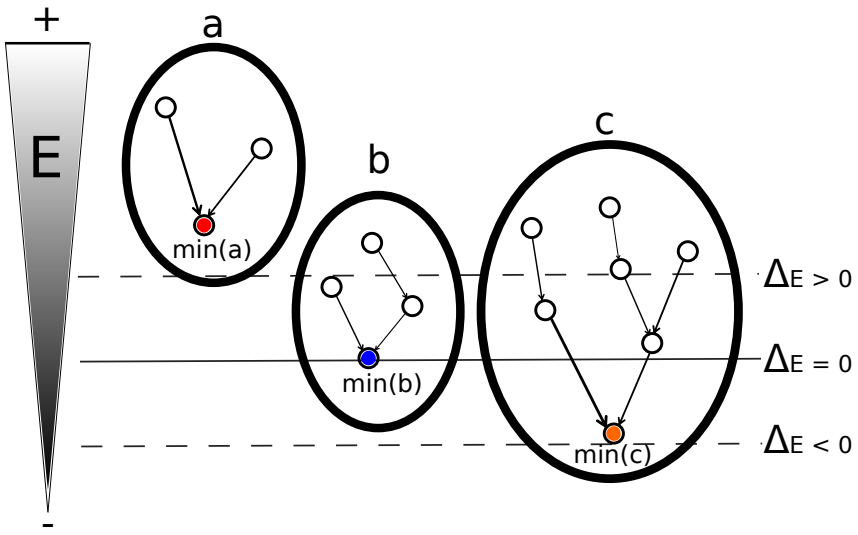


Abbildung 20: Wirkungsweise des DeltaMinE Filters: Die großen Ellipsen symbolisieren die Bassins, die kleinen Kreise sind die Microstates. Die oberen Microstates haben eine größere freie Energie, die unteren Microstates haben eine niedrigere freie Energie. Die minimale Struktur des aktuellen Bassins b ist min_b. Die horizontale Linie markiert das Energieniveau dieser Struktur. Für $\Delta_E = 0$ markiert die durchgezogene Linie die obere Schranke. Es wird nur noch das Bassin c geflutet. Für die gestrichelte Linie mit $\Delta_E < 0$ würde das Bassin c nicht mehr geflutet werden. Das Bassin a wird für die eingezeichneten Schranken nicht geflutet.

6.3 Filterkombination

Die Auswirkung verschiedener Macrostate-Filter auf die Kinetik wurde schon in der Masterarbeit [14] analysiert. Dort wurde festgestellt, wie stark die Auswirkungen des K-Best-Filters und des DeltaMinE-Filters sind. Für einen Wert von DeltaMinE=1 sind zum Beispiel für die Sequenz „d33“ nur noch 39% der Bassins erreichbar (siehe [14] S.56). In Kombination mit dem K-Best-Filter mit K=3 lässt sich die Anzahl der Bassins auf ca. 2% reduzieren. Für diese Filterkombination sind noch alle Zustände aus der ungefilterten Kinetik enthalten, die eine Populationsdichte größer oder gleich 0,2 aufweisen.

Im Folgenden werden die Auswirkungen der Filter auf die Kinetik, die Laufzeit und die Zustandsdichte untersucht. Dafür wurden verschiedene Kombinationen getestet. Um die Laufzeit für Sequenzen mit großen Bassins stärker zu reduzieren, wurde der Delta-Energie-Filter mit 10 kcal/mol bei diesen Tests hinzugefügt.

Beim Vergleich der Laufzeiten mit DeltaMinE- und K-Best-Filter in Abbildung 21 sieht man, dass die Filter zunächst nur eine kleine Geschwindigkeitszunahme ermöglichen. Durch die Kombination der beiden Filter wird die Ausführungszeit jedoch erheblich kürzer. Für dieses Beispiel wird die Ausführungszeit von 3,5 Minuten auf unter 5 Sekunden reduziert.

Die kurze Ausführungszeit mit der Kombination beider Filter ergibt sich dadurch, dass der DeltaMinE-Filter die Eingabe für den K-Best-Filter reduziert. Der K-Best-Filter führt für jede Transition zwischen Macrostates eine Sortierung aus. Nachdem der DeltaMinE-Filter angewendet wurde, ist die zu sortierende Liste viel kleiner.

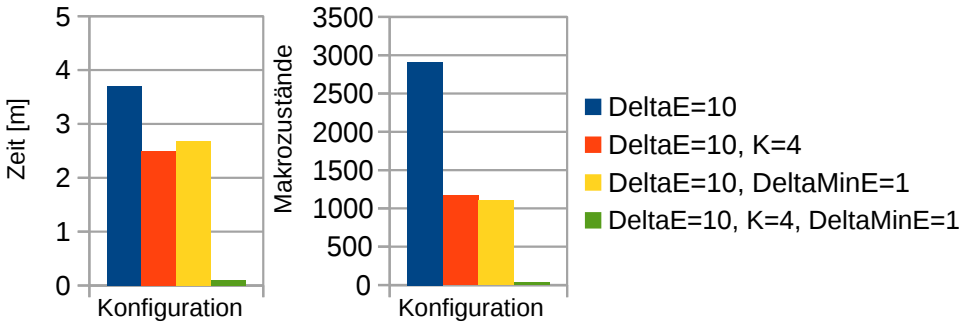


Abbildung 21: Links: Ausführungszeiten für die Filterkombinationen mit der Sequenz „d33“ und 8 Threads. Rechts: Anzahl der Macrostates in der Ratenmatrix.

Die Zustandsdichte für den DeltaMinE-Filter und den K-Best-Filter ist in Abbildung 22 dargestellt. Man erkennt, dass die Filter unterschiedlich wirken. Es sind aber in beiden Fällen noch relativ viele Zustände im Bereich von -10 kcal/mol bis +10 kcal/mol.

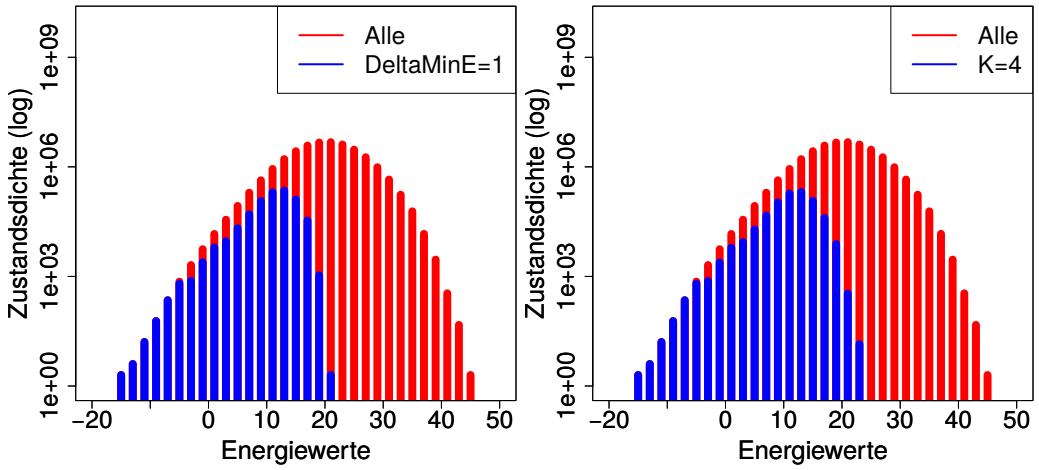


Abbildung 22: Zustandsdichte für die Sequenz „d33“. Links mit DeltaMinE-Filter=1, rechts mit K-Best-Filter=4.

In Abbildung 23 sieht man die Zustandsdichte für die Filterkombination im Vergleich zur ungefilterten Version (rot) und zum Delta-Energie-Filter (grün). Die Filterkombination hat weniger Zustände im Bereich von -10 kcal/mol bis +10 kcal/mol und reicht nicht mehr so weit in den Bereich mit höheren Energien, sondern nur noch bis ca. 11 kcal/mol.

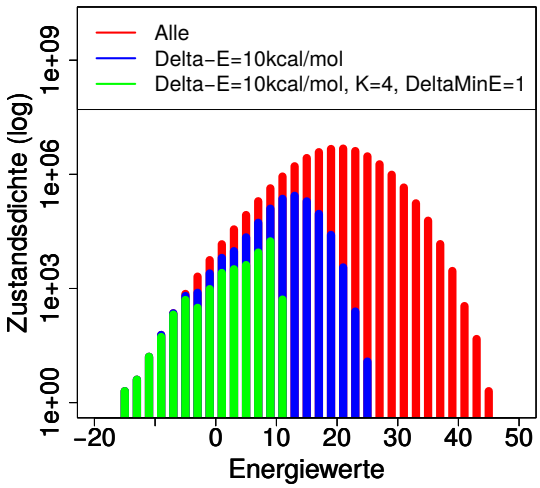


Abbildung 23: Zustandsdichte für die Sequenz „d33“ ohne Filter (rot), mit Delta-Energie-Filter (blau), mit Delta-Energie, DeltaMinE und K-Best-Filter (grün).

Abbildung 24 zeigt die Kinetiken für die vier Filterkombinationen. Diese sehen sich sehr ähnlich, obwohl die vierte Kinetik nur 35 Macrostates enthält und in weniger als einer halben Minute berechnet wurde.

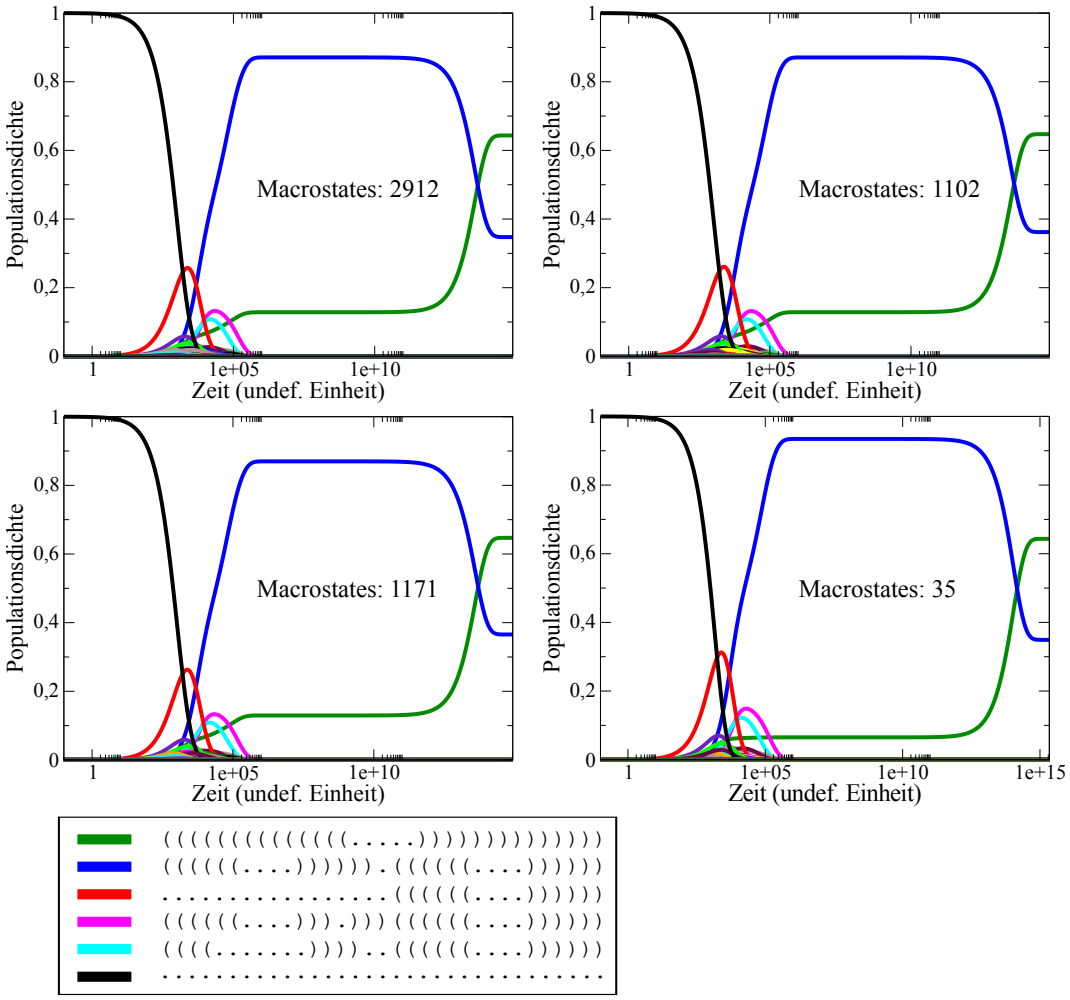


Abbildung 24: Kinetiken der Sequenz „d33“ mit verschiedenen Filterkombinationen, die ähnlich aussehen. Bei allen Kinetiken wurde der Delta-Energie-Filter mit 10 kcal/mol verwendet. Anschließend von links oben nach rechts unten: kein weiterer Filter, DeltaMinE-Filter=1, K-Best-Filter=4, K-Best-Filter=4 mit DeltaMinE-Filter=1.

Bei der Anwendung des K-Best-Filters ist es nicht unwahrscheinlich, dass auch wichtige Zustände, wie z.B die MFE (die Struktur mit der kleinsten freien Energie), entfernt werden. Wenn man zum Beispiel die Sequenz „d33“ mit K=2 filtert, verschlechtert sich die Kinetik stark (siehe Abbildung 25).

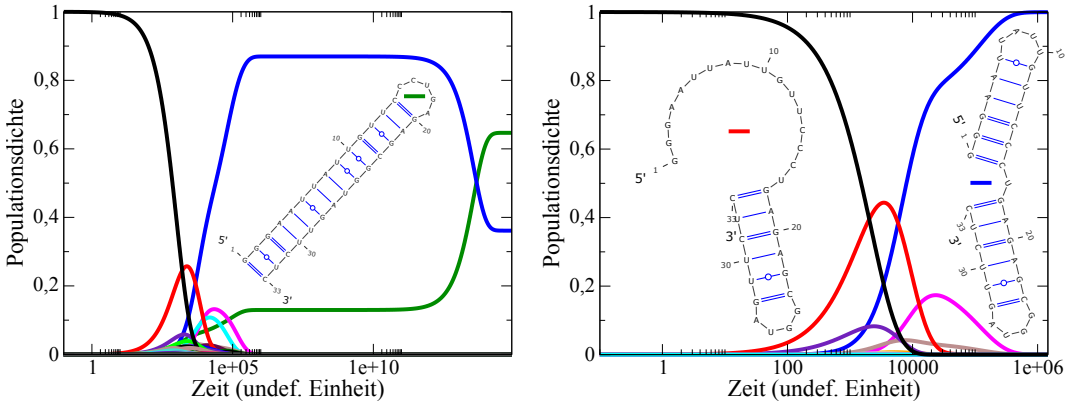


Abbildung 25: Vergleich von zwei Kinetiken der Sequenz „d33“ mit K-Best-Filter. Links ungefiltert und rechts mit K=2. Die MFE (grün) ist nur links enthalten.

Das Programm *ParKin_Explore* soll deshalb um einen Test erweitert werden, um zu bestimmen ob die MFE in der Kinetik enthalten ist. Der K-Best-Filter wird daraufhin automatisch iterativ erhöht, bis die gewünschte Struktur in der Kinetik erscheint. Der Lösungsweg wird im nächsten Kapitel detailliert dargestellt.

6.4 Dynamischer K-Best-Filter

Der dynamische K-Best-Filter erhöht das K automatisch, wenn die MFE nicht in der Kinetik enthalten ist. Das geschieht iterativ so lange, bis die MFE gefunden wurde oder alle erreichbaren Bassins geflutet wurden (wenn andere Filter das Finden der MFE verhindern).

Die IDs der Macrostates werden in einer globalen Liste mit den IDs ihrer zugehörigen Nachbarn verwaltet. Die benachbarten IDs der Macrostates sind nach ihren Energien sortiert. Bevor der explorative Flutalgorithmus mit dem erhöhten K ausgeführt wird, wird für alle Macrostates der Nachbar an Position K+1 ermittelt und auf die To-Do-Liste gesetzt. In der Iteration mit K+1 wird somit die Nachbarschaft der bisher gefundenen Macrostates mit jeweils einem Bassin mehr geflutet. Die Nachbarn der durch das Fluten gefundenen Bassins werden mit dem K-Best-Filter mit K=K+1 gefiltert.

Dadurch erhält man die gleiche Ratenmatrix, die man auch durch direktes Verwenden des K+1 Filters erreicht hätte. Durch den dynamischen Filter kann man sich jedoch sicher sein, dass die MFE gefunden wurde. Die Laufzeit des dynamischen K-Best-Filters ist durch die weitere Liste nur unwesentlich langsamer, als die des K-Best-Filters. Der dynamische K-Best-Filter ist aber wesentlich schneller, als das iterative Ausführen von mehreren K-Best-Filtern bis man die MFE gefunden hat.

In Abbildung 26 ist eine Kinetik mit dem dynamischen K-Best-Filter dargestellt. Die Berechnung mit dynamischem K-Best-Filter benötigte 2,6 Minuten mit 8 Threads. Die gleiche Kinetik mit vorher bekanntem K=4 benötigte 2,4 Minuten. Dieses Beispiel zeigt, dass man mit minimalem Mehraufwand das K automatisch ermitteln kann, anstatt es durch mehrere Versuche mit dem einfachen K-Best-Filter zu erraten.

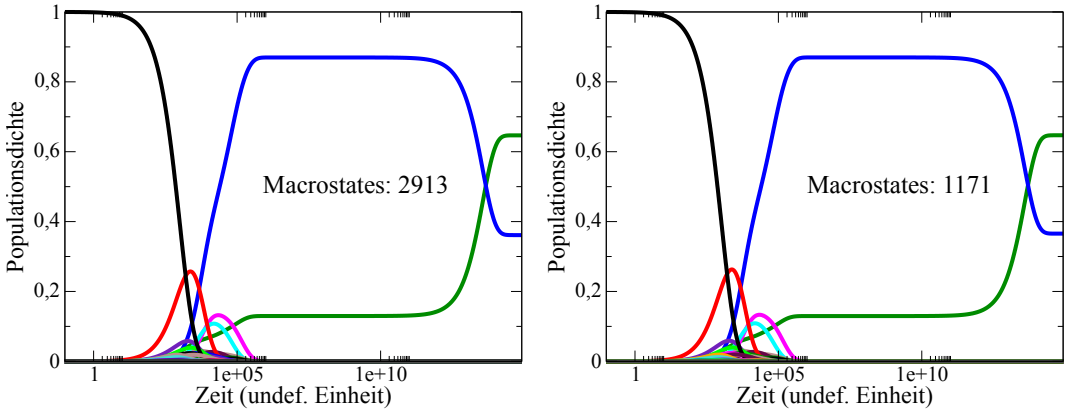


Abbildung 26: Vergleich von zwei Kinetiken der Sequenz „d33“ mit dynamischem K-Best-Filter. Links ungefiltert und rechts mit dynamischem K-Best (gestartet bei K=1) und deltaE=10. Die MFE (grün) wurde links bei K=4 erreicht.

In der Implementierung wird die MFE mit Hilfe des ViennaRNA Pakets ermittelt, wenn das Flag für den dynamischen K-Best-Filter gesetzt ist. Die Schleife, in der die zu flutenden Minima auf der To-do-Liste abgearbeitet werden, ist nun von einer weiteren Schleife umschlossen. In dieser äußeren Schleife wird das K nach jedem Durchlauf erhöht, wenn die MFE noch nicht in den Minima der Done-Liste gefunden wurde oder wenn die To-do-Liste leer ist. Die globale Liste mit den Bassin-IDs und ihren Nachbarn wird nur im ersten Durchlauf erstellt. In den weiteren Durchläufen werden nur die Nachbarn an Position K abgerufen und auf die To-do-Liste gesetzt.

6.5 Vergleichbarkeit von Kinetiken

Es gibt verschiedene Möglichkeiten, ungefilterte Kinetiken mit gefilterten Kinetiken zu vergleichen, jedoch keine Standardmethode. Eine Möglichkeit besteht darin, die Populationsdichten der wichtigsten Macrostates über die Zeit optisch zu vergleichen. Dies wurde hier durch die Abbildungen der Kinetiken veranschaulicht.

Ein festes Maß zum Vergleich ist die Zustandssumme Z der gesamten Energielandschaft. Das Programm *RNAfold* des ViennaRNA-Pakets gibt die freie Energie des Ensembles aus. Diese wird mit Formel 13 berechnet:

$$E(Ensemble) = -RT * ln(Z) \tag{13}$$

Die Zustandssumme ergibt sich durch

$$Z = e^{-E(Ensemble)/RT} \tag{14}$$

Eine andere Möglichkeit zum Vergleich gefilterter Kinetiken sind die Basenpaarwahrscheinlichkeiten für alle Basenpaare in der Energielandschaft. Die Wahrscheinlichkeit für ein Basenpaar (i, j) ist:

$$Pr(i,j) = \frac{\sum\limits_{\substack{x \in \chi_p \\ (i,j) \in x}} e^{-E(x)/RT}}{Z} \tag{15}$$

Diese Wahrscheinlichkeiten können mit dem McCaskill-Algorithmus [28] effizient berechnet werden. Eine übersichtliche Darstellung für die Basenpaarwahrscheinlichkeiten sind DotPlots. Die Wahrscheinlichkeiten und DotPlots können ebenfalls mit dem Programm *RNAfold* berechnet werden.

Diese Methoden wurden für die Sequenzen „d33“ und „Leptomonas“ angewendet, um die Aussagekraft der (mit ParKin_Explore) gefilterten Kinetiken in Abbildung 24 (rechts unten) und Abbildung 33 (unten) zu überprüfen.

Bei den gefilterten Kinetiken werden weniger Strukturen betrachtet. Deshalb gilt für die Basenpaare in den gefilterten Kinetiken $Pr'(i, j) \leq Pr(i, j)$ ($Pr(i, j)$ ist die Wahrscheinlichkeit für ein Basenpaar in der ungefilterten Kinetik. Für die Zustandssummen Z_{approx} aus der gefilterten und Z aus der ungefilterten Kinetik gilt $Z_{approx} \leq Z$.

Die Tabelle 1 zeigt den Vergleich der Zustandssummen für die oben genannten Kinetiken. Die Zustandssummen für die Kinetiken mit „d33“ unterscheiden sich (wie auch die Kinetiken) nur gering. Für die Zustandssummen der Kinetiken mit „Leptomonas“ ist ein größerer Unterschied feststellbar. Aber auch dieser lässt nur einen eher kleineren optischen Unterschied zwischen gefilterter und ungefilterter Kinetik vermuten. Dies müsste man durch eine ausführliche Berechnung der ungefilterten Kinetik zeigen.

Tabelle 1: Zustandssummenvergleich gefilterter Kinetiken[10].

Sequenz	**ungefiltert**	**gefiltert**
d33	2,36e+10	2,34e+10
Leptomonas	1,21e+08	1,05+08

Die Abbildungen 27 und 28 zeigen den Vergleich der DotPlots für die für die oben genannten Kinetiken. Für die Wahrscheinlichkeiten der Basenpaare der MFE ist in beiden DotPlots kein großer Unterschied feststellbar. Der Filtereffekt ist geringfügig bei den Basenpaaren alternativer Strukturen bemerkbar. Das lässt vermuten, dass die Filter nur die weniger relevanten Bereiche der Energielandschaft entfernen.

[10] Parameter: $T = 37°C = 310.15\ K$ und $R = 0.0019871588\ kcal/K * mol$

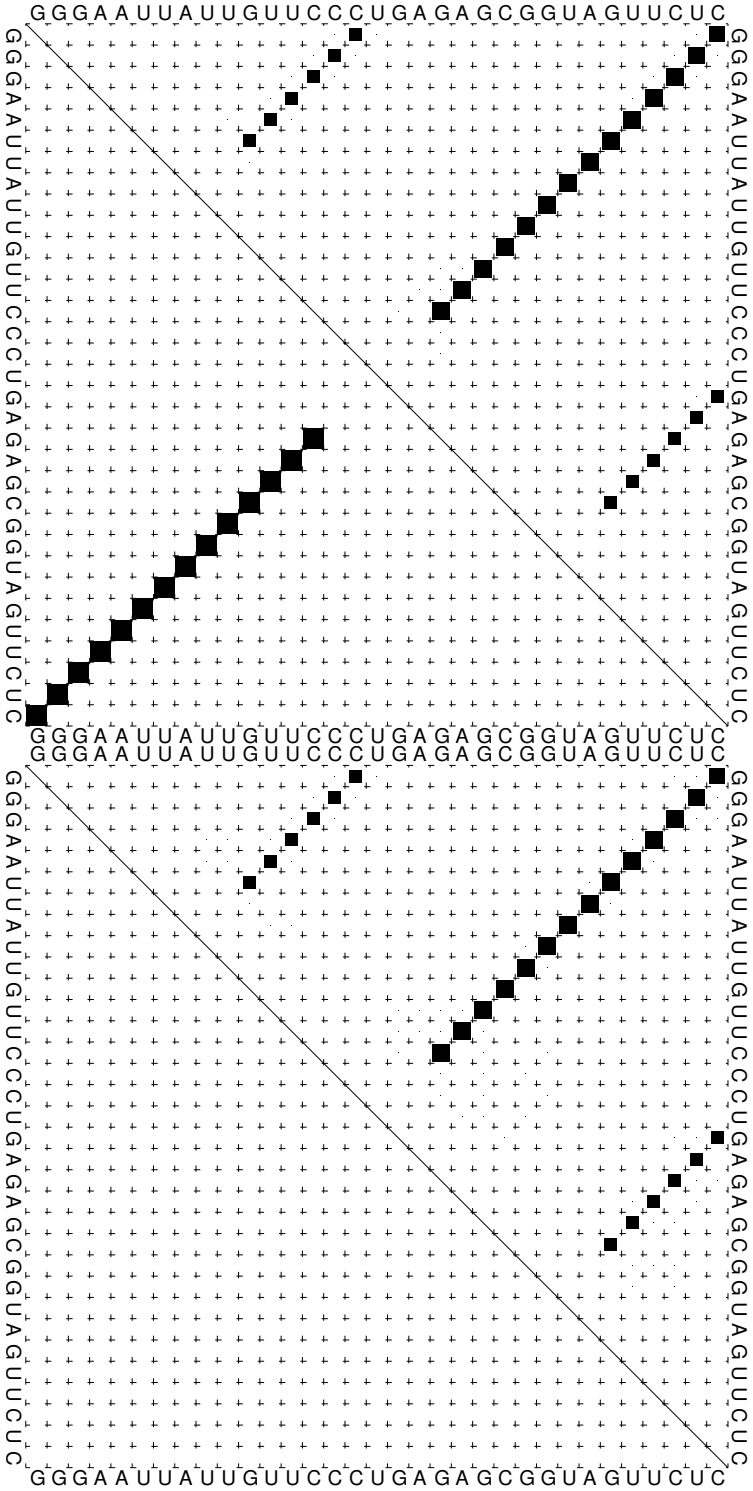


Abbildung 27: Sequenz: „d33“. Die Abbildung zeigt oben das DotPlot für die Kinetik ohne Filter und unten für die Kinetik mit Filter. Die zugehörige Kinetik ist in Abbildung 24 rechts unten dargestellt. Das obere Dreieck enthält die Basenpaarwahrscheinlichkeiten des Ensembles. Die ungefilterte Kinetik enthält zusätzlich die Basenpaarwahrscheinlichkeiten der MFE im unteren Dreieck. (Diese ist für die gefilterte Kinetik gleich und wurde nicht extra angegeben.)

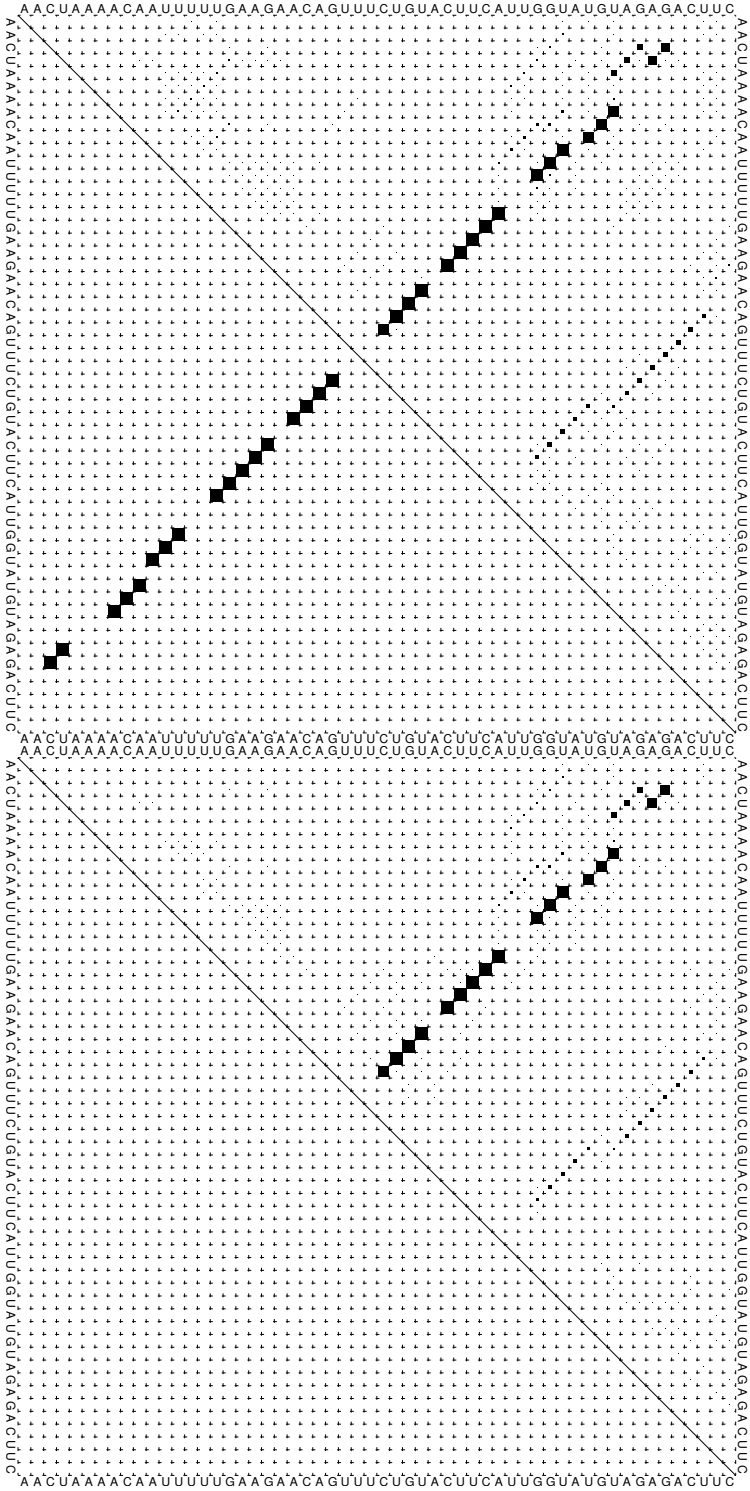


Abbildung 28: Sequenz: „Leptomonas“. Die Abbildung zeigt oben die DotPlots für die Kinetik ohne Filter (in dieser Arbeit nicht berechnet) und unten für die Kinetik mit Filter. Die zugehörige Kinetik ist in Abbildung 33 unten dargestellt. Das obere Dreieck enthält die Basenpaarwahrscheinlichkeiten des Ensembles. Die ungefilterte Kinetik enthält zusätzlich die Basenpaarwahrscheinlichkeiten der MFE im unteren Dreieck. (Diese ist für die gefilterte Kinetik gleich und wurde nicht extra angegeben.)

7 Anwendung

In den vorherigen Kapiteln wurden die Grundlagen für das entwickelte Kommandozeilenprogramm *ParKin_Explore* erläutert und der Aufbau des Programms dargestellt. Dabei wurde auch die Wirkungsweise der implementierten Filter gezeigt.

In diesem Kapitel wird die praktische Anwendung des Programms *ParKin_Explore* ausführlich beschrieben. Um die Anwendung des Programms zu erleichtern, wurde zunächst eine Pipeline erstellt. Diese nimmt die Eingabeparameter für *ParKin_Explore* entgegen und generiert eine Kinetik, die am Ende als skalierbare Grafik ausgegeben wird. Weitere Eigenschaften werden in Kapitel 7.1 erläutert.

Mit Hilfe der Pipeline können die Kinetikplots einfacher erstellt werden. Deshalb wird sie verwendet, um weitere Funktionen von *ParKin_Explore* vorzustellen. Entscheidend für die Faltung von RNA sind die Bindungsstärken, die durch die Energiemodellparameter definiert werden (siehe Kap. 2.2). Die Unterschiede zwischen Kinetiken mit den Turner-1999-Parametern, den Turner-2004-Parametern und den Andronescu-2007-Parametern werden in Kap. 7.2 behandelt. Die Bindungsstärken sind auch temperaturabhängig.

In Kap. 7.3 wird ein RNA-switch untersucht, der in einem thermophilen Bakterium vorkommt. Dieses kann in einem großen Temperaturbereich überleben. Mittels des Temperaturparameters kann untersucht werden, wie sich die Fluktuationen bei verschiedenen Temperaturen ändern. Von biologischer Seite könnte man testen, ob es einen Zusammenhang zwischen dem Switchverhalten bei verschiedenen Temperaturen und den berechneten Strukturänderungen gibt.

Um die Grenzen des Programms *ParKin_Explore* zu zeigen, wurden Tests mit langen RNA-Sequenzen durchgeführt. Diese Sequenzen sind nur mit gröberen Gruppierungsmodellen schnell berechenbar. Die Untersuchung ist in Kap. 7.4 erläutert.

7.1 Pipeline

Das Programm *ParKin_Explore* ist bisher nur über die Kommandozeile ausführbar. Es nimmt eine RNA-Sequenz entgegen und weitere optionale Parameter. Die Ausgabe ist eine Datei mit Übergangsraten für alle Macrostates.

Das Ziel der Verarbeitungspipeline ist es, die Ausgabe in eine grafische Kinetik zu transformieren. Dazu werden zusätzliche Programme in einer bestimmten Reihenfolge nacheinander ausgeführt. Die Ausgabe des vorher ausgeführten Programms ist im Idealfall direkt die Eingabe des nachfolgenden Programms. Sollte das nächste Programm die Ausgabe nicht direkt verwenden können, wird ein Skript dazwischen geschaltet, das die Ausgabe in das gewünschte Format transformiert.

Schritte der Pipeline:

Schritt 1 : Berechnung der Ratenmatrix Zuerst werden die Übergangsraten mit *ParKin_Explore* berechnet. Dabei können verschiedene Parameter eingestellt werden. Die wichtigsten sind in der Tabelle 2 dargestellt:

Tabelle 2: Beschreibung der Parameter von ParKin_Explore.

Parameter	**Beschreibung**
RNA-Sequenz	Die RNA-Sequenz als Zeichenkette. Die erlaubten Buchstaben sind a,c,g,u (klein oder groß).
Startbassin	Die Struktur eines Bassins in dot-bracket Notation. Der Flutalgorithmus beginnt im Bassin dieser Struktur.
K-Best-Filter	K ist die Anzahl der besten Transitionen, die von jedem Zustand ausgehen.
Dynamische K-Best-Filter	Mit diesem Filter wird K erhöht, bis die MFE im Ergebnis enthalten ist.
DeltaMinE-Filter	Siehe Kap. 6.2.2.
MaxToStore	Die maximale Anzahl der Zustände, die im Gradient Walk gespeichert werden. Ein höherer Wert beschleunigt die Bassinzuordnung.
Maximale-Energie-Filter	Hier kann ein Wert in kcal/mol angegeben werden. Die Bassins werden maximal bis zu dieser Grenze geflutet.
Delta-Energie-Filter	Hier kann ein Wert *deltaE* in kcal/mol angegeben werden. Die Bassins werden maximal bis zur Grenze $Minimumenergie + deltaE$ geflutet.
Temperatur	Die Temperatur für die Energieberechnung der Strukturen. Die Populationsdichten hängen von den Boltzmanngewichten ab und diese von den Energien der einzelnen Strukturen (siehe Kap. 2.2). Mit diesem Parameter kann die Temperatur für die Energieberechnung der Strukturen und der Boltzmanngewichte angegeben werden.
Energiemodell	Es können folgende Energiemodelle gewählt werden: Turner 2004, Turner 1999 und Andronescu 2007. Das Energiemodell beeinflusst die Energieberechnung der einzelnen Strukturen (siehe Kap. 2.2).
Anzahl der Threads	Die Anzahl bestimmt wie viele Bassins gleichzeitig geflutet werden. Die Berechnung wird dadurch beschleunigt, benötigt jedoch mehr Speicher.
Zustandssummen	Mit diesem Parameter wird eine Datei angegeben, in welche die Zustandssummen geschrieben werden. Diese können z.B. für eine weitere Zusammenfassung von Zuständen verwendet werden.

Selbstverständlich reicht die RNA-Sequenz als Eingabe, da für die restlichen Parameter Standardwerte eingestellt sind. Die Ausgabe ist eine Datei mit den Übergangsraten, dem Bassin der Startstruktur, dem Bassin der MFE und den Equilibriumswahrscheinlichkeiten aller Macrostates.

Schritt 2 : Berechnung der Populationsdichten Im 2. Schritt werden die Populationsdichten berechnet. Dazu müssen die Übergangsraten aus Schritt 1 in eine komplette Ratenmatrix (mit Null-Einträgen) überführt werden. Eine Datei mit den IDs und Strukturen der Matrixeinträge wird ebenfalls erstellt. Die ID der offenen Kette (OC) wird als Startzustand für die Berechnung verwendet. Diese drei Parameter, die mindestens benötigt werden, werden im Programm *Treekin* angegeben, um die Populationsdichten zu bestimmen.

Als nächstes wird die Zeit bis zum Erreichen des Equilibriums benötigt. Die Equilibriumwahrscheinlichkeiten aus Schritt 1 werden verwendet, um diesen Zeitpunkt zu ermitteln. Um den Zeitpunkt des Equilibriums zu ermitteln, wird das Programm *Treekin* für einen Zeitpunkt aufgerufen. Dadurch erhält man einen Vektor mit der Populationsdichte zu diesem Zeitpunkt. Der erhaltene Vektor wird mit den Equilibriumswahrscheinlichkeiten bis auf die

dritte Nachkommastelle verglichen. *Treekin* wird solange mit Zeitparameter*10 aufgerufen, bis die Vektoren übereinstimmen. Dieser Zeitpunkt wird gespeichert.

Anschließend wird die komplette Kinetik mit *Treekin* berechnet. Diese startet bei Zeitpunkt 0 und endet beim Zeitpunkt, in dem das Equilibrium sicher erreicht ist.

Die Ausgabe ist eine Textdatei mit den Zeitpunkten und dazugehörigen Populationsdichten. Diese Datei wird am Ende noch von einem weiteren Skript soweit gekürzt, bis am Ende keine Zeile mit gleichen Werten zweimal vorkommt. Somit wird sichergestellt, dass wirklich nur Populationsdichten bis zum Equilibrium vorhanden sind.

Schritt 3 : Visualisierung der Populationsdichten Die Textdatei mit den Populationsdichten wird mit dem Programm *grace* visualisiert. Die 10 wahrscheinlichsten Zustände werden von einem Skript ermittelt und (in Dot-Bracket-Notation) in der Legende angezeigt. Die Kurven dieser Zustände werden in verschiedenen Farben ausgegeben. Die anderen Kurven werden in Graustufen ausgegeben. Dadurch kann man die wichtigen Zustände besser von den unwichtigen Zuständen unterscheiden. Die Visualisierung wird als skalierbare PDF Datei ausgegeben.

Anschließend kann noch ein Histogramm über die Zustandsdiche der Energien aller Zustände mit dem Programm *R* berechnet und als PDF ausgegeben werden. Das ist allerdings nur möglich, wenn der entsprechende Parameter bei *ParKin_Explore* (in Schritt 1) angegeben wurde. Er wird per default nicht gesetzt, weil das Speichern der einzelnen Energien für alle Zustände mehr Rechenzeit und vor allem Speicher benötigt.

7.2 Energiemodell

In Kap. 2.2 wurde das Nearest Neighbor Model erläutert. Mit diesem Modell wird die freie Energie einer gesamten RNA-Struktur mit der Summe über die Strukturelemente berechnet, aus denen die Struktur aufgebaut ist. Dabei werden die unmittelbar benachbarten Basenpaare in die Berechnung einzelner Strukturelemente mit einbezogen.

Es gibt verschiedene Messungen über die physikalischen Bindungsenergien zwischen den Basenpaaren. Die bekanntesten sind die Turnerparameter von 1999, die Turnerparameter von 2004 und die Andronescuparameter von 2007.

Um zu zeigen, wie wichtig genaue Energieparameter für die Qualität einer Kinetik sind, wurden drei Kinetiken mit der gleichen Sequenz und verschiedenen Energieparametern berechnet. Die Kinetiken sind in Abbildung 29 dargestellt.

Die Sekundärstrukturen sind, sofern möglich, über die entsprechenden Kurven gezeichnet und farbig markiert. Die Markierungen stimmen mit denen in den anderen Kinetiken überein. Es ist auffällig, dass die MFE (hellblau) bei den Turner-1999-Parametern und den Andronescu-2007-Parametern gleich ist. Mit den Turner-2004-Parametern hat die hellblau gekennzeichnete Struktur einen größeren Wert für die freie Energie. Da sich mit unterschiedlichen Energieparametern sogar die MFE ändert, ist es erforderlich, exakt ermittelte Energieparameter zu finden. In *ParKin_Explore* kann, unter Anwendung der ViennaRNA-Bibliothek, eine beliebige Datei mit Energieparametern verwendet werden.

Die Kinetik mit den Turner2004 Parametern ist vermutlich nicht repräsentativ, da für diese Parameter ein Bug in der (hier noch aktuellen) ViennaRNA Version 2.1.7 vorlag. Dieser sollte in der 2.1.9 Version behoben sein.

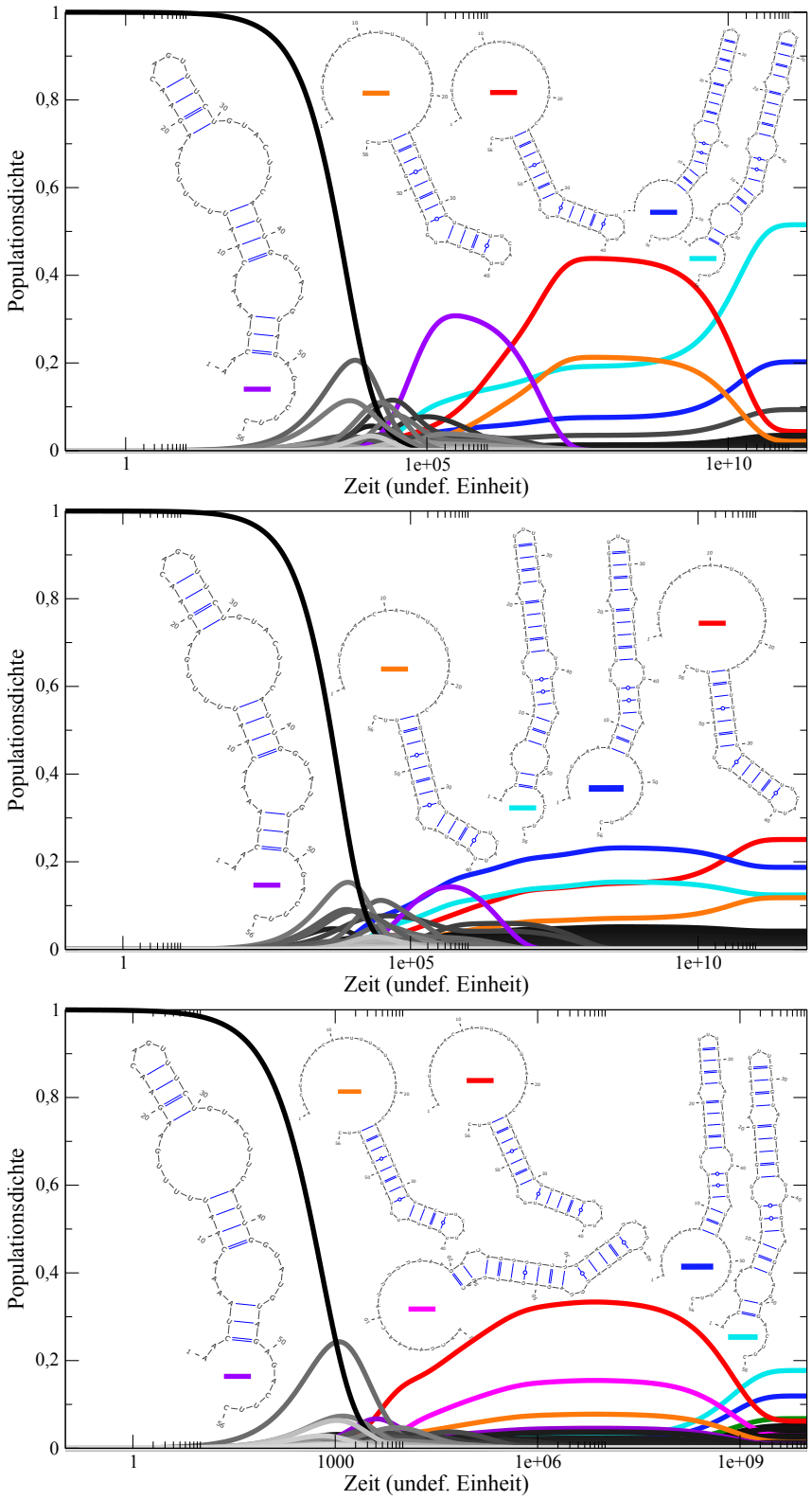


Abbildung 29: Sequenz: „Leptomonas“-Energiemodell-Tests mit den Parametern: Delta-Energie Filter = 5, K-Best-Filter = 3, Threads = 8, Temperatur = $37°C$, Dangles = 2, DeltaMinE-Filter = 0,5.
Energiemodell: 1. Turner 1999, 2. Turner 2004, 3. Andronescu 2007

7.3 Temperaturparameter

Es wurden verschiedene Sequenzen mit unterschiedlichen Temperaturen getestet, weil die Bindungsstärke der Wasserstoffbrücken temperaturabhängig ist. Die RNAs kommen in verschiedenen Lebewesen, bei verschiedenen Temperaturen, vor. Um die Fluktuationen in der Kinetik möglichst realistisch zu berechnen, sollte man diesen Parameter nicht vernachlässigen.

Eine Beispielsequenz ist der „Fluoride Riboswitch“ [30] (RCSB PDB ID: 4ENA) mit 52 Nukleotiden und kommt im Bakterium *Thermotoga petrophila* vor. Dieses wurde im Kubiki-Ölreservoir in Niigata, Japan, entdeckt. Es lebt bei $80°C$ und kann bei Temperaturen von $48°C$ bis $86°C$ überleben [34].

Die Sequenz „4ENA“ ist allerdings nur ein Teil des Riboswitches. Die komplette Sequenz ist 75 Nukleotide lang (siehe Anhang von [30]).

In der Abbildung 30 sind die Kinetiken des „Fluoride Riboswitch“ bei verschiedenen Temperaturen dargestellt. Man kann erkennen, dass die Kinetiken bei $48°C$ und $60°C$ ähnliche Zustände im Gleichgewicht haben. Bei der optimalen Temperatur von $80°C$ ändert sich die Kinetik sehr stark und auch die MFE.

Bei einer Temperaturzunahme kann man erwarten, dass sich die äußeren Basenpaare zuerst lösen. Diese haben weniger benachbarte Paare, die für eine bessere Gesamtstabilität sorgen. Beim Vergleich der MFEs aus den Kinetiken bei $60°C$ und $80°C$ sieht man, dass die obere Hairpinstruktur und die InteriorLoop noch vorhanden sind. Die linke Hairpinstruktur hat sich „aufgelöst“ und dadurch wurde die Helixstruktur unterhalb der InteriorLoop erweitert.

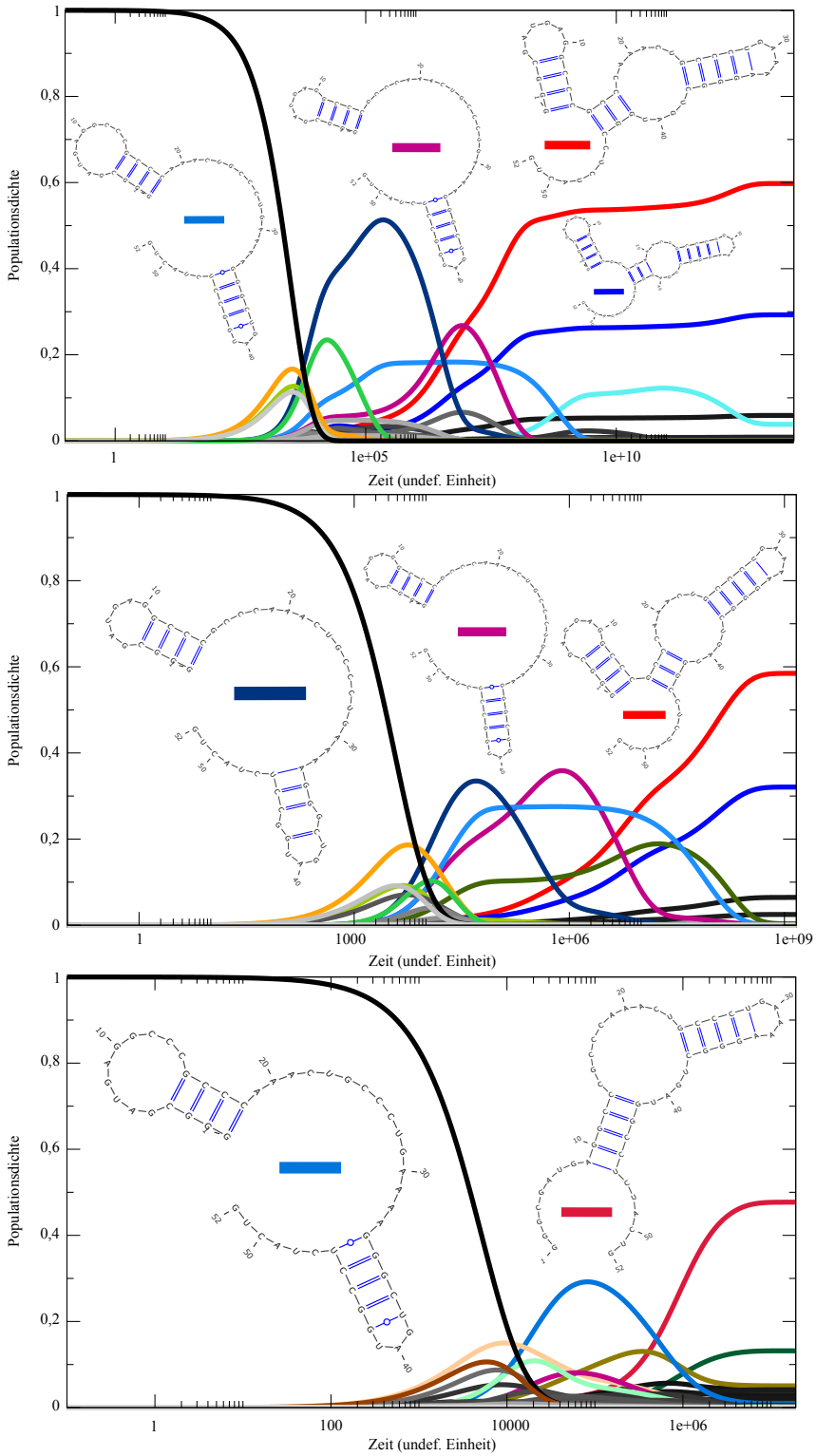


Abbildung 30: Sequenz: „4ENA“ mit K-Best Filter mit K=3, DeltaMinE-Filter mit E=0.5 und Delta-Energie-Filter mit $\Delta E = 10$ und 8 Threads.
1. Bei $48°C$, 2. Bei $60°C$, 3. Bei $80°C$
Die Kinetiken wurden in 40 Sekunden, 55 Sekunden und 7 Minuten berechnet.

Es ist auffällig, dass in den Kinetiken unterschiedliche MFE Strukturen vorkommen. Diese haben eine Haarnadelstruktur gemeinsam, sind aber sonst unterschiedlich. Um einen Eindruck zu bekommen, inwieweit die Kinetiken mit anderen Methoden übereinstimmen, wurde zum Vergleich eine 3D Struktur des RNA-switches analysiert, die mit Röntgenkristallographie ermittelt wurde. Diese ist in Abbildung 31 dargestellt.

```
GGGCGAUGAGGCCCGCCCAAACUGCCCUGAAAAGGGCUGAUGGCCUCUACUG
.[[[[..((((((]]]].....(((((....)))))....))))))).....
```

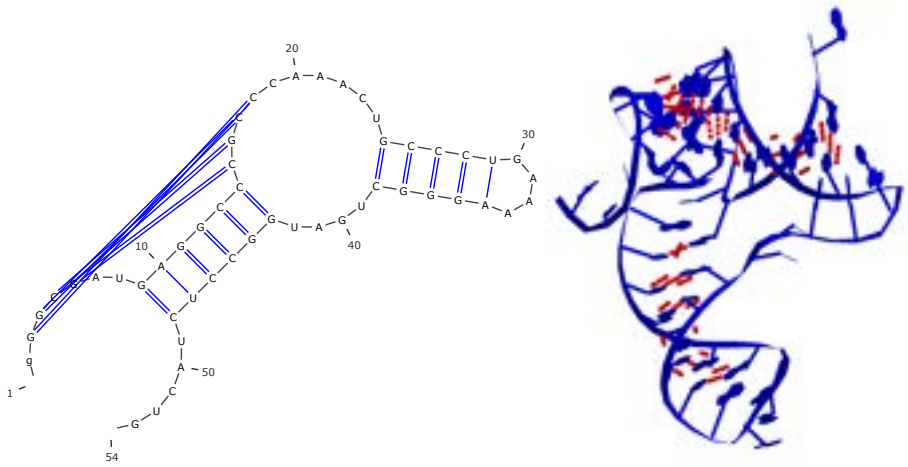


Abbildung 31: Sequenz: „4ENA“ mit Primär-, Sekundär- und Tertiärstruktur. Die Tertiärstruktur entstammt der Datei 4ENA.pdb[11] und wurde mittels *Jmol-14.2.12*[12] erstellt und mit *Povray-3.7*[13] gerendert. Durch den Webservice *RNApdbee* [14] wurde aus der 3D Struktur die wahrscheinlichste 2D Struktur mit Pseudoknoten („[]“) ermittelt.

Wenn man die Pseudoknotenstruktur mit dem Webservice *K2N* (Knotted to Nested) [15] in eine pseudoknotenfreie Struktur umwandelt, werden mit allen Methoden (außer IL und IR) nur die eckigen Klammern entfernt. Diese Struktur entspricht auch der MFE aus der Kinetik, die in Abbildung 30 mit $80°C$ berechnet wurde. Die Tertiärstruktur wurde durch Röntgenkristallographie von einer echten RNA ermittelt. Weil die daraus extrahierte Sekundärstruktur identisch mit der MFE aus der berechneten Kinetik ist, kann man davon ausgehen, dass diese Kinetik nahe an der Realität ist.

Allerdings ist die 52 Nukleotide lange Sequenz „4ENA“ nur der fluoridbindende Teil des 74 Nukleotide langen Riboswitches. Die komplette Sequenz ist im Anhang von [30] zu finden. Laut dieser Quelle bildet sich zur MFE noch eine weitere lange Haarnadelstruktur aus, wenn kein Fluorid gebunden ist.

Das Beispiel sollte aber primär dazu dienen, die Wichtigkeit des Temperaturparameters zu demonstrieren. Denn die berechnete Struktur, die am nächsten der durch Röntgenkristallographie ermittelten Struktur entspricht, kommt nur in der $80°C$ Kinetik als MFE vor. Diese Temperatur entspricht auch der optimalen Temperatur, bei der das Bakterium *Thermotoga petrophila* lebt und in welchem der Riboswitch vorkommt.

[15] http://www.ibi.vu.nl/programs/k2nwww/

7.4 Lange Sequenzen

In diesem Kapitel wird das Programm *ParKin_Explore* mit langen Sequenzen getestet und mit anderen Programmen verglichen. Die Energielandschaft wird bei Sequenzen von ca. 100 Nukleotiden so groß, dass eine Zusammenfassung zu lokalen Minima (trotz Filter) nicht mehr ausreicht, um die Ratenmatrix auf eine berechenbare Größe zu reduzieren. Daher werden teilweise gröbere Arten von Macrostates benötigt. Ansätze dafür werden im folgenden Teil vorgestellt. Anschließend werden Kinetiken für lange Sequenzen miteinander verglichen.

7.4.1 Macrostate Alternativen

Bei Macrostate Kinetiken mit langen Sequenzen wird die Ratenmatrix i.d.R. sehr groß. Bei einer RNA Länge von ca. 52 Nukleotiden werden viel mehr als 3000 lokale Minima gefunden. Eine Kinetikberechnung wäre zu rechenintensiv.

Eine Lösung bietet der helixbasierte Ansatz in [16]. Dabei werden die Microstates zu Helix-Index-Shapes (kurz: HI-Shapes) zusammengefasst. Helix-Index-Shapes beschreiben die RNA-Struktur durch die Helix-Indices ihrer Strukturelemente. Für Hairpins berechnet man den Helix-Index, indem man die Summe der Indices des innersten Basenpaars der Hairpin durch zwei teilt. In Abbildung 32 ist das Zusammenfassen einzelner Strukturen zu ihren HI-Shapes dargestellt. Das HI-Shape [6.5] berechnet sich durch $(4+9)/2$. Das sind die Indices des inneren Basenpaars C-G der Hairpin, der linken Struktur in der Abbildung.

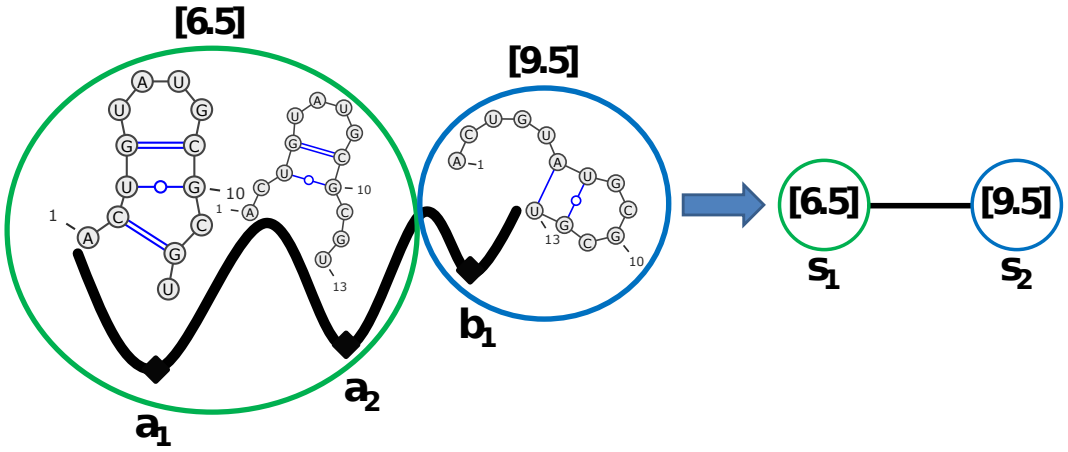


Abbildung 32: Cluster mit HI-Shapes.

Das Programm *HiKinetics*, welches auf Helix-Index-Shapes basiert (siehe Kap. 1.1), verwendet einen barrierenbasierten Ansatz um die einzelnen HI-Shapes zu verbinden. Das ist nötig, weil die HI-Shapes aufgrund der Basenpaare nicht direkt benachbart sein müssen. Die Barriere ist die Energie des energetisch höchst-liegenden Zustands, der auf einem Pfad mit Single-Moves liegt, der zwei HI-Shapes (oder Macrostates) miteinander verbindet. Dieser „Barrier-Tree-Ansatz“ wurde schon vorher im Programm *Barriers* [9] verwendet.

Eigentlich sind „Barrier-Tree-Kinetiken“ mit den hier verwendeten Kinetiken nur schwer vergleichbar, weil unterschiedliche Teile der Energielandschaft betrachtet und die Raten unterschiedlich berechnet werden. In Kap. 7.4.2 wird dieser Vergleich primär dazu durchgeführt, um zu zeigen, dass man mit dem explorativen Flutalgorithmus mit Filtern noch schneller approximative Kinetiken für mittellange Sequenzen berechnen kann.

Eine alternative Lösung zum weiteren Zusammenfassen von Macrostates basiert auf der Fluthöhe der einzelnen Bassins. Dabei wird eine „Aktivierungsenergie“ verwendet. Das ist die Energie, die auf die Energie eines lokalen Minimums addiert werden muss, um eine Sattelenergie zu überwinden und damit in ein anderes Minimum zu gelangen. Diese Lösung hat den Vorteil, dass die zusammengefassten Macrostates nebeneinander liegen (was beim Ansatz mit HI-Shapes nicht garantiert ist). Ein ähnliche Methode, die benachbarte Macrostates mit niedriger Energiebarriere weiter zusammenfasst, ist auch schon in *Barriers* implementiert [9]. Der Ansatz wurde in dieser Arbeit jedoch nicht weiter verfolgt.

7.4.2 Kinetikvergleiche mit langen Sequenzen

Um die Grenzen des Programms *ParKin_Explore* auszuloten, wurden Tests mit langen Sequenzen durchgeführt. In Abbildung 33 sieht man, dass durch den Ansatz mit Helix-Index-Shapes von *HiKinetics* ähnliche Sekundärstrukturen hohe Populationsdichten haben, wie beim Ansatz mit lokalen Minima von *ParKin_Explore*. Beide Kinetiken zeigen das typische Verhalten von RNA-switches. Dieses ist in der Kinetik dadurch erkennbar, dass über ein langes Zeitintervall eine andere Struktur als die MFE, ebenfalls eine hohe Populationsdichte hat. Die Populationsdichte weicht in diesem Intervall nicht stark von ihrem Wert ab, sondern erst kurz bevor sich das Gleichgewicht einstellt und die MFE die höchste Populationsdichte erreicht.

Die untere Kinetik in Abbildung 33 unterscheidet sich auf den ersten Blick stark von der oberen Kinetik. In der unteren Kinetik sind sich jedoch die rote und orange Kurve sehr ähnlich. Das Gleiche gilt für die blaue und hellblaue Kurve. Die Strukturen der roten und blauen Kurve (MFE) sind in beiden Kinetiken identisch. Durch die detaillierte Betrachtung der Energielandschaft mit *ParKin_Explore* erhält man viel mehr Macrostates, als in der groben Betrachtung mit *HiKinetics*. Das liegt daran, dass mit *HiKinetics* Strukturen mit ähnlichen Helixstrukturen zusammengefasst werden.

Durch das Verwenden von Filtern und explorativem globalen Fluten konnte die untere Kinetik mit 1,6 Sekunden in kurzer Zeit berechnet werden, obwohl sie eigentlich viel mehr Macrostates, als die obere Kinetik enthält. Mit *HiKinetics* wurde durch Filtern in [16] (hier nicht gezeigt) eine Kinetik mit ähnlicher Aussagekraft in 14 Minuten erzeugt. Das Programm *ParKin_Explore* kann also bei mittellangen Sequenzen (hier 57 Nukleotide) noch schneller aussagekräftige Ergebnisse liefern.

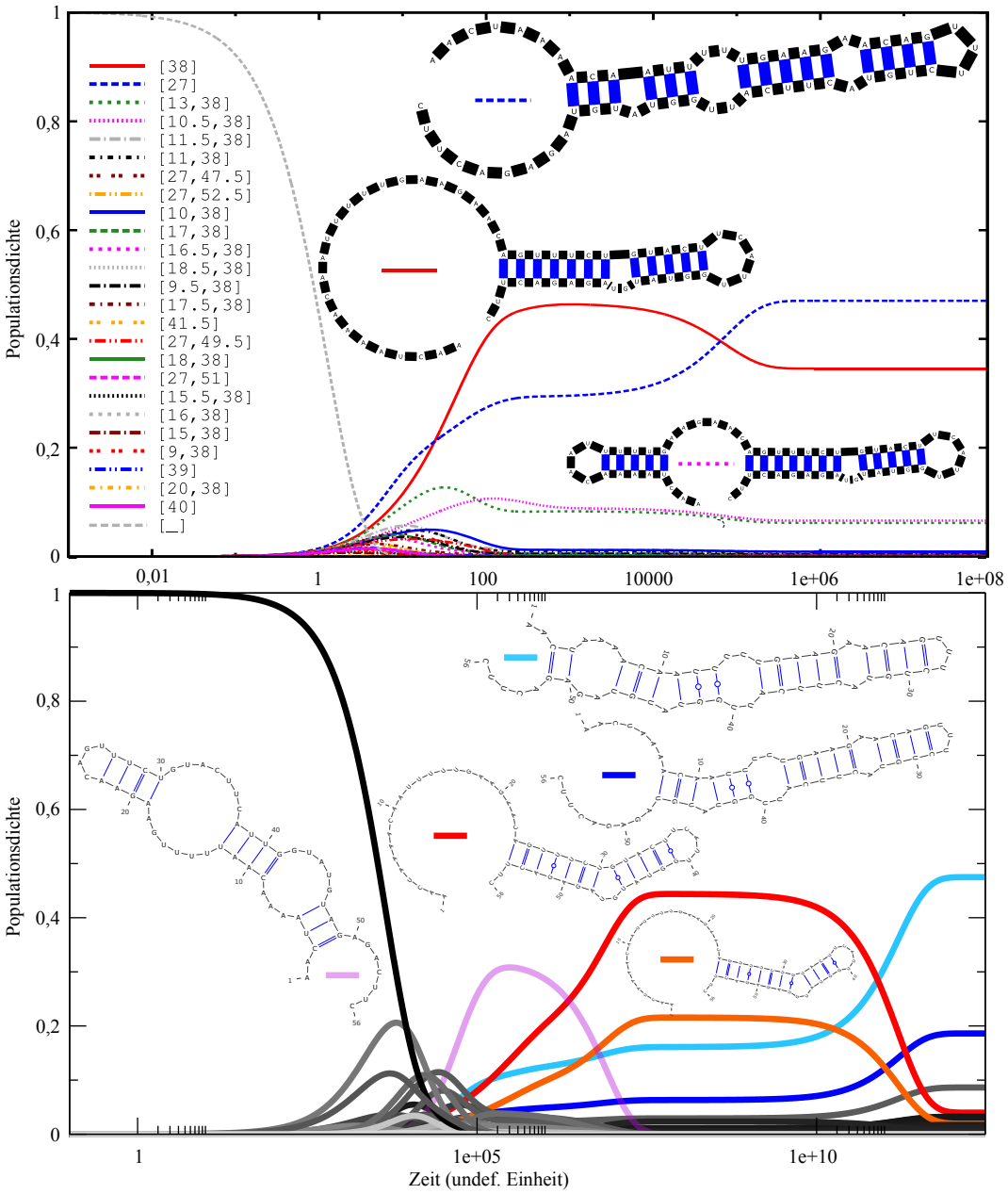


Abbildung 33: Sequenz: „Leptomonas“. Die obere Kinetik wurde mit *HiKinetics* (mit Helix-Index-Shapes, siehe Kap. 1.1) berechnet; das Bild ist aus [16] entnommen. Die Sekundärstrukturen wurden vergrößert. Die Simulation benötigte 253 Minuten mit: 64 Kernen einer 4 x AMD Opteron 6282 SE Maschine, 512 GB RAM und dem Betriebssystem openSuSE12.2. Der Programmaufruf war: „/HiKinetics.rb-i Input/spliced_leader.seq-oOutput/spliced_leader -k100-t4“.
Die untere Kinetik wurde mit *ParKin_Explore* in 1,6 Sekunden berechnet und enthält 93 Macrostates (Parameter: Delta-Energie-Filter = 5 kcal/mol, Best-K Filter = 3, Threads = 8, Temperatur = 37°C, Dangles = 1, DeltaMinE-Filter = 0.5, Energiemodel = Turner 1999). Die verwendete Hardware ist ein Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz mit 4 Kernen und Hyperthreading und 14 GB RAM.

Für lange RNA-Sequenzen steigt die Anzahl der lokalen Minima trotz Filterung sehr schnell. Für die 125 Nukleotide lange Sequenz des C-Di-GMP Riboswitch [16] konnte in 3,3 Stunden eine Macrostate-Ratenmatrix berechnet werden[16] (im Vergleich zu 24 Stunden mit *HiKinetics*). Diese enthält jedoch ca. 14981 Macrostates. Eine Kinetikberechnung würde lange dauern und viel Speicher in Anspruch nehmen. Um dennoch eine Kinetik zu bekommen, kann man die erhaltenen Macrostates zu gröberen helixbasierten Macrostates zusammenfassen. Die HI-Shape-Ratenmatrix der helixbasierten Macrostates wird über die Zustandssummen gemeinsamer Transitionen und Macrostates neu berechnet. Diese Berechnung ist in Formel 16 dargestellt:

$$\frac{\sum_{A \in S_1, B \in S2} Z_{AB}}{\sum Z_A} \tag{16}$$

Die Macrostates in dieser Formel sind A und B, die HI-Shapes sind S_1 und S_2.

Für die Sequenz „Leptomonas“ konnte mit dem HI-Shape Ansatz eine Kinetik berechnet werden (siehe Abbildung 42 im Appendix).

Bei der Berechnung der Sequenz „C-Di-GMP“ wurde die Macrostate-Ratenmatrix von 14981 Macrostates auf 395 HI-Shapes reduziert. Mit 395 Zuständen konnte eine Kinetik berechnet werden. Die Berechnung wurde allerdings vom Programm *Treekin* nach einiger Zeit abgebrochen, weil der Rundungsfehler zu groß wurde. Die Kinetik ist in Abbildung 34 dargestellt. Zum Vergleich wird zunächst eine Kinetik gezeigt (Abbildung 35), die aus [16] S.15 entnommen ist. Diese wurde mit *HiKinetics* in 24 Stunden erzeugt und enthält 26 HI-Shapes.

Die erste Kinetik in Abbildung 34 wurde mit *ParKin_ Explore* in 3,3 Stunden erzeugt. Die Macrostate-Ratenmatrix wurde auf HI-Shapes reduziert. Die zweite Kinetik resultiert aus der ersten. Die HI-Shape-Ratenmatrix wurde mit dem Best-K-Filter mit $K = 4$ gefiltert. Für diesen Filter wurde ein separates Shellscript erzeugt. Man sieht, dass sich die Kinetiken sehr ähnlich sehen. Man kann also durchaus die vorgestellten Macrostate Filter auch mit anderen Zusammenfassungen der Energielandschaft verwenden.

Beim Vergleich von Abbildung 35 und Abbildung 34 fällt auf, dass die Sekundärstrukturen für die farbig markierten Kurven sehr unterschiedlich sind. Die Strukturen aus der ersten Abbildung sind nicht unbedingt lokale Minima. Die Raten wurden nicht aus der Kontaktfläche ermittelt, sondern nur aus der Energie des größten Zustands auf dem Pfad, der je zwei Zustände verbindet. Die Kinetik in der zweiten Abbildung verwendet im Bezug darauf ein genaueres Modell und wurde schneller berechnet.

Ein Vergleich der wichtigsten Strukturen aus beiden Kinetiken in einem Benchmark, mit anderen Programmen, ist im Appendix in 4 dargestellt. Bei diesem Vergleich werden die „ON“ und „OFF“ Strukturen des RNA-switches dargestellt, die von verschiedenen Programmen ermittelt wurden. Die zugehörigen HI-Shapes wurden hinzugefügt. Man erkennt, dass beide Kinetiken wichtige Zustände erzeugen, die allerdings nur mit dem „OFF“ Zustand aus anderen Programmen übereinstimmen. Um weitere Schlüsse aus diesem Vergleich zu ziehen, müsste man jedoch zusätzliche Einstellungen der Programme überprüfen, da z.B. die Energieparameter eine große Auswirkung auf die Kinetik haben.

[16] Parameter: siehe Abbildung 34.

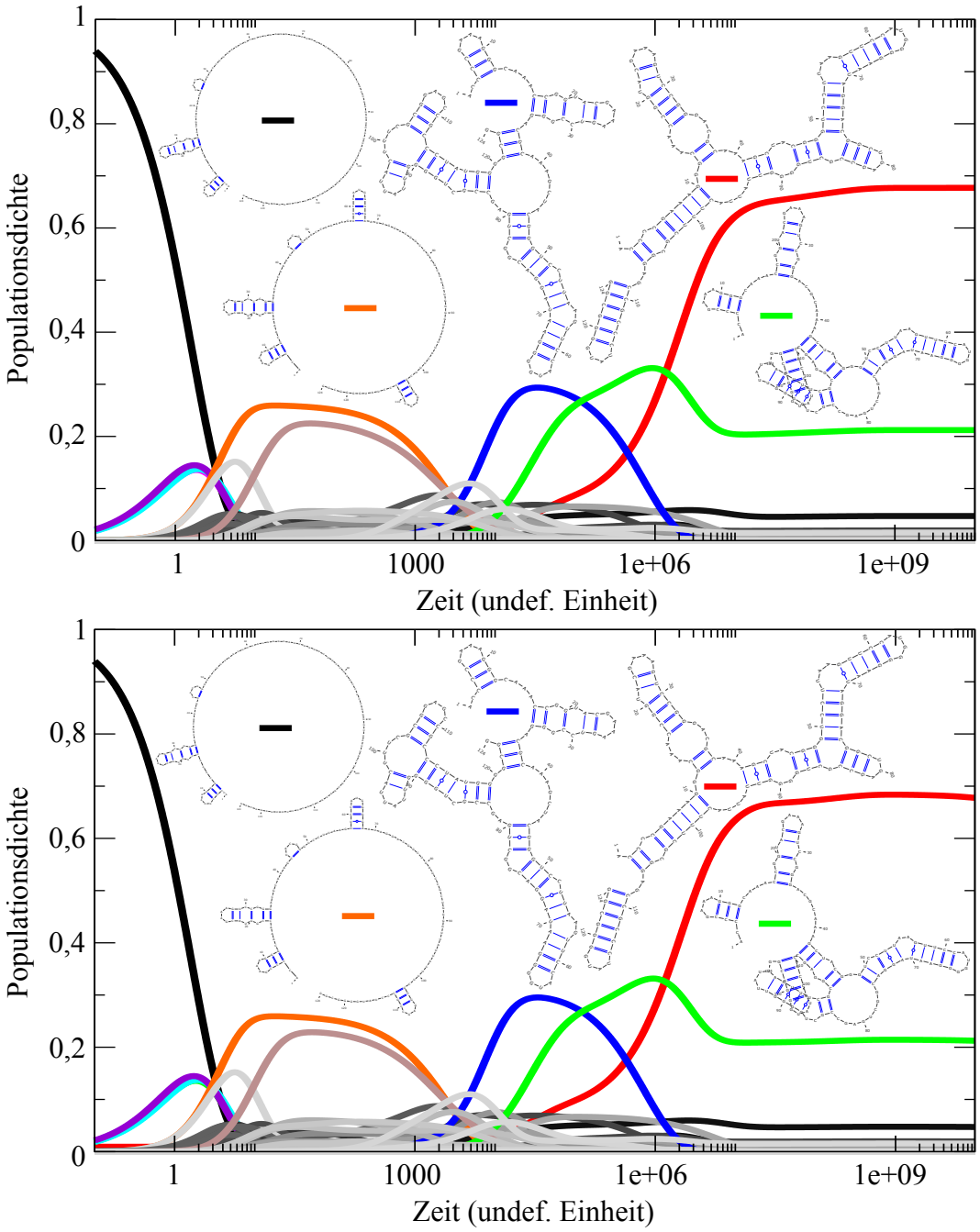


Abbildung 34: Sequenz: „C-Di-GMP“. Die erste Kinetik wurde mit *ParKin_Explore* in 3 Stunden und 20 Minuten berechnet und enthält nach Zusammenfassen der Macrostates 395 HI-Shapes. (Parameter: Delta-Energie-Filter = 10 kcal/mol, Maximale-Energie-Filter = 0 kcal/mol, Best-K Filter = 3, Threads = 8, Temperatur = 37°C, Dangles = 2, DeltaMinE-Filter = 0, Energiemodel = Turner 1999). Die verwendete Hardware ist ein Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz mit 4 Kernen und Hyperthreading und 14 GB RAM. Die zweite Kinetik resultiert aus der gefilterten, auf HI-Shapes übertragenen mittleren Kinetik. Die Ratenmatrix mit HI-Shape wurde mit Best-K = 3 gefiltert und enthält 324 HI-Shapes.

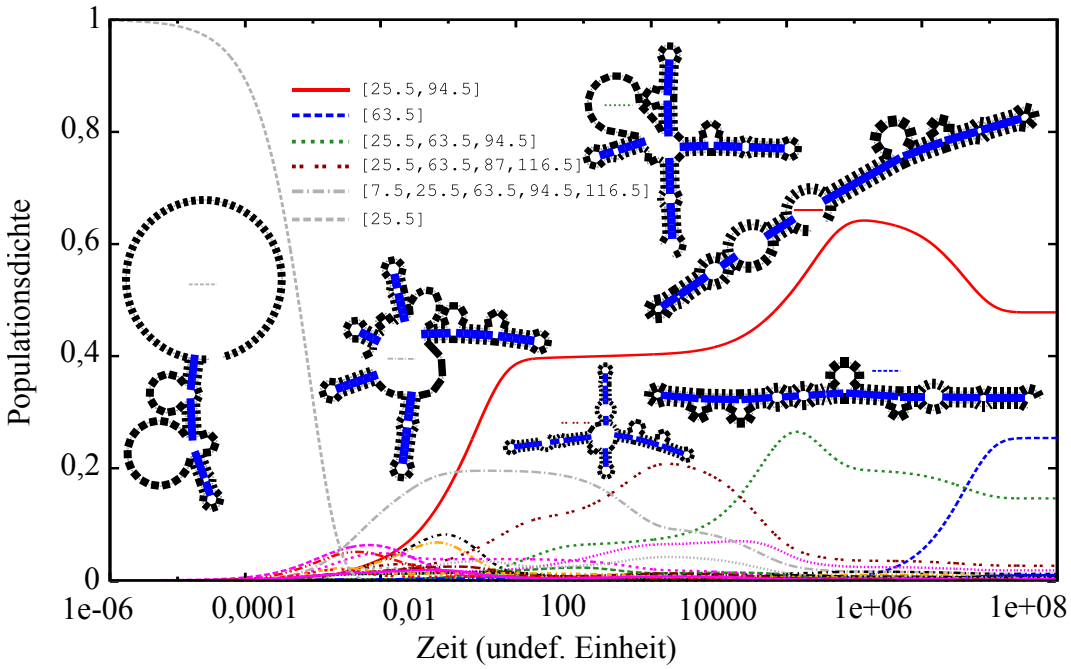


Abbildung 35: Sequenz: „C-Di-GMP“. Die Kinetik wurde mit *HiKinetics* (mit Helix-Index-Shapes, siehe Kap. 1.1) berechnet, das Bild ist aus [16] S.15 entnommen. Die Sekundärstrukturen wurden vergrößert. Die Simulation benötigte 24 Stunden mit: 64 Kernen einer 4 x AMD Opteron 6282 SE Maschine, 512 GB RAM und dem Betriebssystem openSuSE12.2. Der Programmaufruf war: „/HiKinetics.rb-i Input/c_di_GMP_riboswitch.seq-o Output/c_di_GMP_riboswitch_SN -k100 -t4 -s -p [25.5]“.

8 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war, zunächst den explorativen Flutalgorithmus für Energielandschaften mit RNA-Sequenzen zu implementieren und zu parallelisieren. Dieser eröffnet neue Möglichkeiten für effiziente Filterstratiegien, die den Konformationsraum verkleinern. Somit lassen sich die Ratenmatrizen auch für mittellange RNA-Sequenzen in kurzer Zeit approximativ berechnen.

Es wurde gezeigt, dass für geeignete Filterparameter nur sehr geringe Unterschiede zwischen den approximativen Kinetiken und den Kinetiken mit komplettem Konformationsraum feststellbar sind. Der Aufwand für die Berechnung der Ratenmatrizen kann also von mehreren Stunden auf einige Sekunden reduziert werden, ohne dass in den Kinetiken optische Unterschiede erkannt werden können.

Da die Filter unterschiedlich wirken, können unterschiedliche hochenergetische und niedrigenergetische Bereiche entfernt werden. Es ist daher schwer zu sagen, wie gut die Fluktuationen zwischen verschiedenen Strukturen erhalten bleiben und die gefilterten Kinetiken eine volle Kinetik widerspiegeln. In Kapitel 6.5 wurden ein paar Ansätze zum detaillierteren Vergleich gezeigt. Das Erforschen aussagekräftiger gefilterter Kinetiken wird jedoch weiterhin Bestandteil tiefergehender Untersuchungen bleiben.

Das entwickelte Programm *ParKin_Explore* eröffnet, durch viele optionale Ausgaben von Merkmalen der Energielandschaft, alternative Möglichkeiten zur Berechnung und grafischen Aufbereitung der Kinetiken. Ein Beispiel dafür sind die Equilibriumswahrscheinlichkeiten, die zum rechtzeitigen Terminieren der Kinetikberechnung verwendet werden können. Ein weiteres Merkmal ist die Ausgabe der Zustandssummen, welche für die Berechnung diverser Definitionen von Macrostates (u.A. helixindexbasierte Muster) verwendet werden können.

Der hier verwendete Ansatz beschleunigt das Erforschen von RNA-Sequenzen mit Macrostates, die auf lokalen Minima basieren. Es ist jedoch theoretisch möglich, die Filterstrategien auch auf andere Ansätze zu übertragen.

Eine denkbare Erweiterung des Programms besteht darin, alternative Berechnungen für die Ratenmatrix zu verwenden. Kirkpatrick [18] stellt verschiedene Ratenberechnungen vor. Diese ermöglichen zum Teil auch bessere Vergleiche zwischen kompletten und approximativen Kinetiken, weil sie die Proportionalität des Equilibriums erhalten.

Weitere Entwicklungsmöglichkeiten liegen in der Optimierung der Parallelisierung und in der Erstellung eines Webinterfaces, welches die hier erstellte Pipeline zur Berechnung der Kinetiken verwendet. Dadurch kann die Bedienbarkeit des Programms noch komfortabler gestaltet werden.

Diese Arbeit hat gezeigt, dass die Erforschung von RNA-Kinetiken zum besseren Verstehen von Fluktuationen in Faltungsabläufen von RNA-Molekülen ein noch nicht abgeschlossenes Thema ist, weil die exponentiell wachsende Anzahl der RNA-Strukturen entweder nur teilweise oder nur für kurze Sequenzen bei Kinetikberechnungen verwendet werden kann. Deshalb ist es erforderlich, für verschiedene Sequenzlängen individuelle Programme zu entwickeln, welche die Kinetiken so genau und schnell wie möglich berechnen können.

A Appendix

Im Appendix werden die durchgeführten Tests mit Kinetiken auch für andere Sequenzen gezeigt. Um die Versuche jederzeit wiederholen zu können, werden die verwendeten Sequenzen dargestellt und auch die Programmeinstellungen.

A.1 Verwendete Sequenzen

In der Tabelle A.1 sind alle RNA-Sequenzen dargestellt, die in dieser Arbeit verwendet wurden. Diese sind zum Teil künstlich erzeugt oder kommen in Zellen verschiedener Lebewesen vor.

Tabelle 3: Verwendete RNA-Sequenzen.

Name	**Sequenz**	**Länge**	**Quelle**
boris1	GACCGGAAGGUCCGCCUUCC	20	[10]
d33	GGGAAUUAUUGUUCCCUGAGAGCGGUAGUU-CUC	33	[25]
ire	CUGUCUCUUGCUUCAACAGUGUUUGGACGGAA-CAG	35	[25]
4ENA	GGGCGAUGAGGCCCGCCCAAACUGCCCUGAAA-AGGGCUGAUGGCCUCUACUG	52	[30]
Leptomonas	AACUAAAACAAUUUUUGAAGAACAGUUUCUGU-ACUUCAUUGGUAUGUAGAGACUUC	56	[16]

A.2 Verwendete Programme und Einstellungen

In diesem Abschnitt werden die Einstellungen verschiedener Programme gezeigt, wie sie in dieser Arbeit verwendet wurden. Die Kommandozeilenaufrufe sind sowohl für die Compilierung als auch für die Ausführung der Programme wichtig. Hierbei werden nur die Einstellungen gezeigt, die von den Standardparametern abweichen.

Alle Tests wurden auf dem Linux Betriebssystem *Fedora release 20 (Heisenbug)* durchgeführt. CPU: Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz (2 Kerne mit Hyperthreading (mit 4 Threads). Arbeitsspeicher: 8 GigaByte Geschätzte Rechenleistung: 6184.36 bogomips (million instructions per second)

- *Barriers v1.5.2* [17] configure: –with-hash-bits=27 (24 funktioniert nicht mit den Sequenzen ire und d33)
- *ViennaRNA v2.1.7* [18]
- *BIU Version vom 16.09.2014*
- *ELL Version vom 16.09.2014*

Kommandozeilenaufrufe:
Zeitmessungen mit `‘date +%s‘`

Test mit *RNA_barriers*: `cat sequence.txt | RNAsubopt -e 1000 -d2 -s | RNA_barriers -useInE -in=STDIN -transProb=transitions.txt`

Test mit *Barriers*: `cat Sequences/ire/seq.txt | RNAsubopt -e 1000 -d2 -s | barriers -G RNA -M noShift -max=5000 -minh=0 -rates > transitions.txt`

Test mit *ParKin_Explore*: `ParKin_Explore -seq=‘cat sequence.txt‘ -maxThreads=8 -maxToStore=21474836470 -maxToHash=21474836470 > transitions.txt`

[17] http://www.tbi.univie.ac.at/RNA/Barriers/ letzter Besuch: 22.10.2014
[18] http://www.tbi.univie.ac.at/RNA/index.html letzter Besuch: 22.10.2014

A.3 ParKin_Explore-Tests

Die verwendeten Programme wurden mit verschiedenen Einstellungen und Sequenzen getestet. Einige Tests werden in diesem Abschnitt mit anderen Sequenzen gezeigt. Die folgenden Abbildungen verdeutlichen nochmal, dass die vorher durchgeführten Tests keine „Ausreißer“ sind. Die Wirkungsweise der Filter und die Effekte der Parallelisierung sind auch mit anderen Sequenzen bemerkbar.

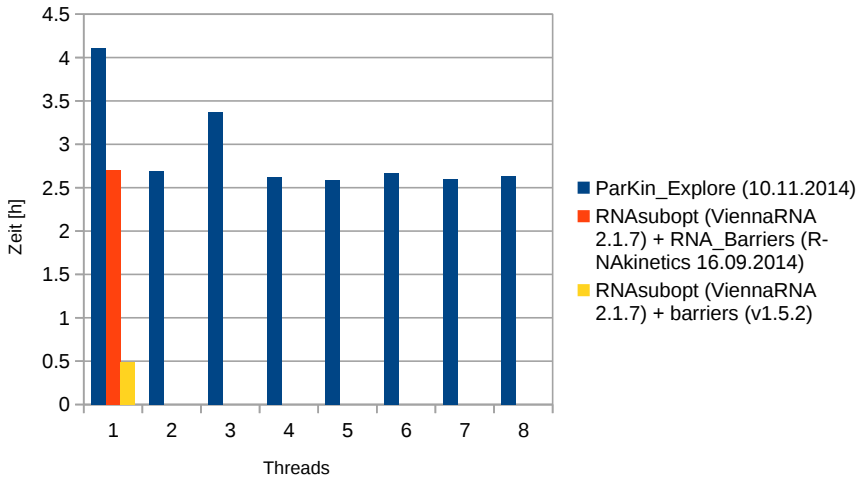


Abbildung 36: Sequenz: „d33“. Die Abbildung zeigt die Ausführungszeit für jedes Programm. Dabei ist zu beachten, dass beide Implementierungen von *Barriers* keine eigene Energieberechnung durchführen. Diese wird von *RNAsubopt* erledigt.

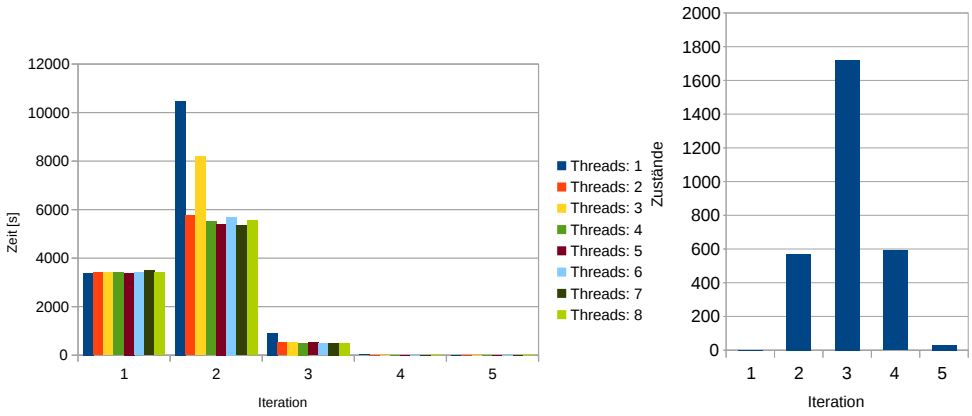


Abbildung 37: Sequenz: „d33“. Die linke Abbildung zeigt den Zeitverbrauch für die To-do-Liste in jeder Iteration von *ParKin_Explore*. Die rechte Abbildung zeigt die Anzahl der Minima in der To-do-Liste in jeder Iteration.

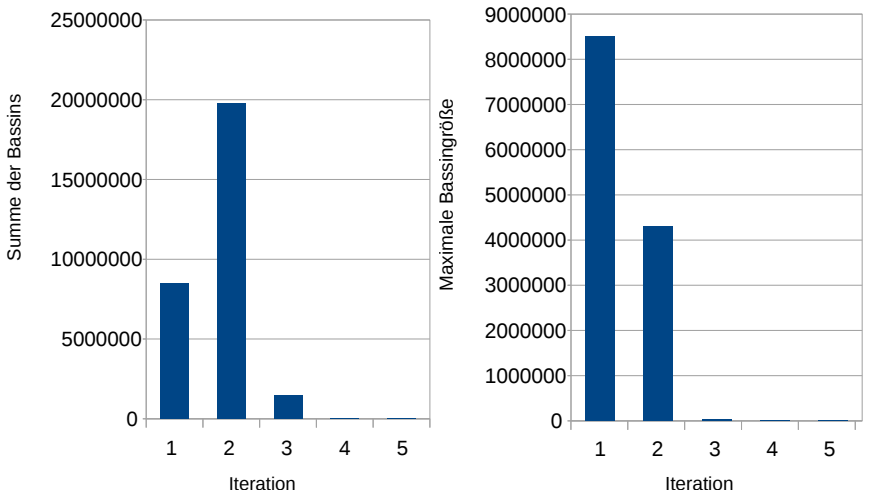


Abbildung 38: Sequenz: „d33“. Die linke Abbildung zeigt die kumulativen Bassingrößen in jeder Iteration. Sie stellt eine obere Schranke für die Ausführungszeit im parallelen Betrieb von *ParKin_Explore* dar. Die rechte Abbildung zeigt die Menge der Microstates des größten Bassins in jeder Iteration. Dies repräsentiert die untere Schranke für den Zeitverbrauch von *ParKin_Explore* im parallelen Betrieb.

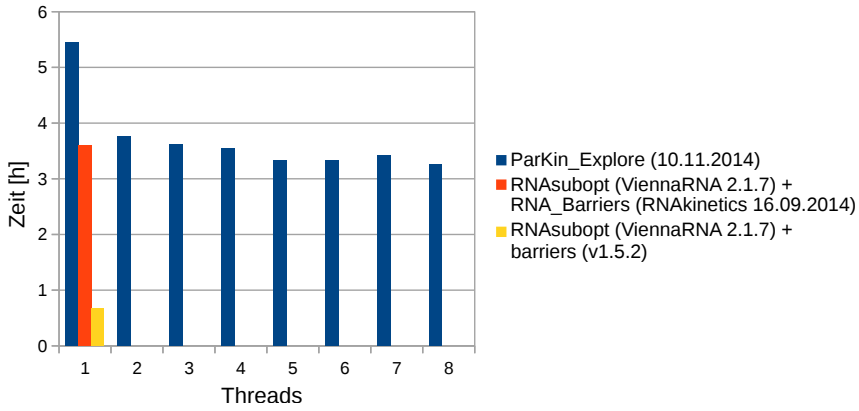


Abbildung 39: Sequenz: „ire“. Die Abbildung zeigt die Ausführungszeit für jedes Programm. Dabei ist zu beachten, dass beide Implementierungen von *Barriers* keine eigene Energieberechnung durchführen. Diese wird von *RNAsubopt* erledigt.

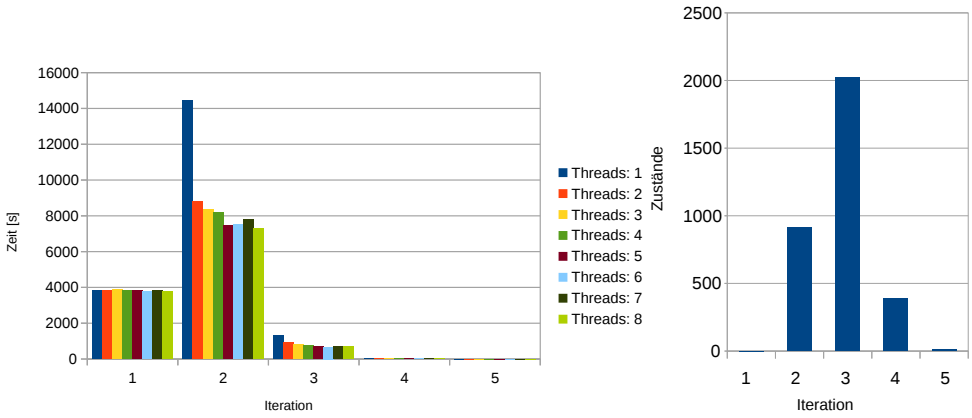


Abbildung 40: Sequenz: „ire“. Die linke Abbildung zeigt den Zeitverbrauch für die To-do-Liste in jeder Iteration von *ParKin_Explore*. Die rechte Abbildung zeigt die Anzahl der Minima in der To-do-Liste in jeder Iteration.

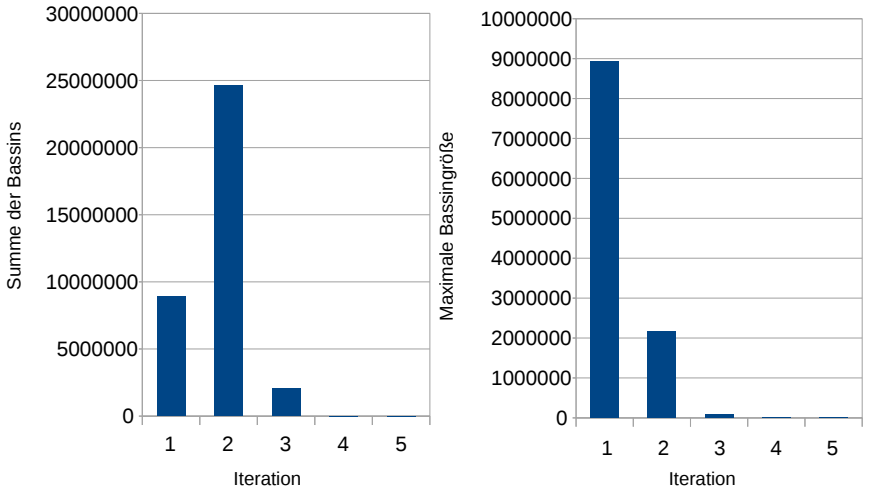


Abbildung 41: Sequenz: „ire“. Die linke Abbildung zeigt die kumulativen Bassingrößen in jeder Iteration. Sie stellt eine obere Schranke für die Ausführungszeit im parallelen Betrieb von *ParKin_Explore* dar. Die rechte Abbildung zeigt die Menge der Microstates des größten Bassins in jeder Iteration. Dies repräsentiert die untere Schranke für den Zeitverbrauch von *ParKin_Explore* im parallelen Betrieb.

A.4 ParKin_Explore HI-Shape Tests

Da die Ratenmatrix auch für mittellange Sequenzen, trotz Verwendung von Filtern, zu groß für eine Kinetikberechnung werden kann, wurde eine Lösung mit HI-Shapes entwickelt. Diese wurde in Kap. 7.4 diskutiert. In Abbildung 42 ist die Kinetik für die Sequenz „Leptomonas“ dargestellt. Dabei wurden 93 Macrostates auf 10 HI-Shapes reduziert. Im Vergleich mit der reinen HI-Shape Kinetik aus Abbildung 33 fällt auf, dass für die HI-Shapes [27] und [38] andere Sekundärstrukturen gefunden wurden. Die Sekundärstrukturen in Abbildung 42 haben jedoch eine kleinere freie Energie. Das typische RNA-switch Verhalten ist in dieser Kinetik nicht so stark ausgeprägt. Die Ursache könnte sein, dass zu viele wichtige Minima durch das Filtern noch nicht erfasst wurden. Eine andere Vermutung ist, dass durch die HI-Shape Abstraktion Minima zusammengefasst wurden, die in der Energielandschaft nicht benachbart sind.

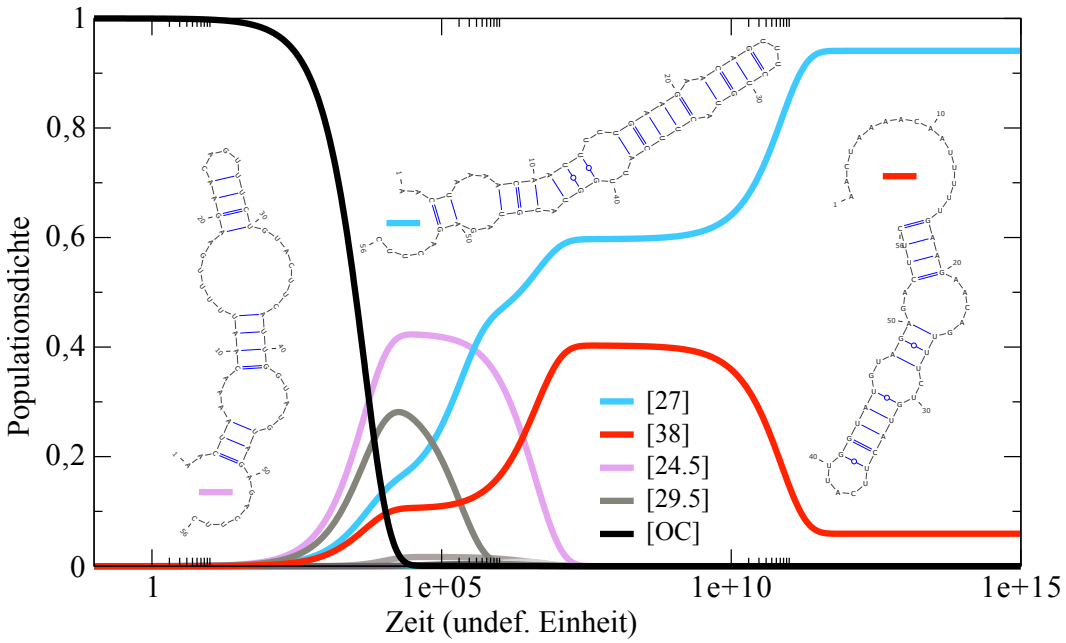


Abbildung 42: HI-Shape Kinetik mit „Leptomonas“ und gefilterter Berechnung mit ParKin_Explore. Die Kinetik wurde mit *ParKin_Explore* in 1,6 Sekunden berechnet und enthält 10 Macrostates (Parameter: Delta-Energie-Filter = 5 kcal/mol, Best-K Filter = 3, Threads = 8, Temperatur = 37°C, Dangles = 1, DeltaMinE-Filter = 0.5, Energiemodel = Turner 1999). Die verwendete Hardware ist ein Intel(R) Core(TM) i7-2630QM CPU @ 2.00GHz mit 4 Kernen und Hyperthreading und 14 GB RAM. Die Macrostates wurden zu HI-Shapes zusammengefasst.

Tabelle 4: Strukturenvergleich von „C-Di-GMP“ mit divers ermittelten Strukturen. Das Benchmark ist aus dem Anhang von [21]. Die HI-Shapes wurden hinzugefügt um die Ergebnisse mit *ParKin_Explore* und *HiKinetics* zu vergleichen.

Switch state	**Name**	**HI-Shape**	**Sequenz / Struktur**
	Sequenz		ACCCCGAAAGGGCAAACCGGUACGAAAGUCCGGGACGCAAAGCUACGGGUCCUUAAGUUCCAUGGGGAAUAGGACGGCUGAGCCGCUGGGGUUAUUACUUUCGCGGAGCCGCCCUAUGGGGCGG
OFF	Native	[25.5,63.5,116.5]	((((((.....((...((((.((....))))))...))...(((.((((((.....((((....))))...))).))))))...)))))).....
	mfold	[25.5,63.5,116.5]	((((((...((((((...((((.((....))))))).....((((....((((((...((((((....)))))))))))))..))))..))))))).....
	RNAShapes	[25.5,63.5,103,116.5]	((((((...((((((...((((.((....))))))).....((((....((((((...((((((....)))))))))))))..))))..))))))).....
	RNAlocopt	[7.5,25.5,63.5,94.5,116.5]	...((....))((...((((.((....))))))...))..((((....((((((...((((((....)))))))))))))...((((..((((....)))).))))..((((((....))))))
	RNASLOpt	[25,63.5,116.5]	((((((....(((...((((.........))))).....((((....((((((...((((((....)))))))))))))..))))..))))))).....
ON	Native	[25.5,94.5]	..........((...((((.((....))))))...)).....(((((.(((((...((((.(((((....)))))..))))))))))).))))).....
	mfold	[25.5,54,94.5]	.(((((.....((...((((.((....))))))...)).....(((.....))).)))))..(((((.(((((...((((..((((....)))).))))))))))).))))).....
	RNAShapes	[25.5,94.5]	..((((.....((...((((.((....))))))...)).....))))..(((((((((((.....(((((...((((..((((....)))).)))))))))))))))).....
	RNAlocopt	[25.5,94.5]	..((((.....((...((((.((....))))))...)).....))))..(((((((((((.....(((((...((((..((((....)))).)))))))))))))))).....
	RNASLOpt	[25,63,94.5]	.((((..(((((((...(((((.........))))).....))))))).)))...((((...)))))(((((.(((((...((((.(((((....)))))..))))))))))).)))))...
Most optimal	HiKinetics (mit Filter)	[25.5,94.5]	..((((.....((...((((.((....))))))...)).....))))..(((((.(((((...((((.(((((....)))))..))))))))))).))))).....
		[63.5]	N.A.
		[25.5,63.5,94.5]	N.A.
	ParKin_Explore (mit Filter)	[25.5,63.5,80,116.5]	.((.((((((.((...((((.((....))))))...))..(((..(((((((((...((((((....)))))))))))(((...)))..)))....)))))).))..((((((....))))))
		[7.5,25.5,63.5,94.5,116.5]	..(((....)))....((.(.((....)).).))..(((.((.((.(((..((((((....))))))))))..(((..((((((....)))))))))..)))(((((....))))).

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] I. Aviram, I. Veltman, A. Churkin, and D. Barash. Efficient procedures for the numerical simulation of mid-size RNA kinetics. *Algorithms for Molecular Biology : AMB*, 7:24–24, 2012.

[2] P. N. Borer, B. Dengler, I. Tinoco, and O. C. Uhlenbeck. Stability of ribonucleic acid double-stranded helices. *Journal of Molecular Biology*, 86(4):843–853, 1974.

[3] S.-J. Chen and K. A. Dill. RNA folding energy landscapes. *Proceedings of the National Academy of Sciences of the United States of America*, 97(2):646–651, 1999.

[4] P. Clote, E. Kranakis, D. Krizanc, and B. Salvy. Asymptotics of canonical and saturated RNA secondary structures. *Journal of Bioinformatics and Computational Biology*, 7(5):869–893, 2009.

[5] H. DeVoe and I. Tinoco. The stability of helical polynucleotides: Base contributions. *Journal of Molecular Biology*, 4(6):500–517, 1962.

[6] I. Dotu, W. A. Lorenz, P. Van Hentenryck, and P. Clote. Computing folding pathways between RNA secondary structures. *Nucleic Acids Research*, 38(5).

[7] C. Flamm. *Kinetic Folding of RNA*. PhD thesis, Universität Wien, August 1998.

[8] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. RNA folding at elementary step resolution. *Rna*, 6(03):325–338, 2000.

[9] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger. Barrier trees of degenerate landscapes. *Zeitschrift für Physikalische Chemie*, 216(2/2002), 2002.

[10] B. Fürtig. *NMR-Study of Dynamic Structural Transitions in RNA-Molecules*. PhD thesis, Johann Wolfgang Goethe-Universität Frankfurt am Main, Mai 2007.

[11] M. Geis, C. Flamm, M. T. Wolfinger, A. Tanzer, I. L. Hofacker, M. Middendorf, C. Mandl, P. F. Stadler, and C. Thurner. Folding kinetics of large RNAs. *Journal of Molecular Biology*, 379(1):160–173, 2008.

[12] W. Gilbert. Origin of life: The RNA world. *Nature*, 319(6055):618–618, 1986.

[13] C. M. Grinstead. *Introduction to probability*. American Mathematical Society, Providence, RI, 2nd rev. ed edition, 1997.

[14] B. Hübner. Kinetiken großer Energielandschaften. Master's thesis, Albert-Ludwigs-Universität Freiburg, Oktober 2013.

[15] I. L. Hofacker. Energy-directed RNA structure prediction. In J. Gorodkin and W. L. Ruzzo, editors, *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*, volume 1097, pages 71–84. Humana Press, Totowa, NJ, 2014.

[16] J. Huang and B. Voß. Analysing RNA-kinetics based on folding space abstraction. *BMC Bioinformatics*, 15(1):60, 2014.

[17] H. Isambert and E. D. Siggia. Modeling RNA folding paths with pseudoknots: Application to hepatitis delta virus ribozyme. *Proceedings of the National Academy of Sciences*, 97(12):6515–6520, 2000.

[18] B. Kirkpatrick, M. Hajiaghayi, and A. Condon. A new model for approximating RNA folding trajectories and population kinetics. *Computational Science & Discovery*, 6(1):014003, 2013.

[19] H. Kochniß. Ein Hybridkinetik Ansatz für RNA Faltungswahrscheinlichkeiten. Master's thesis, Friedrich-Schiller-Universität Jena, August 2008.

[20] N. B. Leontis and E. Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA (New York, N.Y.)*, 7(4):499–512, 2001.

[21] Y. Li and S. Zhang. Finding stable local optimal RNA secondary structures. *Bioinformatics*, 27(21):2994–3001, 2011.

[22] Y. Li and S. Zhang. Predicting folding pathways between RNA conformational structures guided by RNA stacks. *BMC Bioinformatics*, 13(Suppl 3):S5, 2012.

[23] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

[24] R. B. Lyngsø and C. N. Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 7(3-4):409–427, 2000.

[25] M. Mann and K. Klemm. Efficient exploration of discrete energy landscapes. *Physical Review E*, 83(1), 2011.

[26] M. Mann, M. Kucharik, C. Flamm, and M. T. Wolfinger. Memory-efficient RNA energy landscape exploration. *Bioinformatics*, 30(18):2584–2591, 2014.

[27] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288(5):911–940, 1999.

[28] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–1119, 1990.

[29] J. R. Proctor and I. M. Meyer. COFOLD: an RNA secondary structure prediction method that takes co-transcriptional folding into account. *Nucleic Acids Research*, 41(9):e102–e102, 2013.

[30] A. Ren, K. R. Rajashankar, and D. J. Patel. Fluoride ion encapsulation by mg2+ ions and phosphates in a fluoride riboswitch. *Nature*, 2012. 00022.

[31] K. Schmatzer. Berechnung der RNA-Partitionsfunktion mit algebraischer dynamischer Programmierung. Master's thesis, Albert-Ludwigs-Universität Freiburg, Februar 2012.

[32] N. Schnorr. Energy landscapes using crossing structure models. Master's thesis, Albert-Ludwigs-Universität Freiburg, 2014.

[33] P. Sibani, R. van der Pas, and J. C. Schön. The lid method for exhaustive exploration of metastable states of complex systems. *Computer Physics Communications*, 116(1):17–27, 1999.

[34] Y. Takahata, M. Nishijima, T. Hoaki, and T. Maruyama. Thermotoga petrophila sp. nov. and thermotoga naphthophila sp. nov., two hyperthermophilic bacteria from the kubiki oil reservoir in niigata, japan. *Int. J. Syst. Evol. Microbiol.*, 51(Pt 5):1901–1909, 2001.

[35] D. H. Turner and D. H. Mathews. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic Acids Research*, 38(Database):D280–D282, 2010.

[36] M. T. Wolfinger, W. A. Svrcek-Seiler, C. Flamm, I. L. Hofacker, and P. F. Stadler. Efficient computation of RNA folding dynamics. *Journal of Physics A: Mathematical and General*, 37(17):4731–4741, 2004.

[37] A. Xayaphoummine, T. Bucher, and H. Isambert. Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. *Nucleic Acids Research*, 33(Web Server):W605–W610, 2005.

[38] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science (New York, N.Y.)*, 244(4900):48–52, 1989.

[39] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.