Bachelorarbeit

RNA-RNA Interaktionsvorhersage für lange Moleküle

Sebastian Holler

Gutachter: Prof. Dr. Rolf Backofen
Betreuer: Dr. Martin Raden

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Lehrstuhl für Bioinformatik

27. August 2018

Bearbeitungszeit

01. 06. 2018 – 27. 08. 2018

Gutachter

Prof. Dr. Rolf Backofen

Betreuer

Dr. Martin Raden

Erklärung

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

Ort, Datum	Unterschrift

Kurzzusammenfassung

Das Programm IntaRNA schafft es dank einer ausgetüftelten Heuristik, RNA-RNA Interaktionen schnell und treffsicher vorherzusagen. Für lange Inputsequenzen steigt dabei allerdings der benötigte Speicherplatz massiv an und erreicht schnell eine kritische Größe.

Zur Bewältigung des Problems wird in dieser Arbeit der Ansatz der Fensterzerlegung für IntaRNA eingeführt. Hierbei werden die Inputsequenzen während der Präprozessierung in kleine Fenster fester Größe aufgeteilt, wodurch der Speicherverbrauch von der Länge des Inputs entkoppelt wird und konstant niedrig bleibt. Dies geschieht auf Kosten der Laufzeit, denn der Ansatz verursacht durch überlappende Bereiche zwischen Fenstern redundanten Rechenaufwand. Darüber hinaus können in jenen Bereichen Vorhersageduplikate auftreten, die gesondert behandelt und aussortiert werden müssen. Die Fensterzerlegung ermöglicht es, die Interaktionsvorhersage für einzelne Fensterpaare zu parallelisieren, wodurch sogar eine klare Verbesserung der Laufzeit im Vergleich zur originalen Implementierung IntaRNAs erreicht werden kann.

Inhaltsverzeichnis

1 Einleitung

Die *Ribonukleinsäure (Ribonucleic Acid, RNA)* ist für pro- und eukaryotische Organismen von eminenter Bedeutung und erfüllt eine Vielzahl essentieller Funktionen in Zellen. In der Variante als *messenger RNA (mRNA)* dient sie als Vorlage für Proteine und spielt somit eine zentrale Rolle bei der Proteinbiosynthese.

Im Kontrast dazu steht eine Vielzahl *nicht-kodierender RNAs (Noncoding RNA, ncRNA)*, welche meist eine regulatorische Funktion ausüben. Die Forschungsaktivitäten zu ihrer Bedeutung und Vielfältigkeit sind dabei noch verhältnismäßig jung - entsprechend gibt es auf diesem Gebiet auch heute noch offene Fragen [1]. Unter den Oberbegriff der ncRNA fällt auch die *small RNA (sRNA)*. Hierbei handelt es sich um etwa 200 *nt* lange, prokaryotische RNA-Moleküle, die in der Regel durch die Bindung an eine Ziel-mRNA genregulatorische Funktionen erfüllen [2]. Wie exemplarisch in Abbildung 1 veranschaulicht ist, kann sRNA die Genexpression durch Aktivierung oder Repression sowohl auf Tranksriptions- als auch auf Translationsebene beeinflussen. Im Jahr 2005 gelang ein Durchbruch auf dem Gebiet der Identifikation von sRNA-Genen, denn mittels neuer Methoden konnten viele jener Gene in *Escherichia Coli* gefunden werden [3].

Für die Erforschung der genauen Wirkungsweise einer bestimmten sRNA ist die Zielsequenz, an welche diese bindet, von größtem Interesse. Für die Vorhersage solcher Zielsequenzen spielen zwei Kernaspekte eine wesentliche Rolle: Auf der einen Seite ist die durch die Hybridisierung gewonnene Energie zwischen sRNA und Ziel von Bedeutung, worüber Rehmsmeier et al. sowie Tjaden et al. bereits Forschungsergebnisse publiziert haben [5, 6]. Auf der anderen Seite sollte die Erreichbarkeit (*Accessibility*) der Zielsequenz Berücksichtigung finden. Einzelsträngige RNA kann intramolekulare Bindungen ausbilden und somit beispielsweise Schleifenformen annehmen. Liegt die Zielsequenz einer sRNA in solch einer Region, wird Energie benötigt, um die Bindungsstelle für eine intermolekulare Bindung zugänglich zu machen. Dieses Phänomen konnte bereits in diversen Untersuchungen im Zusammenhang mit *micro RNA (miRNA)*, dem eukaryotischen Pendant zur sRNA, beobachtet werden, lässt sich aber ebenso auf letztere übertragen [7, 8].

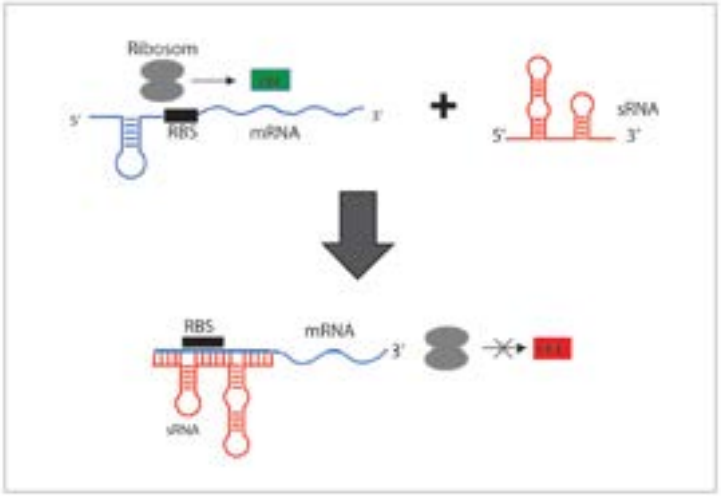

Abbildung 1: Beispiel für die Funktionsweise einer sRNA Im Ausgangszustand ist die Ribosomenbindestelle (RBS) der mRNA frei zugänglich, sodass dort ein Ribosom mit der Translation beginnen kann. Kommt nun eine sRNA hinzu, die an die RBS bindet und diese blockiert, so kann das Ribosom nicht mehr andocken und die Translation wird unterbunden. Abbildung nach [4].

Mit *RNAup* führten Mückstein et al. im Jahr 2006 erstmals einen Ansatz ein, der basierend auf diesen beiden Prinzipien RNA-RNA Interaktionen vorhersagen kann [9]. Da die Berechnung der Interaktionen exakt erfolgt, ist die Software sehr rechenintensiv und benötigt viel Speicher.

2008 entwickelten Busch et al. darauf aufbauend das Programm *IntaRNA*, welches 2017 zu Version 2.0 überarbeitet und erweitert wurde [10, 11]. Durch die Verwendung einer Heuristik zur Berechnung der Interaktion konnten Laufzeit und Speicherbedarf im Vergleich drastisch reduziert werden. Dem Präzisionsverlust der Ergebnisse hat man mittels Verwendung des sogenannten *Seed-Constraints* effektiv entgegengewirkt. Mit dem *Seed* kann die Initialreaktion aufeinandertreffender RNAs modelliert werden [12]. Bildet sich solch eine etwa 5-8 Basenpaare lange Bindung einmal aus, so ist es wahrscheinlicher, dass sich die Bindung weiter zu den Seiten hin ausdehnt, als dass sie sich wieder zurückbildet. Für fast alle sRNA-Bindungen konnten in biologischen Experimenten Seeds gefunden werden [13, 14].

Das Thema dieser Arbeit ist die Implementierung und Evaluation einer zusätzlichen, optionalen Strategie im Rahmen der Präprozessierung in IntaRNA. Bei diesem Ansatz sollen Inputsequenzen vor der Interaktionsvorhersage und dafür benötigten Berechnungen in kleinere, einfacher zu handhabende Teilstücke aufgesplittet werden. Durch

die kürzeren Teilsequenzen benötigen im Programmablauf diverse Matrizen weniger Speicherplatz. Eine angepasste Vorgehensweise wird benötigt, da eine Interaktion im Randbereich einer solchen Teilsequenz liegen könnte. Aus diesem Grund müssen die Teilsequenzen überlappend sein, was sich negativ auf die Laufzeit auswirkt und potentiell Duplikate im Endergebnis verursachen kann, wenn eine Interaktion gerade im Überlappungsbereich lokalisiert ist. Eine Evaluierung dieses Ansatzes soll zeigen, wie sich in der Praxis der verringerte Speicherbedarf im Vergleich zu einer etwas höheren Laufzeit darstellt.

Im kommenden Kapitel werde ich auf formale Grundlagen sowie Details und Hintergründe der Implementierung von IntaRNA eingehen. Daraufhin folgt mit dem Thema RNA-RNA Interaktionsvorhersage der Kernaspekt dieser Arbeit. Nach der genauen Einordnung und Beschreibung des Problems werde ich schrittweise die praktische Umsetzung erläutern. Abschließend erfolgt eine Evaluation der angepassten Implementierung im Vergleich zur originalen.

2 Hintergründe/IntaRNA

Die Abschnitte 2.1 und 2.3 orientieren sich vor allem am Paper "Interactive implementations of thermodynamics-based RNA structure and RNA-RNA interaction prediction approaches for example-driven teaching" von Martin Raden et al. sowie an der Zusammenfassung der theoretischen Grundlagen und der Rekursion IntaRNAs, die Rick Gelhausen für seine Masterarbeit "Constrained RNA-RNA interaction prediction" verfasst hat [15, 16].

2.1 Formale Grundlagen

Ein RNA-Molekül ist eine Aneinanderkettung von n Nukleotiden. Seine *primäre Struktur* kann als Sequenz $S \in \{A, C, G, U\}^n$ beschrieben werden, wobei die Buchstaben A, C, G und U die Basen Adenin, Cytosin, Guanin und Uracil repräsentieren.

Zwischen zwei Nukleotiden können diese Basen Wasserstoffbrücken ausbilden, die sogenannten *Basenpaare*. Dabei sind diverse Kombinationen von Basen möglich, wobei man aufgrund ihrer hohen Bindungsstärke in der Regel nur die Watson-Crick Basenpaare $A-U$ und $C-G$ sowie das Wobble Basenpaar $G-U$ betrachtet, welche *komplementär* sind.

Solche Basenpaare können sich intramolekular zwischen Nukleotiden desselben RNA-Moleküls bilden. Diese Bindungen bestimmen die dreidimensionale Struktur der RNA, die insbesondere bei ncRNAs essentiell für die Ausübung der Funktion ist. Beispielsweise kann die spezifische Form es einer ncRNA ermöglichen, eine Ziel-RNA durch RNA-RNA Interaktion, der Ausbildung intermolekularer Basenpaare zwischen beiden Molekülen, zu regulieren.

Unter einer *sekundären Struktur* D einer RNA-Sequenz S versteht man eine Menge von Basenpaaren:

$$D \subseteq \{(i,j) \mid 1 \leq i < j \leq n \wedge S_i \text{ und } S_j \text{ sind komplementär}\}, \tag{1}$$

wobei $n = |S|$ und für alle $(i,j), (i',j') \in D$ gilt:

$$(i = i' \Leftrightarrow j = j') \wedge i \neq j' \tag{2}$$

Von einer nicht-kreuzenden, sekundären Struktur spricht man, wenn darüber hinaus für alle $(i,j),(k,l) \in D$ mit $(i,j) \neq (k,l)$ gilt:

$$(i < k < l < j) \ \vee \ (k < i < j < l) \tag{3}$$

Die *freie Energie* entspricht der Energiemenge, die in der sekundären Struktur eines RNA-Moleküls gespeichert ist. Ein negativer Wert steht dabei für die Energie, die benötigt wird, um alle intramolekularen Basenpaare der RNA aufzutrennen, während bei einem positiven Wert sogar Energie freigesetzt wird. Folglich ist eine RNA-Struktur stabiler, je kleiner die freie Energie ausfällt. Generell handelt es sich bei der stabilsten Struktur in aller Regel auch um die Form, in der das RNA-Molekül seine Funktion ausübt.

Mithilfe des *Nearest Neighbor* Energiemodells lässt sich die freie Energie von nicht-kreuzenden RNA-Strukturen approximieren [17]. Das Modell beruht auf einer Schleifen-basierten Dekomposition letzterer in einzelne, strukturelle Elemente: *hairpin loops*, *stacks*, *bulges*, *internal loops* und *multi-loops*. Somit kann die Energie $E(D)$ einer nicht-kreuzenden, sekundären Struktur D unter Berücksichtigung aller Einzelteile der Dekomposition abgeschätzt werden:

$$E(D) := \sum_{(i,j) \in D} \begin{cases} e^H(i,j) & \text{falls hairpin loop vorliegt} \\ e^{SBI}(i,j,k,l) & \text{falls stack, bulge oder internal loop vorliegt} \\ e^M(i,j,x,x') & \text{falls multi-loop vorliegt} \end{cases} \tag{4}$$

Dabei liefern e^H, e^{SBI} und e^M den Energiebeitrag des entsprechenden Strukturelements, der von Umwelteinflüssen wie der Temperatur abhängig ist. Bei e^{SBI} repräsentiert (k,l) das umschlossene Basenpaar. Weiterhin stehen bei e^M x und x' für die Anzahl ungepaarter Basen beziehungsweise für die Anzahl umschlossener Helices. e^{SBI} ist gegeben durch:

$$e^{SBI} := \begin{cases} e^S(i,j,i+1,j-1) & \text{falls stack vorliegt} \\ e^B(i,j,i+1,j') \vee e^B(i,j,i',j-1) & \text{falls bulge vorliegt} \\ e^I(i,j,i',j') & \text{falls internal loop vorliegt} \end{cases} \tag{5}$$

Heutzutage steht eine Vielzahl an Sätzen solcher Energieparameter zur Verfügung. In IntaRNA kommen die Turner-Parameter zum Einsatz, die in der "Nearest Neighbor Data Base" (NNDB) eingesehen werden können [18, 19].

Die Accessibility einer bestimmten Subsequenz entspricht der benötigten Energie, um diese in eine einzelsträngige Form zu überführen. Beispielsweise müssen hierfür intramolekulare Bindungen, in welche diese Subsequenz involviert ist, gelöst werden. Zur Berechnung der Accessibility wird die Differenz zweier Energien herangezogen: Zum einen die Energie der Strukturensembles, das heißt aller Strukturen $\mathcal{P}$, welche S natürlicherweise ausbildet, zum anderen dieselbe Energie der Kombination $\mathcal{P}^u$ mit dem Unterschied, dass hier die Interaktionssubsequenz i bis k einzelsträngig bleibt.

Die freie Energie $E^{ens}(\mathcal{P})$ eines Strukturensembles $\mathcal{P}$ ist definiert als:

$$E^{ens}(\mathcal{P}) := -RT \cdot ln(Z_{\mathcal{P}}) \text{ mit } Z_{\mathcal{P}} := \sum_{Q \in \mathcal{P}} e^{-\frac{E(Q)}{RT}} \tag{6}$$

Hierbei ist $Z_{\mathcal{P}}$ die Partitionsfunktion von $\mathcal{P}$ nach einem Ansatz von McCaskill, $E(Q)$ die freie Energie einer Sequenz S, die in eine sekundäre Struktur Q gefaltet ist, R die universelle Gaskonstante und T die Temperatur [20].

Damit ergibt sich die Energiedifferenz $ED(i,k)$ wie folgt:

$$ED(i,k) := E^{ens}(\mathcal{P}^u_{i,k}) - E^{ens}(\mathcal{P}) \tag{7}$$

2.2 Übersicht über den Programmablauf IntaRNAs

Im Folgenden wird der Programmablauf von IntaRNA thematisiert. Zur Veranschaulichung dient Algorithmus 1.

Bei der Ausführung IntaRNAs werden zunächst die Kommandozeilenparameter eingelesen. Darunter zählen neben Query- und Target-RNA-Sequenzen auch diverse zusätzliche Einstellungsoptionen wie die Angabe des optionalen Seeds oder die Auswahl der Anzahl zu verwendender Threads. Dann erfolgt in den Zeilen 3 und 5 für jeweils alle Query- und Targetsequenzen die Berechnung der Accessibility Werte ED wie in Kapitel 2.1 beschrieben.

Im Anschluss wird in Zeile 6 für jede Kombination aus Query- und Targetsequenzen die Interaktionsenergiefunktion E definiert, wobei dafür die jeweiligen Accessibility Werte herangezogen werden. Die Berechnung basiert auf dem Energiemodell, welches in *RNAHybrid* Verwendung findet [5]. Vor der eigentlichen Interaktionsvorhersage werden die beiden Sequenzen in den Zeilen 7 und 8 optional in Teilstücke, sogenannte

Algorithmus 1 Schematischer Programmablauf von IntaRNA in Pseudocode

```
 1: P ← parseParameters()
 2: foreach target sequence T in P do
 3:     AccT ← computeAccessibility(T)
 4:     foreach query sequence Q in P do
 5:         AccQ ← computeAccessibility(Q)
 6:         E ← getEnergyFunction(AccQ, AccT)
 7:         foreach range t in T do
 8:             foreach range q in Q do
 9:                 I_qt ← predictInteraction(E, q, t)
10:                 updateBestInteractions(I, I_qt)
11:             end for
12:         end for
13:     end for
14: end for
15: return I
```

Ranges, partitioniert. Per Default entspricht die Range dabei der kompletten Länge der Sequenz, allerdings kann der Benutzer mittels der Kommandozeilenparameter auch festlegen, dass nur bestimmte, nicht überlappende Teilstücke der Sequenz für die Vorhersage herangezogen werden sollen.

Schließlich wird in Zeile 9 für alle Kombinationen von Teilstücken standardmäßig mittels einer Heuristik und unter Berücksichtigung der Interaktionsenergie die RNA-RNA Interaktionsvorhersage ausgeführt. In einer Liste werden die k vielversprechendsten bzw. energetisch optimalsten Interaktionen für alle Kombinationen aus Query- und Targetsequenz gesammelt und schlussendlich ausgegeben.

2.3 IntaRNAs Rekursion

2.3.1 Exakte Berechnung

Wie bereits einleitend erwähnt, basiert die Interaktionsvorhersage IntaRNAs vor allem auf zwei Kernaspekten: der Hybridisierungsenergie $H(i, j, k, l)$ und der Erreichbarkeit der an der Interaktion beteiligten Stellen der Sequenzen S^1 von Index i bis k und S^2 von j bis l.

Hierbei wird die Hybridisierungsenergie mittels des Nearest Neighbor Energiemodells berechnet, in welchem die minimale, freie Energie zweier paarender Subsequenzen, deren Basen am linken Ende ein komplementäres Basenpaar bilden, repräsentiert wird. Für zwei Subsequenzen $S_i^1...S_k^1$ und $S_j^2...S_l^2$ ergibt sich:

$$H(i,j,k,l) := \min\{E(P) \mid (i,j) \in P \wedge (k,l) \in P\} \tag{8}$$

Nun kann die Hybridisierungsenergie mit einer Rekursion berechnet werden, die der aus Zukers Algorithmus ähnelt [21]:

$$H(i,j,k,l) := \min \begin{cases} E_{init} \\ \qquad (S_i^1, S_j^2) \text{ können paaren} \wedge i = k \wedge j = l \\ \min\limits_{r,s}\{e^{SBI}(i,j,r,s) + H(r,s,k,l)\} \\ \qquad (S_i^1, S_j^2) \text{ sowie } (S_r^1, S_s^2) \text{ können paaren} \wedge i \neq k \wedge j \neq l \\ \infty \\ \qquad \text{sonst} \end{cases} \tag{9}$$

Dabei entspricht e^{SBI} der Energie, die durch Interaktionen der beiden Sequenzen in Form von stacks, bulges und internal loops zustande kommt.

Schließlich erhält man die erweiterte Hybridisierungsenergie $C(i,j,k,l)$ als Kombination aus der Accessibility und der einfachen Hybridisierungsenergie:

$$C(i,j,k,l) := \begin{cases} H(i,j,k,l) + ED_1(i,k) + ED_2(j,l) \\ \qquad (S_i^1, S_j^2) \text{ sowie } (S_k^1, S_l^2) \text{ können paaren} \wedge i \neq k \wedge j \neq l \\ \infty \\ \qquad \text{sonst} \end{cases} \tag{10}$$

Alle dafür benötigten Energiebeiträge sind in Abbildung 2 skizziert.

Darüber hinaus wird in IntaRNA optional ein vom Benutzer übergebener Seed berücksichtigt. Da dieser jedoch nicht für den eigentlichen Inhalt dieser Arbeit relevant ist, wird an dieser Stelle nicht näher darauf eingegangen.

2.3.2 Heuristik

Die exakte Berechnung eignet sich aufgrund ihrer Laufzeit und ihres Speicherverbrauchs in der Praxis nicht für größere, genomweite Untersuchungen. Beschränkt man die Länge berücksichtigter intermolekularer Schleifen, so liegen sowohl Laufzeit- wie auch Speicherkomplexität in $O(n^2m^2)$, wobei n der Länge der Query- und m

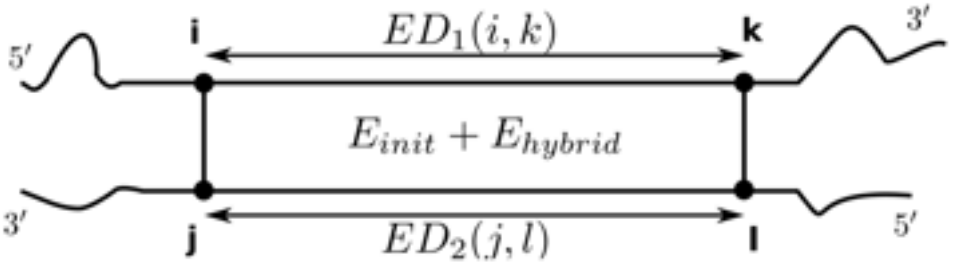

Abbildung 2: Für die Rekursion benötigte Energiewerte Neben der intermolekularen Initialisierungsenergie E_{init} und der Hybridisierungsenergie E_{hybrid} wird für die rekursive Berechnung der erweiterten Hybridisierungsenergie die Accessibility Energiedifferenz ED benötigt. Abbildung nach [16].

der Länge der Targetsequenz entspricht. Um dieses Problem des Zeit- und Speicherverbrauchs zu bewältigen, wurde in IntaRNA eine Heuristik eingeführt, die anstelle der exakten Rekursion benutzt werden kann. Dabei ist die Kernidee, dass für ein gegebenes, linkes Interaktionsende i, j nur eine einzige, lokal optimale Subinteraktion für längere Interaktionen in Betracht gezogen wird. Da bei der exakten Berechnung noch $O(nm)$ viele getestet wurden, profitieren Speicherkomplexität sowie Laufzeit der Heuristik enorm von den Einsparungen und sinken jeweils auf $O(nm)$.

Die heuristische Variante der Matrix $C(i,j,k,l)$ beziehungsweise $C(i,j)$ ist definiert durch:

$$C(i,j) := \begin{cases} E_{init} + ED_1(i,i) + ED_2(j,j) \\ \qquad \text{im Falle einer neuen Interaktion} \\ \min\limits_{p,q}(e^{SBI}(i,j,p,q) + C(p,q) - ED_1(p,K(p,q)) - ED_2(q,L(p,q)) \\ \qquad\qquad +ED_1(i,K(p,q)) + ED_2(j,L(p,q))) \\ \qquad S_i^1 \text{ und } S_j^2 \text{ können paaren} \\ \infty \\ \qquad \text{kein paaren möglich} \end{cases} \tag{11}$$

Hierbei sind $K(p,q)$ und $L(p,q)$ neu eingeführte Matrizen, die die optimalen, rechten Enden k und l für die durch $C(p,q)$ repräsentierte Interaktion liefern. Im Fall, dass S_i^1 und S_j^2 paaren können, werden zunächst die EDs der alten, kürzeren Interaktion abgezogen, bevor dann die EDs der neuen, längeren Interaktion hinzuaddiert werden.

3 RNA-RNA Interaktionsvorhersage für lange Moleküle

3.1 Problemdefinition

Gerade für lange Query- und Targetsequenzen erfordern die zur RNA-RNA Interaktionsvorhersage benötigten Matrizen in IntaRNA viel Speicherplatz - wie in Kapitel 2.3.2 gesehen, beläuft sich dieser auf $O(nm)$, wobei n und m die Sequenzlängen sind. Zerlegt man die Sequenzen vor der Vorhersage in kleinere, einfacher zu handhabende Teilstücke, so verringert sich der Speicherbedarf deutlich, da auch die Matrizen wesentlich kleiner ausfallen. Der Speicherverbrauch lässt sich mittels dieser Vorgehensweise also steuern.

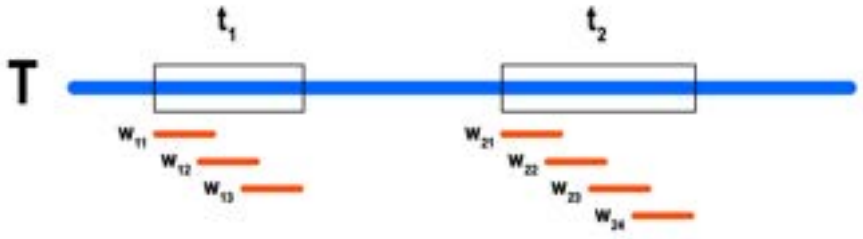

Abbildung 3: Schematische Darstellung der Aufteilung einer Sequenz in Fenster Die benutzerdefinierten Ranges t_1 und t_2 einer Sequenz T sollen automatisch in gleichgroße, überlappende Fenster w_{ij} zerlegt werden.

Das Kernproblem dieser Arbeit ist somit die automatisierte Zerstückelung der Eingabesequenzen in Fenster (*Windows*) bestimmter Größe. Dieser Schritt gliedert sich in die Präprozessierung noch vor der späteren Interaktionsvorhersage ein. In Abbildung 3 ist die Vorgehensweise der Zerlegung skizziert. Für eine Sequenz T werden zunächst durch den Nutzer definierte, nicht überlappende Ranges, hier t_1 und

t_2, festgelegt. Diese sollen dann automatisch in kleine Fenster w_{ij} einer bestimmten Länge zerteilt werden.

Da eine potentielle Interaktion auch im Randbereich zwischen zwei Fenstern stattfinden könnte, müssen die Teilstücke überlappend sein. Dabei muss die überlappende Region so gewählt werden, dass sie mindestens so groß wie die maximale Interaktionslänge ist. Somit wird gewährleistet, dass auch im Grenzbereich alle Interaktionen vorhergesagt werden können. Durch die Überlappung fällt redundanter Rechenaufwand an, der sich im Vergleich zum regulären Programmablauf von IntaRNA negativ in der Laufzeit manifestiert.

3.2 Integration der Fensterzerlegung

Die Umsetzung der Fensterzerlegung gliedert sich im Ablauf IntaRNAs nach Zeile 8 aus Algorithmus 1 ein. Hier greift eine neue, zusätzliche Subroutine ein und verarbeitet die bereits bekannten Ranges t und q.

Dabei zerlegt die Subroutine eine Range in Fenster entsprechend der Darstellung in Abbildung 3. Über die Kommandozeilenparameter können die Fensterlänge sowie die Überlappungsgröße festgelegt werden. Die Anzahl der Fenster $\#_{windows}$ für eine Range der Länge n unter Verwendung der Fensterbreite w und der Überlappung o ergibt sich wie folgt:

$$\#_{windows} := \lceil \frac{n-o}{w-o} \rceil \tag{12}$$

Weiterhin muss $n > o$ und $w > o$ gelten. Schließlich wird der Hauptroutine ein Vektor, bestehend aus allen Fenstern der Range, zurückgegeben.

In der Hauptroutine wird daraufhin mit zwei zusätzlichen For-Schleifen über diese Fenstervektoren von Query- und Targetrange iteriert. Anschließend werden für alle Kombinationen von Query- und Targetfenstern Zeile 9 und 10 aus dem Pseudocode ausgeführt, also die Interaktionsvorhersage vorgenommen. Weitere Änderungen am IntaRNA Algorithmus sind für diese erste Umsetzung der Fensterzerlegung nicht notwendig.

3.3 Behandlung von Vorhersageduplikaten

Befindet sich eine optimale Interaktion vollständig im überlappenden Bereich zweier Fenster w_1 und w_2 einer Sequenz S, so führt dies bei der Version IntaRNAs mit Fensterzerlegung zu einem verfälschten Endergebnis. Wie Abbildung 4 veranschaulicht,

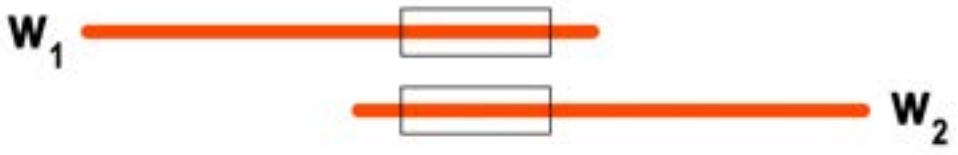

Abbildung 4: Schematische Darstellung der Entstehung eines Duplikats im Vorhersageergebnis Eine potentielle Interaktion ist vollständig im überlappenden Bereich zweier Fenster w_1 und w_2 einer Sequenz S lokalisiert. Ohne weitere Änderungen an IntaRNA würde die Interaktion von beiden Fenstern jeweils einmal vorhergesagt werden, sodass ein Duplikat im Ergebnis zustande käme.

wird die Interaktion in diesem Fall im Rahmen der Vorhersage beider Fenster jeweils einmal gefunden, sodass sie doppelt im kombinierten Endresultat vorkommt.

Abhilfe schafft hier eine Modifikation an der Datenstruktur, die jederzeit eine Liste der n besten Ergebnisse von Interaktionsvorhersagen speichert und verwaltet. Bis dato wurden neue Interaktionen nur dann in die Liste aufgenommen, wenn diese noch weniger als n Elemente enthielt oder die neue Interaktion echt kleiner als das letzte Element der Liste war. Diese Lösung funktioniert fehlerfrei, solange keine Duplikate vorkommen. Um nun auch redundant vorhergesagte Interaktionen auszuschließen und gar nicht erst zur Liste hinzuzufügen, wird ein weiterer Check benötigt. In diesem wird sichergestellt, dass die neue Interaktion nicht gleich derjenigen ist, vor der sie eingefügt werden würde. Durch diese Änderung spiegeln sich Vorhersageduplikate nicht mehr im Endergebnis wider.

3.4 Verbesserung der Laufzeit durch Parallelisierung

In der Originalversion IntaRNAs erfolgt eine mittels der OpenMP API implementierte Parallelisierung nur, sofern dies über einen Parameter aktiviert wird und dem Programm mehrere Query- oder Targetsequenzen auf einmal übergeben werden [22]. Erhält sie jeweils nur eine dieser Sequenzen, greift die Parallelisierung nicht und bietet somit keinen Laufzeitgewinn. Der Ansatz der Fensterzerlegung eröffnet hier eine neue Möglichkeit zur parallelen Berechnung und damit zur Bekämpfung dessen

Laufzeitdefizits. Bei der Abarbeitung einer Kombination aus Query und Target kann die Interaktionsvorhersage jedes Fensterpaares in einem separaten Thread berechnet werden. Nur die Aktualisierung der Liste der besten Ergebnisse muss hierbei gesondert behandelt werden, da diese immer nur von einem Thread gleichzeitig durchgeführt werden darf.

4 Ergebnisse und Diskussion

4.1 Evaluation einer ersten Umsetzung der Zerlegung

Für eine erste Evaluation der soweit implementierten Fensterzerlegung wurde IntaRNA unter Beobachtung der Laufzeit und des Speicherverbrauchs auf zufällig generierten Query- und Targetsequenzen ausgeführt. Getestet wurden diverse Kombinationen verschieden langer Sequenzen sowie unterschiedliche Fenstergrößen, wobei die Fensterüberlappung, wenn nicht anders angegeben, immer 100 *nt* lang war. Die Accessibility wurde für diese Versuche nicht berücksichtigt. Zum Vergleich wurden dieselben Szenarien mit der originalen Umsetzung IntaRNAs ohne Fensterzerlegung durchgerechnet.

4.1.1 Speicher

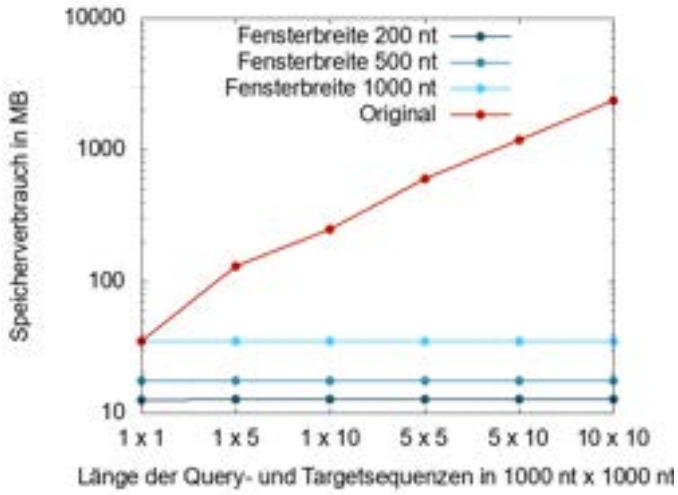

Abbildung 5: Maximaler Speicherverbrauch verschiedener Varianten von IntaRNA Die unveränderte Implementierung von IntaRNA wird hinsichtlich der maximalen Speichernutzung mit angepassten Versionen verglichen, welche die Fensterzerlegung mit unterschiedlichen Fenstergrößen durchführen.

Die Ergebnisse der Messung des Speicherverbrauchs sind in Abbildung 5 dargestellt.

Besonders auffällig ist die Tatsache, dass die Resultate der Versionen mit Fensterzerlegung ungeachtet der Längen der Inputsequenzen lediglich einen konstanten, geringen Speicherbedarf aufweisen. Dies war zu erwarten, da die Zerlegung bewirkt, dass bei der Berechnung viele kleine Matrizen nacheinander anstelle einer einzigen großen zum Einsatz kommen. Diese kleinen Matrizen sind in ihrer Größe nach oben durch die festgelegte Fensterbreite beschränkt. Dahingegen kommen die Nachteile der unveränderten Version IntaRNAs mit zunehmender Länge der Query- und Targetsequenzen immer deutlicher zum Vorschein, denn sie benötigt bedeutend mehr Speicher. Für die Kombination aus 10.000 $nt \times 10.000$ nt liegt der Speicherverbrauch bei über 2.3 GB, wohingegen selbst die ungünstigste Variante der Fensterzerlegung mit einer Fensterbreite von 1000 nt nur auf etwa 35 MB kommt. Bei letzterer Variante greift die Fensterzerlegung für den 1000 $nt \times 1000$ nt Input aufgrund der ebenso großen Fensterlänge noch nicht, weshalb der Speicherverbrauch identisch zur originalen Implementierung IntaRNAs ist. Abgesehen von diesem Sonderfall sind die Versionen mit Fensterzerlegung der ohne im Aspekt Speicherbedarf in jeder Hinsicht überlegen.

4.1.2 Laufzeit

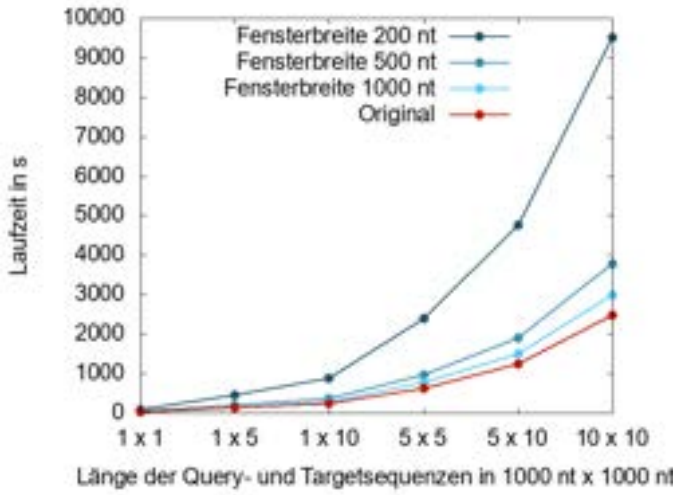

Abbildung 6: Laufzeit verschiedener Varianten von IntaRNA Die unveränderte Implementierung von IntaRNA wird hinsichtlich der Laufzeit mit angepassten Versionen verglichen, welche die Fensterzerlegung mit unterschiedlichen Fenstergrößen durchführen.

Die Resultate der Laufzeitmessung sind in Abbildung 6 veranschaulicht. Wie zu erwarten war, schneidet die originale Implementierung am besten ab, da die Fens-

terzerlegung in jedem Fall einen Overhead für die Laufzeit verursacht. Weiterhin zeigt sich, dass die Laufzeit der neuen Versionen antiproportional zur Fensterbreite zunimmt, was sich besonders deutlich bei der Breite von 200 *nt* widerspiegelt. Je mehr Fenster zur Abarbeitung einer Sequenz benötigt werden, desto höher wird der redundante, rechnerische Aufwand durch überlappende Bereiche, für welche die Interaktionsvorhersage zweimal ausgeführt wird. Diese Problematik betrifft insbesondere die Fensterbreite von 200 *nt*, da jene im Zusammenspiel mit der für dieses Experiment gewählten Überlappungslänge von 100 *nt* dafür sorgt, dass fast die kompletten Query- und Targetsequenzen doppelt durchgerechnet werden. Jenseits dieses Negativbeispiels bleibt der Zugewinn an Laufzeit gegenüber der originalen Version für die Fensterbreiten von 500 *nt* beziehungsweise 1000 *nt* mit etwa 53% respektive 20% überschaubar.

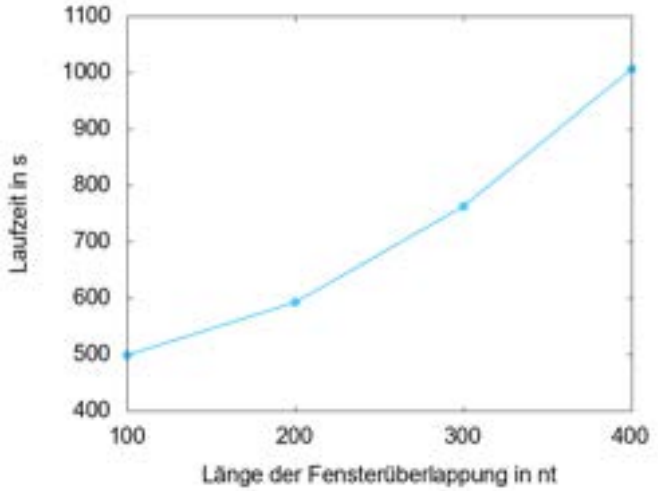

Abbildung 7: Laufzeit IntaRNAs mit Fensterzerlegung in Abhängigkeit der Überlappungsgröße Für einen Input der Größe 5000 *nt* × 5000 *nt* wurde die Laufzeit IntaRNAs bei einer Fensterbreite von 1000 *nt* für unterschiedliche Überlappungsgrößen gemessen. Die Accessibility wurde nicht berücksichtigt.

Der Anstieg der Laufzeit ist nicht nur auf die doppelte Vorhersage überlappender Bereiche zurückzuführen, sondern auch auf einen erhöhten Speicherverwaltungsaufwand. Bei der Fensterzerlegung muss viel mehr Speicher alloziert und wieder freigegeben werden, da Datenstrukturen für die Abarbeitung eines Fensters angelegt und danach wieder gelöscht werden, was das Programm ebenfalls ausbremst. Die in Abbildung 7 dargestellten Messwerte untermauern allerdings, dass vor allem die Länge der Überlappung für die Verschlechterung der Laufzeit verantwortlich ist. Die Vervierfachung

der Überlappungsgröße von 100 auf 400 *nt* schlägt sich im Ergebnis mit ziemlich genau der doppelten Laufzeit nieder.

4.1.3 Vorhersagequalität

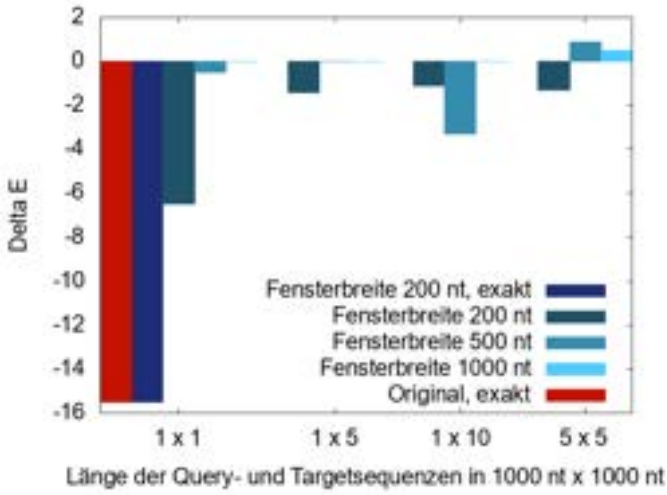

Abbildung 8: Delta der Energieniveaus der besten Vorhersage IntaRNAs mit Fensterzerlegung und der originalen Implementierung
Die beste, heuristisch vorhergesagte Interaktion einer unveränderten Implementierung von IntaRNA wird bezüglich des Energieniveaus mit dem jeweiligen Topergebnis angepasster Versionen verglichen, welche die Fensterzerlegung mit unterschiedlichen Fenstergrößen durchführen. Des Weiteren erfolgt dieser Vergleich auch für exakte Vorhersagen des originalen Algorithmus sowie der Fenstervariante für die Inputgröße 1000 $nt \times 1000$ nt.

Darüber hinaus fällt auf, dass Vorhersageergebnisse der Fenstervarianten und der unveränderten Version IntaRNAs bei selbem Input nicht immer übereinstimmen. In Abbildung 8 ist zur Veranschaulichung das Delta der Energieniveaus dargestellt - die Energie der besten, heuristischen Vorhersage der originalen Implementierung IntaRNAs wird vom besten Resultat verschiedener, anderer Varianten subtrahiert. Dabei wurde für die kleinste Inputgröße auch die exakte, optimale Vorhersage IntaRNAs evaluiert, welche um etwa 25% besser als das Ergebnis der Heuristik abschneidet. Die Variante mit Fensterzerlegung hat im exakten Vorhersagemodus bei einer Fensterbreite von 200 *nt* ebenfalls dieselbe, optimale Interaktion gefunden. Außerdem ist das Ergebnis der heuristischen Vorhersage mit Fensterbreite 200 *nt* stets etwas besser als das originale und näher am exakten, was darauf schließen lässt, dass sich kleine Fenster

positiv auf die Qualität der Ergebnisse auswirken. Für größere Fensterbreiten lässt sich kein klarer Trend ausmachen, teils wird sogar nur ein minimal schlechteres Resultat gefunden. Die beobachteten Abweichungen rühren von der in Kapitel 2.3.2 erläuterten Vorhersageheuristik IntaRNAs her. Ausgehend von einem bestimmten Startpunkt wird nur die beste Interaktion nach rechts mit dem Ende (k, l) berücksichtigt. Liegt solch eine potentielle Interaktion in der Nähe einer Schnittstelle zweier Fenster, so besteht die Möglichkeit, dass die Fensterversion von der Schnittstelle bezüglich der Wahl möglicher k beziehungsweise l beschränkt wird, sodass andere Resultate als bei der originalen Implementierung herauskommen.

Führt man jene Tests erneut unter Berücksichtigung der Accessibility durch, verschlechtert sich das Energieniveau vieler potentieller Interaktionen, da die Interaktionslänge im Mittel wesentlich kürzer ausfällt. Im Experiment war die Topvorhersage für alle Fensterbreiten gleich der originalen. Erst auf weiteren Positionen der Top 10 wurden gerade für kleine Fensterbreiten ab und zu Interaktionen gefunden, die die unveränderte Implementierung nicht aufgelistet hat. Mit der Einbindung der Accessibility treten solche neuen Funde im Allgemeinen deutlich seltener auf.

4.2 Evaluation der Fensterzerlegung mit Parallelisierung

Im Folgenden werden die Auswirkungen der parallelisierten Fensterbearbeitung auf die Laufzeit und den Speicherverbrauch evaluiert. Hierbei wird das originale IntaRNA mit Fensterversionen verglichen, die ihre Berechnungen mit verschieden vielen Threads durchführen. Als Fenstergröße werden 500 nt verwendet und die Fensterüberlappung entspricht 200 nt. Sonstige Parameter sind wie zu Beginn von Kapitel 4.1 gewählt und werden konstant für alle Testdurchläufe beibehalten.

4.2.1 Speicher

In Abbildung 9 sind die Messergebnisse des Speicherverbrauchs dargestellt. Wie bereits in Abschnitt 4.1 beobachtet werden konnte, benötigt die unveränderte Implementierung IntaRNAs am meisten Speicherplatz, insbesondere für wachsende Inputgrößen steigt der Bedarf. Wird die Fensterzerlegung angewandt, so bleibt der maximale Speicherbedarf ungeachtet der Größe des Inputs konstant. Während IntaRNA mit einem Thread etwa 16 MB benötigt, sind es bei zwei Threads bereits 22 MB und bei vier etwa 34 MB. Dies lässt sich vor allem darauf zurückführen, dass bei der parallelen Verarbeitung mehrere Matrizen gleichzeitig berechnet werden, deren Speicherkonsum sich aufsummiert. Darüber hinaus erzeugt Multithreading einen gewissen Overhead,

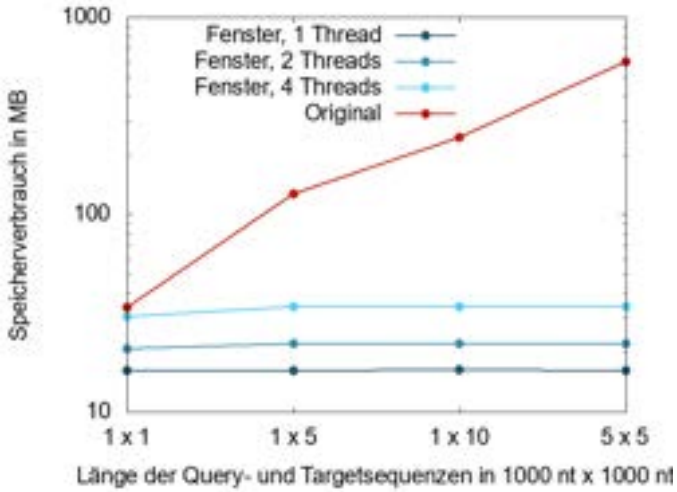

Abbildung 9: Maximaler Speicherverbrauch verschiedener Varianten von IntaRNA Die unveränderte Implementierung IntaRNAs wird hinsichtlich der maximalen Speichernutzung mit einer Version mit Fensterzerlegung verglichen, welche mit einer unterschiedlichen Anzahl von Threads läuft.

der sich auch im Speicherverbrauch widerspiegelt. Somit führt die Parallelisierung zu einem Mehrverbrauch an Speicher, der allerdings in einem steuerbaren Rahmen bleibt und damit immer noch deutlich besser als der Speicherbedarf des originalen IntaRNAs abschneidet.

4.2.2 Laufzeit

Weiterhin werden in Abbildung 10 die Ergebnisse der Laufzeitmessung veranschaulicht. Auch hier wurde in Kapitel 4.1 bereits festgehalten, dass grundsätzlich die Variante mit Fensterzerlegung langsamer ist als die originale Implementierung. Allerdings kann die Laufzeit ersterer durch die Verdopplung der benutzten Threads auf zwei beziehungsweise vier Stück jeweils annähernd halbiert werden. Dass dabei die Verbesserung nicht ganz 50% entspricht, lässt sich darauf zurückführen, dass das Multithreading wie bereits erwähnt einen Overhead mit sich bringt und dass nur ein Thread gleichzeitig die Liste bester Interaktionen aktualisieren kann, sodass sich dort ein kleiner Bottleneck bildet.

Die Verwendung von vier Threads sorgt dafür, dass die Implementierung IntaRNAs mit Fenstern sogar um einen Anteil von etwa einem Drittel schneller läuft als die unveränderte Version. Damit rentiert sich die Verwendung der Fensterzerlegung für

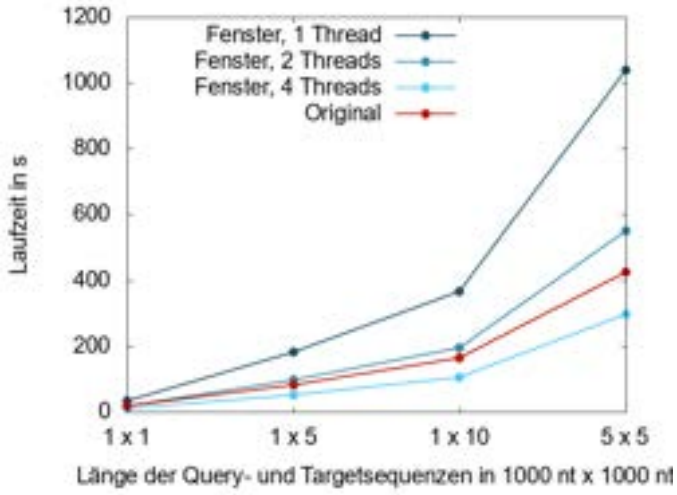

Abbildung 10: Laufzeit verschiedener Varianten von IntaRNA Die unveränderte Implementierung von IntaRNA wird hinsichtlich der Laufzeit mit angepassten Versionen verglichen, welche die Fensterzerlegung mit einer unterschiedlichen Anzahl von Threads durchführen.

solche Inputsequenzen nicht nur, um den Speicherverbrauch konstant klein zu halten, sondern auch, um Laufzeit einzusparen.

5 Zusammenfassung

IntaRNA vermag es aufgrund einer ausgeklügelten Heuristik, RNA-RNA Interaktionen effizient und schnell vorherzusagen. Gerade für sehr lange Query- und Targetsequenzen stellt der Speicherverbrauch allerdings ein Problem dar, da er durch benötigte, riesige Matrizen stark ansteigt und dabei Größen im Gigabyte-Bereich und mehr erreichen kann. Der in dieser Arbeit eingeführte Ansatz der Zerlegung von Sequenzen in Fenster fester Breite ermöglicht es, den Speicherbedarf von der Länge des Inputs zu entkoppeln - dieser ist nunmehr durch die Fenstergröße beschränkt.

Abseits des verbesserten Speicherverbrauchs bringt diese Vorgehensweise grundsätzlich leider auch einen großen Nachteil mit sich: Die Laufzeit im Vergleich zur originalen Implementierung verschlechtert sich je nach gewählten Parametern teils erheblich, vor allem aufgrund von redundantem Rechenaufwand, verursacht durch überlappende Sequenzbereiche. Darüber hinaus zeigte sich in Versuchen, dass Varianten IntaRNAs mit Fensterzerlegung teils vom Original abweichende, tendenziell bessere, optimale Interaktionen finden. Dies lässt sich auf die Vorhersageheuristik IntaRNAs zurückführen, welche von den durch die Fenster definierten Subsequenzen beeinflusst wird. Unter Berücksichtigung der Accessibility nimmt die Anzahl auftretender Abweichungen deutlich ab, da diese die durchschnittliche Länge vorhergesagter Interaktionen deutlich reduziert und Ergebnisdifferenzen so unwahrscheinlicher werden.

Weiterhin ermöglicht es die für die Fensterzerlegung angepasste Hauptroutine IntaRNAs, die Interaktionsvorhersage für eine einzelne Kombination aus Query- und Targetsequenz zu parallelisieren, was zuvor nicht möglich war. Hierbei erhöht sich zwar leicht der Speicherbedarf, allerdings kann bei der Laufzeit eine Verbesserung um einen von der Anzahl der Threads abhängigen Faktor erzielt werden. Damit läuft diese Variante IntaRNAs mit vier Threads auf einem Input, bestehend aus einer langen Query- und einer langen Targetsequenz, bei deutlich verringertem, konstanten Speicherverbrauch sogar um etwa ein Drittel schneller als die originale und ist jener dadurch in jeder Hinsicht überlegen.

In der aktuellen Implementierung wird in jedem Szenario die Fensterzerlegung angewandt. In Zukunft könnte man auch die Originalversion IntaRNAs integrieren

und dann variabel die bestgeeignete Vorgehensweise wählen. Dabei hängt die Wahl von diversen Parametern wie den Wünschen des Benutzers bezüglich Laufzeit und Speicherverbrauch, der Beschaffenheit des Inputs sowie der Anzahl der Threads, die für die Berechnungen zur Verfügung stehen, ab. Darüber hinaus könnte die Laufzeit der Interaktionsvorhersage unter Verwendung der Fensterzerlegung weiter optimiert werden, um diesen Nachteil gegenüber der originalen Implementierung soweit wie möglich wettzumachen. Dazu könnte man beispielsweise für verschiedene Klassen von Inputs gewisse Parameter wie die Fenstergröße so bestimmen, dass die Laufzeit minimiert wird. Außerdem könnte eine spezifische Optimierung für Hyper-Threading implementiert werden. Dieses Feature ist hardwareseitig in fast allen modernen CPUs von Intel verbaut und ermöglicht es, doppelt so viele Threads, wie die CPU Kerne hat, auszuführen, wobei die insgesamt verfügbare Rechenkapazität optimal ausgenutzt wird. So könnte eine Laufzeitverbesserung von bis zu 34% erreicht werden [23]. Des Weiteren könnte man versuchen, die Heuristik IntaRNAs dahingehend anzupassen, dass sowohl die originale Implementierung wie auch Fensterversionen mit verschiedenen Fensterbreiten bei gleichem Input den gleichen Output produzieren. Damit würde die Verarbeitung unterschiedlicher Subsequenzen nicht mehr in voneinander abweichenden, verschieden guten Ergebnissen resultieren.

6 Danksagung

An erster Stelle möchte ich meinem Betreuer Martin Raden meinen größten Dank für die Heranführung an das Thema und das geduldige Beantworten all meiner Fragen zu IntaRNA sowie den theoretischen Grundlagen aussprechen. Darüber hinaus möchte ich mich bei ihm für extrem wertvolle Tipps zur generellen Herangehensweise, die erfolgreiche, gemeinsame Jagd nach Bugs im Code und das ausführliche, hilfreiche Feedback zu meiner Arbeit bedanken.

Des Weiteren möchte ich meiner Mutter danken, die mir während der Bachelorarbeit tatkräftig zur Seite stand, mir den Rücken freigehalten und die Arbeit Korrektur gelesen hat.

Ebenso gilt mein Dank auch meinem Vater, der mich stets mit moralischem Beistand unterstützt hat.

Literaturverzeichnis

[1] G. Storz, "An expanding universe of noncoding RNAs," *Science*, vol. 296, no. 5571, pp. 1260–1263, 2002.

[2] S. Gottesman, "Micros for microbes: non-coding regulatory RNAs in bacteria," *Trends in genetics*, vol. 21, pp. 399–404, 2005.

[3] J. Vogel and C. Sharma, "How to find small non-coding RNAs in bacteria," *Biological chemistry*, vol. 386, pp. 1219–1238, 2005.

[4] "CH391L/S14/SmallRNAs." `http://www.synbiocyc.org/wiki/index.php/File:Figure1review.png`. Zugriffsdatum: 21.08.2018.

[5] M. Rehmsmeier, P. Steffen, M. Höchsmann, and R. Giegerich, "Fast and effective prediction of microRNA/target duplexes," *RNA (New York)*, vol. 10, no. 5, pp. 1507–1517, 2004.

[6] B. Tjaden, S. S. Goodwin, J. A. Opdyke, M. Guillier, D. X. Fu, S. Gottesman, and G. Storz, "Target prediction for small, noncoding RNAs in bacteria," *Nucleic Acids Research*, vol. 34, no. 9, pp. 2791–2802, 2006.

[7] M. Kertesz, N. Iovino, U. Unnerstall, U. Gaul, and E. Segal, "The role of site accessibility in microRNA target recognition," *Nature genetics*, vol. 39, pp. 1278–1284, 2007.

[8] L. Dang, R. Lee, P. Williams, C. Yu Chan, V. Ambros, and Y. Ding, "Potent effect of target structure on microRNA function," *Nature structural & molecular biology*, vol. 14, pp. 287–294, 2007.

[9] U. Mückstein, H. Tafer, J. Hackermüller, S. H. Bernhart, P. F. Stadler, and I. L. Hofacker, "Thermodynamics of RNA–RNA binding," *Bioinformatics*, vol. 22, no. 10, pp. 1177–1182, 2006.

[10] A. Busch, A. S. Richter, and R. Backofen, "IntaRNA: efficient prediction of bacterial sRNA targets incorporating target site accessibility and seed regions," *Bioinformatics*, vol. 24, no. 24, p. 2849–2856, 2008.

[11] M. Mann, P. R. Wright, and R. Backofen, "IntaRNA 2.0: enhanced and customizable prediction of RNA-RNA interactions," *Nucleic acids research*, vol. 45, no. W1, pp. W435–W439, 2017.

[12] T. Künne, D. C. Swarts, and S. J. Brouns, "Planting the seed: target recognition of short guide RNAs," *Trends in Microbiology*, vol. 22, no. 2, pp. 74 – 83, 2014.

[13] K. J. Bandyra, N. Said, V. Pfeiffer, M. W. Górna, J. Vogel, and B. F. Luisi, "The seed region of a small RNA drives the controlled destruction of the target mRNA by the endoribonuclease RNase E," *Molecular Cell*, vol. 47, no. 6, pp. 943 – 953, 2012.

[14] R. Balbontín, F. Fiorini, N. Figueroa-Bossi, J. Casadesús, and L. Bossi, "Recognition of heptameric seed sequence underlies multi-target regulation by RybB small RNA in Salmonella enterica," *Molecular Microbiology*, vol. 78, no. 2, pp. 380–394, 2010.

[15] M. Raden, M. M. Mohamed, S. M. Ali, and R. Backofen, "Interactive implementations of thermodynamics-based RNA structure and RNA-RNA interaction prediction approaches for example-driven teaching," *PLOS Comput. Biol*, 2018.

[16] R. Gelhausen, "Constrained RNA-RNA interaction prediction," Master's thesis, Albert-Ludwigs University of Freiburg, 2018.

[17] P. N. Borer, B. Dengler, I. Tinoco, and O. C. Uhlenbeck, "Stability of ribonucleic acid double-stranded helices," *Journal of Molecular Biology*, vol. 86, no. 4, pp. 843 – 853, 1974.

[18] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner, "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure," *Journal of molecular biology*, vol. 288, pp. 911–940, 1999.

[19] D. H. Turner and D. H. Mathews, "NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure," *Nucleic acids research*, vol. 38, pp. D280–D282, 2010.

[20] J. McCaskill, "The equilibrium partition function and base pair binding probabilities for RNA secondary structure," *Biopolymers*, vol. 29, pp. 1105–1119, 1990.

[21] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic acids research*, vol. 9, pp. 133–148, 1981.

[22] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[23] X. Tian, A. Bik, M. Girkar, P. Grey, H. Saito, and E. Su, "Intel OpenMP C++/Fortran compiler for hyper-threading technology: Implementation and performance," *Intel Technology Journal*, vol. 06, no. 01, pp. 36–46, 2002.