Ein Hybridkinetik Ansatz für RNA Faltungswahrscheinlichkeiten

Diplomarbeit
zur Erlangung des akademischen Grades
Diplom-Bioinformatiker

Friedrich-Schiller-Universität Jena
Fakultät für Mathemathik und Informatik

eingereicht von Hannes Kochniß
geboren am 06.12.1980 in Erfurt

Gutachter:	Prof. Dr. Rolf Backofen
	Prof. Dr. Stefan Schuster
Betreuer:	Prof. Dr. Rolf Backofen
	Martin Mann
	Dr. Sebastian Will

Jena, 13. August 2008

Einen besonderen Dank an

Prof. Dr. Backofen, Prof. Dr. Schuster, Martin Mann, Dr. Anke Busch, Dr. Sebastian Will, Andreas Richter, Steffen Heyne, sowie Ulrike Kochniß.

Zusammenfassung

RNA(*ribonucleic acid*) wird heutzutage weithin als eines der bedeutendsten Biopolymere angesehen. Schon bevor das Interesse der Wissenschaft an diesem Molekül wieder aufflammte - durch Entdeckung von kleinen regulatorischen RNAs - wurde in der Bioinformatik verstärkt versucht, die RNA Struktur anhand der Sequenz vorauszusagen. Nach heutiger Ansicht bestimmt die Sequenz im wesentlichen die Struktur, und diese Struktur beeinflusst die Funktion der RNA maßgeblich. Die Strukturvorhersage funktioniert bei RNA im Vergleich zu Proteinen besser, da die bei RNA angewandten Energieberechnungen als ausgereift gelten und durch Experimente validiert wurden.

Es werden derzeit zwei wesentliche Ansätze für die Faltungsvorhersage verwendet. Zum einen die direkte Simulation des Faltungsprozesses, bei der über viele Iterationen hinweg stochastisch Wahrscheinlichkeiten für verschiedene Faltungen/Strukturen bestimmt werden. Dies ist die am häufigsten genutzte, aber auch bei weitem aufwendigste Methode. Daher wird oft auf Kinetiken ausgewichen, welche auf einer Vereinfachung der Energielandschaft basieren. Energielandschaften sind hierbei eine diskrete Beschreibung des Strukturraums einer RNA, in dem sich der Faltungsprozeß abspielt. Zum einen werden ganze Teile der Energielandschaft zu sogenannten Macrostates zusammengefasst, zum anderen wird die Landschaft häufig vereinfachend als BarrierTree dargestellt, wodurch Adjazenzinformationen zugunsten einer effizienten Berechnung verworfen werden.

In großen Landschaften kommt meist die Arrhenius Kinetik zum Einsatz, welche nur die Übergangshöhen der Macrostates, durch Sampling gefunden, nutzt. Eine durch Fluten der Landschaft mögliche Macrostatekinetik ist in solchen Fällen aufgrund von Speicherrestriktionen nur sehr eingeschränkt möglich, bietet aber in diesem eingeschränkten Bereich der Landschaft deutlich genauere Resultate.

In dieser Arbeit wird ein Hybridansatz vorgestellt, welcher die beiden Kinetiken miteinander verknüpft. Für den unteren Bereich der Landschaft wird die Macrostatekinetik verwendet, während im oberen Bereich durch ein Sampling der Übergangshöhen eine Arrheniuskinetik möglich wird. Diese beiden Kinetiken arbeiten jedoch mit unterschiedlichen Zeitfaktoren, so dass eine Skalierung der jeweiligen Ratenmatrizen nötig ist, um die resultierende Ratenmatrix zu verwenden.

Die Arbeit untersucht mehrere Möglichkeiten der Hybridisierung beider Kinetiken, und zeigt grundsätzliche Limitierungen des Kinetikansatzes auf. Zudem wird eine Metrik für den Vergleich von Kinetiken vorgestellt, um optische Unterschiede im Faltungsverhalten zu quantifizieren. Dieses Maß wird schließlich verwendet, um die Qualität der Hybridkinetik zu bewerten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

RNA(*ribo nucleic acid*) ist ein wichtiges Biopolymer, welches größere Aufmerksamkeit erlangte, nachdem ihm durch verschiedene Forschungsergebnisse der letzten Jahre ein immer bedeutenderer Funktionsumfang nachgewiesen wurde. Seit der Entdeckung des ersten Ribozyms 1980 begann die ursprüngliche Betrachtung der RNA als simpler Bote zwischen DNA und Proteinen zu schwinden. So stellte Walter Gilbert 1986 die RNA-Welthypothese [Gil86] auf, die besagte, dass RNA-Moleküle in der chemischen Evolution die Vorläufer der heutigen Organismen waren. Diese Theorie wurde seitdem durch verschiedene Befunde unterstützt. Die Entdeckung der RNA Interferenz (RNAi) 1998 [FXM+98] und der microRNA (miRNA) 2001 wandelte schliesslich endgültig die Perspektive dieses Moleküls als „dummer“ Informationsbote zum bedeutenden regulatorischen Element, was schließlich vom Science Magazin als „breakthrough of the year 2002“ [Cou02] bezeichnet wurde. Im Jahr 2006 verlieh man den Nobelpreis der Medizin und Nobelpreis der Chemie für Entdeckungen im Bereich der RNA.

Auch in der Bioinformatik haben die RNA und die damit verbundenen Problemstellungen - Faltung, Alignment, Targetvorhersage - längst einen großen Stellenwert erlangt. Da allgemein bekannt ist, dass die Funktion der RNA direkt an ihre gebildete Struktur (und damit an die Entstehung von Bindungsstellen) gekoppelt ist, wird dem Problem der Faltungsvorhersage eine besondere Bedeutung beigemessen.

Bei der RNA Faltung gilt eines der ersten Probleme - die energetisch optimale Faltung zu finden - längst als gelöst [Zuk89, MSZT99]. Diese gefundene Faltungs ist zwar nur zweidimensional und unterstützt keine Pseudoknoten (welche derzeitiger Forschungsgegenstand sind), jedoch hat sich herausgestellt dass zwischen der berechneten Struktur und der in vitro gefundenen dreidimensionaler Struktur nur einige zusätzliche Bindungen liegen. Es ist zwar möglich, die Wahrscheinlichkeiten aller suboptimalen Faltungen nur auf Grund von Energieparametern zu berechnen, allerdings ist dies ein kurzsichtiger Ansatz. Zum einen ist so nicht klar, ob die Lebenszeit des Moleküls für das Erreichen der Faltung überhaupt ausreicht. Und zum anderen bilden RNAs oft metastabile Zustände, falten dabei nicht nur in eine Struktur sondern wechseln z. B. nach einer gewissen Zeitspanne ihre Konformation, mit danach teils völlig unterschiedlicher Funktion [BSR97, SBS04]. Thermodynamische Berechnungen allein können solch ein Verhalten nicht aufzeigen, da sie den Zeitverlauf nicht betrachten. Dies ist der Grund für die Verwendung von Faltungsdynamiken.

Für diese gibt es zwei wesentliche Ansätze. Zum einen die stochastische Faltungssimulation, welche die qualitativ besten Ergebnisse liefert, allerdings auch sehr aufwendig ist, da für aussagekräftige Resultate sehr viele Simulationsläufe erforderlich sind. Zum anderen

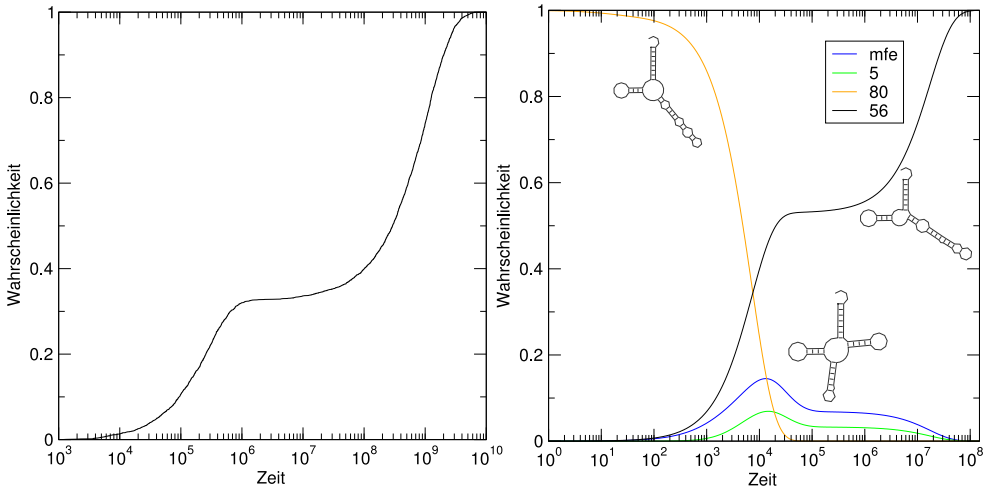


Abbildung 1.1: Vergleich einer stochastischen Simulation (links) und einer berechneten Kinetik (rechts) (aus [WFHS04]); die linke Abbildung nahm drei Monate CPU Zeit auf einem 2,4GHz P4 in Anspruch, das rechte nur 1 Minute, die qualitative Aussage beider ist trotz quantitativer Unterschiede ähnlich.

Faktungskinetiken auf Basis von Markovprozessen, welche die Wahrscheinlichkeitsverläufe über die Zeit berechnen, auf Basis einer meist vereinfachten Energielandschaft. Mit diesem simpleren Ansatz ist es möglich auch für relativ große RNAs in vertretbarer Zeit Faltungswahrscheinlichkeiten zu bestimmen.

Diese Diplomarbeit befasst sich letztendlich mit dem zweiten Ansatz, den Faltungskinetiken. Zum derzeitigen Zeitpunkt wird im Regelfall das Programm `RNAsubopt` [HFS+94, WFHS99] verwendet, um zuerst die gesamte Anzahl an möglichen Strukturen/Faltungen aufzuzählen. Danach werden diese Strukturen mit `barriers` [WFHS04] sequentiell abgearbeitet, um schließlich einen BarrierTree, eine vereinfachte Darstellung der Energielandschaft, zu erzeugen. Anschließend wird mit Hilfe des tools `treekin` [Wol01] eine Kinetik berechnet. Bei diesem Ansatz limitiert der Speicherverbrauch von `RNAsubopt` jedoch die Anzahl der betrachteten Strukturen und somit bei großen Landschaften den nutzbaren Datensatz für die Faltungskinetiken.

Der in dieser Diplomarbeit vorgestellte Hybridkinetikansatz umgeht diese Speicherrestriktion. Zuerst wird wie bei `barriers` die Landschaft soweit betrachtet wie die Speicherrestriktion es zulässt, um dann jedoch für höhere Bereiche der Energielandschaft den in [Ric07] vorgestellten, nicht speicherlimitierten Samplingansatz zu verwenden. Dadurch kann ein größerer Bereich der Energielandschaft betrachtet werden und somit auch zu einer besseren Faltungskinetik führen. Die Kombination der beiden erzeugten Datensätze (durch das sog. „Fluten“ und das darauf folgende Sampling) erfordert jedoch auch eine Kombination zweier Kinetikalgorithmen. Diese sinnvoll zu vereinen ist die Hauptaufgabe, um letztendlich bessere Ergebnisse als z. B. `barriers` zu erzeugen.

1.2 Vorhergehende Arbeiten

In der Vergangenheit wurden einige für diese Diplomarbeit relevante Arbeiten zu dem Thema RNA Faltung veröffentlicht.

Flamm et al. präsentierten in [FFHS00] einen stochastischen Faltungsalgorithmus zur Simulation der Faltung von RNA Molekülen, welcher im Programm `kinfold` implementiert wurde. Die hierbei benutzte Stochastik ist jedoch extrem rechenintensiv. Dennoch wurde somit ein Faltungsalgorithmus auf Basis von Single Moves (Deletion und Insertion von Basenpaaren) vorgestellt und erfolgreich angewandt. Zudem wurden BarrierTrees verwendet, um deren strukturelle Repräsentation der Energielandschaft erfolgreich für die Abschätzung des Faltungsverhaltens zu nutzen. Dies untermauert die Annahme, dass BarrierTrees sich als Modell für die Berechnung von Faltungskinetiken eignen.

Später wurden von derselben Gruppe in [FHSW02] BarrierTrees auf degenerierten Landschaften wie z.B. bei RNA formal definiert. Das dabei vorgestellte tool `barriers` baut BarrierTrees durch umfangreiche Aufzählung aller validen Strukturen, dem sogenannten Fluten, auf. Dadurch ist es allerdings speicherlimitiert und für große Energielandschaften auf den unteren Teil der Landschaft beschränkt.

2004 wurde mit Hilfe dieses Tools in [WFHS04] gezeigt, wie mit Hilfe von numerischer Integration durch das tool `treekin` Kinetiken auf Basis von BarrierTrees berechnet werden. Die Ergebnisse wurden mit einer Faltungssimulation des tools `kinfold` verglichen und entsprechend qualitativ ähnliche Ergebnisse ermittelt, bei nur einem Bruchteil der Rechenzeit. Der in dieser Diplomarbeit beschriebene Ansatz nutzt die gleiche numerische Integration bzw. das tool `treekin`.

In der Diplomarbeit von Andreas Richter [Ric07] wird beschrieben, wie BarrierTrees einer Energielandschaft ohne vollständige Aufzählung von Konformationen effektiv gesampled werden können. Dies bietet im Gegensatz zum Ansatz in [FHSW02] den Vorteil, dass BarrierTrees beliebig großer Landschaften erzeugt werden können. Dieser Algorithmus wird in der vorliegenden Diplomarbeit genutzt, um BarrierTrees für Arrhenius- und somit auch Hybridkinetiken zu samplen.

Morgan und Higgs zeigten in [MH98] wie mit Hilfe einer heuristischen Methode Barrieren zwischen verschiedenen RNA Strukturen geschätzt werden können. Mit Hilfe dieser Barrieren ist es möglich BarrierTrees zu initialisieren um das Sampling der Landschaft zu beschleunigen und den in [Ric07] vorgestellten Ansatz zu erweitern.

Eine zum BarrierTree alternative Betrachtungsweise der Energielandschaft von RNA beschrieben Klemm, Flamm und Stadler in [KFS08]: „folding funnels“. Diese erlauben eine alternative Modellierung der Energielandschaft, welche anscheinend eine bessere Darstellung des Faltungsverhaltens ermöglichen als z.B. Barrier Trees.

Ein guter Überblick über diese und andere Ansätze ist in [FH08] zusammengefasst.

1.3 Wissenschaftlicher Beitrag

Der hier vorgestellte Hybridansatz soll eine echte Alternative zum von Wolfinger et al. in [WFHS04] beschriebenen Ansatz der vollständigen Betrachtung einer Energielandschaft darstellen. Der dort beschriebene Algorithmus schränkt den resultierenden BarrierTree und die damit verbundene Kinetik stark ein.

Im Gegensatz dazu wird hier durch das in [Ric07] beschriebene Sampling die Einschränkung auf den unteren Teil der Energielandschaft aufgehoben. Außerdem wird ein Flutalgorithmus verwendet der speichereffizienter ist und daher ein vollständiges Betrachten größerer

Landschaften erlaubt als `barriers`. Beide Ansätze, Sampling und Fluten, werden für eine bessere Topologie der Energielandschaft genutzt, mit exakten Daten bis zur Flutgrenze und gesampleten Daten darüber.

Die qualitativ höherwertige Macrostate Kinetik, welche nur durch das Fluten der Energielandschaft möglich ist, wird mit einer mit Samplingdaten arbeitenden Arrheniuskinetik zu einer Hybridkinetik verbunden. Deren erklärtes Ziel ist es, die einzeln angewendeten Arheniuskinetiken und durch Speicherrestriktionen beschränkten Macrostatekinetiken vergangener Publikationen zu übertreffen. Dies ist möglich, da mit genaueren Daten der Energielandschaft gearbeitet wird und die beiden Ansätze, die Verwendung der genauen Macrostatekinetik im niedrigenergetischen Bereich und der schlechteren Arrheniuskinetik im hochenergetischen Bereich, miteinander kombiniert werden.

Zusätzlich wird ein Maß zum Vergleich der Kinetiken vorgestellt, welches auf den Wahrscheinlichkeitsverläufen beruht und invariant gegenüber einer Skalierung der Ratenmatrizen ist. Es beschreibt den qualitativ sichtbaren Unterschied der Kinetiken.

Die beschriebenen Algorithmen sind allesamt mit Hilfe der **Energy Landscape Library** [MWB07] implementiert und in diese integriert.[1]

1.4 Übersicht

In Kapitel 2 wird die theoretische Basis für die späteren Algorithmen gebildet. Das Kapitel klärt zuerst die Grundkonzepte der Energielandschaft und Walks. Dann werden die Energielandschaftsmodelle BarrierTree und SaddleNet erläutert, um letztendlich die bekannten und relevanten Kinetiken näher zu betrachten. Algorithmen um Daten für die Transitionsraten zu gewinnen. Danach geht es um die Berechnung der einzelnen Ratenmatrizen für die verschiedenen bekannten Kinetiken und es werden schließlich die verschiedenen Ansätze beschrieben, die aus diesen Matrizen eine Hybridkinetik modellieren.
Kapitel 4 fängt mit einer Betrachtung zum verwendeten Fehlermaß an und vergleicht später die einzelnen Kinetiken sowie deren Eigenschaften. Schließlich analysieren wir die Effizienz der Hybridkinetik und inwiefern sie den bisherigen Kinetiken überlegen ist.
In Kapitel 5 werden die Ergebnisse letztendlich bewertet und offene Probleme sowie mögliche Erweiterungen des Themas diskutiert.

[1] Die ELL ist unter http://www.bioinf.uni-freiburg.de/SW/ELL/ frei verfügbar

Kapitel 2

Grundlagen

In diesem Kapitel werden die grundlegenden Konzepte definiert und erläutert. So wird zuerst in Kapitel 2.1 die RNA behandelt, danach in 2.2 alles relevante zu Energielandschaften und in Abschnitt 2.3 deren Repräsentationen, BarrierTrees und SaddleNets, erklärt. Letztendlich wird auf die bekannten Faltungskinetiken in Kapitel 2.4 näher eingegangen.

2.1 RNA (Ribonukleinsäure)

Die in der vorliegenden Arbeit beschriebenen Algorithmen werden hier ausschließlich auf RNA angewendet. Hauptsächlich da die umfassende Betrachtung der Energielandschaft durch eine ausgereifte und effiziente Energieberechnung für RNA Sekundärstrukturen (auf Basis von [McC90]) möglich ist und die bewährten tools des **Vienna RNA Packages**[1] verfügbar sind [HFS+94].
Der Ansatz lässt sich jedoch generell auf beliebige Modelle anwenden (Ising Spin, Protein, etc.), sofern für die entsprechenden Strukturen des Modells eine Energieberechnung, Aufzählung und Nachbarschaftsfunktion gegeben sind.

Zur Beschreibung der RNA gibt es vielfältige Modelle, wie in [SS01] beschrieben. Diese variieren zwischen sehr detailliert, die 3D Positionen einzelner Atome wiedergebend, bishin zu einfachen Darstellungen der Sekundärstruktur oder im simpelsten Fall nur der Darstellung der Sequenz ohne Strukturinformation. Zuerst wird der generelle Aufbau und im darauffolgenden Teil das später verwendete Darstellungsmodell der Sekundärstruktur und die zugrundeliegenden Primärstruktur näher betrachtet.

Aufbau

Die RNA ist ein Polynukleotid. Jedes der einzelnen Nukleotide der Kette besteht aus einer Ribose (einem Zucker mit fünf C-Atomen), einer organischen Base sowie einem Phosphatrest. Abbildung 2.1 zeigt den grundlegenden Aufbau der Basen und ihre Verbindung untereinander.

Die Ribose der RNA hat im Gegensatz zur Desoxyribose der DNA eine Hydroxyl-Gruppe an der 2'-Position des Pentose-Rings. Dieser Unterschied macht RNA weniger stabil als DNA, da es eine Hydrolyse durch Basen ermöglicht. Somit ist die RNA ein vergleichsweise kurzlebiges Molekül, ein wesentlicher Grund für die Anwendung von Faltungskinetiken, da diese Faltungswahrscheinlichkeiten zu jedem Zeitpunkt bieten.

[1] Das Vienna RNA Package steht unter http://www.tbi.univie.ac.at/RNA/ zur freien Verfügung.

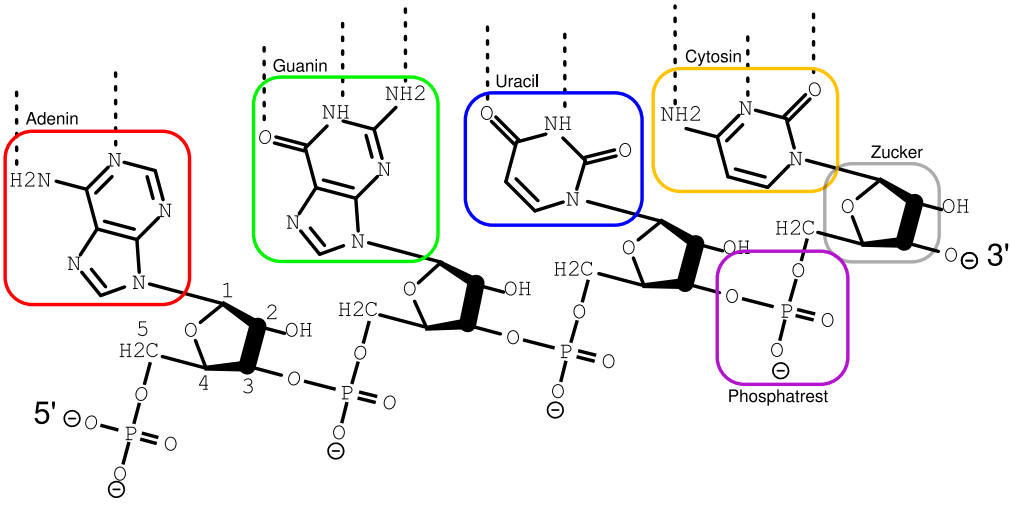


Abbildung 2.1: Detaillierte Betrachtung der RNA Sequenz **AGUC**; links das 5' Ende (5. Kohlenstoffatom) welches den Start der Sequenzangabe festlegt; gestrichelte Linien zeigen *mögliche* Bindungen zu anderen Basen.

Die Basen, die in der RNA verwendet werden, sind zum einen die Purine Adenin (A) und Guanin (G), zum anderen die Pyrimidine Cytosin (C) und Uracil (U). Die Existenz der weiteren möglichen Basen Xanthin und Hypoxanthin, welche sich nur unter Einwirkung von Mutagenen bilden, wird im verwendeten Modell vernachlässigt. Im folgenden werden die Basen nur noch durch ihre Anfangsbuchstaben repräsentiert.

RNA-Moleküle liegen im Gegensatz zur doppelsträngigen DNA in der Regel als Einzelstrang vor. Diese Eigenschaft führt zu einer Vielzahl an unterschiedlichen dreidimensionalen Strukturen und erlaubt ihr im Gegensatz zur zweisträngigen DNA spezielle chemische Reaktionen, einer der Gründe für die vielfältigen Funktionen innerhalb der Zelle. So wird sie zur Informationsübertragung, Katalyse von Stoffwechselwegen und zur Regulation der Genexpression gebraucht. Ein vielversprechender Ansatz der Gentherapie basiert auf der Verwendung von small interfering RNA (siRNA).

Primärstruktur

Die Primärstruktur der RNA lässt sich wie folgt formal definieren:

Definition 2.1.1 (Primärstruktur p)
Sei Σ die Menge der relevanten Nukleotide, $\Sigma = \{A, C, G, U\}$. So sei die Primärstruktur p einer RNA eine Sequenz von Nukleotiden, $p \in \Sigma^* \backslash \emptyset$,

$$p = \langle p_1, p_2, \ldots, p_n \rangle \text{ mit } p_i \in \Sigma,\ 1 \leq i \leq n,\ \text{Länge } |p| = n \in \mathbb{N}_1 \tag{2.1}$$

Die Primärstruktur gibt also lediglich die Sequenz wieder und ist die Basis der Sekundärstruktur. Die Begriffe Sequenz und Primärstruktur werden im folgenden synonym verwendet. Viele Algorithmen operieren nur auf der Primärstruktur, da angenommen wird, dass diese die daraus folgende Sekundärstruktur alleinig bestimmt und daher z.B. zum Vergleich von Sequenzen ausreicht. Da wir jedoch speziell RNA Faltung behandeln, brauchen wir ein Modell welches diese auch widerspiegelt.

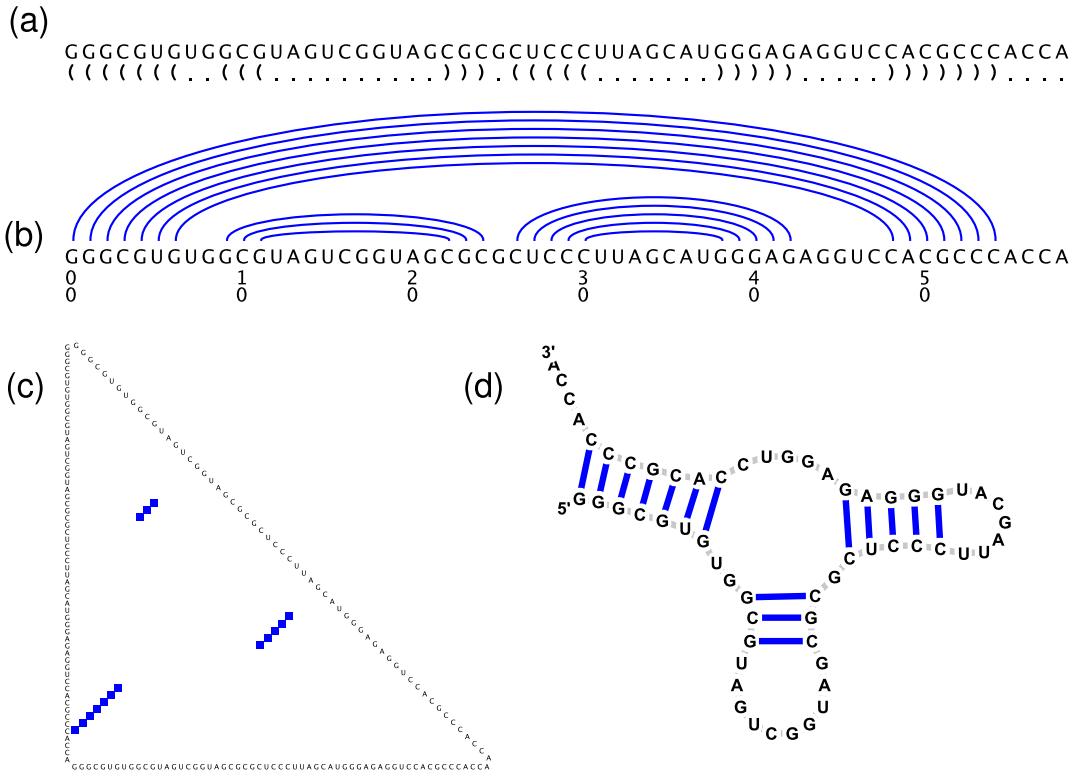


Abbildung 2.2: Verschiedene Darstellungen einer künstlich generierten Sekundärstruktur: (a) dot-bracket Notation, (b) Lineares Feynman Diagram, (c) DotPlot, (d) Klassischer Strukturplot; die verwendete Struktur besitzt offensichtlich keine Pseudoknoten (keine Überkreuzungen in (b)).

Sekundärstruktur

Wie schon erwähnt liegt RNA im Regelfall einsträngig vor und besitzt viele Möglichkeiten 3D Strukturen auszubilden, sprich verschiedene Faltungen anzunehmen. Dabei bildet sie Wasserstoffbrückenbindungen zwischen eigenen Basen aus. Es sind Bindungen zwischen A-U und G-C (Watson-Crick Basenpaarung) möglich als auch - seltener - zwischen G-U (eine energetisch schwache wobble Basenpaarung). Die daraus resultierende 2D-Repräsentation dieser internen Brückenbildung nennen wir Sekundärstruktur.

Definition 2.1.2 (Sekundärstruktur s, offene Kette $\hat{s}_{oc}$)
Sei p eine gegebene Primärstruktur. Sei Ω *die Menge der erlaubten Basenpaarungen,*

$$\Omega = \{(A,U),(U,A),(G,C),(C,G),(G,U),(U,G)\} \tag{2.2}$$

Dann sei s eine zu p kompatible Sekundärstruktur wenn gilt

$$s = \{(i,j) \mid 1 \leq i < j \leq |p|, (p_i, p_j) \in \Omega\} \tag{2.3}$$

wobei $(i,j) \in s$ *eine Basenpaarbindung respräsentiert und zusätzlich gilt*

$$\forall (i,j),(i',j') \in s : i = i' \Leftrightarrow j = j' \tag{2.4}$$

also eine Base nur an maximal einer Basenpaarung beteiligt ist.
Wenn ausserdem gilt

$$\forall (i,j),(i',j') \in s : \neg(i < i' < j < j' \text{ oder } i' < i < j' < j) \tag{2.5}$$

bezeichnet man die Sekundärstruktur als ***„noncrossing“*** *bzw.* ***pseudoknotenfrei****, andernfalls als* ***„crossing”*** *Struktur oder* ***Pseudoknotenstruktur****. Die Attribute „noncrossingünd „crossing”beziehen sich dabei auf ein Überkreuzen jeglicher Bindungslinien in einer planaren Graphendarstellung wie in* Abbildung 2.2(b)*. Eine* ***erlaubte*** *Sekundärstruktur ist eine pseudoknotenfreie Sekundärstruktur für die zusätzlich gilt*

$$\forall(i,j) \in s : |j-i| > 3 \tag{2.6}$$

sprich dass zwischen öffnender und schließender Base mindestens drei weitere Basen liegen müssen, was für das „Umbiegen“ der Kette notwendig ist.
Als ***offene Kette*** $\hat{s}_{oc}$ *bezeichnen wir diejenige Sekundärstruktur welche keine Basenpaarbindungen besitzt,* $\hat{s}_{oc} = \emptyset$*.*

Wenn wir von Sekundärstrukturen sprechen, meinen wir ab sofort nur noch **erlaubte** Sekundärstrukturen, siehe Definition 2.1.2. In Abbildung 2.2 werden einige übliche Darstellungsweisen von Sekundärstrukturen gezeigt. Die dortige Sekundärstruktur enthält keine Pseudoknoten (dies wären Überkreuzungen in Abbildung 2.2(b)) sondern ist eine noncrossing bzw. pseudoknotenfreie Struktur. Pseudoknoten wurden oft bei Algorithmen auf RNA-Sekundärstrukturen „außen vor“ gelassen, da die meisten Probleme auf solchen Strukturen NP-vollständig sind. So auch bei den hier beinhalteten Problemen, denn die Bestimmung der optimalen crossing Struktur mit Pseudoknoten ist ein NP-vollständiges Problem [LP00, JLMZ02]. Zudem sind, im Gegensatz zu „noncrossing“ Strukturen, keine genauen Energieparameter bekannt, um die freie Energie einer Struktur zu berechnen.

Betrachtet man die Faltung der RNA im 3D Raum so spricht man von der Tertiärstruktur der RNA. Diese wäre ein besseres Modell für die Faltungsanalyse, jedoch ist keine wissenschaftlich fundierte Nachbarschaftsgenerierung (siehe die spätere Definition 2.2.2 von MoveSet) vorhanden.

Energieberechnung

Die von uns in dieser Arbeit verwendete Energiefunktion f_E ist aus dem **Vienna RNA Package** [HFS+94] entnommen. Diese basiert im wesentlichen auf dem Algorithmus von McCaskill (siehe [McC90]), mit einem verfeinerten Parameterset aus [WTK+94] und [MSZT99]. Gemäß dieser Parameter besitzt die offene Kette, also die Struktur ohne Basenpaarbindungen, die Energie $0\frac{kcal}{mol}$.

2.2 Die Energielandschaft

Das Modell der Energielandschaft wird schon seit langem verwendet, in erster Linie bei der Untersuchung von Faltungen bzw. Faltungseigenschaften. Es ermöglicht die Simulation eines Faltungsvorganges als „Wanderung“ bzw. Traversierung der Landschaft. Dabei sind einige aus 3D Landschaften wie Gebirgen bekannte Begriffe wie „Sattel“ und „Minimum“ auch in die komplexere Energielandschaft übertragbar. Für die Erklärung der hier betrachteten Landschaftseigenschaften ist dies ausreichend, jedoch sollte man sich bewusst sein dass durch die nicht konsistente Nachbaranzahl zwischen verschiedenen Strukturen nicht von einer fixen n-dimensionalen Landschaft gesprochen werden kann. Eine bildliche Vorstellung einer 3D Landschaft ist somit teilweise irreführend, denn benachbarte Strukturen

können völlig unterschiedliche Energien besitzen (die „Höhe“ in einer 3D Landschaft und somit dort ein Maß für Nachbarschaft). Somit ist auch die Nutzung einer Art „Koordinatensystem“ basierend auf einem Hypercube nicht umsetzbar.
Im folgenden werden die Begriffe „Faltung“, „Konformation“, „Sekundärstruktur“ und „Struktur“ synonym verwendet.

Um die Energielandschaft definieren zu können, müssen vorher die Begriffe **Konformationsraum**, **Nachbarschaftsrelation** und **Energiefunktion** erläutert werden.

Der **Konformationsraum** umfasst alle erlaubten Konformationen bzw. Strukturen.

Definition 2.2.1 (Konformationsraum $\mathcal{X}_p$)
Sei p eine Primärstruktur. Der Konformationsraum $\mathcal{X}_p$ der Primärstruktur p sei wie folgt definiert:

$$\mathcal{X}_p = \{s : s \text{ ist eine erlaubte Sekundärstruktur auf } p\} \tag{2.7}$$

Desweiteren basiert die Energielandschaft auf einer **Nachbarschaftsrelation**, welche wiederum durch ein **MoveSet** definiert wird. Ein solches ist eine Menge von erlaubten Operationen bzw. **Moves**, wobei immer eine dieser Operationen auf eine Struktur angewandt diese verändert und somit eine „Nachbarstruktur“ generiert. Diese Änderung der Struktur kommt einem „Schritt“ in der Energielandschaft gleich.

Definition 2.2.2 (Move, MoveSet $\mathcal{M}$)
*Gegeben eine Struktur $s \in \mathcal{X}_p$. Sei ein **Move** $\boldsymbol{m}$ definiert als*

$$m : \mathcal{X}_p \to \mathcal{X}_p, m(s) \neq s \tag{2.8}$$

*also eine Abbildung einer Struktur auf eine andere. Die Menge aller erlaubter Moves wird als **MoveSet** $\mathcal{M}$ bezeichnet. Wenn zusätzlich gilt*

$$m(s) \in \{s \backslash (i,j) \mid (i,j) \in s\} \cup \{s \cup (i,j) \mid (i,j) \notin s\} \tag{2.9}$$

*also $\boldsymbol{m}$ eine Deletion oder Insertion einer Basenpaarbindung darstellt, so wird $\boldsymbol{m}$ als **SingleMove** bezeichnet und $\mathcal{M}$ als **SingleMoveSet**. Wenn das SingleMoveSet um Verschiebungen von Basenpaarendung (**ShiftMoves**) erweitert wird, so nennen wir dies **ShiftMoveSet**.*

Das ShiftMoveSet wird in [Fla98] näher beschrieben. In den folgenden Algorithmen wird ausschließlich ein SingleMoveSet verwendet. Das Programm `barriers` kennt sowohl ShiftMoveSet als auch SingleMoveSet. Es sind aber auch noch weitere MoveSets bekannt, z. B. die Erstellung und Auflösung größerer Sekundärstrukturelemente wie *bulges*, *hairpin loops* u. ä., jedoch werden diese Ansätze in dieser Arbeit nicht näher betrachtet. Die später vorgestellten Algorithmen lassen sind jedoch ohne weiteres mit beliebigen MoveSets verwenden.

Die **Nachbarschaft** lässt sich dank des MoveSets einfach definieren. Zwei RNA Strukturen gelten innerhalb einer Energielandschaft als benachbart, wenn die eine Struktur aus der anderen Struktur durch Anwendung eines Moves des MoveSets erzeugt werden kann. Daraufhin lässt sich eine symmetrische Relation, die **Nachbarschaftsrelation**, definieren.

Definition 2.2.3 (Nachbarschaft $\mathcal{N}$, Nachbarschaftsrelation $\sim$)
Gegeben zwei Strukturen $s_i, s_j \in \mathcal{X}_p$ und ein MoveSet $\mathcal{M}$. Die ***Nachbarschaft*** *$\mathcal{N}(s)$ ist definiert durch*

$$\mathcal{N}(s) = \{m(s) \mid m \in \mathcal{M}\} \tag{2.10}$$

Die symmetrische ***Nachbarschaftsrelation*** *$\sim$ sei wie folgt definiert:*

$$s_i \sim s_j \Leftrightarrow s_i \in \mathcal{N}(s_j) \text{ oder } s_j \in \mathcal{N}(s_i) \tag{2.11}$$

Eine **Energiefunktion** weisst jeder Struktur des Konformationsraumes $\mathcal{X}_p$ ihre freie Energie zu, also eine reelle Zahl.

Definition 2.2.4 (Energiefunktion f_E)
Gegeben ein Konformationsraum $\mathcal{X}_p$. Sei f_E die ***Energiefunktion*** *für die gilt*

$$f_E : \mathcal{X}_p \to \mathbb{R} \tag{2.12}$$

und es gelte ebenso, dass $f_E(s)$ die freie Energie der Struktur s darstellt. Demzufolge steht ein relativ niedriger Energiewert für eine relativ hohe strukturelle Stabilität.

Die von uns verwendete Energiefunktion für RNA basiert auf den **Turner Regeln** [XSB+98, MSZT99]. Dies bedeutet unter anderem, dass die Struktur $\hat{s}_{oc}$ ohne jegliche Basenpaarbindungen, auch die „offene Kette“ genannt, immer Energie 0 besitzt, und verschiedene Strukturen einen gleichen Energiewert besitzen können.

Nun ist es uns möglich mit Hilfe der vorherigen Definitionen die **Energielandschaft** formal zu beschreiben.

Definition 2.2.5 (Energielandschaft $\mathfrak{E}_p$)
Sei $\mathcal{X}_p$ als Konformationsraum der Primärstruktur p gegeben. Sei $\sim$ eine Nachbarschaftsrelation und f_E eine Energiefunktion. So sei die Energielandschaft $\mathfrak{E}_p$ der Primärstruktur p definiert als Tupel

$$\mathfrak{E}_p = (\mathcal{X}_p, \sim, f_E) \tag{2.13}$$

Wenn Strukturen gleicher Energie vorkommen, wird die Landschaft als **degeneriert** bezeichnet, ansonsten als **nichtdegeneriert**. Aufgrund der in dieser Arbeit verwendeten Energiefunktion für RNA Sekundärstrukturen Abschnitt 2.1, sind die hier betrachteten Energielandschaften generell degeneriert.

Definition 2.2.6 (Degeneriertheit einer Energielandschaft)
Gegeben der Konformationsraum $\mathcal{X}_p$ und eine Energiefunktion $f_E : \mathcal{X}_p \to \mathbb{R}$. Wenn gilt

$$\exists s_i, s_j \in \mathcal{X}_p : s_i \neq s_j \text{ und } f_E(s_i) = f_E(s_j) \tag{2.14}$$

so bezeichnet man die Energielandschaft $\mathfrak{E}_p$ als ***degeneriert****, sonst als* ***nichtdegeneriert****.*

Wir benötigen später jedoch eine **strenge Ordnung** auf Strukturen. Da dies allein durch die für RNA Sekundärstrukturen verwendete Energiefunktion nicht möglich ist muss eine solche strenge Ordnung definiert werden.

Definition 2.2.7 (Ordnung $\prec$)
Die Ordnung $\prec$ ist wie folgt definiert:

$$s_i \prec s_j \Leftrightarrow f_E(s_i) < f_E(s_j) \text{ oder } (f_E(s_i) = f_E(s_j) \text{ und } (s_i <_{lex} s_j)) \tag{2.15}$$

wobei $<_{lex}$ die lexikographische Ordnung über der dot-bracket-Repräsentation (siehe Abbildung 2.2) *der Strukturen ist.*
Die Ordnung $\succ$ ist entsprechend definiert als

$$s_i \succ s_j \Leftrightarrow s_i \neq s_j \text{ und } \neg(s_i \prec s_j) \tag{2.16}$$

Walks

Walks sind „Wanderungen“ innerhalb der Energielandschaft. Sie werden durch eine Sequenz von Strukturen definiert, wobei zwei in dieser Sequenz aufeinanderfolgende Strukturen auch in der Landschaft benachbart sein müssen. Ausserdem ist bei einem **Walk** (im Gegensatz zu einem Pfad) ein mehrfaches Vorkommen derselben Struktur erlaubt.

Definition 2.2.8 (Walk w, Walklänge, Walkhöhe)
*Sei ein **Walk** w eine Sequenz von Strukturen mit **Walklänge** $|w| \in \mathbb{N}_1$, $w \in \mathcal{X}_p^*$*

$$w = \langle w_1, w_2, \ldots, w_{|w|} \rangle \text{ mit } w_i \in \mathcal{X}_p \text{ für alle } 1 \leq i \leq |w| \tag{2.17}$$

$$s_{i+1} \in \mathcal{N}(s_i) \quad \text{für alle } 1 \leq i < |w| \tag{2.18}$$

*Die **Walkhöhe** $f_E(w)$ eines Walks w ist definiert als*

$$f_E(w) = \max_{1 \leq i \leq |w|} f_E(w_i) \tag{2.19}$$

Es existieren verschiedene Walktypen, welche sich nur durch die Strategie bei Selektion der nächsten Struktur unterscheiden bzw. dadurch evtl. zusätzliche Bedingungen für aufeinanderfolgende Strukturen haben.

Als **Gradient Walk** oder auch **steepest decent walk** (steilster Abstieg) bezeichnen wir ab sofort jeden Walk, der bei der Selektion der nächsten Struktur immer den Nachbarn mit niedrigster Energie wählt. Da wir degenerierte Landschaften betrachten, und eine Eindeutigkeit des Gradient Walk für spätere Definitionen gewährleistet sein muss, nutzen wir die vormals erwähnte strenge Ordnung $\prec$ für folgende Definition.

Definition 2.2.9 (Gradient Walk)
*Sei w ein Walk, dann ist w ein **Gradient Walk** wenn folgende zusätzliche Bedingung erfüllt ist:*

$$\forall w_i, w_{i+1} \in w : \nexists s \in \mathcal{N}(w_i) : s \prec w_{i+1} \tag{2.20}$$

Demzufolge endet ein nicht längenbegrenzter Gradient Walk unter Garantie in einer lokal energieminimalen Struktur. Der Beweis dafür wird nach der Defintion 2.2.11 des Minimum erbracht. Da für den Gradient Walk für jede beinhaltete Struktur alle Nachbarn betrachtet werden müssen, ist dies ein sehr aufwendiger Prozess.

Ein **Adaptive Walk** wählt einen Nachbarn niedrigerer Energie entlang einer gegebenen Reihenfolge. Je nachdem ob diese Reihenfolge zufällig ist oder nicht sprechen wir von einem **Random Adaptive Walk** oder einem **Successive Adaptive Walk**.

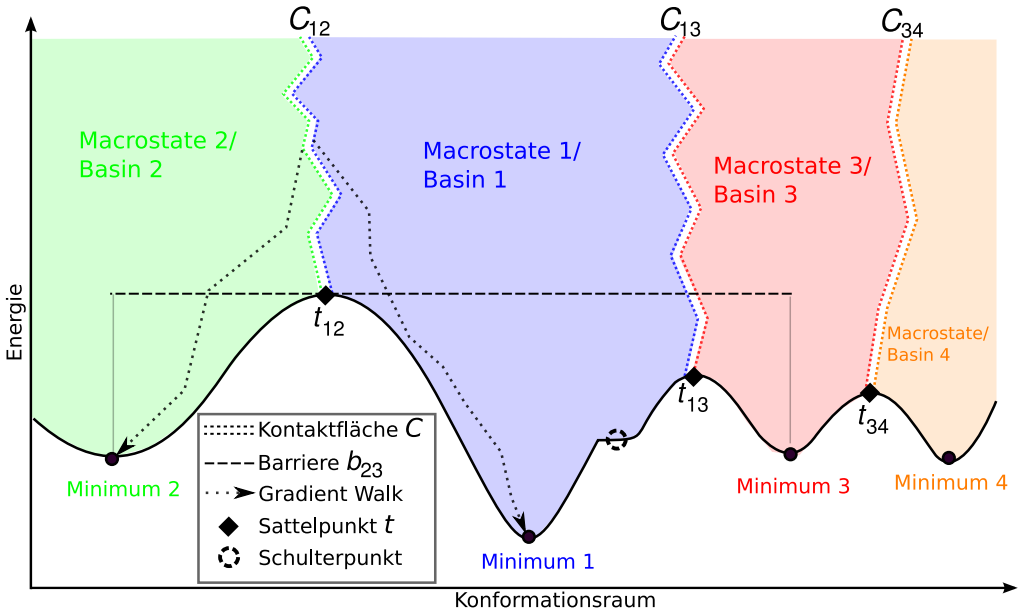


Abbildung 2.3: Elemente einer Energielandschaft zusammengefasst; zwischen allen Minima existiert eine Barriere(nur eine eingezeichnet), aber nur zwischen bestimmten Minima ein Sattel bzw. eine Kontaktfläche.

Definition 2.2.10 (Adaptive Walk)
*Sei w ein Walk, dann ist w ein **Adaptive Walk** wenn folgende weitere Bedingung erfüllt ist:*

$$w_{i+1} \prec w_i \text{ für alle } 1 \leq i < |w| \tag{2.21}$$

Sobald eine Nachbarstruktur niedrigerer Energie gefunden wurde, wird die Betrachtung der aktuellen Nachbarn abgebrochen. Somit ist der Adaptive Walk im Vergleich zum Gradient Walk deutlich schneller.
Auch dieser Walktyp endet bei unbegrenzter Länge garantiert in einem Minimum, der Beweis dafür erfolgt nach Definition 2.2.11.

Als **Random Walk** wird ein Walk bezeichnet, dessen Wahl des nächsten Nachbarn zufällig erfolgt. Eine formale Definition ist demnach nicht nötig.

Elemente der Energielandschaft

Eine RNA Energielandschaft besitzt im Regelfall mehrere charakteristische Elemente, welche die später eingeführten Topologien der Energielandschaft definieren. Solche Elemente sind z.B. Minima, Sättel, Basins und Kontaktflächen. Abbildung 2.3 verdeutlicht diese Begriffe.

Als wichtigstes Element der Energielandschaft ist ein **Minimum** eine Struktur für die gilt, dass kein Nachbar existiert welcher $\prec$ dieser Struktur ist.

Definition 2.2.11 (Minimum $\hat{s}$, Minimamenge $\mathfrak{M}_{\mathfrak{E}_p}$, mfe)
*$\hat{s}$ ist ein **lokales Minimum** wenn gilt*

$$\nexists s \in \mathcal{N}(\hat{s}) : s \prec \hat{s} \tag{2.22}$$

$\hat{s}$ ist zusätzlich ein ***globales Minimum*** *wenn zusätzlich gilt*

$$\nexists s \in \mathcal{X}_p : s \prec \hat{s} \tag{2.23}$$

Daraus folgt dass durch die strenge Ordnung $\prec$ auf den Strukturen (siehe Definition 2.2.7) keine zwei Minima benachbart sein können.
Die Energie des globalen Minimum sei die minimal free energy oder ***mfe****.*
Die ***Menge aller Minima*** *$\mathfrak{M}_{\mathfrak{E}_p}$ ist definiert als*

$$\mathfrak{M}_{\mathfrak{E}_p} = \{\hat{s} \mid \hat{s} \in \mathcal{X}_p \text{ und } \hat{s} \text{ ist ein Minimum}\} \tag{2.24}$$

Ein Gradient Walk endet immer in einem lokalen Minimum. Dies gilt es zu beweisen.

Beweis 2.2.1 (Gradient Walk und Adpative Walk enden in lokalem Minimum)
Die Gleichungen 2.20 und 2.21 ergeben für beide Walktypen:

$$w_{i+1} \prec w_i \text{ für alle } 1 \leq i < |w| \tag{2.25}$$

In jedem Schritt zum neuen Element w_{i+1} wird mindestens das derzeitige Element, w_i, unerreichbar, sonst würde Gleichung 2.25 verletzt:

$$w_{i+k} = w_{i-l} \Rightarrow \neg(w_{i+k} \prec w_i \prec w_{i-l}) \Rightarrow 2.25 \text{ nicht erfüllt} \tag{2.26}$$

Da der Konformationsraum $\mathcal{X}_p$ endlich ist, bedeutet dies, dass auch Gradient Walk und Adaptive Walk ***endlich*** *sein müssen, da die Menge der noch erreichbaren Strukturen nach endlichen Schritten der leeren Menge entspricht.*
Wann das Ende w_n des Gradient/Adaptive Walks erreicht ist, leitet sich aus 2.25 und der Nachbarschaftsbedingung der Walkelemente (Gleichung 2.18) ab:

$$\text{Walk beendet} \Leftrightarrow \nexists w_{n+1} \in \mathcal{N}(w_i) : w_{n+1} \prec w_n \tag{2.27}$$

Die Definition 2.27 für das Walkende eines Adaptive oder Gradient Walks ist jedoch äquivalent zur Definition 2.22 eines lokalen Minimum:

$$\nexists s \in \mathcal{N}(w_n) : s \prec w_n \tag{2.28}$$

Also ist das Walkende eines Adaptive oder Gradient Walks immer ein lokales Minimum. q.e.d.

Desweiteren ist anzumerken, dass die **offene Kette** $\hat{s}_{oc} = \emptyset$, sprich die Struktur mit keinerlei Basenpaarungen, bei Verwendung von ShiftMoveSet oder SingleMoveSet immer ein lokales Minimum ist.
Dies rührt daher, dass jeder Nachbar der offenen Kette nur eine Basenpaarung besitzt, und laut **Turner Regeln** Strukturen mit einzelnen Basenpaarungen eine geringere strukturelle Stabilität bzw. höhere intramolekularen Enthropie besitzen. Da die Energiefunktion die Enthropie des Moleküls ausdrückt, besitzen daher die Nachbarn der offenen Kette durchweg grössere Energiewerte.

Ein **Basin** bzw. **Macrostate** eines Minimums ist die Menge aller Strukturen von denen aus beginnend ein Gradient Walk in diesem Minimum endet [WFHS04].

Definition 2.2.12 (Basin $\mathcal{B}$, Macrostate)
Sei $\mathcal{W}_{x\hat{y}}$ die Menge aller Walks in $\mathcal{X}_p$ die in Struktur x beginnen und in Minimum $\hat{y}$ enden. Dann sei das ***Basin*** *$\mathcal{B}_{\hat{y}}$ von Minimum $\hat{y}$ definiert als*

$$\mathcal{B}_{\hat{y}} = \{\ x \mid \exists w \in \mathcal{W}_{x\hat{y}} : w \text{ ist Gradient Walk } \} \tag{2.29}$$

wobei $\mathcal{B}_{\hat{s}}$ auch als ***Macrostate*** *bezeichnet wird.*

Dass Basins/Macrostates und ihre Grenzen eindeutig definiert sind, beruht also auf der Tatsache, dass ein Gradient Walk eine Struktur eindeutig einem „zugehörigen“ Minimum zuweist, indem er von der Struktur beginnend **immer** im selben Minimum endet. Dies gilt es zu beweisen.

Beweis 2.2.2 (Gradient Walk weisst Struktur s eindeutig lok. Minimum $\hat{x}$ zu)
Wie aus Beweis 2.2.1 ersichtlich ist, endet ein Gradient Walk garantiert in einem lokalen Minimum. Es gilt nur noch zu beweisen, dass der Gradient Walk von einer Struktur ausgehend immer in das ***gleiche*** *Minimum absteigt. Wir beweisen zuerst, dass es für jede Struktur nur* ***einen*** *kleinsten Nachbarn gibt.*

Gehen wir von der gegenteiligen Annahme aus. Sei der kleinste Nachbar einer Struktur nicht eindeutig definiert. Dann kann es für eine Struktur s zwei verschiedene minimale Nachbarstrukturen gemäß der Ordnung $\prec$ geben. Für die zwei angenommenen Minima m_i und m_j gilt

$$\exists m_i, m_j \in \mathcal{N}(s) : m_i \neq m_j \text{ und } \neg(m_i \prec m_j) \text{ und } \neg(m_i \succ m_j) \tag{2.30}$$

da sonst entweder $(m_i \prec m_j)$ oder $(m_j \prec m_i)$, wodurch m_i oder m_j nicht mehr minimal wären. Nach der Definition 2.16 der verwendeten Ordnung ergibt sich jedoch

$$\neg(s_i \succ s_j) \text{ und } \neg(s_i \prec s_j) \rightarrow s_i = s_j \tag{2.31}$$

was jedoch in Widerspruch zur anfänglichen Aussage, s_i und s_j seien unterschiedlich, steht. Daraus folgt: Jede Struktur besitzt einen einzigen kleinsten Nachbarn, dieser ist also eindeutig.

Demzufolge ist bei einem Gradient Walk die Struktur w_{i+1} eindeutig durch w_i bestimmt. Daher ist, da diese Eindeutigkeit transitiv ist, bei einem Gradient Walk das Ende eindeutig durch die Anfangsstruktur des Walks bestimmt. Da ein Gradient Walk laut Definition 2.2.1 immer in einem Minimum endet, ergibt sich:
Ein Gradient Walk, beginnend in einer Struktur s, endet immer eindeutig im gleichen Minimum. q.e.d.

Eine **Kontaktfläche** zwischen zwei Basins ist definiert als die Menge aller Strukturpaare für die gilt, dass eine Struktur zum einen Basin und die andere zum zweiten gehört, wobei beide Strukturen benachbart sein müssen.

Definition 2.2.13 (Kontaktfläche $\mathcal{C}$, Nachbarschaft von Basins)
Seien $\hat{x}$ und $\hat{y}$ zwei verschiedene Minima. So ist die ***Kontaktfläche*** *$\mathcal{C}_{\hat{x}\hat{y}}$ zwischen den Basins $\mathcal{B}_{\hat{x}}$ und $\mathcal{B}_{\hat{y}}$ der beiden Minima definiert als*

$$\mathcal{C}_{\hat{x}\hat{y}} = \{(s_i, s_j) \mid s_i \sim s_j \text{ und } s_i \in \mathcal{B}_{\hat{x}} \text{ und } s_j \in \mathcal{B}_{\hat{y}} \} \tag{2.32}$$

Die beiden ***Basins sind benachbart*** *wenn gilt: $\mathcal{C}_{\hat{x}\hat{y}} \neq \emptyset$.*

Diese Kontaktflächen werden durch den später in 3.1.2 beschriebenen Flutalgorithmus berechnet und für die Macrostate Kinetik (beschrieben in 2.4.4) verwendet.

Die **Barriere** zwischen zwei Minima $\hat{x}$ und $\hat{y}$ ist die Walkhöhe (also größte Energie) des niedrigsten Walks zwischen beiden Minima.

Definition 2.2.14 (Barriere b)
Seien $\hat{x}$ und $\hat{y}$ zwei Minima, $\hat{x} \neq \hat{y}$. Sei $\mathcal{W}_{\hat{x}\hat{y}}$ die Menge aller Walks von $\hat{x}$ nach $\hat{y}$, also

$$\mathcal{W}_{\hat{x}\hat{y}} = \{ w \in \mathcal{X}_p^* \mid w_1 = \hat{x} \text{ und } w_{|w|} = \hat{y} \} \tag{2.33}$$

w_{min} sei ein „global minimaler“ Walk von $\hat{x}$ nach $\hat{y}$ wenn gilt

$$w_{min} \in \mathcal{W}_{\hat{x}\hat{y}} \tag{2.34}$$
$$\nexists w \in \mathcal{W}_{\hat{x}\hat{y}} : \quad f_E(w) < f_E(w_{min}) \tag{2.35}$$

*Dann ist die **Barriere** $b_{\hat{x}\hat{y}}$ zwischen den Minima $\hat{x}$ und $\hat{y}$ definiert als*

$$b_{\hat{x}\hat{y}} = f_E(w_{min}) \tag{2.36}$$

Eine hierarchische Darstellung aller Barrieren ist der in 2.3.1 vorgestellte BarrierTree, ein vereinfachtes Modell der Energielandschaft.

Ein **Sattel** ist nur für Minimapaare definiert deren Basins benachbart sind, siehe Abbildung 2.3. Der Sattel zwischen zwei Minima $\hat{x}$ und $\hat{y}$ ist eine Struktur größter Energie eines minimalen Walks von $\hat{x}$ nach $\hat{y}$, wobei dieser Walk die Kontaktfläche der beiden Basins kreuzen muss. Die Energie des Sattels ist die **Sattelhöhe**.

Definition 2.2.15 (Sattel t, Sattelhöhe)
Seien $\hat{x}$ und $\hat{y}$ zwei Minima, $\hat{x} \neq \hat{y}$, deren Basins benachbart sind und demzufolge eine Kontaktfläche haben, $\mathcal{C}_{\hat{x}\hat{y}} \neq \emptyset$. Sei $\overline{\mathcal{W}}_{\hat{x}\hat{y}}$ die Menge aller Walks von $\hat{x}$ nach $\hat{y}$ welche die Basins der beiden Minima nicht verlassen und demzufolge die Kontaktfläche kreuzen müssen:

$$\overline{\mathcal{W}}_{\hat{x}\hat{y}} = \{ \bar{w} \in (\mathcal{B}_{\hat{x}} \cup \mathcal{B}_{\hat{y}})^* \mid \bar{w}_1 = \hat{x} \text{ und } \bar{w}_{|\bar{w}|} = \hat{y} \} \tag{2.37}$$
$$\overline{\mathcal{W}}_{\hat{x}\hat{y}} = \emptyset \Leftrightarrow \mathcal{C}_{\hat{x}\hat{y}} = \emptyset \tag{2.38}$$

Sei $\bar{w}_{min}$ ein „lokal minimaler“ Walk von $\hat{x}$ nach $\hat{y}$, wenn gilt

$$\bar{w}_{min} \in \overline{\mathcal{W}}_{\hat{x}\hat{y}} \tag{2.39}$$
$$\nexists \bar{w} \in \overline{\mathcal{W}}_{\hat{x}\hat{y}} : \quad f_E(\bar{w}) < f_E(\bar{w}_{min}) \tag{2.40}$$

*So gilt für den **Sattel** $t_{\hat{x}\hat{y}}$: er ist eine Struktur mit höchster erreichter Energie eines lokal minimalen Walks ($\widehat{=}$ Walkhöhe) der die Kontaktfläche kreuzt,*

$$t_{\hat{x}\hat{y}} \in \bar{w}_{min} \tag{2.41}$$
$$f_E(t_{\hat{x}\hat{y}}) = f_E(\bar{w}_{min}) \tag{2.42}$$

er ist Teil der Kontaktfläche,

$$(t_{\hat{x}\hat{y}}, s) \in \mathcal{C}_{\hat{x}\hat{y}} \text{ oder } (s, t_{\hat{x}\hat{y}}) \in \mathcal{C}_{\hat{x}\hat{y}} \tag{2.43}$$

und demzufolge nur definiert wenn eine Kontaktfläche vorhanden ist, also die Basins der Minima benachbart sind:

$$\mathcal{C}_{\hat{x}\hat{y}} = \emptyset \Leftrightarrow t_{\hat{x}\hat{y}} \text{ nicht definiert} \tag{2.44}$$

Demzufolge besitzt nicht jedes Minimapaar einen Sattel.
*Die **Sattelhöhe** sei die Energie des Sattels, also $f_E(t_{\hat{x}\hat{y}})$.*

Demnach entspricht die Sattelhöhe der Walkhöhe des niedrigsten Walks der von $\hat{x}$ nach $\hat{y}$ führt und die Kontaktfläche kreuzt, siehe Gleichung 2.42.

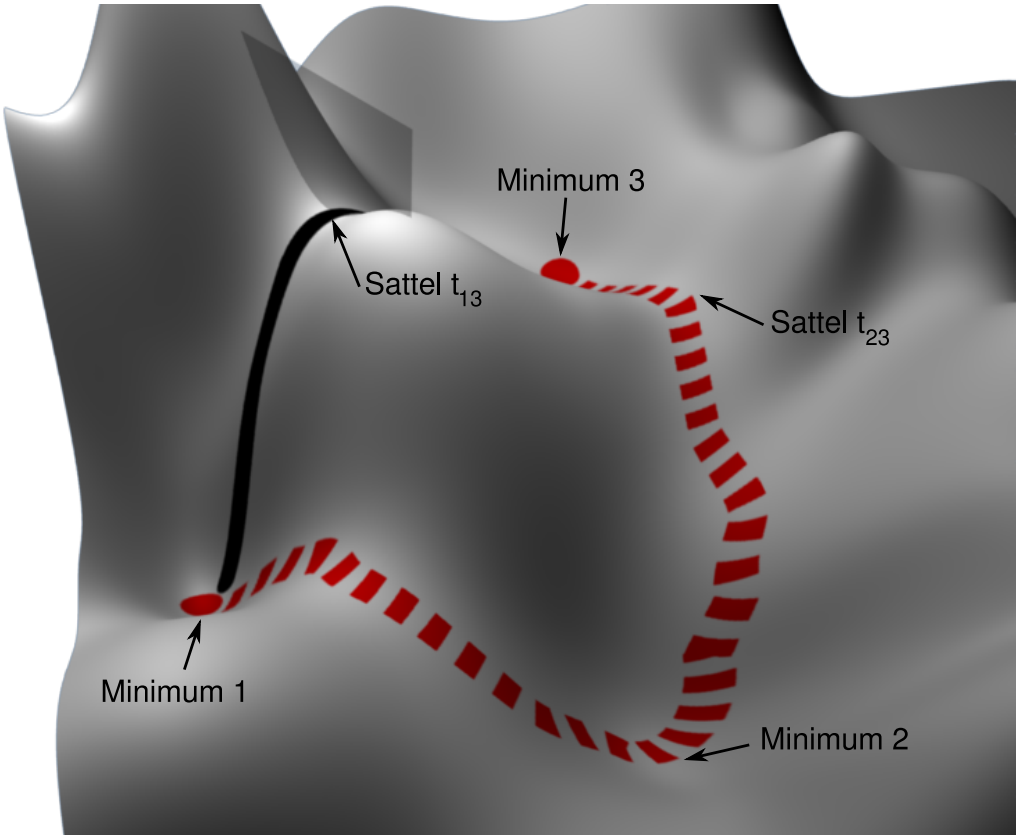


Abbildung 2.4: Stark vereinfachtes 3D Modell einer Energielandschaft. Transparente Ebene stellt unteren Teil der Kontaktfläche $\mathcal{C}_{13}$ dar. Gestrichelte Linie ist ein möglicher global minimaler Walk zwischen Minima 1 und 3. Durchgezogene Linie ist ein lokal minimaler Walk, der somit die Kontaktfläche $\mathcal{C}_{13}$ kreuzt. Der Sattel t_{13} ist Teil dieses Walks. Andere Kontaktflächen (z. B. $\mathcal{C}_{23}$) sind der Übersichtlichkeit wegen nicht eingezeichnet.

Demzufolge ist nicht jede Sattelhöhe eine Barriere, aber jede Barriere eine Sattelhöhe. Zum Verständnis dieser Aussage siehe Abb. 2.4. Die Barriere ist die Walkhöhe eines global minimalen Walks (gestrichelte Linie), die Sattelhöhe dagegen die Walkhöhe des lokal minimalen Walks (durchgezogene Linie).

Ein **Schulterpunkt** ist ein lokales Minimum, das eine Barriere zu einem anderen Minimum hat welche der Energie des Schulterpunktes entspricht. Er hat demnach keinen Nachbarn mit niedrigerer Energie (da lokales Minimum, siehe Def. 2.2.11) und mindestens einen Nachbarn gleicher Energie, sonst wäre die Barriere größer der Energie des Schulterpunktes.

Definition 2.2.16 (Schulterpunkt)
Ein Minimim $\hat{x} \in \mathfrak{M}_{\mathfrak{E}_p}$ *gilt als* ***Schulterpunkt*** *wenn gilt*

$$\exists \hat{s} \in \mathfrak{M}_{\mathfrak{E}_p} \backslash \hat{x} : f_E(\hat{x}) = b_{\hat{s}\hat{x}} \tag{2.45}$$

2.3 Topologien der Energielandschaft

Die Repräsentation und Speicherung von Energielandschaften ist ohne eine Abstraktion bzw. Vereinfachung nicht möglich. Vereinfachung bedeutet, dass die Gesamtheit aller Zustände (Microstates) durch lediglich Sättel und Minima (SaddleNet) oder Barrieren und Minima (BarrierTrees) repräsentiert wird. Dies ist aufgrund der zuvor beschriebenen Basin/Macrostate Partitionierung möglich.
Beide Modelle werden in diesem Abschnitt näher betrachtet.

2.3.1 BarrierTree

Diese Topologie wurde erstmals in [HS88] vorgestellt. Der Name „BarrierTree“ wurde jedoch zuerst in [FFHS00] benutzt. Dort wurde gezeigt, dass die BarrierTree Struktur mit dem Faltungsverhalten korreliert und daher BarrierTrees gut für die Untersuchung von Faltungskinetiken geeignet sind. Eine Umsetzung für bei RNA üblichen degenerierten Landschaften wurde in [FHSW02] genau beschrieben. In [Ric07] wurde schließlich die Implementierung einzelner Operationen auf dem BarrierTree ausführlich dargestellt und wird somit als Referenz empfohlen.
Abbildung 2.5 veranschaulicht den Aufbau eines BarrierTrees und seine Unterschiede zum SaddleNet.

Ein BarrierTree ist eine Baum, also ein zyklenfreier zusammenhängender Graph, mit einem determinierten Wurzelknoten. Die Blätter stellen die Minima der Energielandschaft dar, innere Knoten die Barrieren zwischen diesen. Genauer gesagt repräsentiert der kleinste gemeinsame Vorfahre zweier Blätter bzw. Minima die Barriere zwischen diesen Minima. Ob die inneren Knoten Strukturen sind, welche als Energie die Barriere haben oder reine Barrieren sind, ist implementierungsabhängig. Wir definieren die Knotenmenge des BarrierTrees der Einfachheit halber als Menge von Strukturen. Für eine Darstellung eines Barrier Trees siehe Abbildung 2.5.

Schulterpunkt Problematik

In den vorgenannten Quellen wird explizit die Problematik von Schulterpunkten behandelt. Diese sind bei Verwendung von BarrierTrees problematisch, da ihre Darstellung durch gewisse Graphalgorithmen für BarrierTrees nicht oder nur teilweise unterstützt wird. Zudem werden sie durch ein gleiches Energieniveau mit mindestens einem der Nachbarn oft nicht als „echte“ Minima angesehen, und daher in diesen Ansätzen während des Aufbaus des BarrierTree verworfen.

Unsere Algorithmen arbeiten mit verschiedenen Topologien (z. B. SaddleNet), welche nicht unbedingt die Erkennung von Schulterpunkten in effizienter Form zulassen. Zusäzlich soll ein Qualitätsvergleich zwischen Algorithmen auf verschiedenen Topologien (hier SaddleNet vs. BarrierTree) möglich sein. Wir haben uns daher dafür entschieden, Schulterpunkte *nicht* zu verwerfen.

Definition 2.3.1 (BarrierTree einer Energielandschaft $\mathfrak{E}_p$)
Gegeben eine Primärstruktur p.
Sei $G = (V, E)$ ein Baum, also ein azyklischer, zusammenhängender Graph mit der Knotenmenge $V = V^l \cup V^n$ und der Menge der ungerichteten Kanten $E \subseteq V \times V$, sowie der

Wurzel $r \in V$.
Sei V^l *die Menge aller Blätter und* V^n *die Menge aller inneren Knoten. Sei* $G_k = (V_k, E_k)$ *der* ***Teilbaum mit Wurzel k****, sei* V_k^l *die Menge aller Blätter dieses Teilbaums.*
Dann ist G *ein* ***BarrierTree der Energielandschaft*** $\mathfrak{E}_p$ *wenn gilt*

$$V^l = \mathfrak{M}_{\mathfrak{E}_p} \tag{2.46}$$

$$V^n \subseteq \text{Menge aller Sättel aus } \mathfrak{E}_p \tag{2.47}$$

$$\forall s \in V : f_E(r) \geq f_E(s) \tag{2.48}$$

$$\forall k \in V^n : \forall \hat{x}, \hat{y} \in V_k^l : b_{\hat{x}\hat{y}} \leq f_E(k) \tag{2.49}$$

Abbildung 2.5 verdeutlicht sehr einfach das Konzept des BarrierTrees. Blätter des BarrierTrees sind Minima und werden in der Abbildung als Indizes mit Hilfe ihrer Energien dargestellt, innere Knoten nur als deren Energien bzw. als Barrieren. Es ist ersichtlich, dass eine Barriere mehr als zwei Minima verbinden kann, der BarrierTree ist demzufolge nicht zwingend ein Binärbaum.
Vor allem aber lässt sich in der Grafik ein Trend erkennen: mit steigender Anzahl Minima enthält der BarrierTree für immer weniger Minimapaare Adjazenzinformation, wobei das SaddleNet diese nicht verliert. Schon bei kleinen Sequenzen mit z. B. 34 Minima ist der Prozentsatz an Adjazenzinformationen eines BarrierTrees gegenüber der vollständigen Information die ein SaddleNet besitzt vernachlässigbar.

Dafür bleibt der Speicherverbrauch, durch die Baumstruktur, linear in der Anzahl Minima. Zudem ist das für ihn verwendete Sampling performanter als beim SaddleNet (siehe Algorithmus 3.2).

2.3.2 SaddleNet

Das SaddleNet, auch als Konnektivitätsgraph bezeichnet, ist ein zusammenhängender, ungerichteter und kantengewichteter Graph. Alle Knoten repräsentieren Minima, alle Kanten Sättel und die Gewichte der Kanten sind die Sattelhöhen. Es ist demzufolge ein Energielandschaftsgraph mit vollständiger Adjazenzinformation, basierend auf der Nachbarschaft der Basins, im Gegensatz zum BarrierTree.

Definition 2.3.2 (SaddleNet einer Energielandschaft $\mathfrak{E}_p$)
Gegeben eine Primärstruktur p*. Sei* $G = (V, E, f)$ *ein ungerichteter, zusammenhängender, kantengewichteter Graph mit der Knotenmenge* V *und der Kantenmenge* $E \subseteq V \times V$ *sowie der Funktion* $f : E \to \mathbb{R}$ *welche Kanten ein Gewicht zuordnet. Sei* $t_{\hat{x}\hat{y}} \in \mathcal{X}_p$ *ein Sattel zwischen den Minima* $\hat{x}, \hat{y} \in \mathcal{X}_p$*.*
Dann nennt man G *ein* ***SaddleNet der Energielandschaft*** $\mathfrak{E}_p$ *wenn gilt*

$$V = \mathfrak{M}_{\mathfrak{E}_p} \tag{2.50}$$

$$\forall (\hat{x}, \hat{y}) \in E : \exists t_{\hat{x}\hat{y}} \tag{2.51}$$

$$\forall (\hat{x}, \hat{y}) \in E : f((\hat{x}, \hat{y})) = f_E(t_{\hat{x}\hat{y}}) \tag{2.52}$$

In Abbildung 2.5 sieht man zwei SaddleNets. Alle Knoten werden mit Indizes dargestellt, Kanten entsprechen Sätteln bzw. Kontaktflächen, Kantengewichte sind Sattelhöhen. Man sieht dort in Abbildung (b) wie Sättel vorhanden sind, welche der BarrierTree (und auch die 2D Darsetllung der Landschaft) nicht wiedergeben kann. Adjazenzinformation ist im exakten SaddleNet immer vollständig vorhanden. Dies ist, wie wir noch sehen werden, in auf Adjazenzinformation beruhenden Ratenberechnungen von großer Bedeutung.

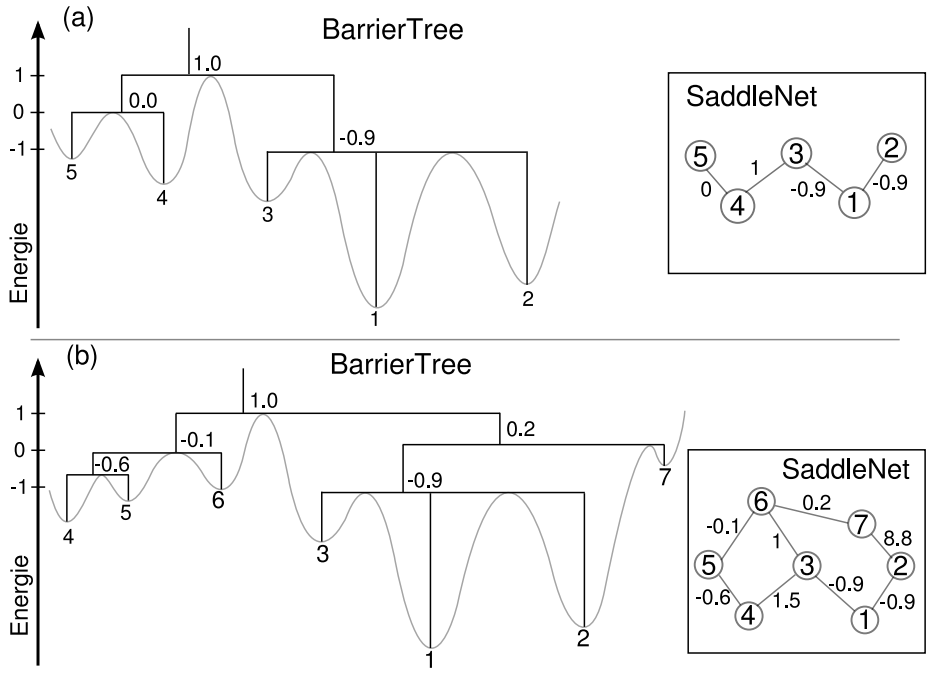


Abbildung 2.5: 2 Energielandschaften, jeweils als BarrierTree und SaddleNet, Indizes der Minima sind nach Energie sortiert; (a) 3 Barrieren, 4 Sättel; jede Sattelhöhe im BarrierTree vertreten, dieser ist in diesem Fall kein Binärbaum, gleichwertige Darstellung wäre gegeben wenn z. B. $b_{12} < b_{13}$; (b) 5 Barrieren, 8 Sättel, komplexere Landschaft; der BarrierTree als quasi „Seitenansicht“ der Landschaft verliert Adjazenzinformation die im SaddleNet als quasi „Draufsicht“ der Landschaft noch sichtbar ist, also die Nachbarschaft der Macrostates/Basins 6 und 7 sowie 3 und 4.

Transformation von SaddleNet zu BarrierTree

Man kann jedes SaddleNet in einen gültigen BarrierTree transformieren, da das SaddleNet unter anderem alle Informationen (Minima, Barrieren) des BarrierTrees implizit enthält.

2.4 Faltungskinetiken

Die Kinetik ist ein eher vager Begriff für die formale Repräsentation des Verhaltens eines Moleküls in einem Gaskontinuum (in diesem Fall einer RNA). Eine Faltungskinetik gibt entsprechend das Faltungsverhalten wieder. Im folgenden werden wir der Einfachheit halber die Begriffe Faltungskinetik und Kinetik synonym verwenden.

2.4.1 Definition

Da keine feste, weit verbreitete Definition für eine Kinetik vorhanden ist, liegt es in unserem Ermessen eine formale Definition zu geben. Diese soll das Verhalten des Systems bzw. Moleküls eindeutig modellieren. Solch eine Repräsentation kann auf verschiedene Arten definiert werden, in unserem Fall wie folgt:
Es sei G eine Topologie einer Energielandschft $\mathfrak{E}_p$. Desweiteren sei $\vec{P}_0$ ein Vektor von

Anfangswahrscheinlichkeiten, welche die Aufenthaltswahrscheinlichkeit des Systems zum Zeitpunkt 0 wiedergegeben. X sei die Menge der durch die Topologie G repräsentierten Strukturen, $X \subseteq \mathcal{X}_p$, bei einem SaddleNet entsprechend $X = V = \mathfrak{M}_{\mathfrak{E}_p}$, bei einem BarrierTree $X = V^l = \mathfrak{M}_{\mathfrak{E}_p}$, beim vollständigen Graphen der Landschaft $X = V = \mathcal{X}_p$. Die Anzahl der durch $|G|$ betrachtbaren Strukturen ist entsprechend $|X|$.
Dann ist für uns eine Faltungskinetik ein Algorithmus zur Berechnung der Wahrscheinlichkeiten p_i^t, dass ein sich faltendes Molekül zu einem Zeitpunkt t eine Struktur im Macrostate des Minimums $\hat{x}_i$ angenommen hat.
Anders ausgedrückt ist für uns eine Kinetik

- ein Basisalgorithmus der $\vec{P}_0$ und eine Ratenmatrix $\mathbf{K}$ verwendet um den Wahrscheinlichkeitsvektor $\vec{P}_t$ für jeden beliebigen Zeitpunkt t mit $0 < t < \infty$ berechnen zu können, $\vec{P}_t = \langle p_1^t, p_2^t, \ldots, p_n^t \rangle$, *und*
- die Ratenmatrix $\mathbf{K}$ mit den Transitionsraten k_{ij} für alle $1 \leq i, j \leq |X|$ bzw. die Algorithmen welche $\mathbf{K}$ aus der Topologie erzeugen

Der erstgenannte Basisalgorithmus ist bei allen Kinetiken gleich: ein zeitkontinuierlicher Markov Prozess. Daher wird dieser hier nur einmal beschrieben. Danach wird vereinfachend als Kinetik nur noch die Ratenmatrix $\mathbf{K}$ und die zu ihrer Berechnung führenden Algorithmen verstanden. Der Markov Prozess, der Vektor $\vec{P}_0$ der Anfangswahrscheinlichkeiten und die Energielandschaft $\mathfrak{E}_p$ werden später als definiert vorausgesetzt. Die verwendete Topologie wird im Regelfall durch die Kinetik vorgegeben, es gibt jedoch einen Spezialfall (Arrheniuskinetik auf SaddleNet) welcher später näher beleuchtet wird.

2.4.2 Zeitkontinuierlicher Markov Prozess

Ein Markov Prozess sei ein Synonym für eine endliche, zeithomogene Markov Kette (oder Markov Modell) im stetigen Zeitraum. Es gelte für unseren Markov Prozess, dass er erster Ordnung sei. Dies bedeutet, dass die Zustandsänderung eines beschriebenen endlichen Systems zum Zeitpunkt t alleinig von dem Zustand des Systems im Zeitpunkt $t \in \mathbb{R}$ abhängt, also nicht von den Zuständen in der Vergangenheit bzw. Zukunft beeinflusst wird, und auch nicht vom Zeitpunkt t selbst, da zeithomogen. Dieses fehlende bzw. eingeschränkte „Erinnerungsvermögen“ nennt man **Markov Eigenschaft**.
Dies bedeutet in unserem konkreten Fall: wir gehen davon aus, dass das weitere Faltungsverhalten eines RNA Moleküls ausschließlich von seiner derzeitigen ausgebildeten Struktur abhängt und nicht vom vorherigen Faltungsprozess.
Die in der folgenden Definition verwendete Zustandmenge Q entspricht bei unserem Markov Prozess der Menge X der in der Topologie vertretenen Strukturen (meist Minima).

Definition 2.4.1 (Markov Prozess, stationäre Verteilung μ)
Sei ein ***zeitkontinuierlicher Markov Prozess*** *ein Tupel*

$$(Q, \vec{P}_0, \mathbf{K}) \tag{2.53}$$

Zeitkontinuierlich *bedeutet dass für eine beliebige Anzahl m an Zeitpunkten t_i gilt:*

$$t_i \in \mathbb{R}^+ \text{ für alle } 1 \leq i \leq m \quad i, m \in \mathbb{N}_1 \tag{2.54}$$

$$0 < t_1 < t_2 < \ldots < t_m \tag{2.55}$$

$Q = \{q_1, q_2, \ldots, q_n\}$ *ist die* ***Zustandsmenge*** *mit Größe* $n = |Q|$.
$\vec{P}_0 = \langle p_1^0, p_2^0, \ldots, p_n^0 \rangle$ *ist die* ***Anfangsverteilung*** *der Wahrscheinlichkeiten eines jeden* q_i *für den Zeitpunkt* 0 *mit Länge* n. $\mathbf{K}$ *ist letztlich die* ***Ratenmatrix****, eine quadratische* $n \times n$ *Matrix mit Transitionsraten* k_{ij} *für den Übergang von Zustand* i *nach Zustand* j*, so dass gilt*

$$\frac{d}{dt}\vec{P}_t = \mathbf{K} \cdot \vec{P}_t \tag{2.56}$$

$$\forall_{i \neq j} : k_{ij} \geq 0 \tag{2.57}$$

$$k_{ii} = -\sum_{j \neq i} k_{ij} \tag{2.58}$$

Formel 2.58 garantiert, dass die Summe der einzelnen Wahrscheinlichkeiten p_i^t *für jeden Zeitpunkt* t *konstant bleibt. Somit gilt der Markov Prozess als* ***konserviert****. Sei* $Z_t \in Q$ *der* ***Zustand*** *des Markov Prozesses zum Zeitpunkt* t*. Dann gilt für jede Verteilung/ jeden Wahrscheinlichkeitsvektor* $\vec{P}_t$

$$\forall q_j \in Q : p_j^t = Prob(Z_t = q_j) \tag{2.59}$$

Es gilt zudem die ***Markov Eigenschaft****, sprich dass der Prozess erinnerungslos ist. Dies bedeutet, dass die Wahrscheinlichkeit, dass der Prozess zum Zeitpunkt* t_n *in einen bestimmten Zustand* j *wechselt nur abhängig („bedingt“) ist vom gegenwärtigen Zustand* Z_t*, aber nicht abhängig von vergangenen Zuständen* $Z_{t_{n-1}}, Z_{t_{n-2}}, \ldots$*, sprich:*

$$\begin{aligned} & \forall n, m \in \mathbb{N}, 0 < t_1 < \ldots < t_m,\ i_1, \ldots, i_n, j \in Q : \\ & Prob(Z_{t_{n+1}} = j \mid Z_{t_n} = i_n, Z_{t_{n-1}} = i_{n-1}, \ldots, Z_{t_1} = i_1) \\ = \quad & Prob(Z_{t_{n+1}} = j \mid Z_{t_n} = i_n) \end{aligned} \tag{2.60}$$

Zudem ist der Markov Prozess ***zeithomogen****, sprich der Wert von* t *hat keinen Einfluss:*

$$Prob(Z_{t+k} = j \mid Z_t = i) = Prob(Z_k = j \mid Z_0 = i) \tag{2.61}$$

Die Verteilung μ *nennt man* ***stationäre Verteilung****, wenn gilt*

$$\mu = \mu \cdot \mathbf{K} \tag{2.62}$$

$$\frac{d}{dt}\vec{P}_t = \mathbf{K} \cdot \mu = \vec{0} \tag{2.63}$$

Gilt zusätzlich

$$k_{ij}\mu_i = k_{ji}\mu_j \tag{2.64}$$

so besitzt der Markov Prozess ein ***detailliertes Gleichgewicht*** *und wird als* ***umkehrbar*** *bezeichnet. Die Markov Theorie garantiert, dass wenn ein Markov Prozess umkehrbar ist, er immer genau* ***eine*** *solche stationäre Verteilung* μ *besitzt.*

Eine detaillierte Beschreibung des Algorithmus und dessen Implementierung sowie Limitierungen erfolgt im Abschnitt 3.4 des nächsten Kapitels. Für alle O-Notationen gelte ab nun $n = |X|$, also Anzahl der durch die Topologie repräsentierten Strukturen, siehe Abschnitt 2.4.1.

Ab jetzt wird vereinfachend als Kinetik nur noch die Ratenmatrix $\mathbf{K}$ und die zu ihrer Berechnung führenden Algorithmen verstanden. Diese sind der Inhalt der nun folgenden Beschreibung der einzelnen Kinetiken.

Name	verwendete Topologie	Speicherverbrauch [byte]	verwendete Algorithmen
Microstatekinetik	vollständige Landschaft	$\approx 2 \cdot (\lvert\mathcal{X}_p\rvert \cdot 8)^2$	`MS, LF, RMB`
Macrostatekinetik	SaddleNet	$\approx 2 \cdot (\lvert\mathfrak{M}_{\mathfrak{E}_p}\rvert \cdot 8)^2$	`MS, LF, RMB`
Arrheniuskinetik	BarrierTree*	$\approx 2 \cdot (\lvert\mathfrak{M}_{\mathfrak{E}_p}\rvert \cdot 8)^2$	`MS, [LF]**, BTS, RMB`

Tabelle 2.1: Vergleich der bekanntesten Kinetikvarianten; Speicherverbrauch bezieht sich auf den speicherintensivsten Zwischenalgorithmus, die Matrixdiagonalisierung.
`MS` - Minima Sampling, `LF` - Landscape Flooding, `BTS` - BarrierTree Sampling, `RMB` - Ratenmatrix Berechnung, alle Algorithmen in Abschnitt 3.1 näher beschrieben.
* Der BarrierTree ist die Standardtopologie der Arrheniuskinetik, die jedoch auch auf einem SaddleNet basieren kann, siehe Ergebnisteil.
** Das LandscapeFlooding ist bei Berechnung der Arrheniuskinetik optional. Durch Verwendung von Macrostate Energien, siehe Gleichung 2.69, anstatt der Energien der Minima, wird das Ergebnis verbessert. Inwieweit die Ergebnisse davon beeinflusst werden wird in Abschnitt 4.2 näher erläutert.

Tabelle 2.1 fasst die wesentlichen Kinetiken kurz zusammen. Zum einen die Microstate Kinetik welche die gesamte Energielandschaft ohne vereinfachende Modellierung betrachtet, aber durch die enorme Anzahl an betrachteten Strukturen nur auf kurze Sequenzlängen (~ 22, dadurch $|\mathcal{X}_p| < 40000$) möglich ist. Dann die Macrostate Kinetik, welche durch das verwendete SaddleNet nur die Minima der Landschaft und die sie verbindenden Sättel repräsentiert, deren Anwendung jedoch neben der typischen Restriktion durch die Matrixdiagonalisierung zusätzlich durch das speicherlimitierte LandscapeFlooding (siehe Abschnitt 3.1.2) eingeschränkt wird. Und schliesslich der qualitativ mit Abstand schlechteste Ansatz, die Arrheniuskinetik, welche wie die Macrostate Kinetik durch die Anzahl Minima limitiert ist, aber sonst keine weiteren Einschränkungen kennt.

2.4.3 Microstate Kinetik

Der simpelste und zugleich qualitativ hochwertigste Ansatz verzichtet auf eine Vereinfachung der Energielandschaft durch Segmentierung in sogenannten „Macrostates“, sondern betrachtet die Gesamtheit aller Strukturen („Microstates“), also den gesamten Konformationsraum $\mathcal{X}_p$. Sie ist daher nur für kleine Sequenzen umsetzbar, da $|\mathcal{X}_p|$ exponentiell in der Länge der Sequenz p wächst. Die Microstate Kinetik bietet, wenn umsetzbar, als eine „Referenzkinetik“ das Maß der Dinge.

Theorie

Die Menge der betrachteten Strukturen X ist der gesamte Konformationsraum $\mathcal{X}_p$, welcher in der Sequenzlänge N exponentiell wächst [HSS98, Wat95]. Entsprechend hoch ist der in 3.4 dargestellte Speicherverbrauch von $O(n^2)$ für die Berechnung der Kinetik. Die Berechnung der Transitionsraten k_{ij} erfolgt nach dem **Metropolis Kriterium**, sofern die Strukturen s_i und s_j benachbart sind:

$$\Delta G_{ij} = f_E(s_j) - f_E(s_i) \tag{2.65}$$

$$s_i \sim s_j \Leftrightarrow k_{ij} = \Gamma_M \cdot \begin{cases} e^{-\Delta G_{ij}/RT} & \text{wenn } \Delta G_{ij} > 0 \\ 1 & \text{wenn } \Delta G_{ij} \leq 0 \end{cases} \tag{2.66}$$

$$s_i \not\sim s_j \Leftrightarrow k_{ij} = 0 \tag{2.67}$$

R und T sind die Gaskonstante und die Temperatur in Kelvin, dies wird in folgenden Formeln nicht mehr erwähnt. Γ_M ist ein Zeitfaktor, der in diesem Fall die Zeit eines einzelnen Moves widerspiegelt. Es wird aber üblicherweise aus Mangel an fundierten experimentellen Daten auf eine Abbildung auf reelle Zeitwerte mittels Γ_M verzichtet und somit $\Gamma_M = 1$ gewählt.

Das **Metroplois Kriterium** (Gleichung 2.66) gibt an, dass die Faltung zu einer energetisch günstigeren Struktur („downhill“) immer mit gleicher Geschwindigkeit erfolgt, unabhängig von der Energiedifferenz. Der Hintergrund hierbei ist, dass die physikalische Annahmne getroffen wird, der Attraktionsbereich einer „favorisierten Bindung“ wäre quasi 0. Die Basen der Kette „spüren“ demzufolge nicht das Verlangen eine energetisch günstige Bindung einzugehen. Dies kann als Difussionsgrenze des Systems interpretiert werden ([Wol01]).

Dieses Kriterium ist nicht die einzige Möglichkeit, Übergangsraten zwischen einzelnen Strukturen zu berechnen, aber es ist das Standardverfahren, um detailliertes Gleichgewicht (siehe Gleichung 2.64) zu ermöglichen [Fla98]. Dies ist nötig für die stationäre Verteilung μ bzw. das thermodynamische Gleichgewicht des Systems, welches durch die Umkehrbarkeit des Markov Prozesses nach endlicher Zeit erreicht wird, siehe Definition 2.4.1.

Umsetzung

Die **verwendete Topologie** ist weder BarrierTree noch SaddleNet, welche nur Minima und Sättel/Barrieren verwalten, sondern ein **kompletter Graph der Energielandschaft**. Dies ist wie schon erwähnt nur möglich für kleine Sequenzen bis zu einer Länge von ~ 22, z.B. für die `xbix` Sequenz. Nähere Informationen zu den in dieser Arbeit verwendeten Sequenzen findet man in Abschnitt 4.1.

Um die Microstate Kinetik zu berechnen, sind folgende Schritte notwendig:

- Minima Sampling: Finden der Minima, Graph G wird erzeugt
- Landscape Flooding: jede Struktur der Landschaft wird G hinzugefügt
- Transitionsmatrix Berechnung: $\mathbf{K}$ wird durch Gleichungen 2.66 und 2.67 erzeugt, Input: G

In Abbildung 2.7(a) ist die Microstatekinetik der `xbix` Sequenz zu sehen. Sie fungiert als Referenzkinetik für diese Sequenz. Bei den anderen verwendeten Beispielsequenzen (siehe Abschnitt 4.1) war eine entsprechende Microstate Kinetikberechnung nicht mehr möglich. Um dies zu verdeutlichen hier eine kurze Rechnung:
Die `xbix` Sequenz mit Länge 20 besitzt $n = 3886$ Strukturen. Die einmalige Diagonalisierung für den Markov Prozess benötigt demnach bei Verwendung von lediglich 32Bit Gleitkommawerten mindestens $2 \cdot 60$MB Speicher für die beiden $n \times n$ Matrizen $\mathbf{N}$ und $\mathbf{M}$ (siehe

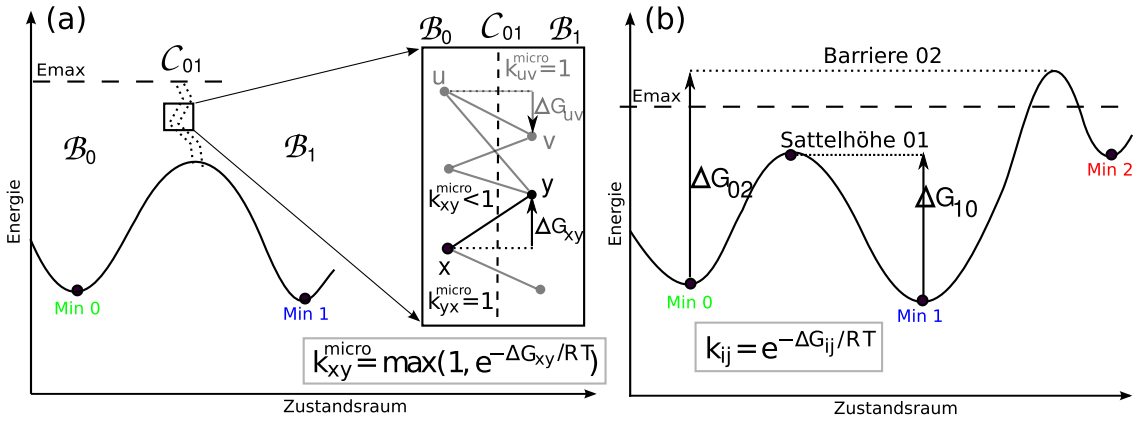


Abbildung 2.6: Darstellung der Berechnung von k_{ij}; (a): Macrostate Kinetik; bis $\mathbf{E_{max}}$ wird Microrate für jedes Strukturpaar der Kontaktfläche $\mathcal{C}_{01}$ berechnet, dann alle Raten aufsummiert; Microraten in „Abwärtsrichtung“ immer 1; Microstate Raten nicht eingezeichnet, sind aber gleich den Microraten definiert; (b) Arrheniuskinetik, ΔG_{02} nur für BarrierTree, ΔG_{10} für SaddleNet und BarrierTree verwendbar.

Gleichung 3.27). Die Sequenz `am282` (siehe 4.1) mit Länge 27 besitzt ca. $n = 250000$ erlaubte Strukturen, was für die Diagonalisierung einer Ratenmatrix selbst mit 32Bit Fliesskommawerten einen Speicherverbrauch von mindestens $2 \cdot 250$GB bedeutet, was weit ausserhalb der maximal 32GB des für die Arbeit verfügbaren Rechenclusters liegt.

2.4.4 Macrostate Kinetik

Eine Macrostatekinetik nutzt die in Definition 2.2.12 beschriebene Unterteilung der Landschaft in Basins bzw. Macrostates. Also rechnet die Macrostate Kinetik nur auf einer Anzahl von Macrostates gleich $|\mathfrak{M}_{\mathfrak{E}_p}|$ und ermöglicht durch diese vereinfachte Betrachtung der Energielandschaft eine deutlich kleinere Ratenmatrix als bei der Microstate Kinetik. Sie ist dafür auf deutlich längere Sequenzen anwendbar. Es werden speziell die Kontaktflächen zwischen Macrostates verwendet, um gute Transitionsraten zu berechnen.

Theorie

Für jedes Minimum $\hat{s}_i$ existiert ein Macrostate bzw. Basin $\mathcal{B}_i$, sowie eine **kanonische Zustandssumme** Z_i für alle Strukturen dieses Basins, genauer:

$$Z_i = \sum_{s \in \mathcal{B}_i} e^{-f_E(s)/RT} \tag{2.68}$$

Die Umkehrfunktion zur Berechnung der **Macrostateenergie** aus Z_i ist

$$f_E(\mathcal{B}_i) = -RT \cdot log(Z_i) \tag{2.69}$$

Die Berechnung der Transitionsraten k_{ij} basiert auf dem Prinzip der Kontaktflächen und nutzt das Metropolis Kriterium für die Strukturpaare der Kontaktfläche.

$$\Delta G_{xy} = f_E(s_y) - f_E(s_x) \tag{2.70}$$

$$s_x \sim s_y \Leftrightarrow k_{xy}^{micro} = \Gamma_M \cdot \begin{cases} e^{-\Delta G_{xy}/RT} & \text{wenn } \Delta G_{xy} > 0 \\ 1 & \text{wenn } \Delta G_{xy} \leq 0 \end{cases} \tag{2.71}$$

$$s_x \not\sim s_y \Leftrightarrow k_{xy}^{micro} = 0 \tag{2.72}$$

$$k_{ij} = \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot Prob(x|\mathcal{B}_i)) \tag{2.73}$$

$$= \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot e^{-f_E(x)/RT}/Z_i) \tag{2.74}$$

$$= \frac{1}{Z_i} \cdot \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot e^{-f_E(x)/RT}) \tag{2.75}$$

$$\mathcal{C}_{ij} = \emptyset \Leftrightarrow k_{ij} = 0 \tag{2.76}$$

Abbildung 2.6(a) verdeutlicht den Ansatz. Gleichung 2.71 ist identisch mit der Formel 2.66. Demzufolge ist auch der Zeitfaktor Γ_M (der sich wie Z_i herausziehen lässt) integriert und besitzt dieselbe Semantik wie bei der Microstate Kinetik: Zeitaufwand für einen Move in der Landschaft, also Deletion oder Insertion eines Basenpaares. Genau wie dort wird er standardmässig auf 1 gesetzt, also ignoriert.
Gleichung 3.7 zeigt, dass die Ratenmatrix Adjazenzinformation der Basins beinhaltet. Wenn zwei Basins nicht benachbart sind, sprich keine Kontaktfläche zwischen ihnen existiert (also laut Def. 2.2.15 auch kein Sattel zwischen den entsprechenden Minima der Basins), so ist die Transitionsrate 0.

Umsetzung

Die **verwendete Topologie** ist daher ein spezielles **SaddleNet**, welches als Kantengewichte die aufsummierte Energie der Paare der Kontaktflächen besitzt und somit die Nachbarschaft der Basins widerspiegelt. Es werden für jede Kante (i, j) während des Flutens der Landschaft zwei separate Werte aufsummiert, einmal für das Paar (i, j) und für (j, i), nennen wir diese Summen u_{ij} und u_{ji}.

Zur Berechnung sind folgende Schritte notwendig:

- Minima Sampling: Finden möglichst aller Minima und Speicherung in SaddleNet G
- Landscape Flooding: gefundene Sättel werden zu Kanten in G; Kantengewichte sind Summen über die Energien der Strukturpaare der Kontaktfläche (näheres in Kapitel 3), kanonische Zustandssummen Z_i werden erzeugt
- Transitionsmatrix Berechnung: $\mathbf{K}$ über Formel 2.75 erzeugen, Input: G und Z_i

Dieser letzte Schritt bedeutet ein einmaliges Teilen der Kantengewichte durch die entsprechende Partitionsfunktion Z_i. Also $k_{ij} = u_{ij}/Z_i$ und $k_{ji} = u_{ji}/Z_j$.

In Abbildung 2.7(b) ist die Macrostatekinetik der `xbix` Sequenz zu sehen. Sie sieht der Microstate Kinetik relativ ähnlich und gilt als qualitativ gut solange $\mathbf{E_{max}}$ nicht zu restriktiv gewählt ist. Der Einfluss der Energieschranke $\mathbf{E_{max}}$ auf die Qualität der Macrostate Kinetik wird in Kapitel 4 analysiert.

2.4.5 Arrhenius Kinetik

Diese Kinetik ist die performanteste der drei hier dargestellten Möglichkeiten. Sie rechnet wie die Macrostate Kinetik nur mit der Menge der Macrostates representierenden Minima, $\mathfrak{M}_{\mathfrak{C}_p}$. Demzufolge haben die Transitionsmatrizen von Arrhenius und Macrostate Kinetiken dieselben Ausmaße. Im Gegensatz zur dortigen Definition nutzt die Arrhenius Kinetik keine Kontaktflächen sondern das schon in Gleichung 2.66 definierte Metropolis Kriterium. Allerdings wird statt für zwei benachbarte Strukturen vereinfachend die Rate für ein Minimum $\hat{x}$ und die Barriere / den Sattel zu einem andere Minimum $\hat{y}$ berechnet. Daher gilt im Unterschied zur Microstate Kinetik immer $\Delta G > 0$ und eine Maximierungsfunktion ist daher überflüssig. Abbildung 2.6(b) verdeutlicht den Ansatz.

Theorie

Für jedes Minimum $\hat{x}$ existiert ein Macrostate bzw. Basin $\mathcal{B}_{\hat{x}}$. Die Berechnung der Transitionsraten k_{ij} basiert auf einer Formel ähnlich dem Metropolis Kriterium. Je nachdem ob die Arrhenius Kinetik einen BarrierTree oder ein SaddleNet nutzt, sind die einzelnen Raten unterschiedlich definiert.
Bei einer Arrhenius Kinetik mit **BarrierTree** wird die Barriere zwischen zwei Minima für die Ratenberechnung verwendet, und diese ist für jedes Minimapaar definiert. Demzufolge ergibt sich eine **vollständig belegte Ratenmatrix**:

$$\Delta G_{xy} = b_{\hat{x}\hat{y}} - f_E(\hat{x}) \tag{2.77}$$

$$k_{\hat{x}\hat{y}} = \Gamma_A \cdot e^{-\Delta G_{xy}/RT} \tag{2.78}$$

Bei einer Arrhenius Kinetik mit **SaddleNet** wird der Sattel zwischen zwei Minima für die Energiedifferenz benutzt. Dieser ist nur für Minimapaare mit benachbarten Basins definiert, demzufolge ergibt sich eine **nicht vollständig belegte Ratenmatrix**:

$$\Delta G_{xy} = f_E(t_{\hat{x}\hat{y}}) - f_E(\hat{x}) \tag{2.79}$$

$$k_{\hat{x}\hat{y}} = \begin{cases} \Gamma_A \cdot e^{-\Delta G_{xy}/RT} & \text{wenn Sattel } t_{\hat{x}\hat{y}} \text{ existiert} \\ 0 & \text{sonst} \end{cases} \tag{2.80}$$

Gleichung 2.78 ähnelt dem Metropolis Kriterium. Die Differenz zwischen Energie des Minimums und der Energie der Barriere die bei einem Walk ins andere Minimum überwunden werden muss, bestimmen die Transitionsrate, welche für jedes Minimapaar definiert ist, unabhängig von der durchschnittlichen Walklänge zwischen den beiden Minima.

Gleichung 3.20 nutzt statt einer Barriere einen Sattel und daher Adjazenzinformation, setzt also zumindest eine Nachbarschaft der Basins bzw. Macrostates voraus. Somit ist die resultierende Ratenmatrix im Aufbau der Macrostate Ratenmatrix sehr viel ähnlicher als der komplett belegten Matrix bei Nutzung des BarrierTrees, was eine qualitativ bessere Kinetik vermuten lässt.

Auch lässt die Adjazenz der Basins eine niedrige durschnittliche Walklänge vermuten, so dass die Chance steigt, dass die berechnete Transitionsrate der tatsächlichen Transitionsrate ähnelt. Dies lässt sich leicht erklären, wenn man das theoretische Extrem der Walklänge 1 betrachtet. Dann würde die Rate einer Microstate Transitionsrate entsprechen und somit die Qualität einer Microstate Kinetik erreicht werden. Dies ist natürlich niemals möglich, da wir Minima betrachten welche per Definition nicht benachbart sein können.

Umsetzung

Wie schon erwähnt gibt es zwei Varianten der **verwendeten Topologien**.
Der BarrierTree gelte hier als Standard, wie er auch in [FHSW02] beschrieben und genutzt wurde. Er hat den Vorteil, dass das für ihn benötigte Sampling Verfahren (siehe Abschnitt 3.2) sehr einfach ist. Das SaddleNet als zweite Variante ist im Gegensatz zum Stattelnetzwerk der Macrostate Kinetik unmodifiziert, das heisst es besitzt als Kantengewichte nur Sattelhöhen und dadurch Nachbarschaftsinformation.
Wie man an den Gleichungen 2.78 und 3.20 erkennen kann und oben schon erwähnt wurde, sehen die Ratenmatrizen sehr unterschiedlich aus.
Allerdings gibt es eine Möglichkeit, die Qualität der Arrheniuskinetik zu verbessern. Hierbei werden die Macrostateenergien anstatt der sonst verwendeten Minimaenergien in die Ratenberechnung einbezogen. Eine erste Vermutung legt nahe, dass die Kinetik dadurch an Qualität gewinnt. Ein Vergleich zwischen Arrhenius Kinetik mit Minima Energien und Macrostate Energien wird in Kapitel 4 diskutiert und ist in Abbildung 4.4 dargestellt.

Zur Berechnung sind folgende Schritte notwendig:

- Minima Sampling: Erzeugung von G mit ersten Barrieren falls BarrierTree
- *optionales* Landscape Flooding: exakte Barrieren/Sättel bis $\mathbf{E_{max}}$, Erzeugen der Z_i, Update auf G
- BarrierTree/SaddleNet Sampling: bessere Barrieren/neue Sättel finden, Update von G
- Transitionsmatrix Berechnung: $\mathbf{K}$ per Gleichung 2.78 oder 3.20, Input: G, *optional* Macrostate Energien durch Z_i (siehe Gleichung 2.69) statt Minimum Energien

In Abbildung 2.7(c) ist die BarrierTree basierte Arrhenius Kinetik der `xbix` Sequenz zu sehen. Sie zeigt qualitative Unterschiede zur Microstate und Macrostate Kinetik im mittleren Zeitverlauf auf. Dies ist nicht verwunderlich, da der Anfangs- und Endteil der Plots/Verlaufs per Definition ähnlich sein müssen (thermodynamisches Gleichgewicht am Ende des Plots ist garantiert und kein qualitatives Maß der Kinetik).

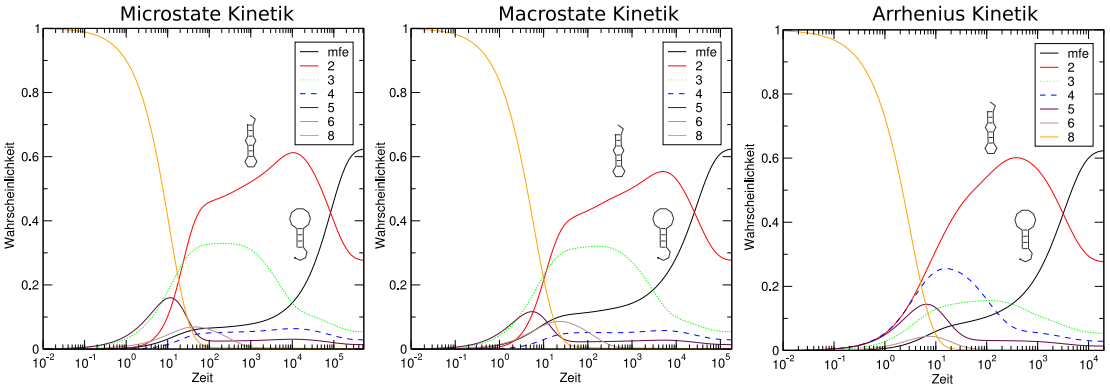


Abbildung 2.7: Die einzelnen Kinetikvarianten im Vergleich, auf Basis der `xbix` Sequenz; Qualität von links nach rechts abnehmend, Microstate Kinetik ist Referenz; man beachte die unterschiedlichen Endzeiten auf der logarithmischen Zeitskala welche das Erreichen des immer gleichen thermodynamischen Gleichgewichts widerspiegeln.

Kapitel 3

Algorithmen

Die in diesem Kapitel vorgestellten Algorithmen sind im wesentlichen unterteilt in vier logisch aufeinander aufbauende Abschnitte:

1. Abschnitt 3.1: Aufbau einer Topologie (SaddleNet oder BarrierTree) via **Minima-Sampling und LandscapeFlooding**.
2. Abschnitt 3.2: Optimierung unvollständiger Topologien via **BarrierTree Sampling und SaddleNet Sampling**.
3. Abschnitt 3.3: Erzeugung der Ratenmatrizen $\mathbf{K}_H$ der Hybridkinetik aus den Ratenmatrizen K_A der Arrhenius und $\mathbf{K}_M$ Macrostate Kinetik via **Matrix Hybridisierung**.
4. Abschnitt 3.4: Beschreibung des zeitkontinuierlichen Markov Prozesses, der zur **Berechnung der Faltungswahrscheinlichkeiten** die Hybrid Ratenmatrix $\mathbf{K}_H$ nutzt.

Diese so entstehenden Faltungswahrscheinlichkeiten für jeden Macrostate können dann z. B. mit `xmgrace` geplottet werden (siehe Abb. 2.7). Schliesslich wird zum Vergleich zweier Faltungswahrscheinlichkeiten bzw. der dazugehörigen Kinetiken ein trajektorienbasierter Algorithmus verwendet, welcher jedoch im Ergebnisteil näher erläutert wird, siehe Abschnitt 4.1.2.

Zum besseren Verständnis der einzelnen Punkte siehe die entsprechenden Abschnitte.

3.1 Algorithmen zur Erzeugung von Topologien

Motivation

Die in diesem Abschnitt beschriebenen Algorithmen dienen dem „Datamining“ der Daten für die späteren Kinetiken. Ihre Effizienz und Qualität bestimmt also direkt die Ergebnisse der herkömmlichen Faltungskinetiken und der darauf basierenden Hybridkinetik.

Der derzeit übliche Ansatz ist ein sehr zeiteffizientes Aufzählen aller Strukturen bis zu einer Energieschranke mittels `RNAsubopt`[1] [WFHS99], welches dynamische Programmierung verwendet. Dieses Programm allein ist sehr speichereffizient, *wenn* es nur um die Generierung aller Strukturen geht.

[1] `RNAsubopt` ist Teil des `RNA Vienna Packages` welches unter http://www.tbi.univie.ac.at/ bezogen werden kann

Da jedoch das im zweiten Schritt genutzte `barriers` Programm [WFHS04] zur Erstellung eines BarrierTrees der Landschaft eine *sortierte* Liste aller Strukturen benötigt, muss `RNAsubopt` entsprechend zu einer sortierten Ausgabe gezwungen werden. Dies führt jedoch zu einer vollständige Verwaltung aller Strukturen im Arbeitsspeicher. Entsprechend groß ist der Speicherverbrauch dieses Verfahrens, da bei RNA die Anzahl der Strukturen der Landschaft exponentiell in der Länge N der Sequenz mit ca. $N^{-\frac{3}{2}} \cdot 1,8^N$ wächst, siehe [Wat95].

Die im folgenden vorgestellten Algorithmen besitzen durchweg eine schlechtere Laufzeit, bieten aber eine geringere bis keine Speicherlimitierung und ermöglichen somit das Erstellen umfangreicherer Topologien in großen Landschaften. Zudem können sie ohne weiteres auch für Proteine und andere Moleküle angewandt werden, sofern eine Energiefunktion und eine Nachbarschaftsgenerierung bereitgestellt wird.

3.1.1 Finden der Minima

Der im nachfolgenden Kapitel vorgestellte, zentrale Landscape Flooding Ansatz benötigt zum Fluten eines Basins das entsprechende Minimum als „Startpunkt“ des Flutens. Ist dieses dem Flutalgorithmus nicht anfangs bekannt, wird das entsprechende Basin ausgelassen und etwaige Kontaktflächen und Sättel zu anderen Basins gehen verloren. Dies bedeutet definitiv einen fehlenden Knoten im Topologiegraph (SaddleNet oder BarrierTree) und somit mindestens eine fehlende Transitionsrate in der resultierenden Ratenmatrix. Zudem würden so alle Strukturen des entsprechenden Macrostates in der späteren Kinetik vernachlässigt.

Unser Landscape Flooding Ansatz ist im allgemeinen in der Lage, neue Minima zu explorieren. Allerdings nur in einem rechenintensiveren Verfahren mit stark erhöhtem Speicherverbrauch und wird im folgenden nicht weiter diskutiert. Zudem ist die Erschliessung neuer Basins während des Flutens für ein konsistentes Flutlevel problematisch, da ein neu erschlossenes Basin zur Senkung der Flutlevels unter die gefundene Sattelhöhe führen kann. In diesem Falle wären die bisher behandelten Basins „zu hoch“ geflutet.

Daher ist es nötig möglichst viele/alle Minima, welche freie Energien bis zum gewünschten maximalen Flutlevel besitzen, dem Flutalgorithmus bereitzustellen, Minima oberhalb dieser Energieschranke werden selbstverständlich nicht benötigt.

Für das Finden der Minima sind zwei Fälle zu unterscheiden:

1. Die gesamte Landschaft ist klein genug, um durch das Programm `RNAsubopt` [WFHS99] erschlossen zu werden. Dieses zählt alle Strukturen innerhalb der Landschaft auf, diese werden dann sequentiell durch eine **Extraktion aller Minima** mittels des tools `RNAtpMinExtract` auf die Minimumeigenschaft (siehe Abschnitt 2.2.11) getestet. Dies ergibt verlässlich alle Minima und dient für kleinere Landschaften lediglich der Sicherheit die ein Samplingansatz nicht bietet, und ist zudem der schnellste Ansatz.

2. Wenn die Landschaft zu groß ist, um alle Strukturen in ihr vollständig aufzuzählen, wird `RNAsubopt` nur bis zu einer Energieschranke verwendet. Ab dieser Energieschranke wird anschliessend ein **Minima Sampling** gestartet, welches explizit Minima oberhalb der Schranke sucht. Dieser Algorithmus basiert auf dem in [Ric07]

vorgestellten Ansatz und ermöglicht ohne Speicherrestriktion das Finden neuer Minima.

Aufgrund dieses Ansatzes, der nicht alleinig auf Minima Sampling basiert, können wir ein vollständiges Fluten der Landschaft bis zu einer Energieschranke garantieren, wobei wir jedoch bezüglich Barrieren etc. nicht auf diesen Teil der Landschaft beschränkt sind.

Minima Extraction

Dieser Ansatz ist trivial, und wir daher nur kurz umrissen. `RNAsubopt` liefert eine Abfolge aller Strukturen, welche dann sequentiell abgearbeitet wird.
In jedem Schritt, sprich für jede Struktur, werden alle Nachbarn generiert. Diese werden, ähnlich wie beim Gradient Walk, komplett analysiert. Wenn kein kleinerer Nachbar, gemäß der Ordnung $\prec$, existiert, dann muss die Struktur ein Minimum sein, und wird gespeichert.

Da für dieses Verfahren, im Gegensatz zu `barriers`, keine sortierte Liste aller Strukturen vonnöten ist, braucht `RNAsubopt` diese nicht zur späteren Sortierung im Speicher halten, und kann diese direkt auf Platte speichern. Es ist also, ähnlich dem Minima Sampling, ein sehr speichereffizientes Verfahren, welches jedoch eine Erschließung aller Minima garantiert. Der Speicherverbrauch bleibt konstant, entspricht also $O(1)$.

Diese müssen schließlich noch sortiert werden, um eine konsistente Indizierung zu gewährliesten. Dies ist aber in $O(nlog(n)$ möglich, und, im Gegensatz zum Speichern und Sortieren der gesamten Energielandschaft durch `RNAsubopt`, kein limitierender Faktor.

Dieser Ansatz ist jedoch, gerade für große Landschaften, durch die Nachbargenerierung für alle Strukturen extrem zeitaufwendig. Genau genommen wird jedes Paar zweier benachbarter Strukturen zweimal betrachtet.

Für das schnelle Finden einer ausreichend großen Anzahl Minima eignet sich daher eher ein Sampling der Landschaft. Dieses ermöglicht im Gegensatz zum Minim Extraction Verfahren auch direkt den parallelen Aufbau eines BarrierTrees.

Minima Sampling

Als **Minima Sampling** wird ein „Abtasten“ der Landschaft bezeichnet, also mehrere Walks von üblicherweise verschiedenen Startpositionen aus. Konkret wird zuerst ein Random Walk (siehe Abschnitt 2.2) verwendet, beginnend in einem gegebenen Minimum, welches durch einen Minimum-Selektions-Algorithmus gewählt wird. Dieser Walk ist wahlweise tiefenlimitiert (keine Unterschreitung der Startenergie), da in den üblichen Fällen, durch das vorher beschriebene **Mimima Extraction** Verfahren zusammen mit `RNAsubopt`, alle Minima bis zu einer gewissen Energieschranke bekannt sind, diese zu unterschreiten also keinen Sinn hat.

Das Ende des Random Walk wird durch ein boolsches Kriterium, z. B. Längen- oder Energieüberschreitung, bestimmt (auch *Walk Abortion Criterion* genannt). Dann wird ein Random Adaptive Walk gewählt, welcher in einem Minimum endet, wie in Beweis 2.2.1 gezeigt.

Der Adaptive Walk ist performant, da er nur einen energetisch niedriger gelegenen Nachbarn finden muss, und nicht, wie der Gradient Walk, das Minimum aller Nachbarn. Vor allem besitzt ein *Random* Adaptive Walk von ein und derselben Startstruktur ausgehend verschiedene Möglichkeiten des Abstiegs. So wird bei mehrfacher Auswahl der gleichen Startstruktur des Walks nicht immer das gleiche Minimum erreicht. Schließlich wird das Minimum, also die Struktur in der der Adaptive Walk endet, zur Menge der Minima $\mathfrak{M}_{\mathfrak{E}_p}$ hinzugefügt, wenn dieses Minimum noch nicht bekannt war.

Zum besseren Verständnis siehe Bild Abbildung 3.1.

Algorithm 1 Minima Sampling

```
1: function RNAtpMinSample(RNA Sequenz S, Minima Menge M)
2:     if M ist leere Menge then
3:         s_mfe ← getMFEstructure(S)                    ▷ berechne MFE Struktur
4:         M ← s_mfe
5:     end if
6:     while |M| nicht konvergiert do
7:         m ← wähle Minimum aus M                       ▷ MinimumSelector
8:         r ← Ende des Random Walk beginnend in m       ▷ WalkAbortionCriterion
9:         m' ← Ende des Random Adaptive Walk der in r startet
10:        if m' ∉ M_{E_p} then
11:            M_{E_p} ← M_{E_p} ∪ {m'}
12:        end if
13:    end while
14: end function
```

Implementierung und Parameterset

Der Minima Sampling Algorithmus ist im tool `RNAtpMinSample` implementiert. Die für den Algorithmus relevanten Parameter sind in Tabelle 3.1 beschrieben. Der hier vorgestellte Algorithmus erzeugt eine Topologie nach Wunsch, also SaddleNet oder BarrierTree, und liest die Minima wahlweise aus einer der beiden Topologien oder einem MinimaSet (reine Menge der Minima ohne Sättel/Barrieren) ein.

	Werte		Beschreibung
`ms`	{1,2*,3}	Minimum Selektion:	1: progressiv, $i = (i+1) \bmod \lvert\mathfrak{M}_{\mathfrak{E}_p}\rvert$ 2: zufällig uniform verteilt 3: zufällig boltzmanngewichtet mit Faktor `bf`
`up`	{1,2*}	RandomWalk Typ:	1: Walk besitzt fixe Länge `len` 2: Walk besitzt Zufallslänge zw. `lenMin` u. `len`
`down`	{1,2*}	DownWalk Typ:	1: Gradient Walk 2: Adaptive Walk
`iter`	$[1 \dots 2^{32}]$	Abbruchkriterium:	Anzahl Samples erreicht
`ncsteps`	$[1 \dots 2^{32}]$	Abbruchkriterium:	letztes Minimum vor Anzahl Samples
`minMax`	$[1 \dots 2^{32}]$	Abbruchkriterium:	Anzahl Minima erreicht

Tabelle 3.1: Für Algorithmus relevante Parameter von `RNAtpMinSample`; mit * versehene Parameterwerte haben sich als optimal herausgestellt.

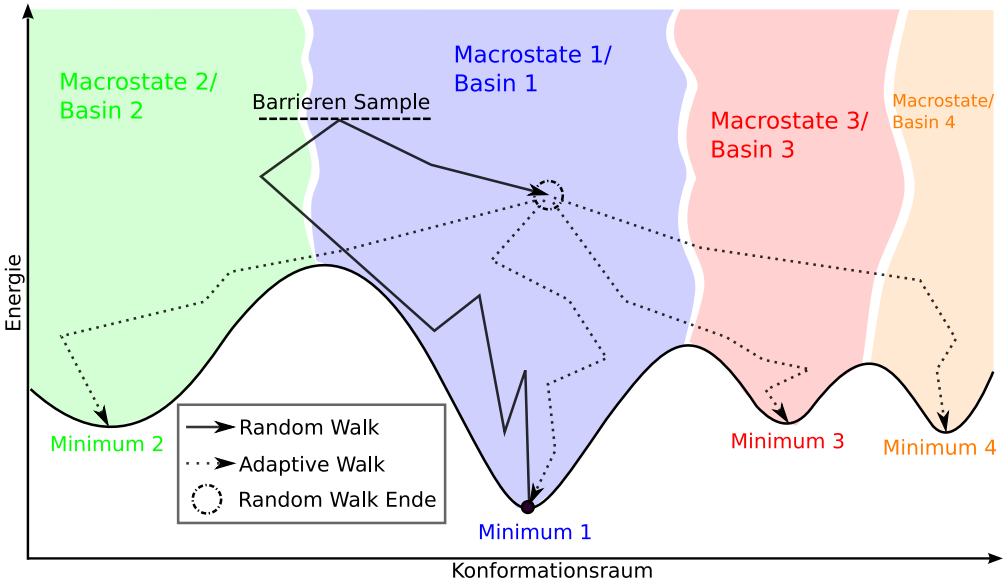


Abbildung 3.1: Minima Sampling einer Energielandschaft
Erster Random Walk startet z.B. in der **mfe** Struktur, hier Minimum 1, Walk endet nach Abbruchkriterium (z. B. maximale Länge), danach Wechsel zu performantem Random Adaptive Walk der per Definition in Minimum endet, dieser hat mehrere Möglichkeiten des Abstiegs; höchster Punkt von Random Walk optional als Barriere für BarrierTree genutzt; für SaddleNet nicht möglich da Nachbarschaft von Start- und Endbasin nicht garantiert (z. B. Random Adaptive Walk von Basin 1 nach Basin 4)

Laufzeit und Speicherverbrauch

Der Speicherverbrauch liegt klar bei $O(1)$ für einen einzelnen Walk. Es wird dauerhaft lediglich die resultierende Menge der Minima gespeichert, was einem Speicherverbrauch von $O(n)$ entspricht.

Da die durchschnittliche Anzahl der Nachbarn exponentiell mit der Länge N der RNA Sequenz ansteigt, ist die Laufzeit mit $O(iter \cdot C^N), 1 < C < 2$ abschätzbar, wobei *iter* die gewählte Anzahl an Samplingiterationen ist.

3.1.2 Fluten der Energielandschaft: Landscape Flooding

Als **LandScape Flooding** oder synonym „Fluten“ wird eine sequentielle Analyse der Landschaft verstanden. Dabei erfolgt im Gegensatz zum Sampling kein zufälliges „Abtasten“ sondern ein (vollständiges) „Füllen“ der Landschaft, welches man sich am besten als steigender Wasserspiegel vorstellen kann. Letzterer wird ab sofort als „Flutlevel“ bezeichnet, also eine Energiegrenze, bis zu der jede Struktur, einschliesslich Strukturen mit dieser Energie, betrachtet wurden. Der Flooding Algorithmus findet dabei exakte Sättel sowie Barrieren und erkennt sogar Teile der Kontaktflächen noch *bevor* das Flutlevel diese erreicht. Bild Abbildung 3.2 verdeutlicht den im folgenden näher beschriebenen Ansatz.

Allgemein betrachtet erfolgt also eine Energie sortierte, sukzessive Betrachtung der Landschaft. Dabei gelten alle Strukturen unterhalb des Flutlevels als schon abgearbeitet und

müssen nicht noch einmal analysiert werden. Unser Ansatz basiert im Gegensatz zum Programm `barriers` nicht auf `RNAsubopt`, er benötigt also nicht eine Aufzählung aller Strukturen vor der eigentlichen Analyse.
Statt dessen beginnen wir mit einer Menge von Minima, welche wir in eine sog. *Priority Queue* geben, und generieren die Energielandschaft „on the fly“, genauer: sukzessiv alle Strukturen welche zu den Basins der gegebenen Minima gehören. Zudem ist der hier vorgestellte Ansatz generischer, da keine sortierte „Vorgenerierung“ aller Strukturen notwendig ist.

Diese sortierte „Warteschlange“ ist, wie der Name verdeutlicht, nach Priorität sortiert (in diesem Fall nach der strengen Ordnung $\prec$, siehe Definition 2.2.7).

Ablauf

Anders ausgedrückt ist das derzeitige „Topelement“ **top** der Queue die Struktur mit niedrigster Energie, welches noch nicht betrachtet wurde. Für diese Struktur **top** werden nur Nachbarn höherer Energie betrachtet, da Nachbarn niedrigerer Energie/Ordnung vorher Topelement der Priority Queue waren und somit schon behandelt wurden. Für jede höher gelegenen Nachbarstruktur **neighbor** von **top** (**top** $\prec$ **neighbor**) gibt es zwei Möglichkeiten:

1. **neighbor** ist noch nicht in der Priority Queue vorhanden, und wird in diese einsortiert. Demzufolge ist das derzeitige Topelement **top** der **niedrigste Nachbar** von **neighbor**, da ansonsten ein vorheriges Toplement niedrigerer Ordnung **neighbor** als Nachbarn gehabt und in die Queue eingefügt hätte. Daher verläuft der Gradient Walk ausgehend von **neighbor** zwangsweise über **top** (siehe Definition 2.2.9), und demzufolge gehört **neighbor** zum gleichen Basin wie **top**. In die Queue wird also neben der Energie und der Struktur von **neighbor** (beides für die Ordnung $\prec$ und somit die Ordnung der Queue benötigt) auch der Basin-/Minimumindex von **top** auf **neighbor** übertragen.

2. **neighbor** ist schon in der Priority Queue vorhanden, besitzt also unter Umständen einen anderen Basinindex als **top**. Wenn dies der Fall ist, bilden **neighbor** und **top** zusammen einen Teil einer Kontaktfläche, die entsprechende Energie des Paares wird zur Energie der Kontaktfläche addiert. Zudem ist **neighbor** ein potentieller Sattel, da **top** $\prec$ **neighbor** und ein Sattel laut Definitionen 2.42 und 2.43 die energetisch höhergelegene Struktur eines Strukturpaares der Kontaktfläche ist. Daher wird geprüft, ob schon ein Sattel niedrigerer Ordnung im entsprechend verwalteten SaddleNet vorhanden ist, und dieser eventuell ersetzt falls **neighbor** $\prec$ dem alten Sattel ist. Der erste gefundene Sattelkandidat entspricht somit nicht unbedingt dem richtigen Sattel. (z.B. 2 Kontaktflächenpaare $s_1^A \sim s_1^B$ und $s_2^A \sim s_2^B$ mit $s_1^A \prec s_2^A \prec s_2^B \prec s_1^B$. Hier bei wird s_1^B zuerst entdeckt, aber s_2^B ist der kleinere Sattelkandidat.)

Also werden mögliche Sättel/Barrieren und demzufolge auch Kontaktflächen entdeckt *bevor* das Flutlevel sie erreicht, aber erst, wenn das Flutlevel der Sattelhöhe bzw. Barriere entspricht, können Sattelhöhe und Barriere als korrekt angenommen werden.

Schließlich wird jedes Topelement, nach Betrachtung all seiner Nachbarn, aus der Queue entfernt und ist für den restlichen Ablauf „verloren“. Nach der Abarbeitung von **top** nimmt das nächste Element höherer Ordnung in der Queue den Platz von **top** ein und das alte **top** Element wird verworfen.

Wie zuvor beschrieben, werden Sättel und Barrieren schon gefunden, bevor das Flutlevel diese erreicht, da bei der Analyse bzw Nachbargenerierung für das derzeitige Topelement der Priority Queue auch Strukturen *höherer* Energie gefunden und einzelnen Basins zugeordnet werden. Dies ist von großer Wichtigkeit, wenn aufgrund von Speicherlimitierungen nur ein Teil der Landschaft geflutet werden kann.

Adaption des Flutlevels aufgrund der Priority Queue Grösse

Wie bei allen Flutalgorithmen, ist der Arbeitsspeicher die limitierende Größe des Ansatzes. Die hier vorgestellte Methode ermöglicht jedoch eine flexible Adaption der maximalen Fluthöhe. Sobald das Speicherlimit bei einer zu großen Energielandschaft erreicht ist, wird die Queue sukzessiv um Strukturen oberhalb der kontinuierlich abgesenkten Energieschranke $Emax$ geleert. Dadurch wird ein Teil der Queue noch abgearbeitet, bevor schliesslich ein einzelnes Absenken der Energie um einen fixen Betrag die gesamte Queue leert. Der hier verwendete Energiebetrag ist $0.01\frac{kcal}{mol}$, gegeben durch die Granularität der verwendeten RNA Energiefunktion. Der Vorteil dieses Ansatzes liegt darin, dass die maximale Fluthöhe mit dem verfügbaren Speicher erreicht werden kann, ohne diese zuvor zu kennen. Zudem können, wie zuvor diskutiert, bereits Informationen über Kontaktflächen und Basinadjazenzen gesammelt worden sein, auch wenn diese Bereiche nicht mehr geflutet werden sollten.

Im folgenden der Algorithmus als Pseudocode:

Algorithm 2 Landscape Flooding

function RNAtpFlood(Minima Menge $\mathcal{M}$, $Emax$, $limit_{PQ}$, $deltaE$) ▷ $\mathcal{M}$ z. B. aus Minima Sampling
Require: $\mathcal{M}$ nicht leer, $deltaE > 0$, $limit_{PQ} \gg 0$

$R \leftarrow$ Gas Konstante
$T \leftarrow$ Temperatur in Kelvin ▷ üblicherweise $310.15 = 273.15 + 37°C$
$SN \leftarrow$ CreateSaddleNet ▷ beinhaltet alle Sättel und Kontaktflächenenergien
$Z[1 \ldots |\mathcal{M}|] \leftarrow 0$ ▷ alle kanonischen Zustandssummen mit 0 initialisieren

for all m_i in $\mathcal{M}$, $1 \leq i \leq |\mathcal{M}|$ **do**
  Priority Queue $PQ \leftarrow m_i$
  $PQ(m_i) \leftarrow i$ ▷ PQ speichert zu jeder Struktur das zugehörige Basin
end for
$E \leftarrow -\infty$
$top \leftarrow PQ.top()$ ▷ top als Struktur kleinster Energie in der PQ eingelesen

while (PQ not empty) and ($E < Emax$) **do**
  $E \leftarrow f_E(top)$ ▷ E ist Energie von top
  $basin_{top} \leftarrow PQ(top)$ ▷ hole Basinzugehörigkeit von top

  $Z[basin_{top}] \leftarrow Z[basin_{top}] + e^{-E/RT}$ ▷ siehe Gleichung 2.68
  $neighbors \leftarrow$ alle Nachbarn von top

  for all k in $neighbors$ **do**
    if $top \prec k$ **then**
      if k not in PQ **then**
        $PQ \leftarrow k$
        $PQ(k) \leftarrow basin_{top}$ ▷ k gehört zu Basin von top
      else ▷ k ist schon in PQ, hat Basin Index
        $basin_k \leftarrow PQ(k)$
        if $basin_{top} \neq basin_k$ **then**
          $SN(basin_{top}, basin_k).E \leftarrow SN(basin_{top}, basin_k).E + e^{-f_E(k)/RT}$ ▷ siehe Abschnitt 3.3.1
          if $f_E(k) < f_E(SN(basin_{top}, basin_k))$ **then**
            $SN(basin_{top}, basin_k) \leftarrow k$ ▷ ersetze Sattelkandidat mit k
          end if
        end if
      end if
    end if
  end for
  $PQ.pop()$ ▷ schmeisse top aus der Queue
  $top \leftarrow PQ.top()$ ▷ hole nächstes top Element
  while $PQ.size() > limit_{PQ}$ **do** ▷ vergleiche Anzahl Strukturen in Queue mit Queue Limit
    $Emax \leftarrow (Emax - deltaE)$
    ResizePQ($Emax$) ▷ alle Strukturen s mit $f_E(s) > Emax$ aus PQ entfernen
  end while
end whilereturn Z, SN
end function

Implementierung und Parameterset

Der Landscape Flooding Algorithmus ist im tool `RNAtpFlood` implementiert. Die für den Algorithmus relevanten Parameter sind in Tabelle 3.1 beschrieben. Der hier vorgestellte Algorithmus erzeugt eine Topologie nach Wunsch, also SaddleNet oder BarrierTree, und liest die Minima wahlweise aus einer der beiden Topologien oder einem MinimaSet (reine Menge der Minima ohne Sättel/Barrieren) ein. Der **besondere Vorteil** von `RNAtpFlood` gegenüber `barriers` besteht in der Möglichkeit, durch den Parameter `-i` eine iterativen Ansatz zu verwenden, welcher, beginnend bei der **mfe**, alle **delta** Energieschritte den Zustand der Landschaft (kanonische Zustandssummen, BarrierTree sowie ein SaddleNet

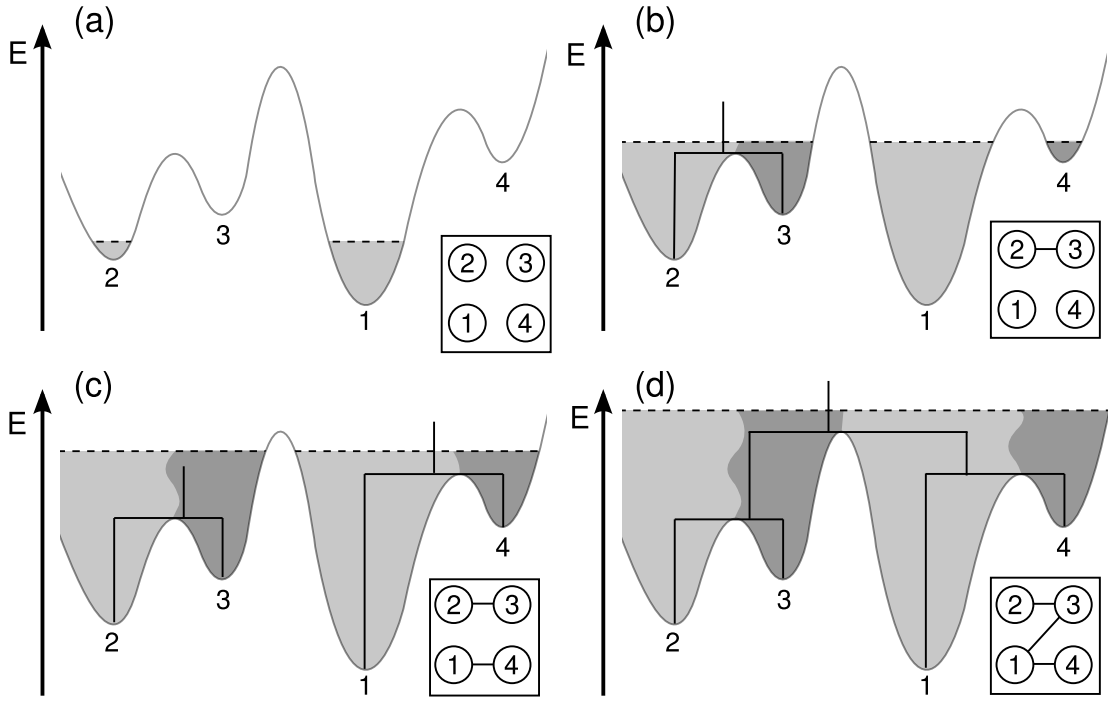


Abbildung 3.2: Landscape Flooding einer Energielandschaft
Gestrichelte Linie ist Flutlevel, Graph im Rechteck ist entsprechendes SaddleNet; (a) paralleles Fluten mehrerer Basins; Energien entsprechender Strukturen werden für kanonische Zustandssumme Z_i aufsummiert (b) erster Sattel gefunden und entsprechend Aktualisierung des SaddleNet oder BarrierTrees; (c) zweiter Sattel gefunden, entstehendes SaddleNet oder BarrierTree nicht zusammenhängend; (d) ab ausreichender Fluthöhe ist entsprechende Topologie zusammenhängend und vollständig.

als Adjazenzlisten) in wahlweise verlustfreier Binär- oder lesbarer ASCII Form auslagert, da die Priority Queue zwischengespeichert wird.

Somit ist für jede Sequenz nur *ein einziger* Flutvorgang vonnöten, und man kann auf diesem Datensatz (in Binärform zwischen 20 und 200 MB für die hier verwendeten Sequenzen) alle hier beschrieben .

	Werte		Beschreibung
`Emax`	[-10^3... 10^3]	max. Flutlevel	kann durch zu kleinen `buffer` nicht erreicht werden
`Emin`	[-10^3... 10^3]	min. Flutlevel	muss < `Emax` sein, nur verwendet wenn `i=true`
`buffer`	[10^5... **max**]	Buffergrösse	Anzahl der erlaubten PQ Elemente
`i`	{true, false	(i)terieren	false: von **mfe** bis `Emax` nur eine Ausgabe für `Emax`
			true: `Emin` bis `Emax` in `deltaE` Schritten, pro Schritt Ausgabe

Tabelle 3.2: Für Algorithmus relevante Parameter von `RNAtpMinFlood`; **max** als maximale Anzahl Elemente der PriorityQueue (PQ) ist durch verfügbaren Arbeitsspeicher beschränkt.

Laufzeit und Speicherverbrauch

Der Speicherverbrauch beträgt $\approx N^{-\frac{3}{2}} \cdot 1,8^N \cdot k$ mit $k < 1$.

Begründung:
Der Speicherverbrauch der Priority Queue über die Zeit lässt sich als Glockenkurve darstellen, wobei der maximale Wert der Glockenkurve dem Speicherverbrauch entpricht und das Integral der Kurve dem Speicherverbrauch aller Strukturen der Landschaft. Diese Kurve wird durch Nutzung eines memory profiling tools, wie z. B. **massif** aus dem toolset **valgrind** (siehe [NS07]) [2], sichtbar.

Wenn wir nun diese Glockenkurve als Dichtefunktion einer Normalverteilung annähern, so lässt sich die Höhe der Kurve, sprich der maximale Speicherverbrauch, direkt als linear abhängig vom Integral der Kurve (der „Fläche“ bzw. Anzahl aller betrachteten Strukturen), abschätzen.
Diese modellierte lineare Abhängigkeit lässt sich als Faktor k mit $k < 1$ beschreiben, wobei das Integral der Kurve als Menge aller Strukturen, wie schon vorher erwähnt, mit $N^{-\frac{3}{2}} \cdot 1,8^N$ abschätzbar ist.
Dieser Faktor k begründet auch den Speichervorteil der Implementierung `RNAtpFlood` gegenüber dem Programm `RNAsubopt`. Dessen Speicherverbrauch entspricht für eine sortierte Ausgabe einer stetig steigenden Funktion, deren Maximim am Ende, bei vollständiger Speicherung aller Strukturen, erreicht wird.

Genauere Betrachtungen des Speicherverbrauches sind in Kapitel 4 zu finden.

Die **Laufzeit** des Landscape Floodings beträgt $\approx (N^{-\frac{3}{2}} \cdot 1,8^N) \cdot C^N \cdot log_2(PQ.size())$.

Begründung:
Wenn wir eine Speicherlimitierung ausschliessen, müssen für jedes Element der Landschaft, also $\approx (N^{-\frac{3}{2}} \cdot 1,8^N)$ mal, alle Nachbarstrukturen generiert werden, von denen es wiederum im Schnitt C^N viele gibt. Zusätzlich müssen diese immer wieder auf Vorhandensein in der Priority Queue geprüft werden, was jeweils $log_2(PQ.size())$ Schritte bei binärer Suche benötigt. Dabei sei $PQ.size()$ die durchschnittliche Größe der Priority Queue, und entspricht dem durchschnittlichen Speicherverbrauch des Algorithmus.

[2] Zu erhalten unter http://valgrind.org/

3.2 Topologie Sampling

Motivation

Dieser Abschnitt befasst sich mit den Sampling Verfahren, welche nicht exakte Topologien aufwerten bzw. vervollständigen. Sie werden meist dann angewandt, wenn eine Topologie für eine Landschaft nur unvollständig vorliegt, sprich z.B. das vollständige Fluten der Energielandschaft nicht möglich ist. Die hier beschriebenen Algorithmen werden also primär für große Energielandschaften genutzt.

Ist durch Speicherrestriktionen ein vollständiges Fluten der Landschaft nicht möglich, so entsteht im schlimmsten Fall eine unzusammenhängende Topologie (siehe Abbildung 3.2(c)), einige Minima sind von anderen Minima abgeschnitten.
Im besten Fall fehlen zumindest Barrieren oder Sättel, was zu unvollständigen Transitionsratenberechnungen (siehe Abschnitt 3.4) führt.

Eine zusammenhängende Topologie, also die Möglichkeit, von jedem bekannten Basin auf irgend einem Pfad in der Landschaft zu jedem anderen zu wandern, ist zwingend erforderlich für die Diagonalisierung der Ratenmatrix, wie sie in Abschnitt 3.4 beschrieben wird. Ohne diese Diagonalisierung ist eine effiziente Berechnung der Faltungswahrscheinlichkeiten für beliebige Zeitpunkte t nicht möglich. Alternativ liese sich die Ratenmatrix auf die untereinander verbundenen Minima einschränken, was für eine Macrostate Kinetik, welche mit den gewonnenen Sätteln/Barrieren der Samplingverfahren kaum rechnen kann, nötig ist, aber die Kinetiken entsprechend drastisch beeinflusst. Zudem würden so große Teile der Energielandschaft ignoriert.

Das Ziel ist somit, möglichst alle Barrieren/Sättel oberhalb des vorherigen maximalen Flutlevels zu finden oder zumindest abzuschätzen. Dies ist für BarrierTrees weitaus effizienter möglich als für SaddleNets, da eine Überprüfung der Adjazenz von Start- und Endbasin eines Samplingschrittes nicht nötig ist. Dies und die einfache speichereffiziente Struktur ist der Hauptgrund für die Nutzung der BarrierTrees. Das hier beschrieben SaddleNet Sampling soll nur dem Versuch dienen, eine Hybridkinetik auf SaddleNet Basis umzusetzen.

3.2.1 BarrierTree Sampling

Der hier beschriebene Ansatz basiert auf der Arbeit von Andreas Richter [Ric07], welcher einen generischen BarrierTree Sampling Algorithmus für beliebige Energielandschaften (nicht auf RNA begrenzt) entworfen und getestet hat. Für eine genauere Beschreibung der Topologie BarrierTree, der internen Verwaltung der Barrieren, und Ergebnissen zum BarrierTree Sampling, siehe dort.

Ablauf

Der Ansatz ähnelt dem Minima Sampling sehr:
Zuerst wird ein Minimum als Ausgangspunkt gewählt, durch einen Selektionsalgorithmus, den wir hier einfach abstrakt *Minimum Selector* nennen. Danach wird ein Random Walk gestartet, welcher nach einem boolschen Abbruchkriterium (Walk Abortion Criterion) beendet wird. Hierbei wird der Random Walk jedoch so modifiert, dass er zumindest

eine gewisse Energieschranke durchbricht, und zwar die maximale Fluthöhe des vorherigen Landscape Floodings. Unterhalb dieser sind alle Barrieren zwischen Basins bekannter Minima vorhanden.
Da alle Minima als bekannt vorausgesetzt werden und so durch das BarrierTree Sampling keine neuen Minima hinzugefügt werden müssen, hat diese Höhenrestriktion des Random Walks keine Nachteile.
Der darauf in diesem Endpunkt des Random Walk startende Random Adaptive Walk endet nach begrenzter Zeit in eine Minimum (siehe Beweis 2.2.1). Die Energie des höchsten Punktes des traversierten Pfades in der Landschaft wird schliesslich als Barriere zwischen den beiden Minima in den BarrierTree übernommen, falls keine bessere (niedrigere) Barriere verfügbar ist. Der Algorithmus wird in Abbildung 3.3 verdeutlicht.

Im Gegensatz zum Minima Sampling arbeitet das BarrierTree Sampling selbstverständlich nicht auf einem Minima Set, sondern auf einem gegebenen BarrierTree. Dieser speichert für nicht vorhandene Barrieren einen `MAX` Wert, der weit ausserhalb der möglichen Energiewerte liegt. Zudem werden andere Parameter verwendet, um separierte Minima (Barriere von diesen zu den meisten anderen Minima liefert `MAX`) möglichst schnell mit den anderen Minima des BarrierTrees über eine Barriere zu verbinden. Wie in [Ric07] gezeigt wurde, ist durch diesen Sampling Ansatz eine stetige Verbesserung des BarrierTrees garantiert.

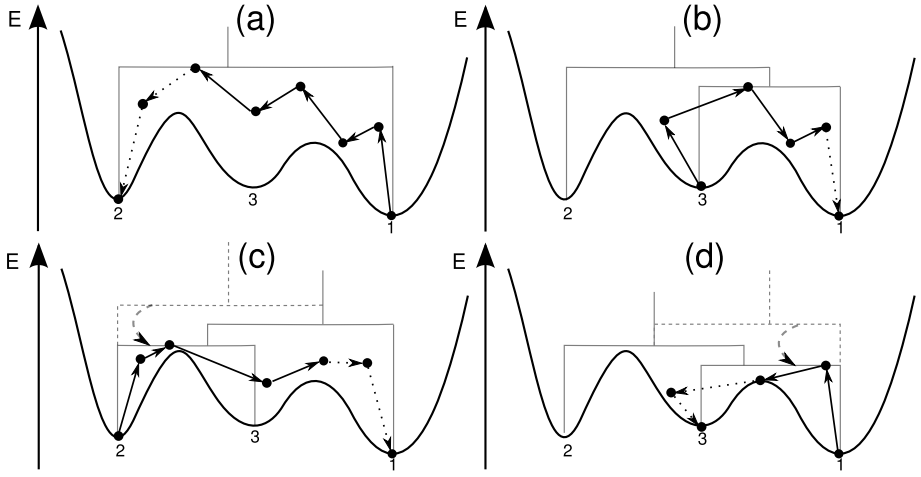


Abbildung 3.3: BarrierTree Sampling
Durchgezogene Linie ist Random Walk, gepunktete Adaptive Walk; (a) MinimumSelektor wählt 1 als Start, Adaptive Walk endet in 2, Barriere b_{12} auf höchste Energie gesetzt; (b) MinimumSelektor wählt 3, zweite Barriere b_{13} gesetzt, gleichzeitig b_{23} aus b_{12} abgeleitet, Hierarchie des BarrierTree ist korrekt; (c) Sampling updatet Barriere b_{12}, Hierarchieänderung und interne Umstellung der Baumstruktur, Hierarchie unkorrekt; (d) erneutes Update, wieder Umstellung der Baumstruktur, korrekte Barrierenhierarchie wiederhergestellt.

Der Algorithmus im Detail:

Algorithm 3 BarrierTree Sampling

function RNAtpBarrierEstim(BarrierTree $\mathcal{B}$, $Emax$)
Require: $\mathcal{B}$ besitzt mindestens 2 Minima

$h \leftarrow$ GetMediumBarrier($\mathcal{B}$)

while h nicht konvergiert **do**
  $m \leftarrow$ wähle Minimum aus $\mathcal{B}$ ▷ MinimumSelector
  $r \leftarrow$ Ende Random Walk der $Emax$ erreicht, in m beginnt ▷ WalkAbortionCriterion
  $r_{max} \leftarrow$ Struktur des Random Walks mit höchster Energie
  $m' \leftarrow$ Ende des Random Adaptive Walk der in r startet
  if $f_E(r_{max}) < \mathcal{B}(m, m')$ **then**
    $\mathcal{B}(m, m') \leftarrow f_E(r_{max})$ ▷ neue Barriere
    $h \leftarrow$ GetMediumBarrier($\mathcal{B}$) ▷ mittlere Sattelhöhe neu berechnen
  end if
end while
end function

3.2.2 SaddleNet Sampling

Wendet man das BarrierTree Sampling ohne Modifikation auf ein SaddleNet an, sprich interpretiert man die gefundene Barriere als Sattelhöhe, und trägt diese entsprechend ins SaddleNet ein, so entsteht eine inkorrekte Topologie. Dies wird deutlich, wenn man Abbildungen 3.3(a) und (c) betrachtet. Die dort gefundene Barriere als Sattel zu interpretieren bedeutet Adjazenzinformation zu „generieren“. Ein so falsch gesampletes SaddleNet

entwickelt sich langsam zum vollständig verbundenen Graphen, bei dem zwischen jedem Knotenpaar eine Kante existiert.

Um dies zu vermeiden, muss in jedem Samplingschritt die Adjazenz des Start- sowie Endbasins gewährleistet werden. Dies ist verschiedenartig möglich, erfordert aber immer mehrere Gradient Walks, welche sehr teuer sind. Statt wie beim BarrierTree Sampling nur die Start- und Endpunkte der Walks zu verwerten, ist beim SaddleNet Sampling zur Kontaktflächen Detektion ein Gradient Walk in jedem Walk Schritt vonnöten. So können jedoch auch mehrere Kontaktflächen in einer Samplingiteration erkannt werden.
Ein konvergierendes Fehlermaß wie beim BarrierTree (durchschnittliche Barrierenhöhe, hier die Sattelhöhe) gibt es nicht, da neue Sättel die durchschnittliche Sattelhöhe vergrößern können. Daher erfolgt das SaddleNet Sampling über eine Anzahl Iterationen.

Algorithm 4 SaddleNet Sampling

function RNAtpSaddleNetEstim(SaddleNet $\mathcal{N}$, $Emax$, iterations)
Require: $\mathcal{N}$ besitzt mindestens 2 Minima

$basin_{old} \leftarrow 0$

for $0 < steps \leq iterations$ **do**
  $m \leftarrow$ wähle Minimum aus $\mathcal{N}$ ▷ MinimumSelector

  while Abbruchkriterium nicht erfüllt **do** ▷ WalkAbortionCriterion
    if $f_E(w_i) < Emax$ **then** ▷ w_i ist i-te Struktur des Random Walk
      wähle zufällig Nachbarn w_{i+1} mit $f_E(w_{i+1}) > f_E(w_i)$ ▷ NeighborSelector
    else
      $basin_w \leftarrow$ Gradient Walk(w_i) ▷ Gradient Walk beginnend in derzeitiger Struktur
      if $(basin_w \neq basin_{old})$ und $basin_{old} \neq 0$ **then** ▷ Kontaktfläche gefunden
        $\mathcal{N}(basin_w, basin_{old}) \leftarrow max(old_w, w_i))$ ▷ Sattel wäre maximales Element
      end if
      $basin_{old} \leftarrow basin_w$
      wähle zufällig Nachbarn w_{i+1} mit $w_{i+1} \sim w_i, f_E(w_{i+1}) < Emax$ ▷ NeighborSelector
    end if
    $old_w \leftarrow w_i$ ▷ speichere Struktur zwischen
  end while
end for
end function

3.3 Hybridkinetik Berechnung

In diesem Abschnitt soll es um den eigentlichen Schwerpunkt gehen: die Verbindung der Arrhenius- sowie Macrostate Kinetik zu einer Hybridkinetik, indem die jeweiligen Ratenmatrizen miteinander verknüpft werden.
Das Ziel ist eine der Arrhenius Variante überlegene Kinetik. Die Macrostate Kinetik wird dabei als Referenz herangezogen, da eine Microstate Kinetik (siehe Abschnitt 2.4.3) für alles außer sehr kurze Sequenzen (z. B. `xbix`) nicht möglich ist.

Es werden im folgenden die beiden ursprünglichen Kinetik Varianten gegenübergestellt. Beide werden noch einmal näher beleuchtet, die einzelnen Formeln mit den jeweiligen tatsächlichen Implementierungen verglichen. Entsprechend werden die Schwächen und

Stärken der beiden Kinetiken erläutert, wobei die experimentelle Betrachtung zum Verhalten der Kinetiken erst in Kapitel 4 erfolgt.

Danach wird die Hybridkinetik erläutert. Der vorgestellte Ansatz wählt einen Skalierungsfaktor zwischen beiden Ratenmatrizen, welcher sich linear aus den einzelnen Macrostate Transitionsraten sowie den äquivalenten Arrhenius Transitionsraten berechnet.

Eine Übersicht der Kinetiken zeigt Tabelle 3.3.

	Arrheniuskinetik	Hybridkinetik	Macrostate Kinetik
Qualität*	niedrig	?	hoch
nutzt Basinadjazenz	nein	ja	ja
Speicherlimitierung	nein	nein	ja
Toplogie	BarrierTree**	BarrierTree** und SaddleNet	SaddleNet
Ratenmatrix	vollständig belegt	vollständig belegt**	sparse

Tabelle 3.3: Vergleich der Kinetiken
* die Qualität im Vergleich zur Microstate Kinetik, siehe Abb. 2.7 des letzten Kapitels, die Microstate Kinetik ist der Macrostate Kinetik ähnlich; ** Arrheniuskinetik auch auf SaddleNet möglich per SaddleNet Sampling, dann Hybridkinetik mit nur SaddleNet, sparse Ratenmatrix wie Macrostate Kinetik, dadurch vermutlich weit bessere Kinetik, auf Kosten eines rechenintensiveren Samplens, siehe Algorithmus 4.

3.3.1 Erzeugung der Macrostate Ratenmatrix $\mathbf{K}_M$

Die Formel aus dem Grundlagenkapitel wiederholend, lässt sich die Macrostate Transitionsrate k_{ij} zwischen zwei benachbarten Macrostates folgendermaßen berechnen:

$$\Delta G_{xy} = f_E(s_y) - f_E(s_x) \tag{3.1}$$

$$s_x \sim s_y \Leftrightarrow k_{xy}^{micro} = \Gamma_M \cdot \begin{cases} e^{-\Delta G_{xy}/RT} & \text{wenn } \Delta G_{xy} > 0 \\ 1 & \text{wenn } \Delta G_{xy} \leq 0 \end{cases} \tag{3.2}$$

$$s_x \not\sim s_y \Leftrightarrow k_{xy}^{micro} = 0 \tag{3.3}$$

$$k_{ij} = \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot Prob(x|\mathcal{B}_i)) \tag{3.4}$$

$$= \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot e^{-f_E(x)/RT}/Z_i) \tag{3.5}$$

$$= \frac{1}{Z_i} \cdot \sum_{\substack{x \in \mathcal{B}_i \\ y \in \mathcal{B}_j}} (k_{xy}^{micro} \cdot e^{-f_E(x)/RT}) \tag{3.6}$$

$$\mathcal{C}_{ij} = \emptyset \Leftrightarrow k_{ij} = 0 \tag{3.7}$$

Nehmen wir nun an, dass nur Strukturpaare der Kontaktfläche betrachtet werden, wie es der Flutalgorithmus 2 entsprechend umsetzt. Demzufolge gilt für jedes betrachtete Strukturpaar $(x, y) \in C_{ij}$ implizit $x \sim y$, und somit ist Gleichung 3.2 implizit gegeben. Dann

lässt sich der Klammerterm in Formel 3.6 dank Gleichung 3.2 in eine Fallunterscheidung umformen:

$$(\ldots) = \Gamma_M \cdot \begin{cases} e^{-f_E(y)+f_E(x)/RT} \cdot e^{-f_E(x)/RT} = e^{-f_E(y)/RT} & \text{wenn } f_E(y) > f_E(x) \\ 1 \cdot e^{-f_E(x)/RT} = e^{-f_E(x)/RT} & \text{wenn } f_E(y) < f_E(x) \end{cases} \tag{3.8}$$

Gleichung 3.8 lässt sich weiter vereinfachen zu

$$(k_{xy}^{micro} \cdot e^{-f_E(x)/RT}) = \Gamma_M \cdot e^{-MAX(f_E(x),f_E(y))/RT} \tag{3.9}$$

Somit vereinfacht sich die Gesamtberechnung von k_{ij} und k_{ji} zu

$$k_{ij} = \frac{1}{Z_i} \cdot \Gamma_M \cdot \sum_{(x,y)\in\mathcal{C}_{ij}} e^{-MAX(f_E(x),f_E(y))/RT} \tag{3.10}$$

$$k_{ji} = \frac{1}{Z_j} \cdot \Gamma_M \cdot \sum_{(x,y)\in\mathcal{C}_{ij}} e^{-MAX(f_E(x),f_E(y))/RT} \tag{3.11}$$

Offensichtlich unterscheidet sich der Summenterm von k_{ij} und k_{ji} nicht, so dass für beide Raten die gleiche Summe genutzt werden kann.

$$SUM_{ij} = \sum_{(x,y)\in\mathcal{C}_{ij}} e^{-MAX(f_E(x),f_E(y))/RT} \tag{3.12}$$

$$k_{ij} = \Gamma_M \cdot SUM_{ij}/Z_i \tag{3.13}$$

$$k_{ji} = \Gamma_M \cdot SUM_{ij}/Z_j \tag{3.14}$$

Somit wird diese Summe SUM_{ij} für jede Kontaktfläche C_{ij} einzeln gespeichert und für je zwei Ratenberechnungen genutzt.

Dies erfordert letztendlich das **Landscape Flooding**, da nur dort die entsprechende Summe für die einzelnen Kontaktflächen berechnet wird. Ein Samplingansatz ist somit für eine Macrostate Kinetik nicht möglich.

Vor dem Landscape Flooding ist entweder ein **Minima Sampling** oder eine **Minima Extraktion** vonnöten, da der angesprochene Floodingalgorithmus nur Basins fluten kann, für welche die Minima bekannt sind.

In Zeile 30 des Pseudocodes zum Landscape Flooding Pseudocodes (siehe 2) wird die Energie des gegenüber *top* höher gelegenen Nachbars aufsummiert, sprich

$$\ldots + e^{-f_E(k)/RT} \mathrel{\widehat{=}} \ldots + e^{-MAX(f_E(x),f_E(y))/RT} \tag{3.15}$$

Letztendlich müssen nach dem Fluten nur noch die einzelnen Kontaktflächensummen SUM_{ij} und kanonischen Zustandssummen Z_i mittels der Gleichung 3.13 zu einer Macrostate Transitionsrate verrechnet werden, ein trivialer Schritt.

Der **Zeitfaktor Γ_M der Macrostate Kinetik** wird dabei üblicherweise aus Mangel an experimentellen Daten ignoriert, also auf 1 gesetzt. Er bekommt jedoch noch eine entscheidenen Bedeutung, wenn wir Arrhenius und Macrostate Kinetik verknüpfen wollen, und wird im Abschnitt 3.3.3 entsprechend noch einmal genauer betrachtet.

Für die resultierende Macrostate Ratenmatrix $\mathbf{K}_M$ gilt:

$$k_{ij} = \begin{cases} > 0 & \text{wenn } \mathcal{B}_i \text{ und } \mathcal{B}_j \text{ benachbart} \\ 0 & \text{sonst} \end{cases} \tag{3.16}$$

Sie ist also keine vollständig belegte Matrix. In der Praxis stellt sich heraus, dass diese Macrostate Ratenmatrizen aufgrund der relativ niedrigen Anzahl von Kontaktflächen großteilig aus Nullen besteht, also auch als **„sparse“ Ratenmatrix** bezeichnet wird.
Demzufolge ist eine Abspeicherung der Matrix als Adjazenzliste sinnvoll.

3.3.2 Erzeugung der Arrhenius Ratenmatrix $\mathbf{K}_A$

Die Formel aus dem Grundlagenkapitel wiederholend, berechnet sich eine Transitionsrate k_{ij} der Arrhenius Ratenmatrix $\mathbf{K}_A$, standardmäßig auf Basis eines BarrierTrees gerechnet, sehr simpel wie folgt:

$$\Delta G_{xy} = b_{\hat{x}\hat{y}} - f_E(\hat{x}) \tag{3.17}$$

$$k_{\hat{x}\hat{y}} = \Gamma_A \cdot e^{-\Delta G_{xy}/RT} \tag{3.18}$$

Es ist offensichtlich, dass, im Gegensatz zur Macrostate Kinetik aus 3.3.1, keine Kontaktflächensumme benötigt wird, sondern nur die Energien der Minima und die Barrieren zwischen ihnen.
Die Energie der Minima ist implizit durch die Minima selbst gegeben, dafür ist ledigleich ein **Minima Sampling** oder die **Minima Extraktion** nötig.
Zur Berechnung der Barrieren sind sowohl **BarrierTree Sampling** als auch **Landscape Flooding** möglich. Im Falle des Landscape Floodings werden exakte Barrieren bis zur Fluthöhe (und eventuell darüber hinaus, siehe Kapitel 3.1.2) erzeugt, indem das entstehende SaddleNet in einen BarrierTree überführt wird, wie es in 2.3.2 beschrieben wurde.

Da wir das Fluten für die Macrostate Kinetik benötigen, wird in unserem Ansatz standardmäßig das Fluten zur Erzeugung der BarrierTrees genutzt, und somit auch für die Arrhenius Kinetik Berechnung. Erst ab einem limitierten Flutlevel wird das BarrierTree Sampling angewandt. Da das Landscape Flooding gleichzeitig die kanonischen Zustandssummen der einzelnen Macrostates berechnet (siehe Pseudocode 2 Zeile 19), und diese sich einfach zu einer Gesamtenergie des Basins umrechnen lassen (siehe Gleichung 2.69), bietet sich eine Nutzung der Macrostate Energien statt der eigentlichen Minima Energien an.

Ob dies einen qualitativen Vorteil ergibt, zeigt Abschnitt 4.2 des Ergebnisteils. Da zwischen jedem Minimum eine Barriere existiert, ist die resultierende **Ratenmatrix vollständig belegt**. Dies ist durch den verwendeten BarrierTree begründet. Es liegt nahe, dass dieser Unterschied im Ratenmatrixaufbau den schon vorhandenen qualitativen Nachteil der Arrhenius Kinetik gegenüber der Macrostate Kinetik noch verstärkt. Eine Lösung dafür wäre die Nutzung eines SaddleNet, wie im nächsten Abschnitt beschrieben.

Auch bei dieser Kinetik wird der **Zeitfaktor Γ_A der Arrhenius Kinetik** aus Mangel an experimentellen Daten auf 1 gesetzt. Seine Bedeutung unterscheidet sich jedoch vom Γ_M der Macrostate Kinetik, was in Abschnitt 3.3.3 näher betrachtet wird.

$\mathbf{K}_A$ auf Basis von SaddleNet

Als Variante der Arrhenius Kinetik ist eine Ratenberechnung auf Basis eines SaddleNet möglich. Dadurch werden nur Raten für adjazente Macrostates berechnet, die resultierende Ratenmatrix wäre eine schwach belegte **„sparse“ Matrix**, ähnlich der Macrostate Ratenmatrix. Entsprechend berechnen sich die eigentlichen Raten k_{ij} aus den Minima und den Sätteln t_{ij}.

$$\Delta G_{xy} = f_E(t_{\hat{x}\hat{y}}) - f_E(\hat{x}) \tag{3.19}$$

$$k_{\hat{x}\hat{y}} = \begin{cases} \Gamma_A \cdot e^{-\Delta G_{xy}/RT} & \text{wenn Sattel } t_{\hat{x}\hat{y}} \text{ existiert} \\ 0 & \text{sonst} \end{cases} \tag{3.20}$$

Das größte Problem dieses Verfahrens ist das dazu nötige **SaddleNet Sampling**. Dieses ist im Gegensatz zum BarrierTree Sampling sehr ineffizient und teuer.

3.3.3 Ratenmatrix Hybridisierung

Nachdem gezeigt wurde, wie die beiden benötigten Ratenmatrizen erzeugt werden, wird nun die mögliche Verschmelzung bzw. Hybridisierung beider Matrizen betrachtet.

Eine offensichtliche Möglichkeit einer Hybridisierung ist die (wo mögliche) Ersetzung der Raten aus der qualitativ schlechteren Arrheniuskinetik mit denen der qualitativ überlegenen Macrostate Kinetik. Dafür müsste aber zumindest der Zeitfaktor Γ beider Kinetiken identisch sein. Dieser wird im nächsten Abschnitt diskutiert, und eine Skalierung der Raten, sowie die Auswirkung auf die Kinetiken, betrachtet. Danach werden verschiedene Umsetzungen der Matrixhybridisierung kurz erläutert, die Ergebnisse befinden sich in Kapitel 4.

Zeitfaktor Γ

Wie schon weiter oben dargelegt, wird im allgemeinen der Zeitfaktor Γ ignoriert, sofern man eine einzelne Kinetik betrachtet und nicht den Versuch unternimmt, die artifizielle Zeitachse eines Kinetikplots in die reale Zeitachse zu transformieren. Dies wäre ohnehin durch fehlende experimentelle Daten derzeit unmöglich.

Im folgenden werden wir jedoch versuchen die Arrhenius Kinetik mit der Macrostate Kinetik verbinden, welche unterschiedliche Zeitskalierungen aufweisen (siehe Abbildung 2.7). Es ist daher sehr wichtig, die Auswirkung des Zeitfaktors Γ auf die Kinetik und die resultierenden Kinetikplots zu betrachten und zu verstehen.

Der Faktor Γ_M der Macrostate Kinetik steht für die verstreichende Zeit einer einzelnen Basenpaardeletion oder -insertion, also für einen SingleMove innerhalb der Landschaft.

Γ_A steht bei Verwendung eines BarrierTrees, im Gegensatz zu Γ_M, für die durchschnittliche Walklänge/Zeit um vom Minimum bis zum die Barriere definierenden Sattel zu gelangen, also mehreren Moves. Dieser Walklängendurchschnitt ist jedoch, durch die kaum vorhandene Adjazenzinformation des BarrierTree, für je zwei Minimapaare starken Schwankungen unterworfen. Somit verstärkt der so modellierte Zeitfaktor auf einem BarrierTree den schon vorhandenen qualitativen Nachteil der Arrhenius Kinetik gegenüber der Macrostate Kinetik, und macht eine konstante Skalierung der beiden Zeitfaktoren schwierig.

Dennoch wird offensichtlich, dass bei einer Vermischung von Transitionsraten eine Skalierung der Zeitfaktoren vonnöten ist, da generell Γ_A ein vielfaches von Γ_M darstellt. Dieser Ansatz der linearen Skalierung scheint bei Nutzung eines SaddleNet für die Arrheniuskinetik erfolgreicher, da die durchschnittliche Walklänge von einem Minimum zu einem adjazenten Sattel geringeren Schwankungen unterlegen ist als der Weg zu einer beliebigen Barriere.

Schließlich ist die Frage zu beantworten, welchen Einfluß der Zeitfaktor, welcher gleichermaßen auf jede Transitionsrate der Matrix Anwendung findet, Einfluß auf den Kurvenverlauf der Faltungswahrscheinlichkeiten nimmt. Dadurch wird verständlich, inwieweit eine Angleichung der Arrheniuskinetik an die Macrostate Kinetik möglich ist, und wo die Limitierungen des Skalierungsverfahrens bestehen.

Wir beweisen im folgenden, dass ein linearer Skalierungsfaktor, angewendet auf jedes Element einer Ratenmatrix $\mathbf{K}$, eine Verschiebung des Kinetikplots auf der logarithmischen Zeitachse bewirkt.

Beweis 3.3.1 (Skalierung von K um f verlagert Kinetikplot um $log_b(f)$)
Betrachten wir zuerst die Differentialgleichung 2.56 für die Wahrscheinlichkeitsverteilung P_t.

$$\frac{d}{dt}\vec{P}_t = \mathbf{K} \cdot \vec{P}_t \tag{3.21}$$

Diese Gleichung lässt sich nun mit einem beliebigen Faktor f versehen, z. B. $f = 2$.

$$2 \cdot \frac{d}{dt}\vec{P}_t = (2 \cdot \mathbf{K}) \cdot \vec{P}_t \tag{3.22}$$

Dieser Faktor bedeutet nun eine „Beschleunigung“ des gesamten Markov Prozesses, da alle Transitionsraten k_{ij} der Matrix $\mathbf{K}$ verdoppelt wurden (Skalarmultiplikation einer Matrix). Es gilt:

$$2 \cdot \frac{d}{dt}\vec{P}_t = \frac{d}{ds}\vec{P}_t \ , \ s = t/2 \tag{3.23}$$

Man kann dies als Überführung der Zeitachse von einer Einheit t in eine andere Einheit s interpretieren. Da die Kinetikplots jedoch auf einer logarithmischen Zeitskala basieren, müssen wir den Logarithmus auf die Gleichung $t = 2 \cdot s$ anwenden: $log_{10}(t) = log_{10}(s) + log_{10}(2)$, also

$$log_{10}(s) = log_{10}(t) - log_{10}(2) \tag{3.24}$$

Bei einer Skalierung der Ratenmatrix $\mathbf{K}$ um f verlagert sich also der Kinetikplot auf einer Zeitachse, welche auf dem Logarithmus zur Basis b basiert, um $log_b(f)$.

In unseren Plots nutzen wir der Übersicht wegen den dekadischen Logarithmus, ein Faktor von 2 würde also den Plot um $log_{10}(2) \approx 0,30$ nach links verschieben.

Hybrid Kinetik mit globaler Skalierung

Die erste Idee, nachdem eine Skalierung des Zeitfaktors vonnöten ist, entspricht einem linearen Skalierungsfaktor. Die Berechnung dieses Faktors erfolgt nach dem folgenden Schema:

Algorithm 5 Matrix Hybridisierung mit globalem Skalierungsfaktor

```
 1: function MergeMatrices(K_A, K_M)
Require: K_A und K_M haben gleiche Dimension > 1
 2:
 3:     count ← 0
 4:     scale ← 0
 5:     for 1 ≤ i ≤ Dimension do
 6:         for 1 ≤ j ≤ Dimension do
 7:             if (K_M[i][j] > 0) then
 8:                 count ← count + 1
 9:                 scale ← scale + (K_M[i][j]/K_A[i][j])
10:             end if
11:         end for
12:     end for
13:     scale ← MIN(1, scale/count)                    ▷ siehe Bemerkung
14:     for 1 ≤ i ≤ Dimension do
15:         for 1 ≤ j ≤ Dimension do
16:             if (K_M[i][j] > 0) then
17:                 K_H[i][j] ← K_M[i][j]
18:             else
19:                 K_H[i][j] ← K_A[i][j] · scale
20:             end if
21:         end for
22:     end for
23:     return K_H
24: end function
```

Zeile 13 des Pseudocodes kappt den verwendeten Faktor bei 1. Dies ist nötig, da im ungünstigsten Fall die ersten wenigen Macrostateraten zufälligerweise größer sein können als die Arrheniusraten. Entsprechend ist dann eine Verschlechterung der Hybridkinetik gegenüber der Arrheniuskinetik zu sehen. Wir wissen jedoch, dass das tatsächliche mittlere Verhältnis zwischen Arrhenius- und Macrostateraten kleiner als 1 sein muss, da der Zeitfaktor der Arrheniusraten Γ_A größer ist als der Faktor Γ_M der Macrostateraten. Daher limitieren wir, aufgrund der anfänglichen numerischen Instabilität des Skalierungsfaktors, diesen auf eine Maximalwert von 1.

3.4 Berechnung der Faltungswahrscheinlichkeiten

Die letztendlich berechnete Ratenmatrix $\mathbf{K}_H$ der Hybridkinetik muss schliesslich zu einer Folge von Wahrscheinlichkeitsvektoren transformiert werden, um am Ende den tatsächlichen Zeitverlauf des Systems betrachten zu können.
Diese Berechnung erfolgt durch den in Kapitel 2 schon erwähnten **Markov Prozess**, der lediglich eine quadratische Ratenmatrix und die Startverteilung P_0 der Wahrscheinlichkeiten nutzt, um die Wahrscheinlichkeitsverteilung P_t für jeglichen Zeitpunkt t zu berechnen.

Hier nun folgt eine genauere Betrachtung des Algorithmus, sowie seiner Umsetzung im tool `treekin`. Er stellt einen wichtigen, weil den Ansatz stark limitierenden, Teil der genutzten Algorithmen dar. Auf die Limitierung wird in einem späteren Abschnitts dieses Kapitel näher eingegangen.

Algorithmus im Detail

Zur näheren Betrachtung des eigentlichen Algorithmus knüpfen wir an die Definition 2.4.1 eines Markov Prozesses aus Kapitel 2 an. Die Lösung der Differentialgleichung 2.56 ist

$$\vec{P}_t = e^{\mathbf{K}t}\vec{P}_0 \tag{3.25}$$

wobei $e^{\mathbf{K}t}$ das **Matrixexponential** der Matrix $\mathbf{K}^t$ darstellt.
Es ist ähnlich der eulerschen Zahl e als Potenzreihe definiert: $e^{\mathbf{K}} = \sum_{l=0}^{\infty} \frac{\mathbf{K}^l}{l!}$. Allerdings wäre diese Berechnungsweise extrem ineffizient und zudem ungenau.

Angenommen $\mathbf{K}$ ist eine Diagonalmatrix, sprich $k_{ij} \neq 0 \Leftrightarrow i = j$, so ist auch $\mathbf{U} = \mathbf{K}^t$ eine Diagonalmatrix, $u_{ii} = (k_{ii})^t$. Die Berechnung von $e^{\mathbf{K}t}$ wäre somit trivial, wenn K eine Diagonalmatrix ist.
$\mathbf{U} = e^{\mathbf{K}t}$ ist dann auch eine Diagonalmatrix mit den Elementen $u_{ii} = e^{k_{ii}t}$, wobei e hier die Eulersche Zahl repräsentiert. Das Matrixexponential $e^{\mathbf{K}t}$ aus einer Diagonalmatrix $\mathbf{K}$ und t zu berechnen ist somit trivial in nur $O(n)$ Zeit mit $O(n)$ Speicher möglich, insofern man die Diagonalmatrizen $\mathbf{K}$ und $e^{\mathbf{K}t}$ jeweils als Vektor der Diagonalenwerte speichert.

Natürlich ist die Ratenmatrix $\mathbf{K}$ aber keine Diagonalmatrix. Durch eine Umformung, die allgemein als „Diagonalisierung“ bezeichnet wird, ist es uns aber dennoch möglich, $\mathbf{K}$ in eine Diagonalmatrix von Eigenwerten und einige Restmatrizen aufzuteilen (siehe [Wol01, ML03]).
Einfach ausgedrückt: $\mathbf{K} = \mathbf{N}\mathbf{\Lambda}\mathbf{M}$ wobei $\mathbf{\Lambda}$ eine Diagonalmatrix mit den Eigenwerten von $\mathbf{K}$ darstellt und auch $e^{\mathbf{\Lambda}t}$ eine Diagonalmatrix ist. Daraus ergibt sich $e^{\mathbf{K}t} = \mathbf{N}e^{\mathbf{\Lambda}t}\mathbf{M}$.
Dann gelten zur Berechnung einer Wahrscheinlichkeitsverteilung P_t für den Zeitpunkt t die Formeln:

$$\begin{aligned}
\vec{P}_t &= e^{\mathbf{K}t}\vec{P}_0 && (3.26)\\
&= \mathbf{N}e^{\mathbf{\Lambda}t}\mathbf{M}\vec{P}_0 && (3.27)\\
&= \mathbf{N}e^{\mathbf{\Lambda}t}\vec{V} && (3.28)\\
&\quad \text{mit} &&\\
v_i &= p_1^0 \cdot m_{i1} + \ldots + p_n^0 \cdot m_{in} && (3.29)\\
p_i^t &= e^{\mathbf{\Lambda}_{11}t} \cdot v_1 \cdot n_{i1} + \ldots + e^{\mathbf{\Lambda}_{nn}t} \cdot v_n \cdot n_{in} && (3.30)\\
&= e^{\lambda_1 t} \cdot v_1 \cdot n_{i1} + \ldots + e^{\lambda_n t} \cdot v_n \cdot n_{in} && (3.31)
\end{aligned}$$

wobei $\lambda_1, \ldots, \lambda_n$ die Eigenwerte der Matrix $\mathbf{K}$ sind.

Implementierung

Der von uns verwendete Markov Prozess ist im tool `treekin` implementiert ([Wol01], dort als `markov` bezeichnet). Dieses Programm verwendet nicht (wie Definition 2.4.1 nahelegt) als Eingabe einen Zustandsraum Q, also die Strukturmenge X der Topologie wie in Abschnitt 2.4.1 definiert, sondern benötigt lediglich die Ratenmatrix $\mathbf{K}$ sowie die Anfangsverteilung $\vec{P}_0$. Dies wird verständlich, wenn man Formel 3.26 betrachtet: außer $\mathbf{K}$, $\vec{P}_0$ sowie der Zeit t werden keine weiteren Daten gebraucht, um eine beliebige Wahrscheinlichkeitsverteilung $\vec{P}_t$ zu berechnen.

Die für die logarithmischen Plots in den in dieser Arbeit dargestellen Abbildungen verwendeten Zeiten werden aus den als Parameter übergebenen Zeitschranken t_{min} und t_{max} berechnet. Es gilt in `treekin`: $\forall_{t_i, t_{min} \leq t_i < t_{max}} : t_{i+1} = t_i \cdot 1{,}02$. Somit wird durch plotten der Werte in fixem Abstand (z. B. durch `xmgrace -log x -nxy < datei`) eine logarithmische Zeitskala realisiert. Für ein Beispiel eines daraus resultierenden Kinetikplots siehe Abbildung 2.7.

Der zugrundeliegende umkehrbare Markov Prozess besitzt eine stationäre Verteilung μ, welche dem thermodynamischen Gleichgewicht entspricht, also $\mu_x = e^{-f_E(x)/RT}/Z$. Dabei ist R die Gaskonstante und T die Temperatur in Kelvin. Z ist die kanonische Zustandssumme, auch als „canonical partition function“ bezeichnet, über alle betrachteten Zustände $x \in X$. Sie lässt sich mittels $Z = \sum_{x \in X} e^{-f_E(x)/RT}$ berechnen. Somit wird das thermodynamische Gleichgewicht unabhängig von der verwendeten Kinetik immer erreicht, siehe Abbildung 2.7. Es ist also **kein** Maß für die Qualität der Kinetik.

Die **Anfangsverteilung** $\vec{P}_0$ hat meist die Semantik

$$p_i^0 = \begin{cases} 1 & \hat{s}_i = \emptyset, \quad \hat{s}_i \text{ ist also die offene Kette} \\ 0 & sonst \end{cases} \tag{3.32}$$

Dies ist natürlich nur möglich, wenn die offene Kette $\hat{s}_{oc} = \emptyset$ Element von X bzw. $\mathfrak{M}_{\mathfrak{E}_p}$ ist. Wenn dies durch große Energielandschaften nicht mehr möglich ist, muss eine alternative Anfangsstruktur gefunden werden, was in dieser Arbeit aber nicht Gegenstand der Untersuchung ist. Wir beschränken uns bei den exemplarischen Untersuchungen auf Landschaften, die über die Energieschranke 0 hinaus geflutet werden können, wodurch die offene Kette mit dem Rest der Topologie verbunden bzw. Teil von ihr ist.

Zeitaufwand, Speicherverbrauch und Limitierung

Die Diagonalisierung einer $n \times n$ Ratenmatrix $\mathbf{K}$ kostet $O(n^3)$ Zeit und $O(n^2)$ Speicher.
Die einmalige Diagonalisierung zur Berechnung des Matrixexponentials $e^{\mathbf{\Lambda}}$ braucht $O(n^3)$ Zeit mit $O(n^2)$ Speicherverbrauch, wobei die Laufzeit grob mit $n^3/3$ geschätzt werden kann (siehe [ML03]). Das Ergebnis der Diagonalisierung von $\mathbf{K}$ ist, wie schon weiter oben erwähnt, der Term $\mathbf{N\Lambda M}$, wobei Λ die Diagonalmatrix der Eigenwerte darstellt (Speicherverbrauch $O(n)$), und $\mathbf{N}$ sowie $\mathbf{M}$ zusammen als eine einzige Matrix gespeichert werden können (Speicherverbrauch $O(n^2)$). Trotz ständigen Fortschritten in den verwendeten Verfahren gilt als gesichert, dass diese Laufzeit- sowie Speicherschranken auch in Zukunft nicht unterschritten werden.

Nach der Diagonalisierung kosteqt die Berechnung eines Vektors P_t $O(n^2)$ Zeit bei $O(n^2)$ Speicherverbrauch.
Siehe wiederrum [ML03]. Teil der Berechnung ist die Multiplikation von P_0 mit einer vollständigen (nicht diagonalen) Matrix. Demzufolge kann die Laufzeit von $O(n^2)$ nicht unterschritten werden.

Der **Ressourcenengpass des gesamten Algorithmus** entsteht demzufolge bei der Diagonalisierung, bei der mehr als eine quadratische Matrix erzeugt wird, siehe Formel 3.27. Der theoretische Speicherverbrauch beläuft sich zwar immer noch auf $O(n^2)$, jedoch stellt die Speicheranforderung der Matrixdiagonalisierung in der Praxis die direkte Limitierung des gesamten Kinetikansatzes dar.

Die Speicherlimitierung des Landscape Floodings scheint auf den ersten Blick das größere Problem, jedoch hat sich in der Praxis herausgestellt, dass die Anzahl gefundener Minima und somit die Größe der handhabbaren Matrizen den größeren Speicherverbrauch darstellen, da auch die Anzahl der Minima exponentiell steigt.

In der Praxis, sprich bei der Erstellung der Kinetikvergleiche im Ergebisteil, ist die Laufzeit ein Vielfaches der einzelnen Diagonalisierung und Berechnung einzelner P_t, da für jeden Vergleich mind. 20 verschiedene Energielevel sowie mindestens 800 Zeitpunkte gewählt wurden.

Zu entsprechenden Laufzeit- und Speicherlimitierungen in der Praxis siehe Abschnitt 4.5 im Ergebnisteil.

Kapitel 4

Ergebnisse

Dieses Kapitel ist in mehrere Abschnitte unterteilt. Zuerst erfolgen **Vorbetrachtungen**, welche die benutzten Sequenzen, sowie das verwendete Fehlermaß zum Vergleich von Kinetiken umfasst. Danach wird der Einfluß der Macrostate Energien auf die Arrheniuskinetik betrachtet, um so die Arrheniuskinetik und somit potentiell auch die Hybridkinetik aufzuwerten. Danach erfolgt ein **Vergleich der Macrostate und Arrheniuskinetik** anhand mehrerer Sequenzen, um die Zielstellung der Hybridkinetik definieren zu können. Dann wird die **Hybridkinetik** auf mehrere Sequenzen angewandt, und betrachtet, inwieweit diese Zielstellung erreicht wird. Schließlich wird noch der in der Praxis beobachtete Ressourcenverbrauch betrachtet, und die praktische **Limitierung** des Ansatzes analysiert.

4.1 Vorbetrachtungen

4.1.1 Verwendete Beispielsequenzen

Verwendet wurden mehrere kurze RNA-Sequenzen. Da der später in 4.5 beschriebene exponentielle Laufzeitzuwachs schon für Sequenzen mit Länge ~ 35 zu einem stark limitierenden Faktor wird, wurden durchweg kürzere Sequenzen untersucht. Im folgenden eine Auflistung:

xbix - `CUGCGGCUUUGGCUCUAGCC`
: Dies ist eine in [WSSF+04] vorgestellte künstlich generierte Sequenz, welche die für eine so kurze Sequenz untypische Eigenschaft besitzt, zwei stems parallel auszubilden. Dadurch entsteht die später zu beobachtende stark variierende Dynamik durch das Ausbilden mehrerer Sekundärstrukturelemente, bei nur sehr geringem Ressourcenverbrauch.

seqB - `ACGCGUACGACACGCAACGCAGU`
: Diese, ebenfalls künstlich generierte, kurze Sequenz stammt aus [Ric07].

am282 - `AAUCUAGCCUCUUCUAGGCUUUGUCUGU`
: Diese Sequenz stammt aus einer microRNA von *Anopheles gambiae*, einer in Afrika existierenden Moskito Art, welche die effektivsten bekannten Malaria Vektoren besitzt. Die Sequenz stellt einen Teil des miR-282 stem-loops dar.

microROSE - `CCAUCUUGCUCCUUGGAGGAUUUGG`
: Diese Sequenz ist, wie auch die folgenden, eine Subsequenz des bekannten ROSE Elementes (repressor of heat shock gene expression), einem RNA Thermometer. Dieses ist derzeitiger Forschungsgegenstand, siehe [CMAN06]. Welche Struktur microROSE in der `mfold` Struktur von ROSE einnimmt zeigt Abbildung 4.1.

subROSE - `GUACCCAUCUUGCUCCUUGGAGGAUUUGGCUAU`
Auch diese Sequenz ist eine Subsequenz des ROSE Elementes. Der genaue Abschnitt ist in Abbildung 4.1 zu sehen.

Tabelle 4.1 fasst die wichtigsten Daten grob zusammen, Abb. 4.1 zeigt die jeweiligen Strukturen innerhalb der ROSE mfe Struktur, bzw. die durch `RNAfold` berechneten mfe Strukturen für `xbix` und `am282`.

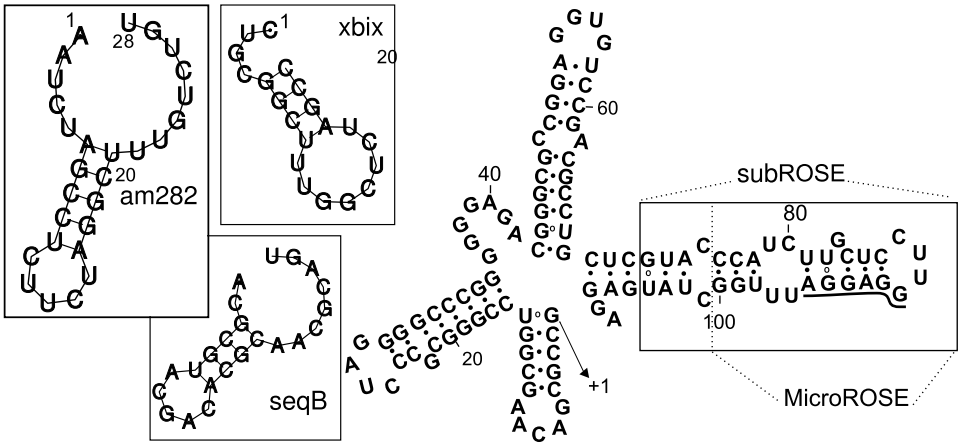


Abbildung 4.1: Verwendete Sequenzmotive und ihre mfe Strukturen
Die **mfe** Strukturen der `xbix` und `am282` Sequenzen wurden mittels `RNAfold` und `RNAplot` generiert [HFS+94], die ROSE Struktur stammt aus [CMAN06], sie wurde mittels `mfold` [ZMT99] erzeugt.

		Anzahl der		Energie		Barriere	
Name	Länge	Strukturen	Minima	mfe	max	min	max
`xbix`	20	3886	34	-4.3	21	-0.4	6.2
`seqB`	23	6226	94	-4.7	23.4	-3.5	6.9
`microROSE`	25	142063	160	-5.9	30.1	-2.4	9.9
`am282`	28	253370	220	-5.1	33.8	-4.1	13.8
`subROSE`	33	7603234	1337	-7	38.4	-5.2	16.8

Tabelle 4.1: Eigenschaften der verwendeten Sequenzen und ihrer Energielandschaften

4.1.2 Fehlermaß für Faltungskinetiken

Die Grundlage des Fehlermaßes ist eine gewichtete mittlere quadratische Abweichung (im englischen root mean square deviation = RMSD). In jedem Fall soll der RMSD mehreren Kriterien genügen: er soll eine Betrachtung des Kurvenverlaufs widerspiegeln, sprich bei einer starken „sichtbaren“ Veränderung des Kurvenverlaufs soll auch der RMSD eine signifikante Änderung aufweisen. Zum anderen soll dabei nur der „relevante Teil“ der Kinetik betrachtet werden, sprich den Teil der Zeitachse, in dem eine Veränderung wahrnehmbar ist (in Abbildung 4.2(a) der Bereich, welcher durch vertikale Linien eingegrenzt ist).

Es gibt jedoch im vorliegenden Fall grundsätzlich zwei Datensätze, aus denen ein RMSD berechnet werden kann. Zum einen die jeweiligen Ratenmatrizen der zu vergleichenden Kinetiken, zum anderen die aus diesen Matrizen resultierenden Trajektorienverläufe der

Wahrscheinlichkeiten. Der erste Ansatz, der **Matrix RMSD**, bietet auf den ersten Blick eine mathematisch sauberere Lösung, da die Matrizen die Grundlage der gesamten Wahrscheinlichkeitsverläufe darstellen und z.B. Artefakte durch das Samplen der Trajektorien keine Rolle spielen. Jedoch ist vorerst unklar, ob dieser Wert ein gleiches oder ähnliches Verhalten wie die Trajektorienabweichung aufweist. Im folgenden werden sowohl der **Trajektorien RMSD** als auch der Matrix RMSD näher vorgestellt. Anschliessend werden beide RMSDs experimentell miteinander verglichen, und ein finales Maß ausgewählt.

Matrix RMSD

Diese Möglichkeit der RMSD Berechnung soll hier nur kurz skizziert werden, da er später keine große Rolle mehr spielt. Wie in Beweis 3.3.1 dargelegt, ist eine Skalarmultiplikation der Ratenmatrix lediglich eine Verschiebung des Kinetikplots. Wir möchten ein- und dieselbe Matrix, mit verschiedenen Faktoren versehen, gleichwertig behandeln, der RMSD beider Matrizen zueinander soll in diesem Fall also möglichst 0 betragen. Entsprechend müssen wir dieser **Faktorinvarianz** innerhalb des RMSD Algorithmus Rechnung tragen. Im Falle des Matrix RMSD erreichen wir das, indem wir einen mittleren Skalierungsfaktor f zwischen beiden Matrizen berechnen. Dies ist identisch zu dem Verfahren der einfach Hybridkinetik mit globalem Skalierungsfaktor (siehe Algorithmus 5).

Schliesslich wird der Matrix RMSD ($final_{rmsd}$) durch die Formel

$$rmsd = \sum_{\substack{1<i\leq dim \\ 1<j\leq dim}} (\mathbf{K}_1[i][j] - f \cdot \mathbf{K}_2[i][j])^2 \tag{4.1}$$

$$final_{rmsd} = 100 \cdot (rmsd/dim^2) \tag{4.2}$$

berechnet. Dabei war $\mathbf{K}_2$ immer die Arrheniusratenmatrix, also eine vollständig belegte Ratenmatrix ohne Nullwerte, und dim die Dimension der Matrizen.

Trajektorien RMSD

Der Datensatz, auf dem der Trajektorien RMSD Algorithmus arbeitet, wird direkt vom Programm `treekin` generiert und besteht aus einer Zeitreihe von Wahrscheinlichkeitsvektoren. Die einzelnen Zeitpunkte der so bereitgestellten Samples sind in logarithmischer Form gegeben. Dies bedeutet, dass der jeweils nächste Zeitpunkt aufgrund eines Faktors (hier 1,02) aus dem vorherigen Zeitpunkt bestimmt wird.

Wie schon erwähnt, soll der RMSD faktorinvariant sein. Das bedeutet, dass eine Verschiebung des Graphen auf der logarithmischen Zeitskala, durch die Skalierung der Ratenmatrix mit einem konstanten Faktor, keine/kaum Auswirkungen auf den RMSD haben sollen. Eine solche Verschiebung ist in Abbildung 4.2 zu sehen. Um dies zu erreichen, wird für jede Kinetik separat ein „Fenster“ bestimmt (in Abbildung 4.2(a) mit vertikalen Linien markiert), aus dem logarithmisch gleichverteilt Samplingpunkte gewählt werden. Die Start- und Endzeitpunkte dieses Fensters werden über einen δ Wert (hier 10^{-4}) bestimmt. Dieser legt die Wahrscheinlichkeitsänderung fest, ab der das Fenster beginnt und wieder endet (mindestens eine Wahrscheinlichkeitsänderung muss jeweils $>$ und $<$ δ sein).

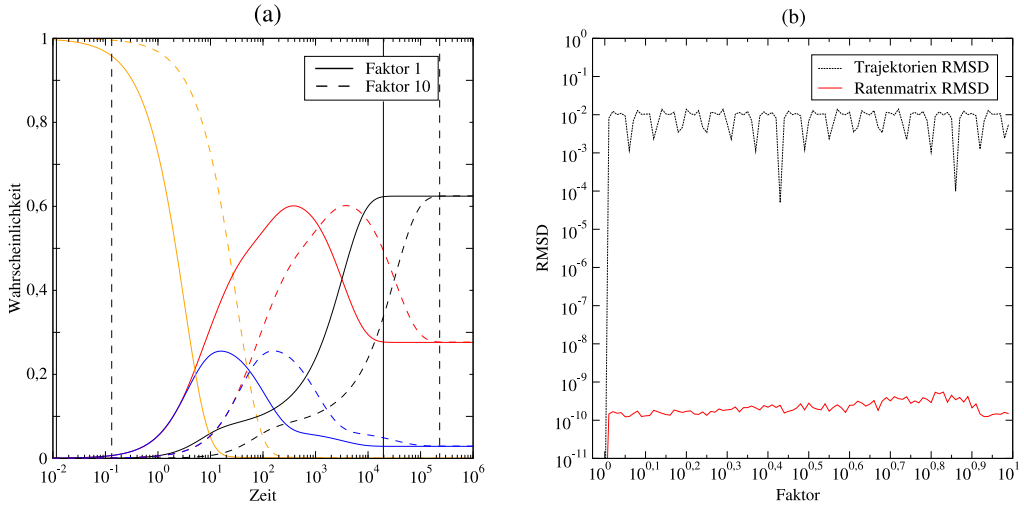


Abbildung 4.2: (a) 2 Varianten derselben Kinetik mit unterschiedlichem Ratenmatrixfaktor, zur Übersichtlichkeit werden nur die 4 bedeutensten Kurven angezeigt, vertikale Linien zeigen das Zeitfenster auf dem der RMSD berechnet wird, zwecks Faktorinvarianz; (b) Qualität der Faktorinvarianz, RMSD als Vergleich zwischen Original und einer um Faktor x modifizierten Ratenmatrix.

Der RMSD Algorithmus nutzt durchgehend 100 Samplingpunkte, welche aus dem beschriebenen Fenster des Datensatzes (und somit der logarithmischen Zeitskala) gleichverteilt bestimmt werden. Genauer gesagt werden diese per Interpolation zwischen verfügbaren, benachbarten Samples berechnet. Die Zahl 100 ist ein Kompromiss zwischen dem durch die Faktorinvarianz enstehenden Fehler (siehe Abbildung 4.2), der mit steigender Samplingdichte wächst, und einer ausreichenden Auflösung zur Detektion schneller Wahrscheinlichkeitsveränderungen. Allgemeine anfängliche Messungen haben zudem gezeigt, dass eine Erhöhung der Samplingdichte auf 1000 mit keinen nennenswerten Veränderungen der RMSD Werte einherging. Eine detailierte Beschreibung der Trajektorien RMSD Berechnung ist in Algorithmus **??** dargestellt.

Beide Arten des RMSDs besitzen somit eine Invarianz gegenüber einem Matrixfaktor f. Abbildung 4.2(b) zeigt, wie gut die Qualität dieser Invarianz ist, im Idealfall bliebe sie wie beim Anfangfaktor $f = 10^0 = 1$ bei 0. Wie aus der Abbildung ersichtlich wird, unterliegt der Matrix RMSD, im Gegensatz zum Trajektorien RMSD, nur wenigen Rundungsfehlern. Dies ist ob der zusätzlichen Rechenschritte um den Trajektorien RMSD zu berechnen (Samplen, Interpolation der Samplewerte, ...) auch nicht überraschend.

Abbildung 4.3 zeigt, wie die Qualität beider RMSD Varianten schließlich zu bewerten ist. In Abbildung 4.3(a) ist der Unterschied zwischen den Fluthöhen -3.1kcal und -3kcal klar zu erkennen. Beide RMSD Varianten zeigen an dieser Stelle auch einen Einbruch nach unten. Der Matrix RMSD jedoch eher gering, einen größeren Einbruch verzeichnet dieser beim Energieübergang 5.6kcal zu 5.7kcal. Wie Abbildung 4.3(b) jedoch zeigt, ist im Trajektorienverlauf kein Unterschied erkennbar (auch die der Übersichtlichkeit wegen ausgeblendeten kleineren Trajektorien zeigen keinen Unterschied, wie der Trajektorien RMSD schon andeutet). Demzufolge ist die Verhältnismässigkeit beim Matrix RMSD nicht gewährleistet, er zeigt andere Verhaltensmuster als die andere Variante und genügt daher nicht dem Anspruch einer Repräsentation der Wahrscheinlichkeitslandschaft.

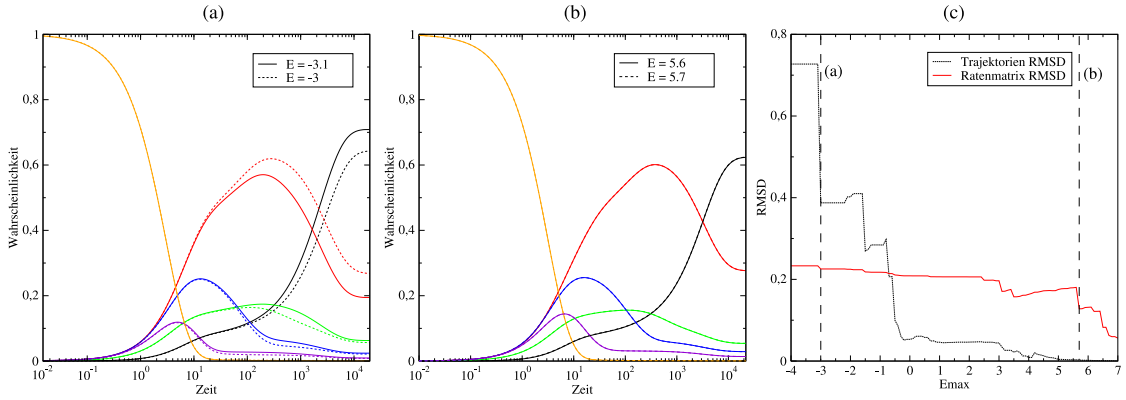


Abbildung 4.3: Vergleich der beiden RMSD Typen anhand einer Arrheniuskinetik: (a) sichtbare Veränderung beim Übergang zwischen -3.1 und -3.0; (b) keine sichtbare Veränderung beim Übergang 5.6 zu 5.7; (c) Die Reaktionen beider RMSD Typen bei den Graphveränderungen in (a) und (b)

Somit wird der trajektorienbasierte RMSD Ansatz gewählt, der hier in Algorithmus **??** kurz erläutert wird.

Algorithm 6 Kinetik RMSD

function RMSD($\mathcal{A}, \mathcal{B}$)
  for all Zeiten t **von** 0 **bis** n **do**
    for all Minima Indizes m **do**
      if $\mathcal{A}(t,m) - \mathcal{A}(t-1,m) < delta$ **then** $start_{\mathcal{A}} \leftarrow t$, break loops
      end if
    end for
  end for
  for all Zeiten t **von** n **bis** 0 **do**
    for all Minima Indizes m **do**
      if $\mathcal{A}(t,m) - \mathcal{A}(t-1,m) < delta$ **then** $start_{\mathcal{A}} \leftarrow t$, break loops
      end if
    end for
  end for
  $tfactor_{\mathcal{A}} \leftarrow exp(log(end_{\mathcal{A}}/start_{\mathcal{A}})/$Anzahl samples$)$
  wiederhole für $\mathcal{B}$

  $rmsd \leftarrow 0, rmsdsum \leftarrow 0$
  $t_{\mathcal{A}} \leftarrow start_{\mathcal{A}}, t_{\mathcal{B}} \leftarrow start_{\mathcal{B}}$
  for all samples i **von** 0 **bis** $samplesize$ **do**
    for all Minima m in Graph $\mathcal{A}$ **do**
      if m hat Äquivalent in $\mathcal{B}$ **then**
        $me \leftarrow$ Index dea äquivalenten Minima in $\mathcal{B}$
        $rmsd \leftarrow rmsd + (\mathcal{A}(t_{\mathcal{A}}, m) - \mathcal{B}(t_{\mathcal{B}}, me))^2$
      else
        $rmsd \leftarrow rmsd + \mathcal{A}(t_{\mathcal{A}}, m)^2$
      end if
    end for
    for all Minima m im Graph $\mathcal{B}$ **do**
      if m hat KEIN Äquivalent in $\mathcal{B}$ **then**
        $rmsd \leftarrow rmsd + \mathcal{B}(t_{\mathcal{B}}, m)^2$
      end if
    end for
    $rmsdsum \leftarrow rmsdsum + \sqrt{rmsd/\text{number of sums}}$
    $t_{\mathcal{A}} \leftarrow t_{\mathcal{A}} \cdot tfactor_{\mathcal{A}}$
    $t_{\mathcal{B}} \leftarrow t_{\mathcal{B}} \cdot tfactor_{\mathcal{B}}$
  end for
  return $100 \cdot (rmsdsum/samplesize)$
end function

Grundsätzlich werden beim RMSD nur Wahrscheinlichkeitswerte gleicher Minima miteinander verglichen. Wenn in Datensatz A ein Minimum existiert, welches in Datensatz B nicht vorhanden ist, so wird „so getan“ als wäre das Minimum doch entsprechend in Datensatz B vorhanden, aber hätte durchweg Wahrscheinlichkeit 0.

4.1.3 Probleme und Ad-Hoc Lösungen

Einige der grundlegenden Probleme, für die keine optimale Lösung verfügbar ist, sind:

- **Problem**: Unvollständige Anzahl Minima bei großen Landschaften:
  Lösung:
  - `RNAsubopt` und **Minima Extraktion** soweit möglich, danach **Minima Sampling** bis Minimazahl konvergiert
- **Problem**: Unzusammenhängende Topologie aufgrund unvollständigen Flutens (keine Diagonalisierung der Ratenmatrix möglich):
  Lösungen:
  - Restriktion der Minimamenge auf alle Minima welche von Ausgangsfaltung (offene Kette) innerhalb der Topologie erreichbare sind. Demzufolge eine Verringerung der Ratenmatrixdimension.
  - Sampling (SaddleNet Sampling/BarrierTree Sampling) bis Topologie vollständig verbunden ist.
- **Problem**: Fluten nur bis Energieschranke $E_{max} < 0$ möglich und somit keine Offene Kette $\hat{s}_{oc}$ als Ausgangsfaltung verfügbar (da $f_E(\hat{s}_{oc}) = 0$):
  Lösungen:
  - Wahl einer alternativen Ausgangsfaltung.
  - Manuelles Hinzufügen der offenen Kette und Verbinden dieser mit Topologie via Sampling (jedoch so keine/kaum Kontaktflächeninformation und daher nur für Arrheniuskinetik geeignet).

Diese Probleme traten bei der Erstellung der Ergebnisse in dieser Arbeit nicht auf, da nur kleine Sequenzen betrachtet wurden.

4.2 Einfluss der Macrostateenergien auf die Arrheniuskinetik

Motivation

Wie in Kapitel 3.3.2 beschrieben, nutzen wir für unsere Arrheniuskinetik einen BarrierTree. Dies ist eigentlich eine Modifikation der ursprünglichen Arrhenius Kinetik, da die Logik hinter der Ratenberechnung (Raten berechnen sich lediglich aus der zu überwindenden Energiedifferenz) offensichtlich auf einer Adjazenzannahme beruhen.

Hier nun werden wir eine weitere Modifikation testen. Wir werden Macrostate Energien anstelle der Energien der Minima für die Arrhenius Kinetik verwenden. Sprich, wir nutzen Information aus dem Flutungsalgorithmus, welcher für die Macrostate Kinetik verwendet werden muss, also Information, die ohnehin generiert wird, wenn eine Hybrid Kinetik

berechnet wird.
Es sollen in diesem Abschnitt zwei Fragen geklärt werden:

- Inwiefern helfen die Macrostateenergien der Qualität der Arrheniuskinetik?
- Wie schnell konvergiert eine Arrheniuskinetik mit Macrostateenergien? Dafür wird mit einer Arrheniuskinetik mit optimnalen, also aus vollständigem Fluten gewonnenen, Macrostateenergien verglichen.

Durch diese Modifikation wird die Qualität der Arrheniuskinetik abhängig von einem Flutlevel. Daher wird auch der RMSD der Arrhenius Kinetik in der folgenden Abbildung über verschiedene Flutlevel geplottet.

Sollte ein Minimum oberhalb der Fluthöhe liegen, wird als „fallback"nur die Energie des Minimums für die Arrheniusraten bzw. das verwendete Metropolis Kriterium (Kapitel 2.4.5) verwendet.

Ergebnisse und Schlussfolgerungen

In Abbildung 4.4 wird gezeigt, inwieweit die Macrostate Energien die Arrheniuskinetik verbessern.

Wie man in Abbildung (a) sieht, konvergiert die Arrheniuskinetik noch lange vor der maximalen Energie der Landschaft (bei beiden Sequenzen ~ 20) gegen 0. Dies liegt sicher an der relativ hohen Strukturdichte im mittleren Energiebereich. Höher gelegene Strukturen haben aufgrund ihres niedrigeren Energiebeitrages zur kanonischen Zustandsumme, auch einen niedrigeren Beitrag zur Macrostateenergie eines Basins, siehe die entsprechenden Formeln 2.68 und 2.69.

Abbildung (b) zeigt, dass die Nutzung der Macrostate Kinetiken kurzfristig zu einer Verschlechterung der Faltungsvorhersage kommen kann. Langfristig jedoch ist die Nutzung der Macrostate Energien immer die bessere Wahl (dies wurde auch durch die restlichen Sequenzen verfifiziert).

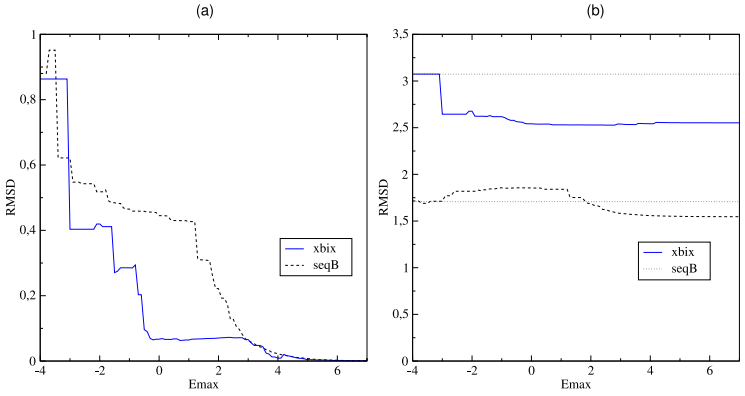


Abbildung 4.4: Jeweils Betrachtung einer Eigenschaft der Arrheniuskinetik mit Macrostateenergien: (a) Konvergenzverhalten der Arrheniuskinetik zu einer Arrheniuskinetik mit „optimalen“ Raten (b) Angleichung der Arrheniuskinetik an eine Macrostatekinetik mit maximaler Fluthöhe. Horizontale gepunktete Linien geben jeweils den RMSD an, der einer Arrheniuskinetik ohne Macrostateenergien entspricht.

4.3 Vergleich zwischen Macrostate und Arrhenius Kinetik

Motivation

Da die Hybridkinetik letztendlich eine Kombination der beiden vorgestellten Kinetik Ansätze darstellt, ist es wichtig, die Stärken und Schwächen beider Kinetiken gegenüberzustellen. Die folgenden Fragen sollen hier beantwortet werden:

- Wie stark ist die Macrostate Kinetik abhängig von der Fluthöhe?
- Wann übertrifft die Macrostate Kinetik die Arrheniuskinetik?
- Wo liegen demzufolge die Erwartungen an die Hybridkinetik?

Vorbetrachtung

Der Startvektor der Wahrscheinlichkeiten setzt immer die Wahrscheinlichkeit der offenen Kette auf 1, da eine in einer ungefalteten Kette beginnende Faltungssimulation erwünscht ist. Diese offene Kette als konstanter Ausgangspunkt jeglicher Simulation muss demnach mit (allen) anderen Minima verbunden sein. Dies ist kein Problem bei der Arrheniuskinetik, da hier ein ausreichend zusammenhängend gesampleter BarrierTree/SaddleNet genügt und als gegeben vorrausgesetzt wird.

Da aber bei der Macrostate Kinetik Kontaktflächen benötigt werden, muss bis zu einem gewissen minimalen Energielevel geflutet werden. Das bedeutet das maximale Flutlevel bestimmt bei der Macrostatekinetik den Aufbau der Topologie - im Gegensatz zur Arrheniuskinetik, bei der die Fluthöhe nur die Macrostateenergien beeinflusst und somit die Übergangsraten verbessert. Die offene Kette besitzt laut Turner Regeln [XSB+98] immer den Energiebeitrag null und ist immer ein lokales Minimum. Somit ist die Berechnung

eines durchweg konsistenten RMSD mit ein und demselben Ausgangsvektor (offene Kette = 1, Rest der Minima = 0) erst bei einer Energie größer null möglich ist. Ab dieser Energie bis zur ersten Barriere zwischen der offenen Kette und einem anderen Minimum ist zumindest eine Berechnung des RMSD möglich, wenn auch nicht sonderlich sinnvoll. (Die offene Kette hat durchweg Wahrscheinlichkeit 1 bis zum Erreichen der Barriere.)

Somit ist ein RMSD Plot mit Macrostate Kinetiken unter einer Fluthöhe von null nicht möglich. Dies verdeutlicht auch die grundlegende Einschränkung der Macrostatekinetik: das Fluten der Landschaft muss bis $E_{max} > 0$ erfolgen. Für lange Sequenzen ist dies, ob der Speicherabhängigkeit des Flutalgorithmus, schlichtweg unmöglich und entsprechend eine Motivation für die später vorgestellte Hybridkinetik.

Ergebnisse und Schlussfolgerungen

Für Beispielsequenzen aus Tabelle 4.1 wurde die Macrostatekinetik und die Arrheniuskinetik berechnet, jeweils mit unterschiedlichen maximalen Fluthöhen E_{max}. Dann wurde der RMSD zu einer „perfekten“, sprich durch vollständiges Fluten erzeugten, Macrostatekinetik berechnet. Die Ergebnisse sind in Abbildung 4.5 zu sehen. Ab $E = 6.2$ (`xbix`) respektive $E = 6.9$ (`am282`) sind alle Minima miteinander verbunden, der BarrierTree komplett. Ab $E = 21$ (`xbix`) und $E = 23.4$ (`am282`) ist die Energielandschaft vollständig geflutet. Der RMSD der teilgefluteten Macrostate Kinetik erreicht allerdings schon vorher 0, was auf Rundungsfehler zurückzuführen ist. Es wird schon früh ein sehr niedriger RMSD erreicht und die Trajektorienverläufe sind auch mit menschlichem Auge kaum zu unterscheiden. Das kurzzeitige Aufsteigen des RMSDs der Arrheniuskinetik ist eine Folge des Hinzufügens neuer Minima im Bereich < 6.4 und von Rundungsfehlern im Bereich nahe 18.

Die wichtigsten Schlussfolgerungen sind:

- Die Macrostatekinetik ist, bis zu einer Energiebarriere deutlich über 0, nicht zu gebrauchen.
- Die Arrheniuskinetik ist von einer beliebig niedrigen Energieschranke bis zu einer Energiebarriere deutlich über 0 weitaus besser als eine teilgeflutete Macrostatekinetik. Der Schnittpunkt der beiden Graphen wird immer über 0 liegen, und ist indirekt mit der Energiebarriere zwischen der Offenen Kette und dem Rest der Topologie korreliert.
- Die Arrheniuskinetik verbessert sich nur gering und stagniert schon recht früh, während die Macrostatekinetik sich stetig verbessert, und selbstverständlich RMSD 0 erreicht, sofern das Flutlevel am Maximum angelangt ist. Auch dies ist nicht verwunderlich, da hier ja mit einer „perfekten“ Macrostatekinetik verglichen wurde, und die „Form“ der Kinetikplots bei Nutzung der Arrheniuskinetik sich ab einem bestimmten Punkt nicht mehr an die Macrostatekinetik angleichen.

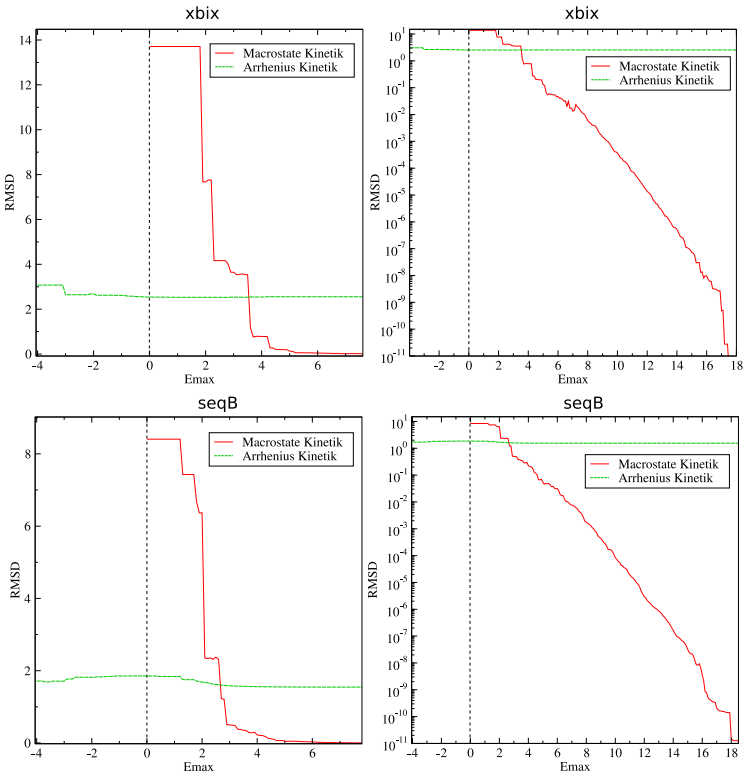


Abbildung 4.5: Vergleich zwischen den beiden Kinetik Varianten für `xbix` und `am282` Sequenz, jeweils als linearer und logarithmischer Plot. Die Macrostate Kinetik beginnt erst ab Fluthöhe 0 da vorher keine offene Kette - Bedingung für die Anfangsverteilung der Wahrscheinlichkeiten - Teil der Minimamenge ist.

4.4 Hybridkinetik

4.4.1 Zielstellung und erste Erwartungen

Durch die im letzten Abschnitt betrachteten RMSD Plots ist die **Zielsetzung** der Hybridkinetik klar definierbar. Der RMSD der Hybridkinetik sollte **für einen möglichst großen Energiebereich** unter den Werten der ursprünglichen Kinetiken bleiben.

Betrachten wir nun die zu erwartenden Resultate. Dafür definieren wir bestimmte Energiegrenzen. Sei b_{min} die niedrigste Barriere des BarrierTrees, und E_{max} die höchste betrachtbare Energie der Landschaft. Sei b_{mfe} die Barriere zwischen **mfe** und Offener Kette. Sei b_{oc} die niedrigste Barriere von der Offenen Kette zu einem beliebigen anderen Minimum. Es gilt: $b_{oc} > 0$, da laut verwendeter Energiefunktion auf Basis der Turner Regeln ([XSB+98, MSZT99]) die Energie der offenen Kette mit 0 definiert ist. Zudem ist die Offene Kette kein Schulterpunkt, somit ist keine Barriere gleicher Energie möglich.

Dann sollten die RMSD Plots der Hybridkinetik in folgende vier Abschnitte unterteilbar sein:

Abschnitt I - Flutlevel mfe bis $< b_{min}$
Das Fluten der Landschaft hat noch keine Kontaktfläche bzw. Sättel oder Barrieren gefunden, demzufolge können noch keine Macrostate Transitionsraten berechnet werden.
In diesem Bereich sind Hybridkinetik und Arrheniuskinetik identisch.

Abschnitt II - Flutlevel $\leq b_{min}$ bis $< b_{oc}$
Das Fluten findet den ersten Sattel bzw. die erste Kontaktfläche (u.U. bevor das Flutlevel diesen erreicht, siehe Begündung in Abschnitt 3.1.2). Somit kann die Matrixhybridierung nun erfolgen, da erste Macrostate Transitionsraten berechnet werden können. Eine Macrostate Kinetik allein kann durch die Isolation der Offenen Kette innerhalb des SaddleNet noch nicht sinnvoll berechnet werden, da keine Transitionsrate zwischen offener Kette und einem anderen Minimum existiert.
In diesem Bereich sollte die Hybridkinetik sowohl der Arrheniuskinetik als auch der Macrostate Kinetik gegenüber überlegen sein, *sofern die grundsätzliche Idee der Verschmelzung beider Ratenmatrizen erfolgreich umgesetzt wurde*, also die Arrheniuskinetik durch Nutzung der Macrostate Raten verbessert werden kann.

Abschnitt III - Flutlevel $\leq b_{oc}$ bis $\approx b_{mfe}$
Es wird, noch bevor das Flutlevel b_{oc} erreicht wird, der erste Sattel zwischen der Offenene Kette und einem beliebigen anderen Minimum gefunden. Nun kann die Macrostate Kinetik sinnvoll berechnet werden, ein Kurvenverlauf wird dargestellt. Der resultierende Kinetikplot ähnelt bei steigendem E_{max} schnell der Macrostate Kinetik auf höchstem Flutlevel, der RMSD der Macrostate Kinetik sinkt rasant. Ein weiteres kritisches Flutlevel ist erreicht, sobald die **mfe** Struktur von der Offenen Kette aus erreicht werden kann (unterhalb von Flutlevel b_{mfe}). Sprich das entsprechende SaddleNet beinhaltet einen Pfad zwischen beiden, bzw. der BarrierTree eine Barriere zwischen beiden. Spätestens hier sinkt der RMSD der Macrostate Kinetik unter den der Hybridkinetik.
Beim Flutlevel E_{hyb} verliert die Hybridkinetik an Bedeutung, da die für sie genutzte Macrostate Kinetik eine bessere Qualität aufweist. E_{hyb} befindet sich innerhalb von Abschnitt III und ist sehr stark von der Topologie der gerade betrachteten RNA

Sequenz abhängig.

Abschnitt IV - Flutlevel $\approx b_{mfe}$ bis E_{max}
Der RMSD der Macrostate Kinetik sinkt auf 0, der RMSD der Hybridkinetik stabilisiert sich, und erreicht keinesfalls 0. Dies ist dadurch begründet, dass die Hybridratenmatrix eine vollständig belegte Matrix ist, die der Macrostate Kinetik jedoch nur dünn besiedelt (sparse) ist. Die identischen Raten der Macrostate Kinetik werden in der Hybridkinetik durch den Skalierungsfaktor zusammen mit den Arrheniusraten „verzerrt“. Zudem ist keine/kaum Adjazenzinformation berücksichtigt.

Die Größe der interessanten Abschnitte II und III, abhängig von b_{min}, b_{oc} und $\sim b_{mfe}$, sind somit unabhängig von der verwendteten Hybridisierungsmethode, und können nicht beeinflusst werden.
Die Qualität der Hybridisierungsmethode wird daher, wie schon initial angekündigt, durch den Vergleich zur Arrhenius Kinetik definiert, nicht über die Macrostate Kinetik.

Jedoch hat somit selbst eine schlechte Matrixhybridisierung einen qualitativen Vorteil in einem recht großen Abschnitt der unteren Energielandschaft, sofern, wie oben erwähnt, die Arrhenius Kinetik in jedem Fall durch die Matrixhybridisierung verbessert wird.

4.4.2 Ergebnisse einer globalen Skalierung

Hier wird nun betrachtet, inwieweit sich eine Hybridkinetik anhand eines einzelnen, auf die Ratenmatrix angewandten Faktors entwickelt. Die Erzeugung der entsprechenden Ratenmatrizen wurde in Abschnitt 3.3.3 beschrieben.

Abbildung 4.6 zeigt den Kurvenverlauf für die vier verwendeten Sequenzen. Es lassen sich folgende Aussagen über die Hybridkinetik mit einzelnem Skalierungsfaktor treffen:

1. Die einfache Hybridkinetik ist zu keinem Zeitpunkt schlechter als die Arrheniuskinetik, speziell nicht in Abschnitt II und III.
2. Die unterste Barriere b_{min} hat in größeren Sequenzen als `xbix` kaum einen Einfluss, ein Unterschied zur Arrheniuskinetik ist beim Übergang von Abschnitt I zu II kaum wahrnehmbar.
3. Es gibt Sequenzen, bei denen die Macrostate Kinetik oberhalb des Flutlevels 0 die Hybridkinetik sehr schnell ad absurdum führt. Diese besitzen einen sehr kleines Abschnitt III.

Diese ersten Ergebnisse zusammenfassend lässt sich sagen, dass diese simple Form der Hybrid Kinetik in jedem Fall besser ist als die häufig verwendete Arrhenius Kinetik.
Die Macrostate Kinetik besitzt den natürlichen Nachteil, erst bei Verbindung der Anfangsstruktur über eine Kontaktfläche mit den restlichen Minima der Kinetik, eine Faltungsentwicklung wiederzugeben.

Eine Abzuschätzung, ob eine Hybridkinetik oberhalb eines $E_{max} = b_{oc}$ gut geeignet ist (Abschnitt III groß) oder nicht geeignet (Abschnitt III klein), scheint dank der Barrier-Trees der Sequenzen (siehe A) möglich.

So ist klar ersichtlich, dass im Falle der Sequenzen `xbix` und `am282` die Macrostate Kinetik eine kurzfristige Stagnation des RMSDs oberhalb der Arrhenius Kinetik aufweist, da die Barriere zwischen **mfe** Struktur und Offener Kette nicht der niedrigsten Barriere der Offenen Kette entspricht.

Somit wird bis zum Erreichen dieser ein völlig anderer stationärer Zustand des Systems, mit den bis dahin von der offenen Kette aus erreichbaren Minima, erreicht. Dies resultiert in einem breiten Abschnitt III und einer Überlegenheit der Hybridkinetik in diesem Bereich. Dieses Verhalten ist jedoch nicht direkt von der Größe der Landschaft abhängig, so dass die Hybridkinetik auch für größere Sequenzen oberhalb der Offenen Kette eine gegenüber der Macrostate Kinetik dominante Qualität zeigen kann.

Im Gegensatz dazu weisen die RMSD Plots für `microRose` und `subRose` einen sehr schmalen Abschnitt III auf. Dies führt dazu, dass die offene Kette sehr schnell mit der den Kinetikplot bzw. die stationäre Verteilung dominierenden **mfe** Struktur und anderen tief liegenden Minima verbunden ist. Somit wird sehr schnell ein ähnliches Zeitverhalten auch für nur teilgeflutete Macrostate Kinetiken möglich.

Es ist jedoch auch anhand der `xbix` Sequenz ersichtlich, dass die Barriere b_{mfe} zwischen der **mfe** Struktur und der Offenen Kette keine feste Aussage liefert.

4.5 Limitierungen in der Praxis

In diesem Abschnitt sollen der in der Praxis wahrgenommene Ressourcenverbrauch der den Hybridansatz limitierenden Tools `RNAtpFlood`, `RNAsubopt` sowie `treekin` näher betrachtet werden.

Speicherverbrauch `RNAtpFlood` vs. `RNAsubopt`

Grafik 4.7 zeigt den durch das tool **massif** (siehe [NS07]) beobachteten Speicherverbrauch von `RNAtpFlood` gegenüber dem tool `RNAsubopt` aus dem Vienna RNA Package. Dies greift die in Abschnitt 3.1.2 erfolgte theoretische Betrachtung zum unterschiedlichen Speicherverbrauch der beiden tools auf. Wie man sieht, ist, unabhängig von der Länge der Sequenz, RNAtpFlood immer im Vorteil, wenn auch nur in geringem Umfang. Dafür ist der von uns genutzte Floodingansatz generisch, also auch auf z. B. Protein-Modelle anwendbar [MWB07], und erlaubt das sequentielle Fluten einzelner Basins oder Basinpaare, um somit, auf Kosten der Laufzeit, einen größeren Abschnitt der Energielandschaft abzudecken. Dieses Verfahren ist jedoch nicht Inhalt dieser Arbeit.

Es soll nicht verschwiegen werden, dass derzeit ie Laufzeit von `RNAsubopt` in Verbindung mit `barriers` durch Verwendung dynamischer Programmierung der stetigen Nachbargenerierung von `RNAtpFlood` weit überlegen ist, da die Algorithmen speziell auf RNA angepasst sind. Allerdings beinhaltet die aktuell verfügbare RNA-Struktur Implementierung der für diese Arbeit verwendeten ELL noch einigen Optimierungsspielraum, sodass dieser Unterschied noch gesenkt werden kann.

Laufzeit `treekin`

Den Großteil der Laufzeit benötigt in der Praxis das tool `treekin`, das für die Berechnung der Diagonalisierung und der einzelnen Wahrscheinlichkeitsvektoren aus der Ratenmatrix K_H zuständig ist. Dies wird spätestens dann klar, wenn man den Ablauf zur Berechnung der RMSD Plots betrachtet, n ist dabei Anzahl der Minima.

1. **einmaliges** Fluten der Landschaft mittels `RNAtpFlood`, Daten für jedes Energielevel werden erzeugt
2. Berechnung der Ratenmatrix K_H durch die Algorithmen aus Abschnitt 3.3 in $O(n^2)$
3. Diagonalisierung der Ratenmatrix K_H in ca. $n^3/3$ Operationen ([ML03]) mittels `treekin`
4. Berechnung eines Wahrscheinlichkeitsvektors P_t für jeden der ~ 1000 Zeitpunkte, in je $O(n^2)$ Zeit mittels `treekin`
5. Vergleich der entsprechenden Wahrscheinlichkeitsvektoren durch Algorithmus **??** in $O(n \cdot 1000)$
6. Wiederholung der Prozedur ab 2., für ca. 20 bis 40 verschiedene Flutlevel/Energien

Die Spitze der Kurve stellt den maximalen Speicherverbrauch dar, wobei das Integral der Kurve (quasi die abgedeckte „Fläche“) den Speicherverbrauch aller Strukturen umfasst, der offensichtlich von unserem Flutalgorithmus nicht erreicht wird, von `RNAsubopt` jedoch durchaus, siehe den zweiten Teil der Abbildung.

Dieser Wert liegt bei Verwendung von `treekin` auf einem Rechner mit 32GB bei ca. 20000, wie .
Für RNA Sequenzen mittlerer Länge (50+) überschreitet die Anzahl der Minima die Grösse der durch die Kinetik berechenbaren Zustandsmenge bei weitem, wodurch die Energielandschaft durch eine obere Energieschranke $\mathbf{E_{max}}$ beschränkt werden muss.

Dazu kommt, dass `treekin` normalerweise ~ 1200 solcher Zeitpunkte berechnet. Dies lässt sich aufgrund des schon vorher erreichten thermodynamischen Gleichgewichtes und dem nur auf den dynamischen Zeitabschnitt fokussierten RMSD Algorithmus (siehe 4.1.2) oft auf 800 Zeitpunkte einschränken, ohne Auswirkungen auf die Ergebnisse.

All dies wiederum wird für ca. 20 verschiedene Flutlevel $Emax$ berechnet, um die Qualität der verschiedenen Kinetiken in Abhängigkeit von diesen Flutleveln darstellen zu können.

Die Gesamtlaufzeit lässt sich in unserem Fall also abschätzen mit:

1. **Diagonalisierung**

mit also mit $O(n)$

// Resultate bzw. Thematisieren dass dies $<$ `RNAsubopt` Limit ? Sonst ja unsinnig

// grundlegende Angaben zur Relation Sequenzlänge/Strukturanzahl vs. Laufzeit eines treekin Durchlaufs

// einfacher Graph mit Angaben zu den 4 kleinsten Sequenzen, in Minuten, evtl. logarithmisch

// Verweis auf die O Notationen aus Kapitel 3

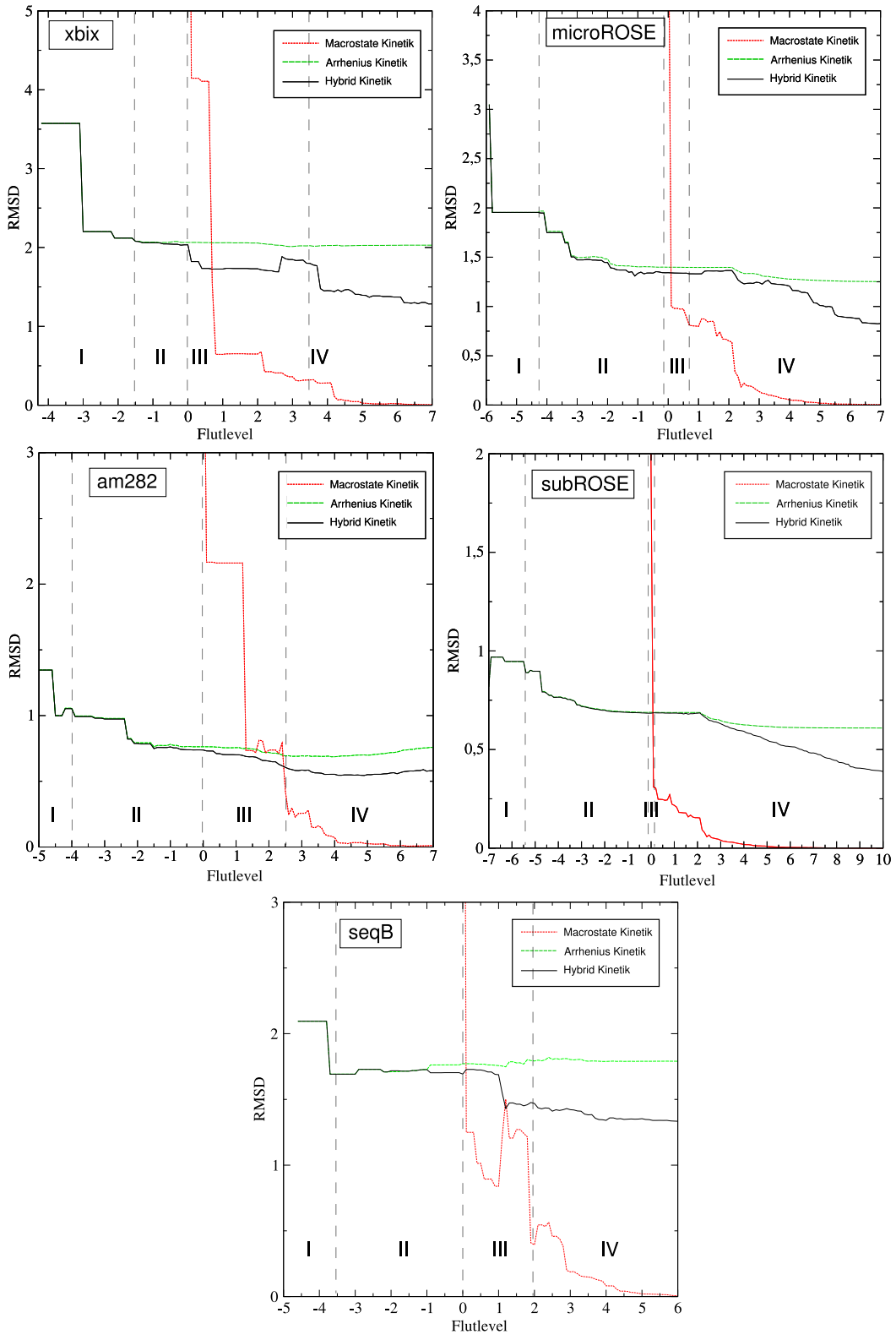


Abbildung 4.6: Die Hybridkinetik in der Praxis
Der dargestellte RMSD aller 3 Kinetikarten entsteht durch den Vergleich mit einer optimalen Macrostate Kinetik (maximales Flutlevel). Die jeweiligen in Abschnitt 4.4.1 beschriebenen Energiebänder sind mittels römischer Zahlen angegeben. Der Plot beginnt jeweils bei der **mfe**, und endet wenn sich die Kinetiken nicht mehr nennenswert ändern.

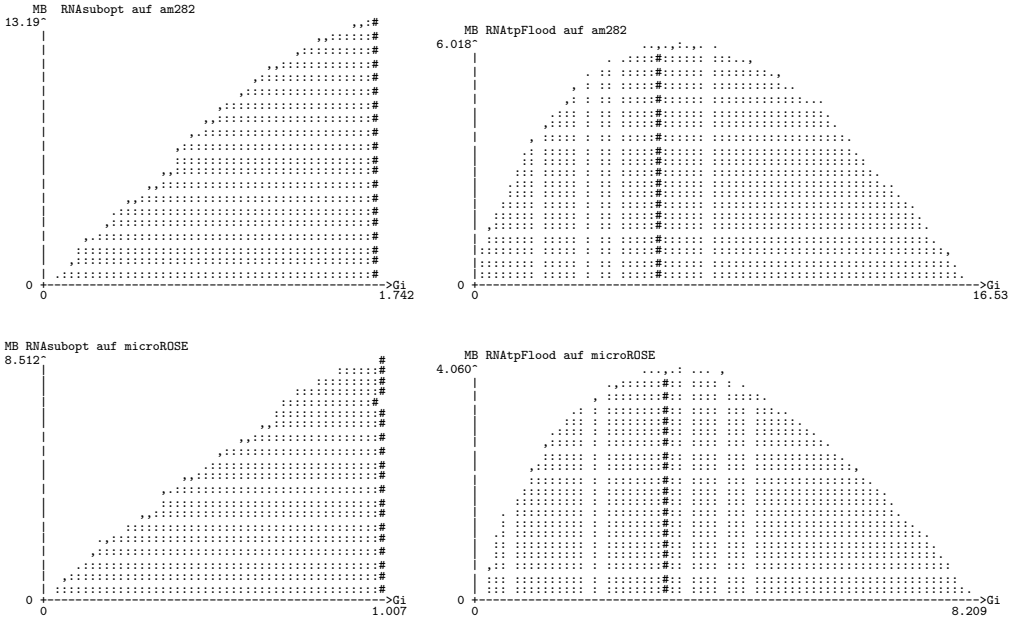


Abbildung 4.7: Vergleich RNAsubopt und RNAtpFlood
Vergleich des Speicherverbrauches zwischen `RNAsubopt` (links) und `RNAtpFlood` (rechts) mittels des heap profiling tools `massif` aus dem valgrind package (siehe [NS07]). Die oberen zwei Grafiken beziehen sich auf die `am282 Sequenz`, die unteren auf die `microROSE` Sequenz. Maßeinheit der Abszisse entspricht Milliarden Instruktionen (Giga instructions).

Kapitel 5

Diskussion

5.1 Zusammenfassung

Faltungskinetiken werden oft den qualitativ überlegenen Faltungssimulationen vorgezogen, da ihre Laufzeit nur den Bruchteil einer Simulation umfasst. Die üblichsten Faltungskinetiken sind dabei die qualitativ hochwertige Macrostate Kinetik, welche jedoch ein speicherlimitiertes Fluten der Landschaft benötigt, und erst ab einer gewissen Fluthöhe sinnvoll berechnet werden kann. Im Falle großer Energielandschaften wird deswegen oft die Arrhenius Kinetik verwendet, welche eine sehr einfach Abstraktion darstellt, und nur gesampelte Barrieren der Energielandschaft benötigt. Diese ist jedoch der Macrostate Kinetik qualitativ unterlegen.

In dieser Diplomarbeit untersuchten wir einen Hybrid Kinetik Ansatz zur Faltunsgvorhersage von RNA Molekülen auf Basis der beiden oben genannten Kinetiken. Es wurden die verschiedenen, zur Erzeugung der Faltungskinetiken verwendeten, Algorithmen vorgestellt und analysiert. Dabei stellten wir auch ein zum tool `barriers` alternativen Flutungsalgorithmus vor, der durch die alleinige Verwendung einer einzelnen Priority Queue einen geringeren Speicherverbrauch aufweist, und somit ein potentiell höheres Flutlevel erreichen kann.

Erstmalig wurde ein Fehlermaß für Faltungskinetiken vorgestellt und umgesetzt, welches mit dem Trajektorienverlauf der Faltungen korreliert und wahrnehmbare Veränderungen des Faltungsverlaufes wiedergibt. Dadurch wurde ein Vergleich der Kinetiken überhaupt erst möglich. Es wurden anschliessend mit Hilfe dieser Metrik die Arrhenius und Macrostate Kinetik im Detail miteinander verglichen, um die Zielstellung der Hybridkinetik zu definieren. Dabei wurde auch eine mögliche Verbesserung der Arrhenius Kinetik durch Verwendung von Macrostate Energien untersucht und entsprechend umgesetzt. Eine Variante zur Hybridisierung der zwei Ratenmatrizen wurde implementiert, und diese mit den jeweiligen ursprünglichen Kinetiken verglichen. Schliesslich wurde der Einfluß der Landschaftstopologie auf die beobachteten Ergebnisse sowie die Qualität der Hybridkinetik besprochen.

Die Ergebnisse zur Hybridkinetik zeigen, dass diese, immer im Vergleich zur Macrostate Kinetik einer vollständig gefluteten Landschaft, nie schlechtere Faltungsvorhersagen als die Arrhenius Kinetik trifft. Sie ist, ab einem gewissen niedrigen Flutlevel, der Arrheniuskinetik sogar dauerhaft überlegen. Demzufolge ist die Hybrid Kinetik bis zu einer Fluthöhe ≈ 0 beiden üblichen Kinetikvarianten, unabhängig der verwendeten Sequenz, vorzuziehen.

Je nach Beschaffenheit der Energielandschaft kann sie auch auf einem Flutlevel größer 0 der Macrostate Kinetik überlegen sein, im wesentlichen abhängig von der Barriere zwischen Offener Kette und **mfe** Struktur.

Eine konkrete Betrachtung der Limitierungen des Ansatzes zeigte schließlich auf, dass die Anwendung auf größeren Sequenzen vor allem durch die verwendete Matrixdiagonalisierung des tools `treekin` eingeschränkt wird.

5.2 Offene Probleme und Ansätze

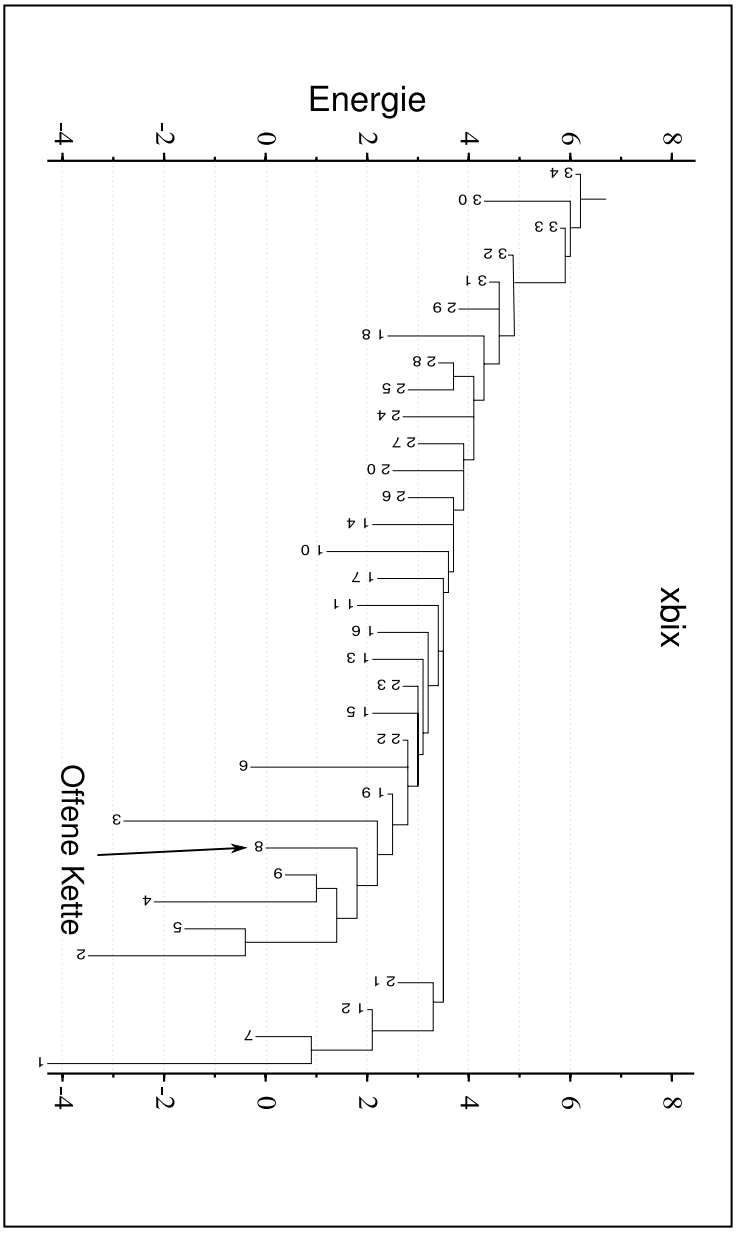
Der Ansatz zur Untersuchung größerer Sequenzen scheitert in erster Linie am Zeitaufkommen. Es wurde gegen Ende der Arbeit vorgeschlagen, den RMSD auf den Eigenwerten der Ratenmatrix, statt auf den Trajektorien, zu berechnen. Entsprechende Betrachtungen zur realen Laufzeit zeigen, dass dadurch die Berechnung des RMSD bei großer Minimazahl um mindestens den Faktor 3 beschleunigt würde.

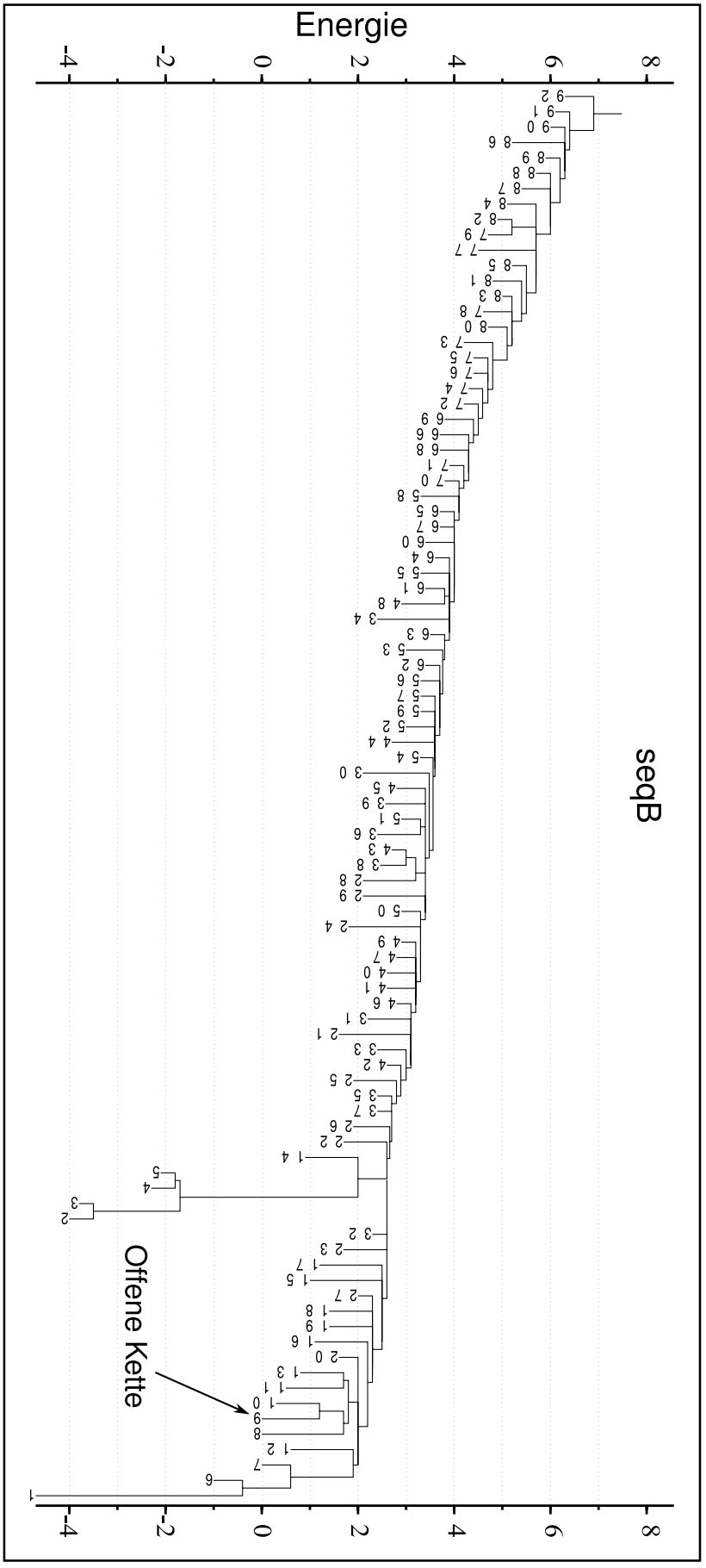
Ein weiterer Punkt ist die verwendete Hybridisierungstechnik, welche nur einen einzelnen Faktor berechnet. Da die Ratenmatrix der Hybridkinetik, durch Verwendung eines BarrierTrees für die Arrhenius Kinetik, vollständig belegt ist, aber die Macrostate Kinetik eine sparse Matrix aufweist, ist ein wesentlicher Qualitätssprung der Hybridisierung bei Verwendung einer SaddleNet basierten Arrhenius Kinetik zu erwarten. Zudem könnte man dann, entsprechend der minimalen Abstände in der Energielandschaft zwischen jedem Minimum und dem entsprechenden Sattel, einen individuellen Skalierungsfaktor pro Rate verwenden.
Dafür müsste jedoch ein teures SaddleNet Sampling erfolgen. Eine Untersuchung zum qualitativen Vorteil einer solchen Hybridkinetik im Vergleich zum dafür nötigen Aufwand wäre erforderlich.

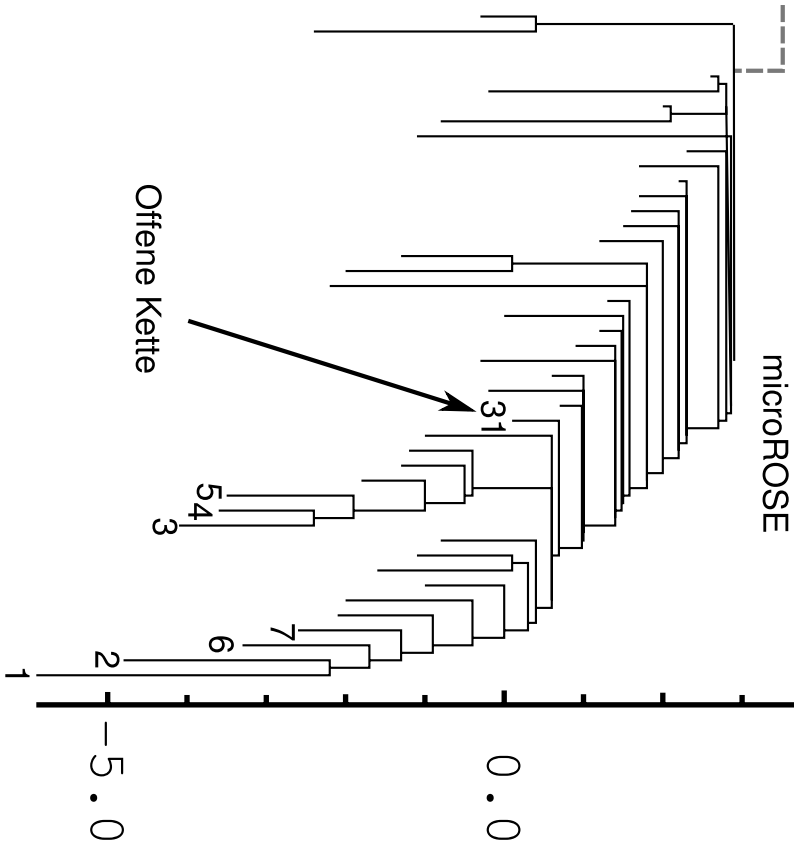
Eventuell könnte auch die Heuristik von Morgan und Higgs [MH98] zur Findung niedriger kürzester Pfade genutzt werden, um das SaddleNet Sampling zu beschleunigen.

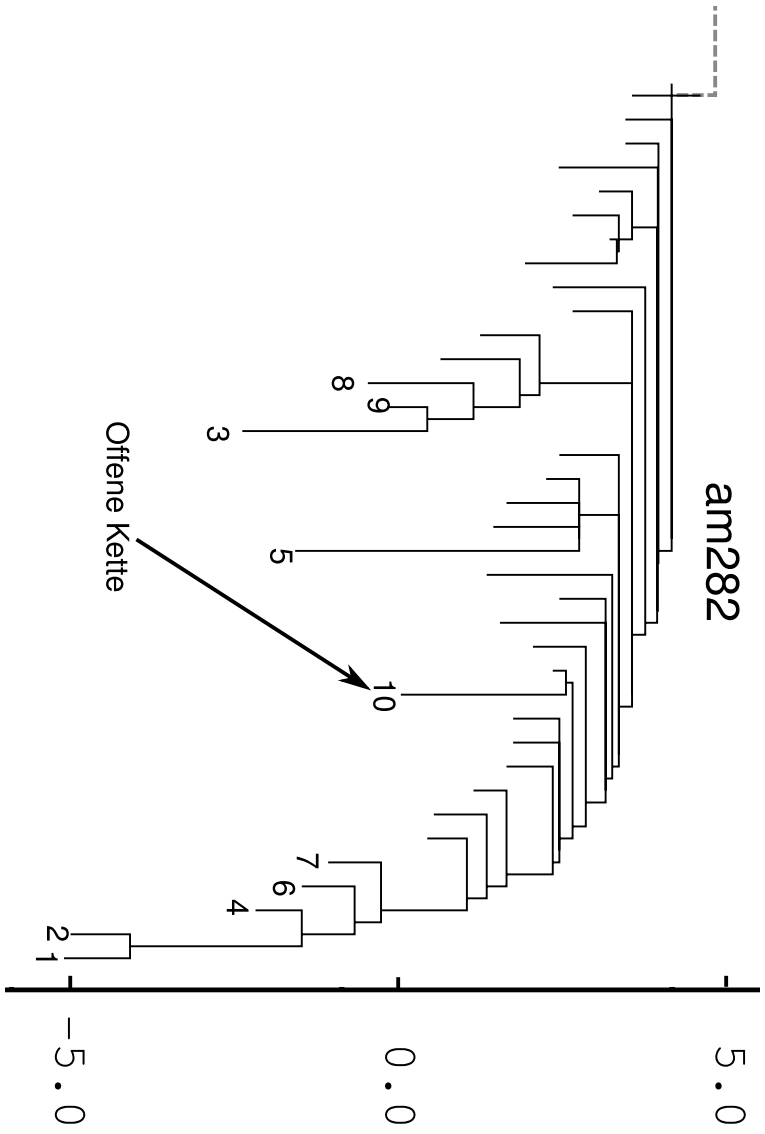
Anhang A

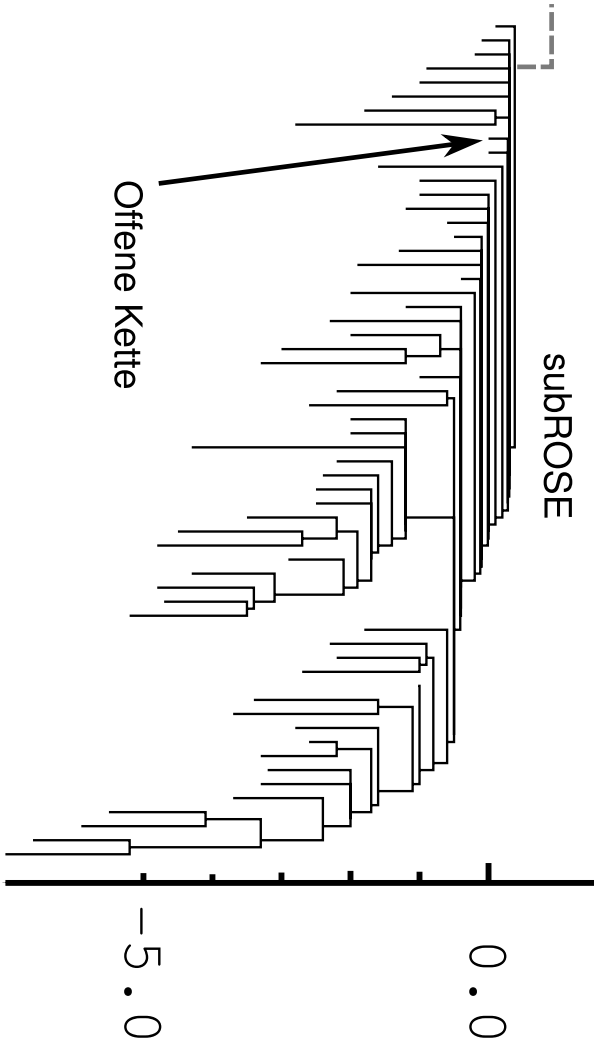
BarrierTrees der Sequenzen




Energie
-4
-2
0
2
4
6
8
seqB
Offene Kette


Offene Kette
microROSE
31
5
4
3
7
6
2
1
−5.0
0.0


am282
Offene Kette
1
2
4
6
7
10
5
3
9
8
−5.0
0.0
5.0


Offene Kette
subROSE
−5.0
0.0

Literaturverzeichnis

[BSR97] Tilman Baumstark, Astrid R. W. Schröder, and Detlev Riesner. Viroid processing: switch from cleavage to ligation is driven by a change from a tetraloop to a loop E conformation. *EMBO Journal*, 16:599–610, 1997.

[CMAN06] Saheli Chowdhury, Christophe Maris, Frederic H-T Allain, and Franz Narberhaus. Molecular basis for temperature sensing by an RNA thermometer. *EMBO J*, 25(11):2487–97, 2006.

[Cou02] Jennifer Couzin. Breakthrough of the year: Small RNAs make big splash. *Science*, 298(5602):2296–2297, 2002.

[FFHS00] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. RNA folding at elementary step resolution. *RNA*, 6(3):325–38, 2000.

[FH08] Christoph Flamm and Ivo Hofacker. Beyond energy minimization: approaches to the kinetic folding of RNA. *Chemical Monthly*, 139:447–457, 2008.

[FHSW02] Christoph Flamm, Ivo L. Hofacker, Peter F. Stadler, and Michael T. Wolfinger. Barrier trees of degenerate landscapes. *Z.Phys.Chem*, 216:155–173, 2002.

[Fla98] Christoph Flamm. *Kinetic Folding of RNA*. Doctor rerum naturalium, University of Vienna, 1998.

[FXM+98] A. Fire, S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver, and C. C. Mello. Potent and specific genetic interference by double-stranded rna in caenorhabditis elegans. *Science*, 391:806–811, 1998.

[Gil86] Walter Gilbert. News and Views: origin of life: The rna world. *Nature*, 319:618–618, 1986.

[HFS+94] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, L. Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of rna secondary structures. *Monatsh.Chem.*, 125:167–188, 1994.

[HS88] Karl Heinz Hoffmann and Paolo Sibani. Diffusion in hierarchies. *Physical Review A*, 38(8):4261–4270, 1988.

[HSS98] I. L. Hofacker, P. Schuster, and P.F. Stadler. Combinatorics of rna secondary structures. *Discr. Appl. Math.*, 88:207–237, 1998.

[JLMZ02] T. Jiang, G. Lin, B. Ma, and K. Zhang. A general edit distance between rna structures. *J. Comput. Biol.*, 9(2):371–388, 2002.

[KFS08] Konstantin Klemm, Christoph Flamm, and Peter F. Stadler. Funnels in energy landscapes. *The European Physical Journal B*, 63(3):387–391, 2008.

[LP00] Rune B. Lyngso and Christian N.S. Pedersen. RNA pseudoknot prediction in energy-based models. *J. Comput. Biol.*, 7(3/4):409–427, 2000.

[McC90] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–19, 1990.

[MH98] Steven R. Morgan and Paul G. Higgs. Barrier heights between ground states in a model of rna secondary structure. *Journal of Physics A: Mathematical and General*, 31(14):3153–3170, 1998.

[ML03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1), 2003.

[MSZT99] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol*, 288(5):911–40, 1999.

[MWB07] Martin Mann, Sebastian Will, and Rolf Backofen. The energy landscape library - a platform for generic algorithms. In *BIRD'07 - 1st international Conference on Bioinformatics Research and Development*, volume 217, pages 83–86. Oesterreichische Computer Gesellschaft, 2007. Poster and Short-Paper.

[NS07] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavy-weight dynamic binary instrumentation. *SIGPLAN Not.*, 42(6):89–100, 2007.

[Ric07] Andreas Richter. Exploration of biopolymer energy landscapes via random sampling. Master's thesis, Friedrich Schiller University Jena, July 2007.

[SBS04] Renée Schroeder, Andrea Barta, and Katharina Semrad. Strategies for RNA folding and assembly. *Nature Reviews Molecular Cell Biology*, 5(11):908–919, 2004.

[SS01] P. Schuster and P. Stadler. Discrete models of biopolymers. In M. J. C. Crabbe, M. Drew, and A. Konopka, editors, *Handbook of Computational Chemistry*. Marcel Dekker, New York, 2001.

[Wat95] Michael S. Waterman. *Introduction to computational biology: maps, sequences and genomes.* Chapman Hall, London, UK, 1995.

[WFHS99] S. Wuchty, W. Fontana, I. L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–65, 1999.

[WFHS04] Michael T. Wolfinger, W. Andreas Svrcek-Seiler Christoph Flamm, Ivo L. Hofacker, and Peter F. Stadler. Exact folding dynamics of RNA secondary structures. *J.Phys.A: Math.Gen.*, 37:4731–4741, 2004.

[Wol01] Michael Thomas Wolfinger. The energy landscape of rna folding. Master's thesis, University of Vienna, March 2001. treekin Algorithm prototype "markov", first RNA kinetics.

[WSSF+04] Michael T. Wolfinger, W. Andreas Svrcek-Seiler, Christoph Flamm, Ivo L. Hofacker, and Peter F. Stadler. Efficient computation of rna folding dynamics. *Journal of Physics A: Mathematical and General*, 37(17):4731–4741, 2004.

[WTK+94] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Muller, D. H. Mathews, and M. Zuker. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *PNAS*, 91(20):9218–22, 1994.

[XSB+98] T. Xia, J. Jr SantaLucia, M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, and D. H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of rna duplexes with watson-crick base pairs. *Biochemistry*, 37(5):14719–14735, 1998.

[ZMT99] Michael Zuker, David H. Mathews, and Douglas H. Turner. Algorithms and thermodynamics for rna secondary structure prediction: A practical guide. In J. Barciszewski and B. F. C. Clark, editors, *RNA Biochemistry and Biotechnology*, pages 11–43. Kluwer Academic Publishers, Dordrecht, NL, 1999.

[Zuk89] M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48–52, 1989.

Abbildungsverzeichnis

Algorithmenverzeichnis

Tabellenverzeichnis

Anhang B

Erklärung

1. Mir ist bekannt, dass die Diplomarbeit als Prüfungsleistung in das Eigentum des Freistaats Thüringen übergeht. Hiermit erkläre ich mein Einverständnis, dass die Universität Jena diese Prüfungsleistung den Studenten der Universität Jena zur Einsicht überlassen darf, und dass sie die Abschlussarbeit unter Nennung meines Namens als Urheber veröffentlichen darf.
2. Ich erkläre hiermit, dass ich diese Diplomarbeit selbständig verfasst, noch nicht anderweitig für andere Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benützt sowie wörtliche und sinngemässe Zitate als solche gekennzeichnet habe.

Jena, 13. August 2008