Bachelorarbeit

Evaluierung von Algorithmen zur Vermeidung Von Sekundärstrukturen in DNA-Speichersystemen

Erstprüfer: Prof. Dr. Bernhard Seeger

Zweitprüfer: Prof. Dr.-Ing. Bernd Freisleben

Tutor: Alex El-Shaikh

Vorgelegt von Oumar Sadio

am 14 April 2025

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Marburg, den 14. April 2025 Oumar Sadio

Inhaltsverzeichnis

1. Einführung

Die Nutzung von DNA als Speichermedium wird als zukünftige Lösung betrachtet, die die Grenzen der aktuellen Technologien überwinden wird. Trotz vielversprechenden Perspektiven, viele Herausforderungen beeinträchtigen seine Entwicklung. Sekundäre Strukturen, wie Hairpins sind viele schwerwiegende Fehler verantwortlich. In diesem Kontext wurden verschiede Algorithmen wie der SSA Coder und der Permutation Coder vorgeschlagen, um dieses Problem zu beheben. Jedoch fehlt es an einer Evaluation ihrer Effektivität. Die vorliegende Studie zielt darauf ab, eine detaillierte Analyse dieser Algorithmen auf der Grundlage diverser Kriterien durchzuführen, um ihre Stärken und Schwächen zu identifizieren und ihre praktische Anwendung zu definieren. Die vorliegende Arbeit liefert somit wesentliche Kriterien für die Auswahl von Codierungsstrategien in DNA-Speichersystemen und trägt dazu bei, die Grundlagen der DNA-Speichersysteme für eine zukünftige Nutzung zu stärken.

1.1 Motivation

In den vergangenen Jahren konnte eine bemerkenswerte technologische Entwicklung beobachtet werden, die mit einer signifikanten Zunahme der Daten- und Informationsproduktion einherging. In zahlreichen Bereichen konnten Fortschritte erzielt werden, welche die Produktion großer Mengen an Informationen unterschiedlicher Art zur Folge hatten. Gemäß einer Studie von IDC nimmt die Menge an Daten seit 2015 und mindestens bis 2025 jährlich um etwa ein Viertel zu [1]. Diese Entwicklung ist insbesondere auf die fortschreitende Digitalisierung, das Internet der Dinge (IoT), Big Data, soziale Medien sowie wissenschaftliche Forschung zurückzuführen [2]. Gleichzeitig wurden Fortschritte bei den Speichermedien erzielt, um die steigenden Informationsmengen bewältigen zu können. In der Konsequenz wurden Speicherkapazitäten wie die Cloud, SSD-Festplatten sowie weitere, noch effizientere Speichermedien weiterentwickelt. So lässt sich beispielsweise eine jährliche Steigerung des SSD-Marktes zwischen 2019 und 2024 um 23,9 % beobachten [3]. Obgleich sie mit höheren Kosten verbunden sind, haben sich SSDs als Standard für Primärspeicher etabliert und beginnen nun auch, den Sekundärspeichersektor zu erobern.

Bedauerlicherweise kann die Entwicklung in diesem Bereich nicht mit der Datenproduktion Schritt halten. Denn neue Speichermedien müssen nicht nur eine größere Speicherkapazität aufweisen, sondern auch die Möglichkeit bieten, verschiedene Arten von Daten auf intelligente und nachhaltige Weise zu verwalten. Verschiedene Studien veranschaulichen das exponentielle Wachstum der Daten sowie die Notwendigkeit von neuen Speichertechnologien [4].Siehe Abbildung 1

Diese Tatsache motivierte die Forscher, sich auf der Suche nach neuen Speichermethoden, die effizienter sind, zu machen . In der Folge wurde das Thema DNA-Speicher angebracht, um den Speicherbedarf zu decken. Dieser Bereich bietet viele Vorteile und gewinnt in der Wissenschaft immer mehr an Bedeutung.

Die Verwendung von DNA als Datenspeicher basiert auf der hohen Informationsdichte [5] und Stabilität des Moleküls. Die Speicherung von Daten in DNA erfolgt durch die Codierung der Sequenz der Basen (Adenin, Cytosin, Guanin, Thymin), welche die digitalen Informationen (0 und 1) repräsentieren. Die Speicherung immenser Datenmengen in einer sehr kompakten Form sowie die Aufrechterhaltung der gespeicherten Daten über lange Zeiträume [6] [7] hinweg sind wesentliche Eigenschaften der DNA .

Zu den Institutionen, die bereits Fortschritte in diesem Gebiet erzielt haben, zählen mehrere große Universitäten sowie Unternehmen, wie zum Beispiel Microsoft Research [8] und die Universität von Illinois Urbana-Champaign [9] und Forscher der ETH Zürich [10].

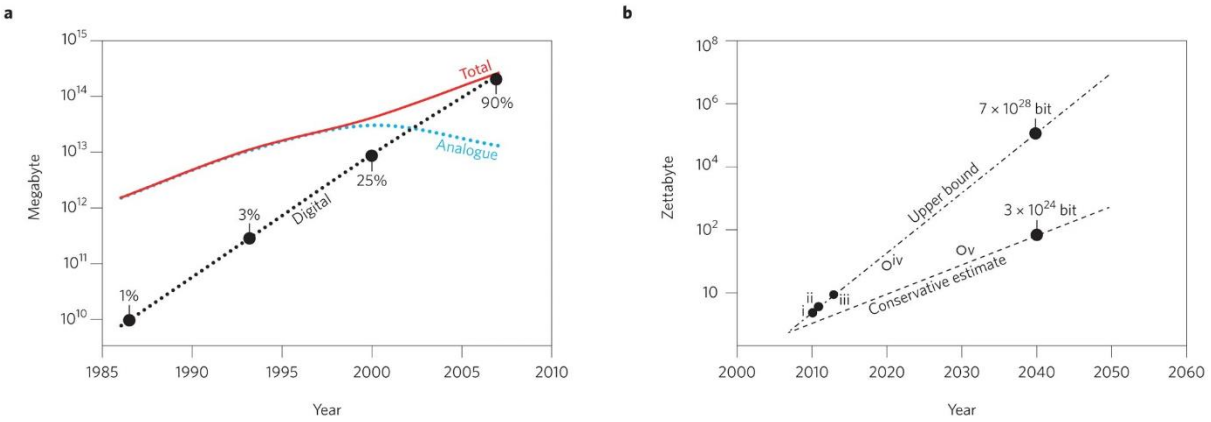


Abbildung 1: Zeitstrahl der gespeicherten analogen, digitalen und gesamten Daten und Prognostizierter globaler Speicherbedarf, basierend auf tatsächlichen Messpunkten und prognostizierten Daten *[4]*

1.2 Problembeschreibung

Die Struktur der DNA sowie die jüngsten Fortschritte in der Synthese und Sequenzierung ebnen den Weg für die Speicherung digitaler Informationen in nahezu unerschöpflichem Umfang. Allerdings ist zu berücksichtigen, dass der DNA-Speicher nicht gänzlich fehlerfrei ist. Die Synthese- und Sequenzierungstechniken weisen mehrere Schwachstellen auf, die zu unbeabsichtigten (nicht selektiven) Hybridisierungen oder Fehlern während des Prozesses führen können [5] [11]. Obwohl zahlreiche vielversprechende Ansätze zur Fehlerkorrektur und Gruppierung von DNA vorgeschlagen wurden, stehen die Forscher vor weiteren Herausforderungen.

Unter diesen Herausforderungen stellt die Sekundärstruktur der DNA einen kritischen Punkt bei der Nutzung von DNA als Speichermedium dar. Die Sekundärstruktur der DNA, welche aus Wasserstoffbrückenbindungen und weiteren Wechselwirkungen zwischen den Basen besteht, bezieht sich auf die Tendenz einer einzelsträngigen DNA-Sequenz, sich auf sich selbst zurückzufalten und manifestiert sich in Form von Doppelhelix, Haarnadelstrukturen oder Schleifen.

Dies verursacht Schwierigkeiten beim Lesen und Schreiben von DNA-Sequenzen und somit irreparablen Fehlern in DNA-Speichersystemen. Damit stellt die Sekundärstruktur der DNA ein wesentliches Problemfeld dar, dessen Lösung erforderlich ist, um die praktische Anwendung von DNA als neuem Speichermedium zu ermöglichen.

Zur Vermeidung solcher Strukturen wurden verschiedene Algorithmen angebracht, die durch unterschiedliche Arten zur Minderung der Fehler beitragen.

1.3 Zielsetzung

Im Rahmen der vorliegenden Arbeit wird sich mit dem Thema DNA-Speicher befasst, wobei ein Schwerpunkt auf der Vermeidung von Sekundärstrukturen in DNA-Sequenzen liegt.

Das Hauptziel unserer Arbeit ist, verschiedene Algorithmen zur Vermeidung sekundärer Strukturen in DNA-Sequenzen zu evaluieren und anschließend ihre praktische Einsatzbarkeit in DNA-Speichersystemen zu bewerten. Dabei werden hauptsächlich zwei Algorithmen in Betracht gezogen: der (m-k) -SSA (Secondary Structure Avoidance) Algorithmus, der auf einen Sequenzersatz basiert und der Permutation Coder, einen Permutations-basierte Ansatz. Um die Evaluierung strukturiert durchzuführen, werden als Teilziele eine theoretische Analyse zur Untersuchung der Unterdrückung von Sekundärstrukturen anhand ihrer Eigenschaften, gefolgt von einer praktischen Evaluierung durch Anwendung auf verschiedene Sequenzen und die Entwicklung praxisrelevanter Leitlinien basierend auf die quantitative Bewertung der resultierenden Strukturstabilität definiert.

Durch diese Arbeit soll ein fundierter Beitrag zur Optimierung der DNA basierten Speichersystemen geleistet werden, indem verschiedene Ansätze miteinander verglichen und bewertet werden. Die Arbeit liefert damit wesentliche Grundlagen für die Nutzung von DNA als Speichermedium der Zukunft.

1.4 Vorgehensweise

Die vorliegende Arbeit ist in sechs Kapitel unterteilt. Im ersten Schritt werden die Grundlagen der DNA und ihre Funktion als Speichermedium erörtert sowie die Grundlagen von DNA-Sekundärstrukturen und ihren Einfluss auf die DNA-Speichersysteme untersucht. In diesem Kapitel werden auch die verschiedenen Herausforderungen vorstellen, denen sich der DNA-Speicher gegenübersieht, und einen Überblick über den aktuellen Stand der Lösungen geben, die bei der Vermeidung der Sekundärstruktur eingesetzt wurden.

Das nachfolgende Kapitel widmet sich der Methodik. In diesem Abschnitt der Arbeit werden die ausgewählten Algorithmen präsentiert und deren jeweilige Implementierung beschrieben. Zudem werden die Kriterien sowie Metriken zur Analyse der gegebenen Algorithmen vorgestellt.

Im Abschnitt "Ergebnisse" erfolgt zunächst eine Erläuterung der durchgeführten Tests sowie im weiteren Verlauf eine Analyse der gewonnenen Erkenntnisse. Letztere resultieren aus der Anwendung der Implementierung der Algorithmen auf den Datensatz. Die Resultate ermöglichen es, die jeweiligen Vor- und Nachteile der einzelnen Ansätze zu evaluieren.

Im letzten Schritt werden die Ergebnisse diskutiert und so aufbereitet, dass daraus eine Bewertung ihrer praktischen Anwendbarkeit und die Identifikation optimaler Einsatzbereiche erfolgen kann. Dies wird mit Verbesserungsvorschlägen und weitere Ansätze ergänzt.

Am Ende der Arbeit erfolgt eine Zusammenfassung der wichtigsten Erkenntnisse und zukünftige Forschungsrichtungen werden vorgeschlagen

2. Grundlagen der DNA-Speicher

2.1 Definition von DNA

Die Desoxyribonukleinsäure (DNA) stellt eine Folge von mehreren Molekülen, den sogenannten Nukleotiden oder Basen, dar, welche eine Information in ihrer Reihenfolge repräsentieren [12]. Die vier Basen Adenin (A), Thymin (T), Guanin (G) und Cytosin (C) bilden die fundamentalen Bausteine der DNA [13].

Ein wesentliches Merkmal der Desoxyribonukleinsäure nach den Arbeiten von Watson und Crick ist die Komplementarität der Basen [12], aus denen sich die DNA zusammensetzt. Dabei ist Adenin komplementär zu Thymin, Guanin komplementär zu Cytosin und umgekehrt. Die Komplementarität der Basen erlaubt die Beobachtung der Wechselwirkung zwischen zwei DNA-Sequenzen und erleichtert somit die Übertragung genetischer Information [14] [15]. Sie ist für die zweite Eigenschaft der DNA, ihre Doppelhelix-Struktur, verantwortlich. Diese entsteht durch die Bindung zweier linearer Sequenzen, die komplementär zueinander sind. Diese zwei wesentlichen Charakteristika der DNA machen sie zu einem stabilen Molekül, welches die Speicherung von Daten begünstigt.

Eine lineare DNA-Sequenz wird als Oligonukleotid bezeichnet. Das 5′-Ende und das 3′-Ende definieren den Beginn bzw. das Ende der Sequenz und sind für Prozesse wie die Replikation und Transkription von zentraler Bedeutung. [13].

2.2 DNA als Speichermedium

Der Prozess der Speicherung in der DNA ist hauptsächlich in vier Teile gegliedert, siehe Abbildung 2:

- Die Umwandlung der Informationen in die DNA.

Die Daten werden in Form von Bits dargestellt und dann mithilfe eines ausgewählten Algorithmus in DNA-Sequenzen kodiert. Ein Beispiel wäre die Konvertierung in Quaternäre Zahlen. Die Basen A, C, T und G entsprechen jeweils den Ziffern 0, 1, 2 und 3. Die codierten Sequenzen werden anschließend synthetisiert und während dieser Methode werden auch mehrere Kopien derselben Sequenz hergestellt. Da DNA-Sequenzen bei der Synthese häufig in kleinere Segmente aufgeteilt werden, werden für eine spätere Rekonstruktion Indizes oder übereinander liegende Chunks benötigt. Die Untersuchungen von Heckel et al. zeigen, dass ein einfaches Schema, das auf einem Index basiert, theoretisch optimal ist. [16] [8]

- Datenerfassung

Die Daten, die bei der Synthese entstehen, werden in Pools aufbewahrt, die laut Organik et al. eine Speicherkapazität von etwa 10^{12} Bytes haben [7]. Eine Bibliothek wird aus diesem Grund verwendet, um große Datenmengen zu verwalten. [11] [17]

-Zufälliger Zugriff (Random Access)

Die benötigten Daten werden durch zufälligen Zugriff aus ihren jeweiligen Pools mit Hilfe von Mechanismen wie PCR-Primern ausgewählt. [18]

- Ablesen der Sequenzen

Die Sequenzen von den Pools werden anschließend geordnet, untersucht und entschlüsselt, um die ursprüngliche Sequenz zu rekonstruieren. [19]

Am Ende des Prozesses werden Errors Codes Algorithmen, wie Reed Solomon eingesetzt, um Fehler zu korrigieren, die während des Sequencings aufgetreten sind. [20] [21]

Im Unterschied zu anderen magnetischen Speichermedien besteht das Speichersystem aus einem Synthesizer, welcher die Daten in einen DNA-Strang codiert, einem Container, welcher die Stränge in Pools speichert und verwaltet, sowie einem Sequenzer, welcher die DNA-Stränge liest und in digitale Daten umwandelt. Ein DNA-Strang besteht aus maximal 200 Nukleotiden. Weil jedes Nukleotid nur eine begrenzte Menge an Daten speichern kann, hat ein DNA-Strang Platz für 100 Datenbits. Ein Datensatz wird in mehreren DNA-Strängen gespeichert, die in Pools mit jeweils einer Adresse gespeichert werden.

Ein Satz codierter DNA-Sequenzen, auch als DNA-Codewörter bezeichnet, wird als Code definiert, wenn er bestimmte spezielle Eigenschaften (oder Beschränkungen) für die Zwecke der DNA-Berechnung erfüllt. Eine umfassende Darstellung der Arten von Constraint-Problemen, die bei der Codierung für den DNA-Computer auftreten, wurde von Milenkovic und Kashyap im Jahr 2006 eingeführt [22], einschließlich der konstanten GC-Inhaltsbeschränkung auf 50 % (die sich auf den Prozentsatz der Nukleotide bezieht, die G oder C sind), der Hamming-Distanzbeschränkung [18](die eine ausreichende Differenzierung der Codewörter der DNA gewährleistet). Hierzu zählen die Vermeidung der Bildung von Sekundärstrukturen, welche eine Inaktivierung der DNA-Sequenz durch Rückfaltung in sich selbst verhindert, sowie die Präsenz von Homopolymeren [23], das heißt konsekutive Wiederholungen desselben Nukleotids.

Aufgrund ihrer Eigenschaften weist die Verwendung von DNA als Speichermedium eine Reihe von Vorteilen auf. Die Speicherkapazität von synthetischer DNA ist theoretisch unerschöpflich. [8] Ein Gramm DNA kann mehrere Petabytes an Daten speichern, was sie zu einem äußerst kompakten Speichermedium macht. Ihre hohe Dichte [5] qualifiziert sie als exzellentes Speichermedium für die Speicherung großer Datenmengen in geringen Mengen. Die theoretische Grenze liegt bei über 1 EB/mm³, was einer acht Größenordnungen höheren Dichte als bei Band entspricht. Ein weiterer Vorteil ist die Langlebigkeit [6], die sich in einer Halbwertszeit von über 500 Jahren manifestiert, wobei diese in rauen Umgebungen beobachtet wurde. Des Weiteren ist DNA nicht auf ein spezifisches Format festgelegt und kann sämtliche Arten von Daten speichern. [11]

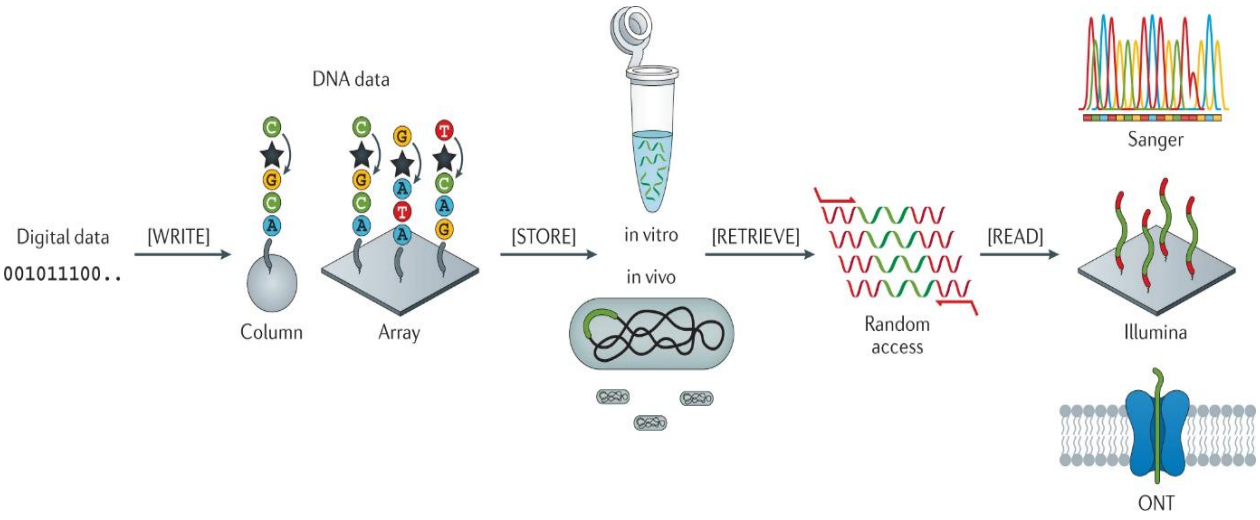


Abbildung 2: Überblick über den Prozess von DNA-Speicher *[11]*

2.3 DNA-Synthese und Sequenzierung

Der Begriff "DNA-Synthese" bezeichnet einen Prozess, bei dem unter Zuhilfenahme diverser Methoden künstliche DNA-Partikel erzeugt werden. Dazu werden die Nukleotide nacheinander in einer spezifischen Abfolge hinzugefügt, um eine bestimmte DNA-Sequenz zu erhalten.
Zu den angewandten Methoden zählen insbesondere die chemische und die enzymatische Synthese von DNA.

Im Rahmen der chemischen DNA-Synthese erfolgt die die sequenzielle Addition von Nukleotiden mithilfe von Phosphoramiditen und Schutzgruppen – beispielsweise durch den Einsatz von 4,4′-Dimethoxytrityl (DMT), um unerwünschte Nebenreaktionen an reaktiven Gruppen zu verhindern. Diese Methode bietet eine Genauigkeit von ca. 99 % und eignet sich diese Methode optimal für die Synthese kurzer DNA-Sequenzen bis zu 200 Nucleotiden. Allerdings führt die geringe Fehlertoleranz in einigen Fällen zu einem falschen Ergebnis, sodass sie bei langen DNA-Sequenzen weniger effektiv ist. [24]

Im Gegensatz dazu zielt die enzymatische Synthese darauf ab, mit Hilfe natürlicher Enzyme zusätzliche Sequenzen zu einer bereits gegebenen Sequenz zu erzeugen [25]. Ein prominentes Beispiel ist die Polymerase Chain Reaction (PCR), bei der DNA-Polymerasen zum Einsatz kommen, um vorhandene Sequenzen exponentiell zu vervielfältigen Diese Methode eignet sich daher insbesondere für das Kopieren oder Vervielfältigen einer bereits gegebenen Sequenz. [9]

Die DNA-Sequenzierung ist ein Verfahren, das die Reihenfolge der Kernbasen in einer DNA-Sequenz bestimmt. Zu den derzeit populärsten Methoden zählen die Next-Generation Sequencing (NGS) Technologien, wie etwa die Illumina-Sequenzierung, Ion-Torrent-Technologie und Pyrosequenzierung. Diese Methoden basieren auf der Verwendung von DNA-Polymerase-Enzymen und werden als "synthetische Sequenzierung" bezeichnet. [26]Das Ziel dieser Methode besteht darin, mit Hilfe von Polymerase und fluoreszierenden Nukleotiden eine komplementäre Sequenz zur Basissequenz zu erstellen. Die Verwendung von fluoreszierenden Nukleotiden ermöglicht eine optische oder chemische Signalerkennung, was die Lesbarkeit der Komplementärsequenz und somit die Rekonstruktion der Originalsequenz erleichtert. Die parallele Sequenzierung von Millionen von DNA-Fragmenten durch NGS stellt eine ideale Methode für die Analyse von langen Sequenzen dar.

Das in der DNA befindliche Speichersystem sieht sich jedoch mit verschiedenen Herausforderungen konfrontiert, die seine zukünftige Nutzung als äußerst komplex erscheinen lassen.
Obschon die Wartungskosten für die Speicherung von Daten in der DNA deutlich geringer sind als bei den bislang verwendeten Methoden, sind die Kosten für die Synthese und Sequenzierung der DNA noch mit einem hohen finanziellen Aufwand verbunden, sodass die Nutzung der DNA-Speicherung verhindert oder zumindest verzögert wird.

Unter Berücksichtigung der derzeitigen Kosten von ca. 10-4 $/Base und einer Kodierungsdichte von 1 Bit/Base lassen sich die Schreibkosten auf 800 Millionen Dollar/TB schätzen, während die Kosten für das Band etwa 16 $/TB betragen. Demgegenüber sind die mit den aktuellen Sequenzierungsmethoden erzielbaren Lesekosten in einer kleineren Größenordnung von ca. 0,01 bis 1 Million Dollar/TB anzusiedeln, was jedoch nach wie vor einen hohen finanziellen Aufwand darstellt. [5] [27] [17]

Eine weitere Herausforderung stellt die Geschwindigkeit des Schreibens und Lesens von Daten dar, welche nach wie vor deutlich langsamer ist als die derzeit verwendeten Mittel. [11]
Die größte Herausforderung besteht jedoch in der Anfälligkeit für Fehler bei der Synthese und Sequenzierung, die auf verschiedene Ursachen zurückzuführen sind (wie beispielsweise Substitutionen, Insertionen und Deletionen von Nukleotiden). [28]

Die Methoden der DNA-Synthese und -Sequenzierung machen große Fortschritte, auch wenn sie für die Datenspeicherung noch nicht ausgereift sind. Ihre Entwicklung verläuft jedoch so exponentiell, dass sie das Moore'sche Gesetz übertrifft. [29] [25]Dank zahlreicher Fortschritte in der Biotechnologie werden diese Methoden bald die Verwendung von DNA als Speichermedium ermöglichen.

3. Grundlagen der DNA-Sekundärstruktur

3.1 Definition von Sekundärstrukturen

Die Sekundärstrukturen der DNA sind das Resultat von Wechselwirkungen zwischen den Basen eines einzelnen DNA-Strangs. Infolgedessen kommt es zu stabilen Faltungen, die lokal begrenzte dreidimensionale (3D) Strukturen bilden. Diese manifestieren sich in der Form von Hairpins, G-Quadruplexe und Cruciforme Strukturen. Siehe Abbildung 3

Hairpins (Schleifen) bezeichnen die Rückfaltung komplementärer Sequenzen innerhalb eines Strangs. Dabei paaren sich inverse Wiederholungen intramolekular, es kommt zur Ausbildung von Wasserstoffbrücken zwischen basenseitig komplementären Abschnitten, was zur Bildung eines doppelsträngigen Stammes und einer einzelsträngigen Schleife führt.
G-Quadruplexe sind Stapelungen von G-reichen Sequenzen In diesen Strukturen ordnen sich vier guaninbasierte Viertelkreise, die durch koordinative Bindungen mit Ionen (zum Beispiel Kaliumionen) zusätzlich stabilisiert werden, übereinander an. [30]
Cruciforme Strukturen sind eine kreuzförmige Faltung von palindromischen Sequenzen. [31]

Die Ursachen für die Entstehung dieser Falten können vielfältig sein. Zu den potenziellen Faktoren zählen das Vorhandensein von inversen Komplementen, eine hohe Konzentration der Basen G und C, die eine bessere Stabilisierung der Strukturen ermöglichen als die Basen A und T, das Vorhandensein von Homopolymeren, die Wiederholungen derselben Base (z. B. AAAA) sind sowie Umweltfaktoren wie niedrige Temperaturen oder hohe Ionenzahlen.

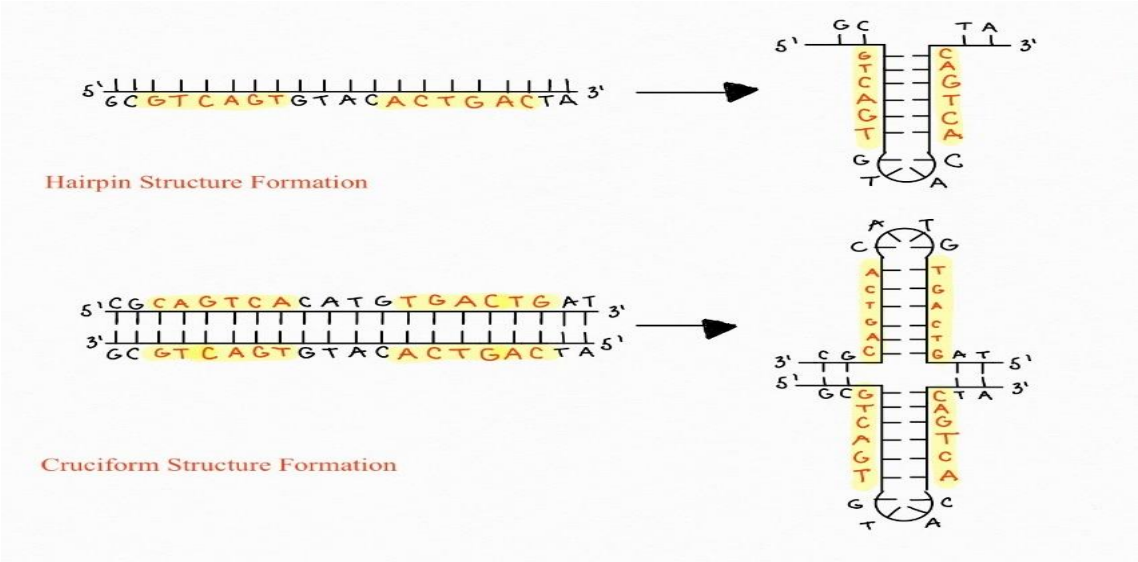


Abbildung 3: DNA Sekundärstrukturen

3.2 Einfluss auf DNA-Speicher

Die Sekundärstruktur der DNA stellt eine große Herausforderung für die Speicherung digitaler Daten in der DNA dar. Diese Problematik manifestiert sich in der Synthese, der Datensequenzierung sowie der Datenintegrität. [33] [34]
Insbesondere führen gefaltete DNA-Abschnitte zu einer Verringerung der Effizienz und Produktivität des chemischen Syntheseprozesses, weil Polymerasen in diesen Bereichen häufiger unterbrechen oder Fehler machen [34]. Ein Problem, das sich aus der Art der Struktur ergibt, ist die Bildung von unerwünschten Bindungsstellen. Dies kann zu Ungleichgewichten führen, die wiederum eine fehlerhafte Durchführung der Basenpaarung zur Folge haben. Dies kann zu Veränderungen und Fehlern in den aufbewahrten Daten führen. In der Konsequenz können synthetisierte DNA-Stränge ungleichmäßig oder fehlerhaft sein und sich folglich nicht für die Konservierung eignen. [32] [20] [11] [33] [30]

Darüber hinaus resultiert die Struktur in mehreren Fehlern im Prozess der DNA-Sequenzierung. Die angewandten Techniken, wie die NGS-Methoden, sind nicht in der Lage, spezifische Abschnitte der DNA-Sequenz zu erreichen. Dies resultiert in Sprüngen von Sequenzabschnitten, Leseabbrüchen oder Fehlinterpretationen der Sequenzen. [34] Dies versursacht Lesefehler und resultiert in einer reduzierten Zuverlässigkeit der Datenwiederherstellung. [35]

Diese Fehler, die während der Synthese und Sequenzierung auftreten, gefährden die Integrität der Daten. Letztlich resultiert die Vervielfachung solcher Fehler – bedingt durch wiederholte Replikationen – in einem erheblichen Informationsverlust. So zeigte eine Studie, dass bei GC-reichen Sequenzen ein Datenverlust von bis zu 15 % auftreten kann [35].

Die Beeinflussungen der Sekundärstruktur sind wegen ihrer thermodynamischen Stabilität schwer. Die bisherigen Labormethoden sind dafür nicht gut einsetzbar [36].

Damit die Informationen korrekt gespeichert werden können, sind komplexe Korrekturmechanismen erforderlich [36]. Eine bessere Darstellung oder Kodierung der DNA könnte verhindern, dass sich selbstkomplementäre Sequenzen bilden. Damit würde sich die Wahrscheinlichkeit einer solchen Selbsthybridisierung verringern. Die Verringerung der Selbstkomplementarität führt jedoch zu einer Verringerung der potenziellen Dichte der Darstellung, so dass ein Kompromiss zwischen Dichte und Zuverlässigkeit gefunden werden muss. [37]

3.3 Aktueller Stand der Forschung

In diesem Abschnitt erfolgt die Vorstellung der verschiedenen Strategien, die zur Behebung der Sekundärstruktur angewendet wurden, sowie eine Erläuterung ihrer jeweiligen Stärken und Grenzen. Eine Zusammenfassung dieser Strategien erfolgt in Abbildung 4.

1. thermodynamische Modelle

Diese Modelle dienen als Grundlage für viele Algorithmen zur Vermeidung der sekundären DNA-Struktur. Die freie Energie von Gibbs (ΔG) wird dabei benutzt, um die wahrscheinlich entstehenden Strukturen sowie deren energetische Stabilität vorherzusagen. Zu den bekanntesten Algorithmen, die auf diesen Modellen basieren, gehören unter anderem:
- Der Nussinov-Algorithmus ist ein einfaches und dynamisches Werkzeug. Damit kann die maximale Anzahl an Basenpaaren in einer RNA- oder DNA-Sequenz und einer oder mehrerer potenzieller Sekundärstrukturen bewertet werden. [38]

- Der Zucker-Algorithmus ist der am weitesten verbreiteter und am häufigsten verwendeter Algorithmus, um die Sekundärstruktur von RNA zu berechnen. Dabei wird die freie Energie so klein wie möglich gehalten. [39]
- Der Sankoff-Algorithmus ist eine Methode, die die Faltung und das Alignment von Sequenzen berücksichtigt. Dadurch wird die Strukturbestimmung genauer [40]

Das ViennaRNA-Paket besteht aus verschiedenen Software- Komponenten, die die Vorhersage und den Vergleich von sekundären DNA-Strukturen auf Basis einer energetischen Optimierung ermöglichen. Die enthaltene Codebibliothek stellt dabei eine wesentliche Komponente dar. [14]

Die Algorithmen und Modelle, die evaluiert wurden, sind sehr leistungsfähig bei der Vorhersage von Sekundärstrukturen. Dabei wurde die energetische Stabilität als Kriterium herangezogen. Das macht sie zu einem guten Werkzeug, um DNA-Sequenzen zu verbessern. Dennoch ist zu berücksichtigen, dass sie häufig einen hohen Rechenaufwand erfordern, was ihre Anwendung in größerem Maßstab erschwert.

2. Heuristische Methoden für das Sequenzdesign

Das sind Methoden zur Problemlösung, die keine detaillierte Analyse führen. Sie basieren auf allgemeinen Regeln oder bereits identifizierten Mustern . Bei dieser Untersuchung wird der GC-Anteil beachtet. Wenn Guanin (G) und Cytosin (C) im richtigen Verhältnis zueinanderstehen, wird die Stabilität der Paarungen verringert. [20]

Des Weiteren wird die Bildung von Palindromen, welche häufig für die Ausbildung von Haarnadelstrukturen verantwortlich sind, verhindert. Dies erfolgt durch ein individualisiertes Design der DNA-Sequenzen. Die Verwendung von Kodierungsbibliotheken [33] ermöglicht die Auswahl optimaler DNA-Sequenzen, welche die Bildung von Haarnadeln verhindern. [41] [7]

Die präsentierten heuristischen Methoden liefern in kurzer Zeit vielversprechende Resultate. Die vollständige Eliminierung von sekundären DNA-Strukturen kann jedoch nicht gewährleistet werden, da die Methoden neue, nicht vorhersehbare Probleme verursachen können. Dies führt zu einer Einschränkung der Robustheit der Algorithmen. [41]

3. Maschinelles Lernen und Optimierung:

In diesem Kontext werden Algorithmen und Ansätze betrachtet, die auf der Nutzung von Trainingsdaten basieren. Das Ziel besteht in der Entwicklung von Modellen, welche die Antizipation und Prävention von Sekundärstrukturen ermöglichen. Neuronale Netze, wie beispielsweise Convolutional Neural Networks (CNNs) [42] oder Random Forests, ermöglichen die Analyse von Sequenzen sowie die Erkennung von Eigenschaften, die die Bildung sekundärer Strukturen begünstigen. [43]

Der Einsatz von maschinellem Lernen in unserem Kontext ermöglicht es, den Prozess flexibel zu gestalten, indem verschiedene Parameter und Umgebungen berücksichtigt werden. Sie ermöglichen die Vermeidung und Vorhersage der Sekundärstruktur der DNA aus verschiedenen Blickwinkeln.

Die Ergebnisse hängen stark von der Qualität der Daten ab, die während des Trainings verwendet werden, sowie von den während des Prozesses berücksichtigten Parametern, die unzureichend oder schlecht sein können. [44]

4. Fraktale und kombinatorische Ansätze:

Fraktale und kombinatorische Methoden versuchen zu verhindern, dass Sequenzen in ungewollter Weise Sekundärstrukturen ausbilden. Dazu werden robuste DNA-Sequenzen erzeugt, die sich durch den Einsatz von mathematisch festgelegten Sequenzen auszeichnen, welche keine Selbstähnlichkeit aufweisen oder bestimmten Beschränkungen unterliegen [20] [45].

Die von den Algorithmen generierten Ergebnisse reflektieren nicht vollständig die tatsächlichen Resultate der Experimente, da diese nicht den diversen biophysikalischen Gegebenheiten und chemischen Interaktionen wie dem pH-Wert, der Ionenstärke und unvorhersehbaren chemischen Wechselwirkungen ausgesetzt sind. [46]

5. Multiobjektive Optimierung und Nutzung von Werkzeugen und Plattformen:

Durch den Einsatz von Software und Open-Source-Tools oder Methoden wie genetische Algorithmen werden mehrere Ziele gleichzeitig verfolgt. Dabei werden auch redundante Strukturen reduziert, um den Schreib- und Leseprozess zu optimieren. [47] [48] [49]

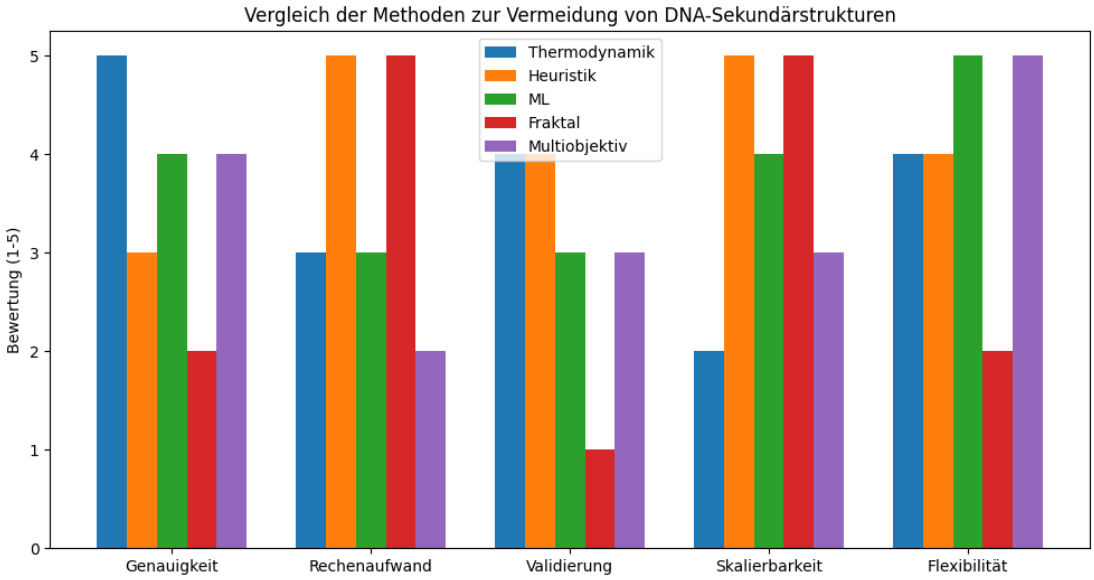


Abbildung 4: Überblick über die aktuellen Methoden mit Stärken und Schwächen

4. Methodik

4.1 Kriterien der Evaluierung

Unsere Bewertung der Algorithmen wird sich auf die folgenden Kriterien konzentrieren, um ihre Rolle bei der effektiven Vermeidung der Bildung von Sekundärstrukturen im Prozess der Informationsspeicherung in der DNA beurteilen zu können.

- **Berechnungszeit/Komplexität**

Das vorliegende Kriterium ermöglicht die Messung der Ausführungszeit eines Algorithmus für eine gegeben DNA-Sequenz oder die Definition der Zeitkomplexität Big-O-Notation. Die Ausführungs- und Programmiergeschwindigkeit ist dabei ein entscheidender Faktor, insbesondere bei der Verarbeitung von Daten in großem Umfang. Im Bereich der Informationsspeicherung in DNA ist der schnelle Zugriff und die schnelle Verarbeitung von Daten entscheidend, damit sie als effizientes und dynamisches Speichermedium angesehen werden kann. Algorithmen, die zu viel Zeit benötigen, werden daher weniger bevorzugt. Daher wird die theoretische Ausführungszeit jedes Algorithmus mit Hilfe bestehender Techniken wie „System.currentTimeMillis()“ in Java untersucht [35] [50].

- **Effizienz in der Vermeidung von Sekundärstrukturen**

Das ist das Hauptkriterium unserer Bewertung, da die Algorithmen speziell für diesen Kontext ausgewählt.
Die Bewertung dieses Kriteriums erfolgt mithilfe einem Sekundärstrukturindex I zwischen 0 und 1. Dieser Index gibt die Wahrscheinlichkeit an, mit der eine bestimmte DNA-Sequenz eine oder mehrere Sekundärstrukturen aufweist. Ein Index-Wert, der 0 nähert, bedeutet also eine geringere Wahrscheinlichkeit für stabile Sekundärstrukturen.

Mit der Messung der freien Energie nach Gibbs (ΔG) ist es möglich, die Auswirkungen der Algorithmen auf die Stabilität der Sekundärstrukturen einer gegebenen DNA-Sequenz zu bestimmen. Ebenso wird die Anzahl der GCs am Ende der Kodierung untersucht, denn zu viele dieser Basen können unerwünschte Strukturen begünstigen. Es besteht ein Zusammenhang zwischen dem GC-Gehalt und der freien Energie, da mehr GCs die Stabilität der Sekundärstruktur erhöhen. [34] [36]

In diesem Zusammenhang wird auch die RC-Rate-Fehlerfunktion eingeführt. Diese Funktion berechnet die Abnahme der inversen Komplementpaare infolge der Kodierung nach dem Algorithmus.

- **Auswirkungen auf die Fehlerrate bei der Sequenzierung**

Untersucht wird, inwieweit die Sequenzen, die durch die Kodierung erhalten werden, den Sequenzierungsprozess, der als besonders fehleranfällig gilt, erschweren oder erleichtern. Besonders das vermehrte Auftreten von Homopolymeren und ein hoher GC-Gehalt können zu einer erhöhten Fehlerrate bei der Sequenzierung führen.

Um diese Zusammenhänge zu evaluieren, werden verschiedene Simulationen – etwa auf Basis von Monte-Carlo-Methoden – herangezogen, die die Wahrscheinlichkeit der Entstehung neuer Homopolymere bewerten. Zudem wird der GC-Gehalt vor und nach der Kodierung gemessen, um die Auswirkungen der einzelnen Algorithmen auf diese Rate zu bewerten.. [50] [51]

- **Robustheit bei variierenden Bedingungen**

Das Kriterium erlaubt eine unmittelbare Evaluation der praktischen Anwendbarkeit und Zuverlässigkeit der Algorithmen. Zu diesem Zweck werden die folgenden Bedingungen überprüft:

Zunächst wird die Erreichung einer stabilen Leistung in Abhängigkeit von verschiedenen Parametern wie der Länge, der GC-Rate oder der Anzahl der Homopolymere der zu kodierenden Sequenzen überprüft, um eine Anpassung an verschiedene Szenarien zu bestätigen.

Ebenso relevant ist eine nahezu lineare Ausführungszeit für kurze und lange Oligonukleotide, da so eine mögliche Skalierbarkeit der Algorithmen abgeleitet werden kann.

Schließlich wird die Reaktion der Implementierung auf Fehler oder Extremsituationen untersucht. Dazu wird ihre Leistung mit sehr reichhaltigen GC- oder Homopolymerdaten getestet.

Mittels einer Analyse der minimalen freien Energie der Ergebnisse kann ermittelt werden, ob die Effizienz der Unterdrückung der Sekundärstruktur unabhängig von der Zusammensetzung der Sequenz ist.

Trotz der Existenz verschiedener physikalischer Faktoren wie Temperatur und Lagerungsort, die sich auf die DNA-Sequenzen und damit auf die Wahrscheinlichkeit der Bildung von Sekundärstrukturen auswirken, wird die vorliegende Studie ausschließlich theoretisch bleiben und sich auf die Ergebnisse der durchgeführten Simulationen und der oben erwähnten Messinstrumente nach der Kodierung stützen. [36] [51]

4.2 Auswahl der Algorithmen

In diesem Abschnitt der Studie werden zwei Algorithmen zur DNA-Codierung fokussiert, die durch unterschiedliche Methoden hauptsächlich dazu beitragen, die Risiken der Bildung von Sekundärstrukturen zu reduzieren.

In diesem Zusammenhang wird der (m-k) -SSA-Algorithmus (Secondary Structure Avoidance) in der vorliegenden Studie berücksichtigt. Die Implementation in Java erlaubt eine detaillierte Analyse deder Effektivität dieses Algorithmus. Daher wurde für diesen eine detailliertere Darstellung vorgenommen.

Der andere Encoder, der in unserer Auswertung berücksichtigt wird, ist der Permutation Encoder, ein Algorithmus, der bereits von Dozent Alex El Shaikh und Dr. Professor Bernhard Seeger in Java entwickelt und codiert wurde. Dieser spielt ebenfalls eine entscheidende Rolle im Prozess der Verschlüsselung von DNA-Sequenzen.

4.2.1 (m-k) -SSA Algorithmus

Mit dem Ziel, eine konkretere Visualisierung der Prozesse zu erhalten, die bei der Vermeidung der Sekundärstruktur der DNA zum Einsatz kommen, wird in der vorliegenden Auswertung die (m, k)-SSA – abgeleitet von der Secondary Structure Avoidance-Codierung – vorgestellt. Dieses Framework ist aus den Arbeiten von Nguyen Tuan Thanh & co. [52] sowie den jüngsten Arbeiten von Rui Zhang & co. hervorgegangen. [53] In unserem Kontext wird sie als ein effektiver Ansatz betrachtet, dessen Prinzip auf dem Sequenzersatz speziell in einzelsträngigen DNA-Sequenzen beruht. Durch diese Methode lassen sich potenzielle Faltungsmuster im Vorfeld identifizieren und gezielt eliminieren, um unerwünschte Sekundärstrukturen zu verhindern.

Die vorliegende Untersuchung wurde mittels einer Java-Implementierung unterstützt, die auf den Forschungsergebnissen der zuvor angeführten Literatur basiert. Die Umsetzung liegt auf GitHub alexelshaikh/dna at sadio und ist ein Teil der DNA-Speicher Forschung von dem Dozent Alex El-Shaikh und Dr. Prof. Bernhard Seeger. Die Implementierung liefert dabei nicht nur einen praktischen Beleg für die theoretischen Konzepte, sondern ermöglicht auch eine anschauliche Darstellung der Performance und Robustheit des Ansatzes.

Es ist essenziell, dass vor der Vorstellung des Algorithmus eine Erläuterung der grundlegenden Begriffe erfolgt, um die Funktionsweise des Verfahrens adäquat zu erfassen.

Definition 1:
Gegeben ein $N \epsilon \mathbb{N}$. Die DNA-Darstellung von N ist die Ersetzung der Symbole in der quaternären Darstellung von N über D = {0, 1, 2, 3} nach folgender Regel: 0 ↔A, 1 ↔T, 2 ↔C und 3 ↔G
Beispiel: Sei n = 15_{10} = 33_4 = GG

Definition 3
Die Komplementarität zwischen zwei Basen x und y mit x = $\bar{y}$ notiert. Für eine gegebene Sequenz x ϵD^n , mit x =$x_1x_2...$ x_n, definieren wir y= $RC(x) = \bar{x}_n \bar{x}_{n-1} .. \bar{x}_3 \bar{x}_2 \bar{x}_1$ als das **inverse Komplement** von x. y und x sind dann **zwei inverse komplementäre Sequenzen**. Beispiel: Seien x = ATCGGGCCAT und y =ATGGCCCGAT. x und y sind zwei umgekehrt komplementäre Sequenzen.

Definition 5
Seien $y = x_i\, x_{i+1} ... x_{i+l-1}$ und $z = x_j\, x_{j+1} ... x_{j+s-1}$ zwei nicht-überlappende Sequenzen mit $j > i + l - 1$, das heißt mit klarer Trennung zwischen ihren jeweiligen Start- und Endpositionen in der Ursprungsfolge x. Der Abstand zwischen y und z wird als d (y, z) = j-i -l definiert. Wenn y =RC(z), ist der Abstand k = d (y, z) die Länge der Schleife oder **Loop** der Sekundärstruktur. Die Sequenzen x und y werden dann als **Stem** bezeichnet. Siehe Abbildung 5

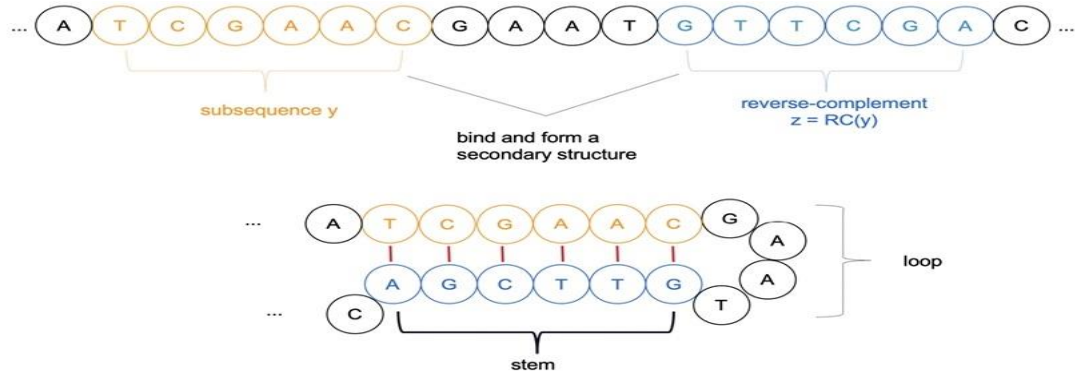


Fig. 1. DNA secondary structure model.

Abbildung 5: Repräsentation von DNA Sekundärstruktur mit Loop und Stem [53]

Definition 6

Es sei 0 < m ≤ n. Eine DNA-Sequenz x ϵD^n wird als **(m, k) -SSA-Sequenz** charakterisiert, wenn jedes Paar von Untersequenzen y und z von x und der Länge s, wobei z die inverse komplementäre Sequenz von y ist (y=$RC(z)$), eine der nachfolgenden Bedingungen erfüllt:

- Die Länge s der Untersequenzen y und z ist kleiner als m **(s<m)**

- **oder** der Abstand zwischen den beiden Teilfolgen y und z d (y, z) ist kleiner als k **(d(y, z) <k**).

Anders ausgedrückt ist eine DNA-Sequenz x eine (m-k)-SSA-Sequenz, wenn es **kein Paar** von Untersequenzen y und z von x mit der Länge s gibt, so dass **s ≥ m und d(y, z) ≥ k** gilt.
In der Folge wird ein Code C ⊆ D^n als (m, k)-SSA-Code bezeichnet, wenn jedes Codewort in C die Kriterien für einen (m, k)-SSA-Sequenzcode erfüllt.

Die Bedingungen, die die Faltung eines Oligonukleotids auf sich selbst bewirken, werden durch die Wahl von m und k bestimmt werden.

- Die Länge des Stems m steht in einem bestimmten Verhältnis zur Anzahl der Sekundärstrukturen innerhalb eines DNA-Moleküls. Wird die Stem-Länge m zu groß gewählt, wie $m \geq 3 \log 2n + 4$, so kann die DNA-Sequenz mehrere kleine Sekundärstrukturen aufweisen. Dies hätte zur Folge, dass bei der Verarbeitung der DNA-Sequenz mehrere Fehler verursacht würden. Wird die Länge m hingegen zu klein gewählt, beispielsweise 2 oder 3, so resultieren daraus kurze palindromische Sequenzen, die keine Sekundärstruktur darstellen.
- Die Berücksichtigung des k-Parameters ist jedoch sinnvoll, da sich ein DNA-Oligonukleotid mit begrenzter Flexibilität nicht über kleine Distanzen oder Regionen falten kann.

Um die zuvor genannten Anforderungen zu erfüllen, implementieren wir unseren (m, k)-SSA-Algorithmus für eine gegebene DNA-Sequenz der Länge n unter den folgenden Bedingungen:
$m \geq 2\log 2\ n + 2$ und $k \geq 4$
Die Parameterwahl erfolgte unter Berücksichtigung konkreter biochemischer Prozessparameter.

Funktionsweise des (m, k)-SSA

Die Funktionsweise des Algorithmus wird basierend auf die gegebenen Definitionen erklärt.
Die Methode zeichnet sich durch die Verwendung eines redundanten Symbolsystems aus, das aus einem Encoder und einem Decoder besteht. Zuerst wird eine Analyse der DNA-Sequenz, um die Einhaltung einer (m-k)-SSA-Sequenz zu bestimmen. Im Falle einer Abweichung von dieser Sequenz wird eine Verschlüsselung durchgeführt, die auf selektiver Substitution und Präfix-Hinzufügen basiert. Das Präfix enthält ein redundantes Symbol sowie verschiedene Substitutionsinformationen der gegebenen Sequenz. Durch diese Maßnahmen wird sichergestellt, dass die resultierende Sequenz fast keine umgekehrten komplementären Sequenzen enthält, die eine sekundäre Struktur bilden können. Die Codierung endet mit einer Optimierung der erzeugten Sequenzen, um die chemische Stabilität zu maximieren.

a. **SSA-Encoder**

Der Encoder minimiert Sekundärstrukturen, indem er Palindrome, Haarspangen und sehr stabile Energiestrukturen vermeidet. Es ist in drei Phasen unterteilt und wird wie im Algorithmus 1 implementiert.

Algorithmus 1: ssaencoder(dna, m, k)

Eingabe: dnaSequence (Zeichenkette), m (Stem-Länge), k (Loop-Länge)
Ausgabe: kodierte SSA-Sequenz

1. Setze x = "T" + dnaSequence
2. Solange x keine (m, k)-SSA-Sequenz ist: // Analysephase
   a) Setze t = ceil(m / 2)
// Substitution Phase
   b) Solange x eine t-komplementäre Sequenz hat:
      i) Finde die t-komplementäre Sequenz
      ii) Ersetze sie mit C-Replacement
      iii) Aktualisiere x
   c) Setze tHalb = t / 2
   d) Solange x eine X1X2-Sequenz hat:
      i) Finde die X1X2-Sequenz
      ii) Ersetze sie mit A-Replacement
      iii) Aktualisiere x
// Erweiterungsphase
   e) Füge ein Suffix hinzu: x = addSuffix("T" + dnaSequence, x)
3. Rückgabe von x

- Analysephase:

In dieser Phase wird evaluiert, ob die DNA-Sequenz bereits eine (m-k)-SSA-Sequenz ist. Der Ablauf dieser Phase ist wie folgt:
Bei einer Sequenz u ϵD^{n-1} wird das Präfix T hinzugefügt, um die Sequenz $x=Tu\ \epsilon D^n$ zu erhalten, womit der Beginn der ursprünglichen Sequenz beim Decodieren markiert wird. Im Anschluss wird mithilfe des folgenden Algorithmus 2 überprüft, ob die Sequenz x eine (m-k)-SSA-Sequenz ist. Ist dies der Fall, wird x als Ergebnis gespeichert und der Algorithmus wird beendet. Ist dies nicht der Fall, wird zur Substitutionsphase übergegangen.

- Substitutionsphase:

In dieser Phase werden Untersequenzen der DNA-Sequenz x identifiziert, die an der Bildung von Sekundärstrukturen beteiligt sein können, und es wird eine geeignete Substitution durchgeführt. Es werden hauptsächlich zwei Fälle betrachtet und je nach Fall wird eine angepasste Kodierung vorgenommen.

Fall 1: C-REPLACEMENT

Der Algorithmus analysiert die DNA-Sequenz x, um Paare von nicht überlappenden Untersequenzen y und z der Länge t zu finden, die die Bedingungen der inversen Komplementarität **z = RC(y)** und des minimalen Abstands zwischen ihren jeweiligen Startpositionen i und j, i-j genannt, erfüllen, so dass **i-j ≥ k + t** ist.

Wird ein Paar nicht überlappender Untersequenzen y und z gefunden, die die oben genannten Bedingungen erfüllen, so dass **x** $=X_1\mathbf{y}X_2\mathbf{z}X_3$, wobei X_1, X_2 und X_3 Untersequenzen von **x** sind, wird eine als **C-Replacement** bezeichnete Substitution durchgeführt. Die Durchführung erfolgt gemäß der folgenden Vorgehensweise:

$$X_1\mathbf{y}X_2\mathbf{z}X_3 \xrightarrow{j\text{-}i\text{-}t\geq k.} X_1yX_2X_3 \rightarrow CK\alpha X_1yX_2X_3$$

Das zuvor beschriebene Prinzip findet ebenfalls Anwendung, wenn die Sequenz x mehrere Sekundärstrukturen aufweist. Bei der Betrachtung von zwei Paaren von umgekehrten Komplementen y1 und z1 sowie y2 und z2 ist zu beachten, dass diese jeweils an den Positionen i1 und j1 sowie i2 und j2 beginnen.

$$X_1y_1X_2y_2X_3z_1X_4z_2X_5 \xrightarrow[j_1\text{-}i_1\text{-}t\geq k]{j_2\text{-}i_2\text{-}t\geq k.} X_1y_1\ X_2y_2X_3X_4X_5 \rightarrow CK_2\alpha_2CK_1\alpha_1X_1y_1X_2y_2X_3X_4X_5$$

<u>Erläuterung des Pointers P = CKα</u>
Dieser Zeiger fungiert als Präfix, in dem Informationen bezüglich der Größe des k-Loops und der Position des umgekehrten Komplements verschlüsselt sind.
C repräsentiert ein einzelnes Nukleotid, das das umgekehrte Komplement bezeichnet.
K ist die DNA-kodierte Darstellung der Länge der Schleife k.
α repräsentiert die Position i, an der die Sequenz y beginnt, und wurde ebenfalls mittels DNA-Kodierung dargestellt.
Die Länge von **K** und i beträgt $\mathbf{\log_4 n}$, was impliziert, dass **P** eine Länge von $\mathbf{1 + 2\log_4 n = 1 + \log_2 n}$ hat. Die C-Replacement endet, indem die Sequenz **z** von **x** entfernt und **P** an den Anfang der **x**-Sequenz angehängt wird.

Diese Codierung erhöht die Länge der x-Sequenz nicht, da die entfernte **z**-Sequenz eine Länge von $\mathbf{t = 0{,}5m = 1 + \log_2 n}$ hat und die hinzugefügte **P**-Sequenz eine Länge hat, die kleiner oder gleich der von **z** ist.

- **<u>Fall 2</u>: A-REPLACEMENT**

Es erfolgt eine weitere Analyse, um Untersequenzen in der Form $\mathbf{s = (x1x2)^{t^2}}$ zu finden, mit x1 und x2 $\epsilon\ D^n$, so **dass** $\mathbf{x = X_1 s X_2 = X_1(x1x2)^{t^2}X_2}$. Unter der Voraussetzung, dass diese Bedingung erfüllt ist, wird eine Codierung namens **A-Replacement** wie folgt durchgeführt:

$$X_1(x1x2)^{t^2}X_2 \rightarrow X_1X_2 \rightarrow Ax1x2\beta X_1X_2.$$

<u>Erklärung des Zeigers $Q = Ax1x2\beta$.</u>
A repräsentiert ein einzelnes Nukleotid, welches die Wiederholungssequenz x1x2 der Länge 2 bezeichnet. $\boldsymbol{x1x2}$ steht für die wiederholten Wörter x1 und x2.
$\boldsymbol{\beta}$ steht für die DNA-Darstellung von i, der Stelle, an der s beginnt. Die Länge von β beträgt $\mathbf{\log_4 n}$, was impliziert, dass Q eine Länge von $\mathbf{1 + 2 + \log 4n = 3 + 1/2\ \log 2n}$ hat. Die A-Replacement endet, indem die Sequenz **s** von **x** entfernt und **Q** an den Anfang der Sequenz **x** angehängt wird. Diese Kodierung reduziert die Länge der **x**-Sequenz um mehr als ein Symbol, insbesondere wenn ihre Länge $n > 2^6 = 64$ ist. Genauer gesagt um $\mathbf{\frac{1}{2}\log_2 n - 2}$, was die Differenz zwischen den Längen der hinzugefügten Sequenz Q $\mathbf{(3 + \frac{1}{2}\log_2 n - 2)}$ und der entfernten Sequenz **s** mit einer Länge von größer oder gleich $\mathbf{t\ (1 + \log 2n)}$ ist.

- Erweiterungsphase

Diese Phase findet ihren Abschluss, sobald der Algorithmus sämtliche zuvor genannte Codierungen identifiziert und wiederholt hat, sodass die resultierende Sequenz **x** keine Paare von umgekehrt komplementären Subsequenzen mehr enthält. Die Substitutionsphasen enden in vielen Fällen mit einer Verkürzung der ursprünglichen Länge n, wobei der Versuch unternommen wird, die Eigenschaften einer (m-k)-SSA-Sequenz zu erhalten.

Um die Eigenschaften einer (m-k)-SSA-Sequenz zu erhalten, ist es notwendig, die ursprüngliche Länge **n** der Sequenz **x** wiederherzustellen, indem die Differenz zur Länge **n'** der resultierenden Sequenz, die mit **ns = n-n'** bezeichnet wird, überbrückt wird. Dies wird durch die Hinzufügung eines Suffixes realisiert, welcher die Bedingungen einer (m-k)-SSA-Sequenz gewährleistet und gleichzeitig die Integrität und Effizienz der vorliegenden Codierung konserviert.
Bei einer gerade Anzahl **ns** des Suffixes wird $\mathbf{r = (GT)^{ns/2}}$ an das Ende der aktuellen Sequenz **x** angefügt, andernfalls wird $\mathbf{r = (GT)^{ns-1/2}G}$ hinzugefügt.
Der Algorithmus eliminiert schließlich alle bestehenden oder während des Prozesses gebildeten Sekundärstrukturen, da die Substitutionsoperationen so lange wiederholt werden, bis die Sequenz x die Bedingungen für eine (m-k)-SSA-Sequenz erfüllt.

b. SSA-Decoder

Der Decoder rekonstruiert die ursprüngliche Sequenz der vom Encoder codierten Sequenz.
Am Ende jedes Decodierungsprozesses folgt eine Phase der Verifizierung und Korrektur von Fehlern, die während des Substitutionsprozesses aufgetreten sind.
Die Decodierung von (m-k)-SSA basiert hauptsächlich auf der Identifizierung des ersten Symbols der verarbeiteten x-Sequenz. Anhand dessen kann der Substitutionsfall bestimmt werden und die ursprüngliche Sequenz kann anhand des Zeigers rekonstruiert werden. Der Ablauf des Algorithmus wird explizit in dem untenstehenden Algorithmus 2 erklärt wird.

Es sei eine codierte Sequenz $\mathbf{x} \in D^n$, mit **x = Enc(u)** gegeben.

- Fall 1: Das erste Symbol ist T.

In diesem Fall ist **x = Tu**, wobei **u** eine Länge von **n-1** hat. Der Decodierungsprozess endet mit der Identifizierung und Extraktion von **u**, genauer gesagt der nächsten **(n-1)** Symbole, die als die ursprüngliche DNA-Sequenz der Nachricht betrachtet werden. Diese Sequenz wird dann vom Decoder ausgegeben.

- Fall 2: Das erste Symbol ist C.

Der Decoder extrahiert die ersten $\mathbf{1 + \log_2 n}$ Symbole, die als der Zeiger **P** betrachtet werden, der nach dem ersten Fall der Substitution (**C-Replacement**) hinzugefügt wurde. Da **P** in der Form $\mathbf{P= CK\alpha}$ vorliegt, berechnet der Decoder die numerischen Werte der Loop-Länge **k** sowie der Position **i** von **y** aus ihren DNA-kodierten Darstellungen **K** und $\boldsymbol{\alpha}$. K und $\boldsymbol{\alpha}$ werden aus dem Zeiger **P** anhand ihrer Länge, die jeweils $\mathbf{\log_4 n}$ beträgt, extrahiert.

Der Decoder entfernt das Präfix **P** aus der **x**-Sequenz und berechnet anschließend das inverse Komplement **z** von **y**, bevor er es an der Position **i + k + t** in der **x**-Sequenz einfügt.

$CK\alpha X_1 y X_2 X_3 \rightarrow X_1 y X_2 X_3 \rightarrow X_1 y X_2 z X_3$

- Fall 3: Das erste Symbol ist A.

Der Decoder extrahiert die ersten $\mathbf{3 + 0{,}5\log_2 n}$ Symbole, die als der nach dem zweiten Substitutionsfall (A-Replacement) hinzugefügte **Q**-Zeiger betrachtet werden. Da **Q** in der Form $\mathbf{Q} = \boldsymbol{Ax1x2\beta}$ vorliegt, extrahiert der Decoder **x1** und **x2**, die die Symbole sind, aus denen die Wiederholung der Länge 2 besteht. Daraufhin wird die Startposition **j** berechnet, an der die Wiederholungssequenz $\mathbf{s= (x1x2)^{t/2}}$ begonnen hat, aus ihrer DNA-codierten Darstellung, die $\boldsymbol{\beta}$ ist. $\boldsymbol{\beta}$ wird aus dem Zeiger **Q** aus seiner Länge extrahiert, die $\mathbf{\log_4 n}$ beträgt. Daraufhin entfernt der Decoder das **Q-Präfix** aus der **x**-Sequenz und fügt $\mathbf{(x1x2)^{t/2}}$ an der Position **j** in der **x**-Sequenz hinzu.

$Ax1x2\beta X_1X_2 \rightarrow X1X2 \rightarrow X_1(x1x2)^{t/2}X2.$

-Fall 4: Das erste Symbol ist G.

Der Decoder meldet einen Fehlerfall, da G nicht das erste Symbol gemäß unserer Kodierung sein sollte. In der Folge werden Korrekturcodes erstellt, um diesen Fehler zu beheben.

Die Dekodierung wird als abgeschlossen betrachtet, wenn Fall 1 nach mehreren Scans und Substitutionen erreicht wird und das Resultat als die Daten der ursprünglichen Nachricht interpretiert werden kann

Algorithmus 2: ssadecoder(dna, m, k)
Eingabe: dnaSequence (Zeichenkette), m (Stem-Länge), k (Loop-Länge)
Ausgabe: dekodierte DNA-Sequenz
1. Falls m < 2 * log(n + 2) oder k < 4, werfe eine Ausnahme ("Ungültige Parameter")
2. Setze x = dnaSequence
3. Falls x mit 'T' beginnt:
   a) Entferne das erste Zeichen und gib die verbleibende Sequenz zurück
4. Falls x mit 'C' beginnt:
   a) Extrahiere Präfix p mit Länge $\mathbf{1 + \log_2 n}$ // P=CK$\boldsymbol{\alpha}$
   b) Berechne K aus einem Teil von p
   c) Berechne $\boldsymbol{\alpha}$ aus einem anderen Teil von p
   d) Berechne t = m / 2
   e) Berechne j = $\boldsymbol{\alpha}$ + K+ t
   f) Entferne p aus x
   g) Füge das Reverse Komplement von x[$\boldsymbol{\alpha}$: $\boldsymbol{\alpha}$ +t] an Position j ein
   h) Setze x auf die aktualisierte Sequenz
5. Falls x mit 'A' beginnt:
   a) Extrahiere Präfix q mit Länge 3 + 0.5 * log(n) in Basis 2 // q= $\boldsymbol{Ax1x2\beta}$
   b) Extrahiere x1x2 aus q
   c) Berechne $\boldsymbol{\beta}$ aus dem Rest von q
   d) Berechne t = m / 2
   e) Entferne q aus x
   f) Füge x1x2 t/2-mal an Position $\boldsymbol{\beta}$ ein
   g) Setze x auf die aktualisierte Sequenz
6. Falls x mit 'G' beginnt:
   a) Werfe eine Ausnahme ("Ungültige DNA-Sequenz, beginnt mit G")
7. Rufe rekursiv ssadecoder(x, m, k) auf, um weiter zu dekodieren
8. Rückgabe von x

4.2.2 Permutation Coder

Der DNA-Sequenz-Permutationsalgorithmus, auch als Permutation Coder bezeichnet, ist das Resultat der gemeinsamen Forschungsarbeit von Dozent Alex El-Shaikh und Dr. Prof. Bernhard Seeger von dem Fachbereich Mathematik und Informatik der Universität Marburg, Deutschland. [32]

Das Ziel des Algorithmus besteht darin, den Anteil der in einer DNA-Sequenz vorhandenen Homopolymere zu verringern und gleichzeitig durch die Kontrolle der GC Gehalt die Entstehung von Strukturen zu verhindern, die bei der Speicherung von Informationen in der DNA Fehler verursachen können. In unserem Fall bilden sich Sekundärstrukturen. Durch eine gezielte Permutation der Sequenzabschnitte wird die Homogenität aufgebrochen, wodurch die Wahrscheinlichkeit der Ausbildung solcher Strukturen signifikant reduziert wird, zusammen mit den Sequenzierungsfehlern. Dazu wird gezeigt, dass der Algorithmus die Integrität der Informationsspeicherung verbessert.

Konkret beruht der Algorithmus auf der Anwendung des Fisher-Yates-Algorithmus, um verschiedene Permutationen einer DNA-Sequenz zu erzeugen, die zu einer zufälligen Anordnung der Basen führen. Mittels eines Zufallszahlengenerators (RNG) werden k-1 zufällige Indexpaare für eine Sequenz der Größe k erzeugt, die anschließend innerhalb der Sequenz ausgetauscht werden. Die Zufälligkeit dieser Permutationen wird durch einen "Seed" kontrolliert, der von der Startsequenz abgeleitet wird.

Der Algorithmus vereint Randomisierung, Seed und Jaccard-Distanz-basierte Optimierung, um DNA-Sequenzen zu generieren, die den Beschränkungen einer DNA-Sequenz entsprechen. Die Permutationen ermöglichen den Ausgleich der GC-Rate einer Sequenz sowie die Auflösung vorhandener Homopolymerketten, indem lange, aufeinanderfolgende Wiederholungen derselben Base neu verknüpft werden. Darüber hinaus wird eine Differenzierung der permutierten Sequenzen mit Hilfe der Jaccard-Distanz gewährleistet. Somit wird der Permutation Coder als ein randomisiert-konstruktiver Hybridalgorithmus mit starkem Fokus auf Constraint-Satisfaktion betrachtet .

Um eine permutierte Sequenz zurückzusetzen, wird der Offset entfernt und dann die gleiche RNG-Initialisierung genutzt. Anschließend werden die Tauschpaare in umgekehrter Reihenfolge angewendet.

Eine ausführlichere Erklärung findet sich in der Publikation „El-Shaikh, A., & Seeger, B. (2023). Content-based filter queries on DNA data storage systems. *Scientific Reports*, *13*(1), 7053“.

Die Forschungsarbeit basiert auf einer Java-Umsetzung, die die Theorie des Permutationsalgorithmus praktisch bestätigt und ist auf GitHub verfügbar.

Der Prozess des Kodierers wird im Algorithmus 3 zusammengefasst.

Algorithmus 3 : Permutation Coder(q, m):

Eingabe: DNA-Sequenz q, Anzahl der Permutationen m
Ausgabe: Permutierte Sequenz, die alle Bedingungen erfüllt

1. Berechne den Seed der Sequenz q:
   seed_q = |q|_A * |q|_C * |q|_T * |q|_G

2. Initialisiere beste_permutation = None
   Initialisiere max_jaccard = -1

3. Für i = 0 bis m - 1:
   a) Initialisiere RNG mit (seed_q + i)
   b) Erzeuge eine Permutation $\hat{q}$ von q mit Fisher-Yates:
      - Erzeuge k-1 zufällige Tauschpaare
      - Wende sie auf q an
   c) Hänge Offset i an $\hat{q}$ an

   d) Überprüfe Homopolymer-Bedingung:
      - Falls $\hat{q}$ Homopolymere enthält, fahre mit nächster Iteration fort

   e) Berechne die Jaccard-Distanz zwischen q und $\hat{q}$:
      jaccard_dist = Jaccard(q, $\hat{q}$)

   f) Falls jaccard_dist > max_jaccard:
      - Setze beste_permutation = $\hat{q}$
      - Setze max_jaccard = jaccard_dist

4. Falls keine gültige Permutation gefunden wurde:
   a) Falls möglich, erhöhe m und wiederhole Algorithmus
   b) Andernfalls passe den Payload-Encoder an

5. Rückgabe beste_permutation

Um die Effektivität des (m, k)-Algorithmus und des Permutationskodierers effektiv zu bewerten, wird sich unsere Analyse auf verschiedene DNA-Sequenzen stützen, die aus der Datenbank erstellt wurden, die in Java von Dozent Alex El Shaikh und Dr. Prof. Bernhard Seeger eingerichtet wurde.

Da eine öffentliche DNA-Datenbank, mit der man die Sekundärstruktur der DNA testen oder vorhersagen kann, noch nicht verfügbar ist, werden unsere Ergebnisse aus der Anwendung unserer Methoden auf DNA-Sequenzen stammen, die nach verschiedenen Kriterien erstellt wurden, wie z. B. Länge und GC-Basenanteil.

Eine Analyse und ein Vergleich der erhaltenen Ergebnisse wird gemäß den in 4.1 festgelegten Kriterien durchgeführt, um die Vor- und Nachteile jedes Verfahrens zu bestimmen und Vorschläge für Verbesserungen zu generieren oder auch um ihren Anwendungsfall zu definieren.

5.1 Analyse

a. Berechnungszeit/Komplexität

Der (m-k)-SSA-Algorithmus ist dahingehend spezifisch, dass die Zeit, die für die Kodierung und Dekodierung einer bestimmten Sequenz erforderlich ist, maßgeblich von den angewandten Substitutions- und Analysetechniken abhängt. Diese dienen dazu, die Authentizität einer (m-k)-SSA-Sequenz zu verifizieren. Die Komplexität kann, je nach Implementierung und verwendeter Programmiersprache, zwischen O(n) und $O(n^2)$ variieren, d. h. von linearer Einfachheit bis hin zu quadratischer Komplexität.

Gemäß dem vorliegenden Pseudocode ist die zeitliche Komplexität des Encoders und Decoders, die mit einem Codewort der Länge n verbunden ist, proportional zu n. Diese Schlussfolgerung basiert auf der Analyse der Substitutionsprozesse der Teilsequenzen in der Ersetzungsphase, deren Anzahl n-m nicht übersteigt. Darüber hinaus erfordern sekundäre Methoden, wie die Berechnung und das Inkrementieren des Präfixes, die Umwandlung einer DNA-Sequenz in eine quaternäre Darstellung und umgekehrt, eine konstante zeitliche Komplexität von O(1). Dies hat keinen Einfluss auf die allgemeine lineare Komplexität des (m-k)-Algorithmus. Es wird jedoch darauf hingewiesen, dass umfangreichere Strukturen durch kompetente und geschickte Verarbeitung erleichtert werden und somit eine effiziente Datenspeicherung und -abfrage in großem Umfang begünstigt werden.

Wie bereits erwähnt, wird in unserer Arbeit die Programmiersprache Java verwendet, die aufgrund der verwendeten Methoden und Strukturen eine höhere zeitliche Komplexität aufweist. Folglich hat der Encoder eine Komplexität von $O(n^2)$, während unser Decoder eine Komplexität von $O(\mathbf{nlog_2}\boldsymbol{n})$ aufweist.

Eine ähnliche Situation zeigt sich beim Permutations-Encoder, bei dem die effektive Ausführungszeit je nach verwendeter Hardware und JVM-Implementierung variiert. Der Algorithmus, welcher die effektive Ausführungszeit maßgeblich beeinflusst, führt abhängig von den gegebenen Parametern, wie der Anzahl der Permutationen, zu einer hohen Anzahl an erforderlichen Operationen. Die Analyse des Pseudocodes, unterstützt durch die in Java durchgeführte Implementierung, liefert folgende Informationen. Es konnte festgestellt werden, dass jede Permutation und jede Bewertung der Punktzahl eine Komplexität von O(n) aufweist, was für k Permutationen in O(k – n) übersetzt werden kann. Die Suche nach der höchsten Punktzahl dauert O(k), was bedeutet, dass die gesamte zeitliche Komplexität des Algorithmus auf O(k – n) geschätzt wird.

Die Gegenüberstellung der Ausführungszeiten, siehe Abbildung 6 zeigt, dass der SSA Coder deutlich weniger Reaktionszeit benötigt als der Permutation Coder, um eine bestimmte DNA-Sequenz zu codieren.

Die Analyse der Leistungsbewertung aus der Abbildung 6 ergibt, dass :
Der (m-k)-SSA-Coder zeigt eine nahezu lineare Entwicklung (≈ O(n log n)) und erhält seine Effizienz auch bei 2048 Basen (8.971 Millisekunden).
Demgegenüber leidet der Permutations-Coder unter einer exponentiellen Verschlechterung der Leistung (≈ O(n^3)) und weist eine extreme Verlangsamung bei n=2048 (320290 Millisekunden) auf.
Der SSA-Coder ist daher besonders für die Echtzeitverarbeitung geeignet und fördert seinen Einsatz bei großen Datenmengen. Der Permutation Coder benötigt wesentlich mehr Zeit, was für große Datensätze mit mehr als 500 Datenbanken nicht geeignet ist.

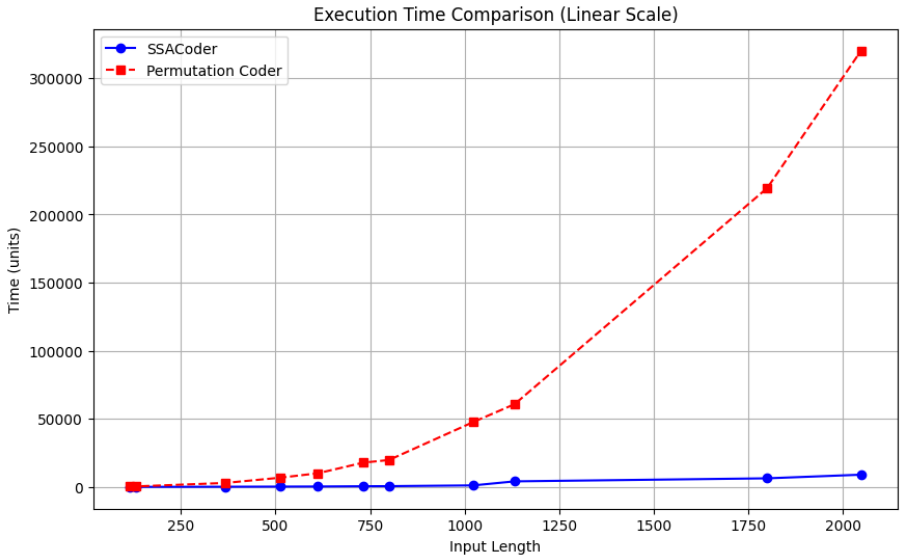


Abbildung 6: Laufzeit der SSA Coder und der Permutation Coder

b. Effizienz in der Vermeidung von Sekundärstrukturen

○ Sekundärstrukturindex

Seine Formel lautet wie folgt:

$$\boldsymbol{I} = \left(\frac{\boldsymbol{GC\ Content}}{\boldsymbol{n}}\right) * \left(\frac{\boldsymbol{actualRC}}{\boldsymbol{maxPossibleRC}}\right) * \left(\frac{\boldsymbol{m}}{\boldsymbol{m+k}}\right)$$

n steht für die Länge der DNA-Sequenz, m für die Länge der doppelsträngigen Stammoberfläche und k für die Länge der einzelsträngigen Schleife.
GC Content ist der GC-Gehalt: Die Anzahl von Guanin (G) und Cytosin in der DNA-Sequenz. Ein höherer GC-Gehalt kann auf stabilere Sekundärstrukturen hinweisen.
(actualRC / maxPossibleRC): Gibt das Verhältnis zwischen der Anzahl der in der Sequenz gefundenen invers komplementären Sequenzen (z. B. Haarnadelschleifen) und der maximal möglichen invers komplementären Sequenzen an, die eine bestimmte Sequenz aufweisen kann. Eine höhere Anzahl kann zu einer höheren Wahrscheinlichkeit der Bildung einer größeren Sekundärstruktur führen.
Stem / Loop-Verhältnis (m/(k + m)): Dieses Verhältnis gleicht den Einfluss der Parameter m und k aus und sorgt gleichzeitig für eine Normalisierung zwischen 0 und 1.
Ein Indexwert, der näher an 0 liegt, zeigt eine geringere Tendenz zur Bildung einer Sekundärstruktur.

Die vorliegende Untersuchung, siehe Abbildung 7 wird mit Sequenzen, die Indizes zwischen 0,011 und 0,129 haben, durchgeführt. Die Ergebnisse nach der Kodierung zeigen, dass der(m-k)-SSA-Kodierer (grün) eine konstante Leistung aufweist mit Indizes, immer niedriger als die der originalen Sequenzen (Rückgang des Index von 0,117 (Original) auf 0,022 (-81 %)). Bei längeren Sequenzen zeigt sich eine signifikant höhere Leistung, bei kurzen Sequenzen mit einem GC-Anteil von bis zu 50 % eine geringere Effizienz.
Der Permutationskodierer (rot) reduziert den Index aktiv, was zu einer marginalen Verbesserung führt (von 0,125 (Original) auf 0,062 (-50%) bei einer Länge von 1932 Basen).

Der Permutationskodierer (rot) reduziert den Index aktiv, was zu einer marginalen Verbesserung führt (von 0,125 (Original) auf 0,062 (-50 %) bei einer Länge von 1932 Basen). Bemerkenswert ist seine Leistung auch bei kurzen Sequenzen mit einem GC-Anteil von 50 % oder weniger, wo er sogar (m, k)-SSA bei Längen von 115, 512 und 800 übertrifft. In der Folge wird die Eignung für einfache/wenig komplexe und kurze Sequenzen deutlich. Es ist zu beachten, dass die Ergebnisse des Permutationskodierers nach vier Permutationen erzielt wurden. Es ist anzunehmen, dass sich die Ergebnisse bei einer höheren Anzahl von Permutationen ändern, was jedoch einen deutlich längeren Zeitraum für die Ausführung in Anspruch nehmen würde. Dieser Aspekt ist jedoch erwünscht, da die Ausführungsgeschwindigkeit im Prozess der Speicherung in der DNA als wesentlicher Faktor betrachtet wird.

Zusammenfassend lässt sich festhalten, dass zur Minimierung sekundärer Strukturen der Einsatz des SSA-Encoders vorteilhaft als der permutationsbasierte Ansatz ist, da dieser in der Regel eine konsistente Leistung aufweist, selbst bei stark strukturierten Sequenzen. Der Permutationskodierer führt eine besondere Optimierung des Index durch, ohne dabei die GC-Rate zu verringern.

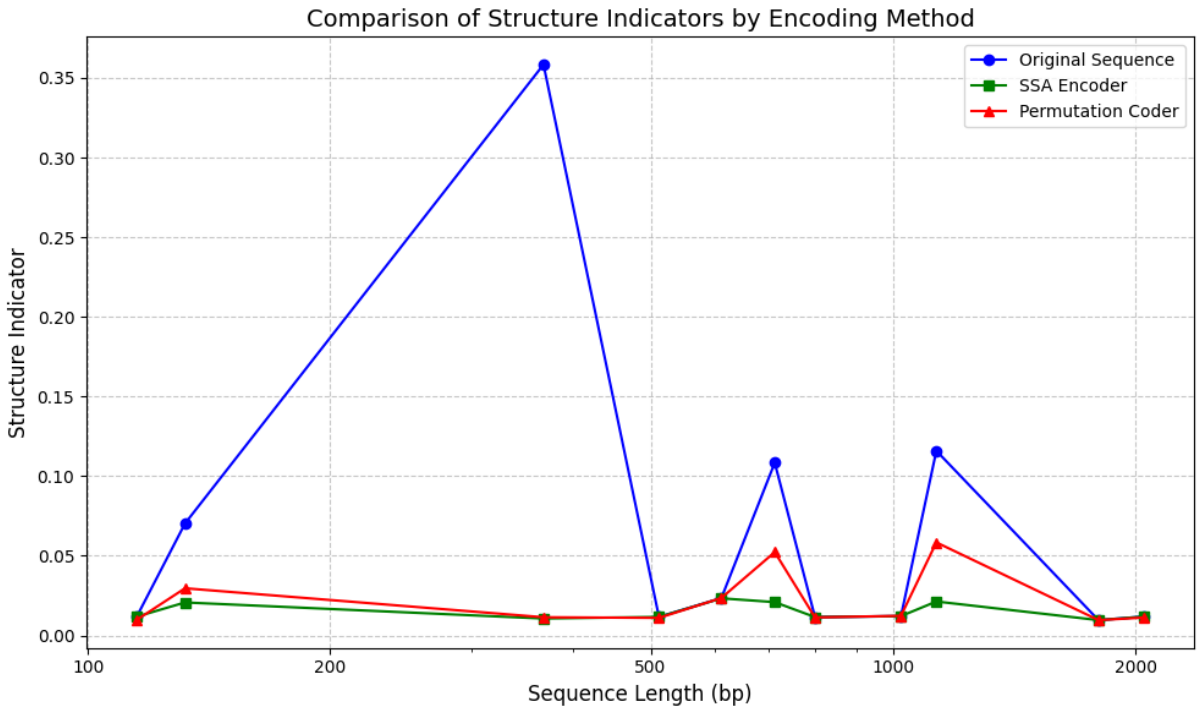


Abbildung 7: Sekundärstrukturindex von SSA Coder und Permutation Coder nach der Kodierung

- **RC Rate Fehlerfunktion**

Diese Funktion dient dazu, die Verringerung der inversen Komplementpaare nach der Codierung gemäß dem unten genannten Algorithmus zu messen und wird in Java ausgeführt. Die Funktion steht auf GitHub https://github.com/alexelshaikh/dna/tree/sadio. Die durchgeführten Tests haben eindeutig auf eine vollständige Entfernung von inversen Komplementsequenzen mit einer Länge von m >= 2*logn + 2 hingewiesen. Dies zeigt die Wirksamkeit von Algorithmen, die die Verringerung oder sogar Verhinderung der Bildung von Sekundärstrukturen erleichtern.

- **Freie Energie von Gibbs (ΔG)**

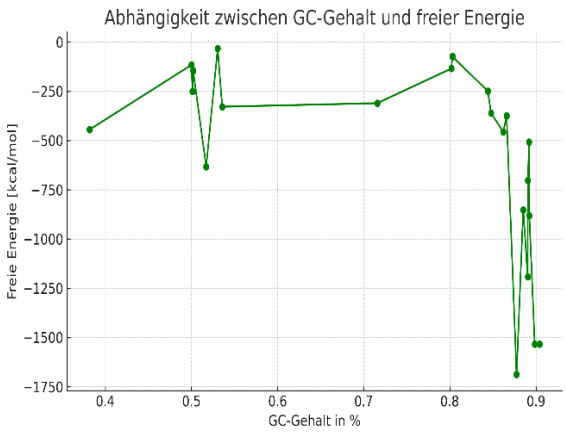
Der online RNAFold-Mechanismus [54] wurde verwendet, um die freie Energie von Gibbs ΔG für jede Sequenz vor und nach der Kodierung berechnen zu können. Siehe Abbildung 9

Die verwendeten Originalsequenzen wiesen eine ausgeprägte Stabilität in der Sekundärstruktur auf, mit freien Energien, die zwischen ΔG: -41.1 und -3195.1 kcal/mol schwankten. .

Es sei darauf hingewiesen, dass eine hohe Korrelation zwischen der GC-Rate und der freien Energie einer Sequenz besteht, siehe Abbildung 8 ersichtlich. Je höher der GC Gehalt, desto stärker die Stabilität der Struktur. Somi

Der (m-k)-SSA-Codierer reduziert die Stabilität der freien Energie der gegebenen Sequenzen erheblich, indem er Werte erzielt, die näher an 0 als die Originalsequenz liegen, in der Regel zwischen 50 und 70%. Dadurch erhöht er die Instabilität der Sekundärstrukturen in den Sequenzen.

Der Permutationskodierer wirkt in ähnlicher Weise, jedoch mit einer weniger signifikanten Energieabnahme als der (m-k)-SSA.



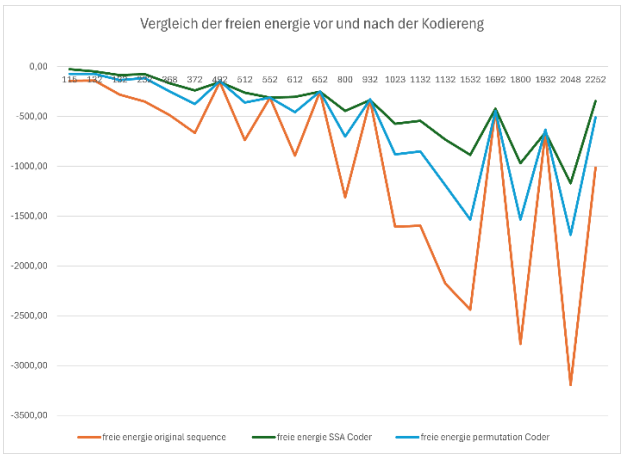


Abbildung 8: Abhängigkeit zwischen GC Gehalt und freier Energie

Abbildung 9: Vergleich der freien Energie vor und nach der Kodierung

c. Auswirkungen auf die Fehlerrate bei der Sequenzierung

○ Analyse von Homopolymeren

Die Untersuchung von Homopolymeren ist von essentieller Bedeutung, da ihre Existenz ein entscheidendes Element darstellt, das die Entstehung von Sekundärstrukturen und auch Anomalien bei der Speicherung von Informationen in der DNA auslösen kann. Diese Analyse darf nicht außer Acht gelassen werden, wenn berücksichtigt wird, dass während der Substitutionsphase des (m-k) -Kodierers Operationen wie das Hinzufügen von Präfixen und das Löschen von Teilsequenzen durchgeführt werden. Die vom Permutationskodierer vorgenommenen Permutationen können die Struktur der verarbeiteten Sequenz verändern und folglich zum Auftreten neuer Homopolymere führen.

Im Rahmen der vorliegenden Studie wurden diverse Simulationen durchgeführt, die auf dem nachfolgend genannten Algorithmus basieren. Mithilfe der Monte-Carlo-Simulation konnte die Wahrscheinlichkeit der Entstehung neuer Homopolymere nach der Kodierung abgeschätzt werden.

Die Ergebnisse des Encoders (m-k) , siehe Abbildung 10, Abbildung 11 zeigen, dass die Wahrscheinlichkeit, dass Homopolymere eine Länge von 6 Basen überschreiten, praktisch null ist. Die Wahrscheinlichkeit des Auftretens von Homopolymeren mit 4 Basen bleibt sehr gering, aber quantifizierbar (1,5–2,0 %). Kürzere Sequenzen (64/128 Basen) bergen ein etwas höheres Risiko (2,0 %) im Vergleich zu längeren Sequenzen (1,5 % bei 255/512 Basen). Längere Abfolgen von mehr als 255 Basen werden etwas besser minimiert als kürzere. Wenn eine vollständige Vermeidung von Homopolymeren unerlässlich ist, können Sequenzen mit 512 Basen aufgrund ihres ausgewogenen Verhältnisses zwischen Speicherkapazität und Fehlerkorrektur bevorzugt werden.

Die Permutationscodierung zeigt eine effektive Reduktion von 4-mers, kann deren Häufigkeit jedoch nicht unter etwa 48 % senken, wie es auf der Abbildung 10 zu sehen ist. Zudem scheint sie das Auftreten von Homopolymeren der Länge 8 zu verhindern, deren Auftreten mit zunehmender Länge der ursprünglichen Sequenz unter 5 % liegt. Der Defekt bei mittleren Längen (6-mer) wird jedoch bei längeren DNA-Sequenzen zunehmend bedeutsamer.
Die Resultate der durchgeführten Tests demonstrieren eindeutig eine geringe Wahrscheinlichkeit für die Bildung von Homopolymeren nach jeder (m-k) -Codierung. Bei Verwendung des Permutation-Codierers ist jedoch eine erhöhte Wahrscheinlichkeit für dieses Risiko zu beobachten, wenn gleich diese bei langen Homopolymersequenzen geringer ist. Bei mittellangen Sequenzen bis zu 6 Basen ist diese Wahrscheinlichkeit dagegen etwas höher.

Dies verdeutlicht die Effizienz des (m-k) -Algorithmus-Codierers, der das Auftreten neuer Homopolymere verhindert, die bei der Sequenzierung zu Fehlern führen könnten.

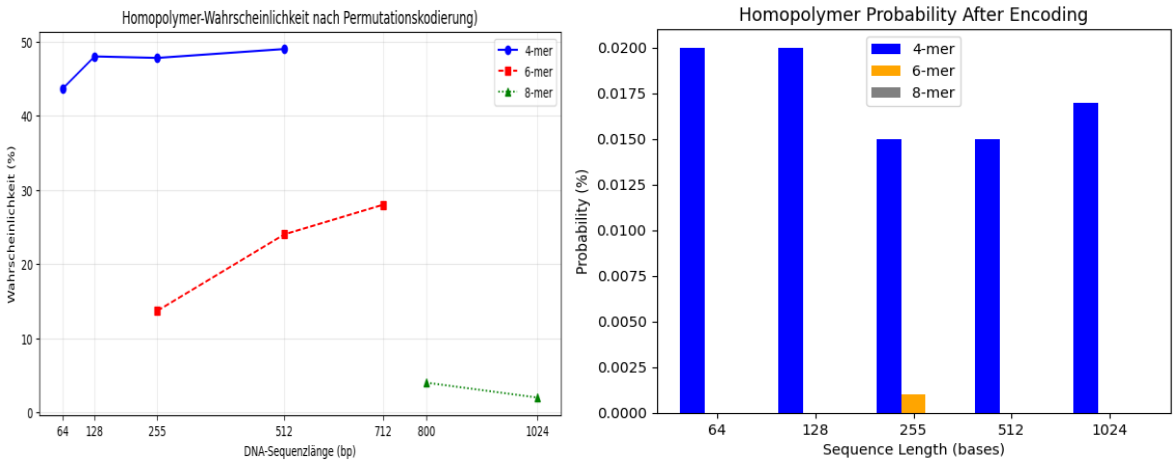


Abbildung 10: Wahrscheinlichkeit der Bildung neuer Homopolymeren nach Permutation

Abbildung 11: Wahrscheinlichkeit der Bildung neuer Homopolymeren nach (m-k) -SSA Kodierung

Algorithmus 4: simulateHomopolymerProbability (N, L, m, k, trials, encoder)
Eingabe: Lange der Sequenz N, läange des homopolymers L, Stem Länge m, Loop Länge k, Anzahl der versuche trials, encoder instanz
Ausgabe: Geschätzte Wahrscheinlichkeit, dass es mehr Homopolymer der Länge L
in einer DNA-Sequenz der Länge N nach der Kodierung generiert wurde

```
count = 0
FOR i = 1 TO trials:
   // Generiere eine zufällige DNA-Sequenz der Länge N
   dna = generateRandomDNA(N)

   // Zähle die Homopolymere der Länge L in der ursprünglichen DNA
   count1 = countHomopolymers (dna, L)
   IF encoder is SSAEncoder
   // Kodiere die DANN-Sequenz mit den Parametern m und k
       dnamk = SSAEncoder (dna, m, k)
  IF encoder is SSAEncoder
   // Kodiere die DANN-Sequenz mit den Parametern m und k
     dnamk = PermutationCoder (dna, k)

   // Zähle die Homopolymere der Länge L in der kodierten DNA
   count2 = countHomopolymers (dnamk, L)

   // Vergleiche die Anzahl der Homopolymere
   IF count1 < count2:
     count = count + 1

// Berechne die geschätzte Wahrscheinlichkeit
estimatedProbability = count / trials

// Gib das Ergebnis als String zurück
RETURN estimatedProbability
```

- **GC-Content Analyse**

Zur Durchführung der Studie über den Anteil der GC-Basen werden Sequenzen mit einem GC-Anteil zwischen 38 % und 89 % ausgewählt, um die Auswirkungen der verschiedenen Algorithmen auf diese zu untersuchen.

Das Diagramm, siehe Abbildung 12 zeigt, dass der SSA-Encoder auf eine signifikante Reduktion der durchschnittlichen GC-Rate zusteuert (maximale Reduktion: 69 % → 55 % bei einer Länge von 1976 Basen). Seine besondere Leistungsfähigkeit zeigt sich insbesondere bei langen Sequenzen oder solchen, die eine Rate von mehr als 55 % aufweisen.

Der Permutationskodierer zeigt eine Tendenz, das GC-Niveau beizubehalten, und führt nur selten eine Verringerung durch (mit einer maximalen Reduktion von 18,0 % bei 115 bp).

In der Gesamtschau tragen beide Algorithmen zur Reduzierung von Sequenzierungsfehlern bei, indem sie den GC-Wert bei Bedarf senken, wenn er zu hoch ist, oder ihn beibehalten, um einen Datenverlust in einer bestimmten Sequenz zu verhindern.

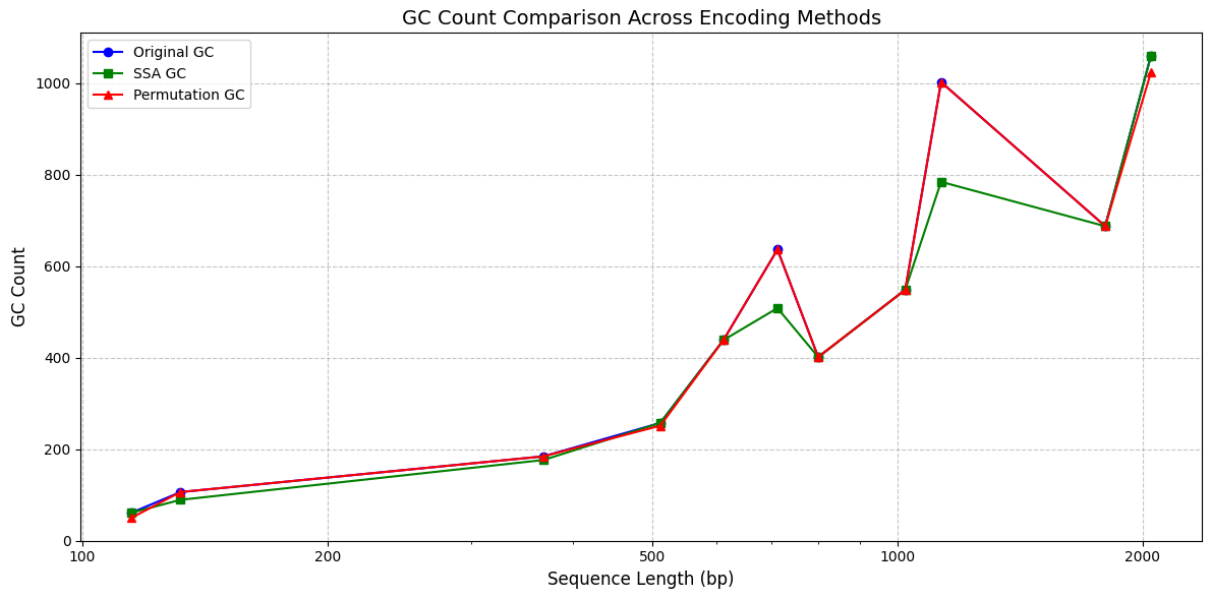


Abbildung 12: Vergleich der GC-Content nach der Kodierung mit SSA Coder und Permutation

d. Robustheit bei variierenden Bedingungen

Die Tests belegen, dass der SSA Coder stake und konstante Leistung liefert, unabhängig von verschiedenen Parametern wie der Länge der zu codierende Sequenz, dem Wert von Stem und Loop oder dem GC-Anteil.
Insbesondere bei der effizienten Entfernung komplementärer Sequenzen und der Reduzierung des GC-Gehalts in Sequenzen mit einem zu hohen GC-Gehalt spielt der Algorithmus eine wesentliche Rolle.
Die Robustheit gegenüber Fehlern wurde in Tests in Java evaluiert.
Dabei wurde festgestellt, dass die gleichen Ergebnisse für die gleiche Sequenz nach der Kodierung erhalten werden. Außerdem werden Fehlern wie einer falschen DNA-Sequenz oder bei falscher Auswahl der Parameter m und k mittels Exceptions aufgehoben.

Darüber hinaus zeichnet es sich durch eine schnellere Entwicklung aus (geringere Anfälligkeit für Wartezeiten) und ist in der Lage, sich an umfangreiche Daten anzupassen. Infolgedessen eignet sich das Verfahren als effektives Instrument im Kodierungsprozess und zur Prävention der Bildung von Sekundärstrukturen.

Der Permutationskodierer steuert die Konzentration von GC in verschiedenen Sequenzen effizient und induziert nicht mehr Homopolymere, die unerwünschte Falten bilden könnten, die die DNA-Sequenzierung erschweren. Es wird jedoch empfohlen, den Permutationskodierer vorzugsweise für kurze Sequenzen oder Sequenzen einzusetzen, bei denen die Empfindlichkeit gegenüber der GC-Konzentration entscheidend ist, um eine Zerstörung oder Veränderung der gespeicherten Daten zu vermeiden.

Der Algorithmus ist mit einem gravierenden Zeitproblem verbunden. Der benötigte Zeitrahmen während der Codierungen signifikant ist hoch und nimmt mit der Länge der Sequenz zu. Dies könnte auf die Anzahl der Operationen zurückzuführen sein, die bei den verschiedenen Permutationen durchgeführt werden. Eine Optimierung des Verfahrens oder die Codierung in einer anderen Sprache könnten jedoch zu einer Verbesserung der Situation führen.

5.2 Vor und Nachteile der Algorithmen

Auf Basis der Resultate, die jeder Algorithmus in den vorangegangenen Tests erzielt hat, lassen sich die folgenden Schlussfolgerungen bezüglich ihrer Vor- und Nachteile ableiten:

- **Der (m-k) -Algorithmus**

Es handelt sich um ein Werkzeug, das sich perfekt für den Prozess der Speicherung in der DNA eignet und mehrere Vorteile bietet.

+ Skalierbarkeit:

Der Kodierer bietet eine recht stabile Leistung in Abhängigkeit von den verschiedenen ausgewählten Sequenzen, mit einer fast linearen Ausführungszeit (≈O (n log n)), selbst bei sehr großen Sequenzen. Darüber hinaus weist er eine bemerkenswerte Resilienz gegenüber Zeitverlusten bei großen Datenmengen auf, wie beispielsweise eine Sequenz mit 2048 Basen, die lediglich 8,971 Millisekunden benötigt. Seine Schnelligkeit könnte ihn für genomische Datenströme oder skalierbare Stapelverarbeitung nützlich machen.

+ Effektive Vermeidung von Sekundärstrukturen:

Dies wird durch Experimente wie die Analyse von Indizes für Sekundärstrukturen, die Berechnung der GC-Rate oder auch die Bewertung der freien Energie nach der Kodierung deutlich. Der Kodierer trägt signifikant zur Verringerung der Wahrscheinlichkeit für ungewollte Falten in der DNA bei. Selbst bei langen Sequenzen zeigt der Kodierer einen stabilen und niedrigen Strukturindikator (I) und begrenzt gleichzeitig den Anteil an GC-Inhalten.

Der Algorithmus gewährleistet zudem, dass umgekehrte Komplementpaare, welche die Kriterien Stem m und Loop k erfüllen, nahezu vollständig eliminiert werden (RC-Rate nahezu 100 %), wodurch das Risiko von Sekundärstrukturen vollständig eliminiert wird.

Die starke Reduzierung des GC-Gehalts durch den Algorithmus trägt ebenfalls dazu bei, Fehler bei der DNA-Sequenzierung zu verhindern.

+ Robustheit

Die Robustheit des Algorithmus zeigt sich in der konstanten Leistung mit einheitlichen Ergebnissen unter verschiedenen Bedingungen (Länge, GC-Inhalt). Darüber hinaus ist die Wahrscheinlichkeit, dass neue Homopolymere mit unterschiedlichen Längen produziert werden, so gering, dass der Algorithmus als effizient und robust eingestuft werden kann.

Trotz der evidenten Stärken des Verfahrens sind auch einige Mängel zu konstatieren.

- Eine zu starke Reduktion des GC-Gehalts könnte zu einer Beeinträchtigung biologischer Muster führen und somit die Qualität der aufgezeichneten Daten verschlechtern. Aus diesem Grund wird empfohlen, sie für lange, AT-reiche Sequenzen zu verwenden.
- Andererseits stellen kurze Sequenzen eine Herausforderung für ihn dar, da die Codierung nur bei Sequenzen funktioniert, die inverse Komplemente mit einer Länge von m>= 2logn + 2 und mit einem Mindestabstand von k besitzen. In der Folge kommt es häufig vor, dass bestimmte Sequenzen nicht codiert werden können, auch wenn sie das Risiko unerwünschter Faltenbildung bergen. Dies ist ein Indikator für eine gewisse Einschränkung der Flexibilität des Algorithmus.

- **Permutation Kodierer**

  + GC-Erhalt

Der größte Vorteil der Nutzung der Permutation ist seine Fähigkeit, die Anzahl der GCs in einer zu codierenden Sequenz zu erhalten. Die meisten Tests zeigen, dass die GC-Rate erhalten bleibt, während die Anzahl der GCs nur selten signifikant abnimmt. Der Encoder ist daher ein ideales Werkzeug für die Erhaltung biologischer Strukturen und für Anwendungen im Zusammenhang mit GC.

  + Effektivität mit kurzen Sequenzen

Darüber hinaus erweist er sich bei kurzen Sequenzen als effektiv, wie der in den Experimenten erzielte Strukturindex zeigt. In einigen Fällen mit einer Rate von 50% bietet er eine bessere Leistung als der (m-k) -Algorithmus und erweist sich als überlegen, wenn es darum geht, die Bildung von Sekundärstrukturen zu verhindern.

  + Verhinderung von langen Homopolymeren

Der Encoder verhindert effektiv die Bildung von langen Homopolymeren (>8 bp) mit einer Auftretenswahrscheinlichkeit von weniger als 5% und trägt damit zur Fehlervermeidung bei der Sequenzierung bei.

Trotz dieser positiven Aspekte weist der Algorithmus in mehreren Bereichen Schwächen auf.

- Hohe Ausführungszeit: Die zeitliche Komplexität des Permutationskodierers scheint weniger effizient zu sein, da seine Ausführungszeit mit der Größe des Inputs schneller ansteigt. Außerdem macht ihn seine langsame Codierung für große Datenmengen (320290 ms für 2048 Basen) für große Eingaben praktisch unbrauchbar.
- Der Permutationskodierer behält in der Regel die Anzahl der CGs bei, allerdings kann er lokale Wiederholungen erzeugen, was zur Entstehung neuer Sekundärstrukturen führen könnte. Dies könnte die DNA-Sequenzierung komplexer machen.
- Vermeidung inkohärenter Strukturen: Der Codierer kann bei Sequenzen mittlerer Länge manchmal schlecht abschneiden und sogar das Risiko von Homopolymeren mit 6 Basen erhöhen, was die Stabilität der freien Energie bei bestimmten Gelegenheiten in Frage stellt. Darüber hinaus zeigt der Algorithmus im Allgemeinen eine durchschnittliche Leistung, was nicht für seine Zuverlässigkeit gegenüber Sekundärstrukturen spricht. Daher wird sein Einsatz in praktischen Kontexten wie Live-Anwendungen oder Hochgeschwindigkeits-Cloud-Speichersystemen schwierig.

5.3 Grenzen der Evaluierung

Allerdings ergeben sich bei der Analyse der von den verschiedenen vorgestellten Algorithmen erzeugten codierten Sequenzen gewisse Limitierungen.

Diverse Faktoren, wie beispielsweise die Temperatur oder der pH-Wert, können die Bildung von Sekundärstrukturen oder sogar des DNA-Moleküls beeinflussen. Zudem tragen weitere Umweltelemente wesentlich zu diesen Prozessen bei, deren Validierung durch chemische Experimente erfolgen muss. [35] [50] Die Untersuchung der Auswirkungen dieser Faktoren war nicht Gegenstand der vorliegenden Bewertung, da diese ausschließlich auf den in Java durchgeführten Implementierungen basiert.

Darüber hinaus wäre es von Interesse, die Möglichkeit der Verwendung einer anderen Programmiersprache zu untersuchen, da alle Tests und Algorithmen, die für die vorliegende Bewertung implementiert wurden, in Java durchgeführt wurden. Python als Programmiersprache verfügt über Bibliotheken und Pakete wie Nupack oder ViennaRNA, [37] [55] die speziell für die Analyse der Sekundärstrukturen von DNA-Sequenzen entwickelt wurden. Diese Tools könnten aussagekräftigere Ergebnisse liefern als andere Methoden oder eine effizientere Version des (m-k) - SSA-Codierers bieten. Ein Nachweis der Reproduzierbarkeit der Algorithmen sowie eine Stärkung des Urteils über ihre Robustheit wäre somit erbracht.

Andererseits fokussieren sich die meisten Algorithmen, die bisher im Zusammenhang mit der Sekundärstruktur entwickelt wurden, eher auf die Vorhersage als auf die Verhinderung der Sekundärstruktur. Dies limitiert den Umfang des Vergleichs, da die Resultate der eigenen Implementation lediglich mit denen des Permutationskodierers vergleichbar sind. Da der Prozess bei umfangreichen Sequenzen sehr zeitaufwendig ist, erweist sich der Vergleich von Sequenzen, die mehr als 5000 Basen umfassen, als kompliziert.

Die Testdaten wurden nicht aus Datenbanken bezogen, da es keine öffentlichen DNA-Datenbanken gibt, die eine Verifizierung der Vermeidung von Sekundärstrukturen ermöglichen würden. Die mit der Java-Implementierung erstellten Sequenzen ermöglichen jedoch die Beobachtung von Parametern wie der Sequenzlänge, aus der auch die Mindestlänge des Stems abgeleitet werden kann, den Anteil der enthaltenen GCs und ihre jeweilige minimale freie Energie. [36]

Die Dekodierungsphase von beiden Algorithmen muss noch weiterimplementiert, um getestet zu werden. Die gegebenen Algorithmen sind theoretisch korrekt, aber weisen Fehler in der Implementierung. Dies könnte leider in der gegebenen Zeit für unsere Arbeit nicht verbessert werden.

Darüber hinaus ist eine Bewertung der Kompatibilität mit Labortechnologien erforderlich, um die praktische Anwendbarkeit der aktuellen Synthese-/Sequenzierungsmethoden zu ermitteln.

Diskussion

6.1 Praktische Anwendbarkeit und optimale Einsatzbereiche

Um unerwünschte Sekundärstrukturen (wie Haarnadeln oder Stammschleifen) in DNA-Sequenzen, die mit SSA- oder Permutations-Codierern codiert wurden, auf ein Minimum zu reduzieren, werden die folgenden Hauptstrategien empfohlen:

- Der (m, k) -SSA-Coders wird für lange DNA-Sequenzen mit mehr als 500 Basen bevorzugt, da er eine bessere Leistung liefert.
  Er stellt eine optimale Option dar, wenn Sekundärstrukturen stark minimiert werden müssen (wie bei PCR-Primern) und der GC-Anteil flexibel angepasst werden kann.
  Darüber hinaus ist er ein ideales Werkzeug für Echtzeitanwendungen und hochskalierbare DNA-Speicherlösungen.
- Der Permutationskodierer findet vornehmlich bei kurzen Sequenzen Anwendung, deren Länge 500 Basen nicht überschreitet. Zudem wird sein Einsatz bei Sequenzen empfohlen, bei denen der Prozentsatz des CG-Ursprungs als kritisch oder unzureichend einzustufen ist oder erhalten werden muss. In der Folge ist der Permutationskodierer auch für thermostabile DNA-Anwendungen geeignet. Es empfiehlt sich daher, den Algorithmus spezifisch für stabile und kurze Sequenzen zu konzipieren, anstatt ihn für allgemeine Anwendungsbereiche zu bestimmen.

6.2 Optimierungsmöglichkeiten und weitere Ansätze

Es könnten jedoch einige Verbesserungen an beiden Codierern vorgenommen werden, um eine bessere Nutzung zu gewährleisten.

- Die Entwicklung einer adaptiven Version für (m, k) -SSA erscheint sinnvoll, um eine effizientere Verwaltung der GC-Basis zu erreichen. Einerseits sollte es dem Programmierer möglich sein, eine kontrollierte Abnahme der GC zu erreichen, um die GC-reichen Funktionsbereiche zu erhalten. Wird der GC-Gehalt jedoch zu hoch und eine Reduzierung hat keine Auswirkungen auf die gespeicherten Daten, ist ein Eingreifen notwendig, um Strukturen zu verhindern, die einen destabilisierenden Effekt aufweisen könnten. Darüber hinaus ist es ratsam, die Codierung kritischer Bereiche zu vermeiden und sie stattdessen als "nicht komprimierte oder neutrale Bereiche" zu bezeichnen, um die biologische Funktion zu erhalten. Eine Optimierung des Algorithmus mit Parallelverfahren könnte die Ausführungszeit weiter senken. Eine mögliche Integration mit anderen Algorithmen könnte Unzulänglichkeiten beheben.
- Für den Permutation Coder könnten die Permutationen auf kleine Fenster beschränken. Dann könnten weitere Wiederholungen verhindert und die benötigte Zeit für die Programmierung reduziert werden.
  Mit bioinformatischen Werkzeugen kann eine weitere Möglichkeit der Filterung nach der Kodierung genutzt werden. Eine Kombination mit energetischen Methoden könnte lokale Wiederholungen entfernen und Sekundärstrukturen noch besser unterdrücken.
  Des Weiteren könnte eine Implementierung in einer Sprache wie C++ oder Rust die Ausführungsgeschwindigkeit verbessern..
- Die praktische Anwendbarkeit könnte durch die Validierung der beiden Algorithmen mittels Langzeitanalysen oder der direkten Integration von Fehlerkorrekturcodes (z. B. Reed-Solomon) in die Implementierung weiter erhöht werden.

6.3 Einordnung in den Forschungsstand

Die beobachteten Ergebnisse tragen dazu bei, dass Fortschritte bei der Prävention von Sekundärstrukturierung erzielt werden können. Die bisher eingesetzten Algorithmen fokussierten sich in der Regel auf die Vorhersage oder das Risiko der Bildung dieser unerwünschten Struktur. Die Anwendung der beiden Algorithmen erleichtert die Kodierung von Sequenzen, die diese Strukturen enthalten.

Darüber hinaus bietet diese Studie durch die systematische und quantitative Vergleichsanalyse dieser Methoden eine glasklare Bewertung ihrer Leistungsfähigkeit. Dies ermöglicht es, insbesondere den gefundenen Kompromiss zwischen der Vermeidung der Struktur und der Erhaltung der Sequenz hervorzuheben, was die Auswahl der optimalen Strategien für die DNA-Codierung unterstützt.

Es ist jedoch anzumerken, dass eine praktische Laborstudie erforderlich ist, um die theoretischen Schlussfolgerungen der Analyse zu bestätigen.

Die in diesem Abschnitt diskutierten Punkte zeigen, dass die Auswahl der Algorithmen stark kontextabhängig ist und dass eine unabhängige optimale Lösung noch entdeckt werden muss. Die gewonnenen Erkenntnisse bilden eine robuste Grundlage für zukünftige Fortschritte in diesem sich schnell entwickelnden Forschungsfeld.

7. Fazit

7.1 Zusammenfassung

Ziel der vorliegenden Studie war die Untersuchung verschiedener Algorithmen, die im Kontext der Prävention der Bildung von Sekundärstrukturen innerhalb von DNA-Informationsspeichersystemen entwickelt wurden.

Diese Fehler, die aus diesen Strukturen resultieren können die Prozesse der chemischen Synthese und der DNA-Sequenzierung stören, weshalb es von entscheidender Bedeutung ist, das Ausmaß zu beurteilen, in dem die verschiedenen eingesetzten Methoden diese Fehler minimieren.

Zu diesem Zweck wurden verschiedene Ansätze erforscht, wobei für das Projekt zwei Algorithmen in Betracht gezogen wurden. Diese wurden anhand verschiedener Kriterien beurteilt, die in Kapitel vier der vorliegenden Studie ausgearbeitet wurden.

Die vorliegende Studie fokussiert sich daher in erster Linie auf den (m-k) SSA-Algorithmus und den Permutationskodierer. Die Implementierung beider Algorithmen erfolgte in der Programmiersprache Java. Im Rahmen dessen wurden Tests durchgeführt, um die Robustheit, die Leistung bei Sekundärstrukturen, die zeitliche Komplexität sowie die Auswirkungen auf die Fehler beim Lesen und Sequenzieren der DNA zu bewerten.

Die Tests zeigen, dass beide Algorithmen zufriedenstellend sind, obwohl der SSA-(m-k)-Codierer eine deutlich bessere Leistung als der Permutations-Codierer bietet. Eine Analyse der Daten zeigt, dass der SSA-Codierer eine stabile und konsistente Leistung mit einer nahezu linearen Ausführungszeit bietet, selbst bei erweiterten Sequenzen. Dies beweist, dass der SSA-Algorithmus skalierbar ist, wodurch er sich für große Datensätze eignet. Er ermöglicht es auch, den Grad der CG in einer Sequenz zu regulieren und beweist damit seine Effektivität bei der Vermeidung von Sekundärstrukturen.
Seine Fähigkeit, Homopolymere zu minimieren und damit Fehler im DNA-Syntheseprozess zu

reduzieren, sowie seine Fähigkeit, unter verschiedenen Bedingungen konstant zuverlässige Ergebnisse zu liefern, zeugen ebenfalls von seiner Effektivität.
Der Permutationskodierer bewahrt die GC-Rate während der Kodierung nahezu perfekt, was ihn zur idealen Wahl für biologische Anwendungen macht, bei denen genomische Stabilität erforderlich ist. Zudem entfernt er auch sehr effizient unerwünschte DNA-Segmente, wie seine hervorragende Leistung bei kurzen Sequenzen zeigt und die Tatsache, dass er den (m-k)-Algorithmus in einigen Fällen übertrifft. Darüber hinaus minimiert seine effektive Kontrolle der Anzahl von Homopolymeren mit großer Länge Sequenzierungsfehler, was für die Zuverlässigkeit der Daten von entscheidender Bedeutung ist.

Nichtsdestotrotz ist auch diese Methode nicht so perfekt. Der Permutationskodierer weist eine Ausführungszeit auf, die mit zunehmender Datengröße rapide ansteigt. Darüber hinaus besteht die Möglichkeit, dass die getroffenen Anordnungen zu lokalen Wiederholungen führen, was wiederum neue, unvorhergesehene Unregelmäßigkeiten zur Folge haben könnte. Obwohl der Algorithmus so konzipiert ist, dass er Sekundärstrukturen entgegenwirkt, bleibt seine Effizienz mittelmäßig und macht ihn daher für kritische Anwendungen nicht ausreichend zuverlässig.

Der (m, k) -SSA-Coder reduziert die GC-Rate in einem übermäßigen Maße, was sich nachteilig auf die funktionellen biologischen Muster auswirken und zu einer Verschlechterung der gespeicherten Daten führen könnte. Darüber hinaus ist der Coder bei kurzen Sequenzen aufgrund von Einschränkungen bei der Parameterimplementierung nur eingeschränkt einsetzbar. Er erfordert eine Sequenz mit einer Mindestlänge n von mehr als 64 und gilt nur für inverse Komplemente mit einer Länge m, die größer oder gleich $2\log_n + 2$ mit einem Mindestabstand von k gleich oder größer 4 ist. Dies resultiert in einer eingeschränkten algorithmischen Flexibilität.

7.2 Praktische Einsatz

Aus praktischer Perspektive kann der (m, k) -SSA-Coder für die Verarbeitung von Genomströmen in Echtzeit oder für die Archivierung biologischer Daten in großem Umfang eingesetzt werden. [35] [50] Aufgrund seiner Stärken ist er ein vielversprechendes Instrument, das Schnelligkeit, Stabilität und Genauigkeit für die Zukunft der molekularen Speicherung vereint und potenziell in industrielle DNA-Speichersysteme eingebaut werden könnte. Im medizinischen oder biotechnologischen Bereich könnte der Algorithmus zur Verringerung von Fehlern in der synthetischen Biologie und in DNA-basierten Schaltkreisen sowie zur sicheren Speicherung von Patientendaten innerhalb der DNA beitragen. [50] [36]

Um den Einsatzbereich zu erweitern, sind jedoch noch Verbesserungen des Algorithmus erforderlich. Aufgrund der starken Reduktion von GC-Basen ist es für die Analyse von biologischen Sequenzen, die zu diesen Basen sehr empfindlich sind, nicht geeignet. Dies begrenzt in gewissem Maße die Einsatzmöglichkeiten des Algorithmus im Hinblick auf die Speicherung komplexer Daten.

Der Permutationskodierer erweist sich hingegen als nützlich in Bereichen wie der synthetischen Biologie, in denen die Erhaltung der Stabilität der codierenden Sequenzen von entscheidender Bedeutung ist. Zudem ist er ein ideales Instrument für Anwendungen, bei denen die Erhaltung des GCs im Vordergrund steht, da er eine zuverlässige Speicherung der DNA mit weniger Artefakten beim Ablesen gewährleistet. [32]

Allerdings ist zu bemerken, dass der Permutationskodierer zeitliche Einschränkungen aufweist, die seine Anwendung in Echtzeitanwendungen wie der Verarbeitung von Genomströmen, Sofortdiagnosen oder Hochgeschwindigkeits-Cloud-Speicherung begrenzen.

Des Weiteren besteht die Möglichkeit, dass die Variabilität des Instruments die Wahrscheinlichkeit von Fehlern in den Phasen der DNA-Synthese/-Sequenzierung erhöht. Zudem sind die technischen Beschränkungen des Verfahrens zu berücksichtigen, die dessen Nutzung erheblich einschränken. Um mit den aktuellen Techniken konkurrieren zu können, sind radikale Verbesserungen unerlässlich.

7.3 Abschließende Bemerkung und Ausblick

Die für das Projekt festgelegten Ziele wurden erreicht. Die Analyse im Anschluss an die Messungen lieferte eine systematische Bewertung der Fähigkeit des (m, k) SSA-Encoders und des Permutations-Encoders, die Sekundärstruktur der DNA zu vermeiden. Anhand mehrerer Kriterien und Indikatoren wurden die Stärken und Schwächen der einzelnen Encoder bereitgestellt, die einen Einblick in ihre Verwendung im praktischen Bereich geben. Die Ergebnisse der Studie zeigen, dass der SSA-Kodierer bei langen Sequenzen deutlich bessere Ergebnisse liefert. Der Permutationskodierer ist dagegen für kritische GC-Anwendungen besser geeignet. Eine Optimierung beider Verfahren im Hinblick auf ihre jeweiligen Defizite wäre wünschenswert, um den Fortschritt im Bereich der DNA-Speicherung weiter voranzutreiben

Für zukünftige Studien wurde eine hybride Strategie vorgeschlagen, die die Vorteile der einzelnen Methoden vereint und in mehreren Sprachen umgesetzt werden könnte, um die Implementierung mit der besten Leistung zu identifizieren.

Darüber hinaus könnte die Kombination der Algorithmen mit einer experimentellen Validierung in Betracht gezogen werden. Obwohl die Ergebnisse der Forschung theoretisch optimal sind, könnten praktische Experimente ihre Qualität besser belegen. Es wäre von Interesse zu überprüfen, ob die Ergebnisse tatsächlich im Labor oder in bereits vorhandenen DNA-Speichersystemen anwendbar sind. Dadurch könnte konkret bestätigt werden, ob der Prozess des Lesens und Schreibens der verschiedenen Sequenzen verbessert wird. [35] [50] Dies könnte eine entscheidende Brücke zwischen Theorie und Praxis darstellen.

Weiterhin können die Ergebnisse je nach Programmiersprache variieren. Die Implementierung könnte daher in einer anderen Sprache erfolgen.
Mit modernen Technologien, die auf künstlicher Intelligenz basieren, kann die Leistungsfähigkeit bestehender Algorithmen erweitert oder die Implementierung neuer, leistungsfähigerer Algorithmen erleichtert werden.

Modelle des maschinellen Lernens ermöglichen eine präzisere Vorhersage von Sekundärstrukturen und die Optimierung von Kodierungsparametern auf der Grundlage von lernbasierter Verstärkung. Allerdings ist in diesem Zusammenhang eine kritische Haltung gegenüber den Herausforderungen im Zusammenhang mit der Zugänglichkeit der Daten und dem Verständnis der Modelle erforderlich.

Literaturverzeichnis

[1] „The Digitization of the World: From Edge to Core," IDC, 2021.

[2] „Annual Internet Report (2018–2023)," Cisco, 2022.

[3] „Solid-State Drive (SSD) Market Size Growth Rate Worldwide from 2019 to 2024," Statista, 2023.

[4] R. N. H. R. P. M. P. D. &. S. W. J. Grass, "Nucleic acid memory," *Nature Materials,* no. https://doi.org/10.1038/nmat4594, pp. 15(4), 366–370, 2016.

[5] „Next-generation digital information storage in DNA," *Science,* Bd. 337, p. 1628, 2012.

[6] „Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Nature Biotechnology,* Bd. 33, p. 731–737, 8 2015.

[7] „Random Access in Large-scale DNA Data Storage," *Nat. Biotechnol.,* Bd. 36, p. 242–248, 3 2018.

[8] „DNA Fountain enables a robust and efficient storage architecture," *Science,* Bd. 355, p. 950–954, 3 2017.

[9] „Enzymatic DNA synthesis for scalable data storage," *ACS Synthetic Biology,* Bd. 8, p. 1241–1248, 6 2019.

[10] „A DNA-of-things storage architecture to create materials with embedded memory," *Nature Nanotechnology,* Bd. 17, p. 731–742, 8 2022.

[11] „Molecular Digital Data Storage using DNA," *Nature Reviews Materials,* Bd. 6, p. 456–466, 6 2021.

[12] „Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid," *Nature,* Bd. 171, p. 737–738, 1953.

[13] „Desoxyribonukleinsäure (DNA)," 2025.

[14] „ViennaRNA Package 2.0," *Algorithms Mol. Biol.,* Bd. 6, p. 26, 2011.

[15] „DNA Aufbau – Struktur und Funktion," 2025.

[16] „DNA-based Storage: Models and Fundamental Limits," *IEEE Trans. Inf. Theory,* Bd. 64, p. 3671–3685, 2018.

[17] „DNA Synthesis Pricing," 2023.

[18] „A DNA-based Archival Storage System," *ACM SIGOPS Oper. Syst. Rev.,* Bd. 50, p. 637–649, 2016.

[19] „The Oxford Nanopore MinION: Delivery of Nanopore Sequencing to the Genomics Community," *Genome Biol.,* Bd. 17, 2016.

[20] „Error Correction for DNA Data Storage," *IEEE Trans. Inf. Theory,* 2018.

[21] „Toward a DNA-based Archival Storage System," *IEEE Micro,* Bd. 37, p. 98–104, 2017.

[22] M. &. Kashyap, „Trans. Inf. Theory," in *On the Design of Codes for DNA Computing*, 2006.

[23] „DNA Data Storage Robustness," *Sci. Rep.,* 2017.

[24] „Nucleic acid synthesis via phosphoramidite chemistry," *Tetrahedron,* Bd. 48, p. 2223–2311, 1992.

[25] „Forward Error Correction for DNA Data Storage," *Nat. Biotechnol.,* Bd. 36, p. 645–650, 2018.

[26] „Nanopore Sequencing," *Nat. Biotechnol.,* Bd. 34, p. 518–524, 2016.

[27] „Spectrum Storage Report," 2022.

[28] „Forward Error Correction in DNA Data Storage," *Bioinformatics,* 2020.

[29] „Nat. Methods," 2022.

[30] „G-quadruplexes in DNA," *Chem. Soc. Rev.,* Bd. 37, p. 1375–1384, 2008.

[31] „Cruciform Structures in Palindromic DNA," *Biopolymers,* Bd. 29, p. 1279–1287, 1990.

[32] „Permutation Coder: Preventing Secondary Structures in DNA Storage," in *Proc. IEEE Symp. on Advanced DNA Storage Techniques*, 2023.

[33] „Forward Error Correction for DNA Data Storage," in *Proc. IEEE ICC*, 2016.

[34] „Monte Carlo Simulations for DNA Coding Error Reduction," *IEEE Trans. Comput. Biol.,* Bd. 22, p. 256–264, 2021.

[35] „Algorithmic Foundations of DNA Data Storage," *IEEE Trans. Biotechnol.,* Bd. 15, p. 123–132, 2023.

[36] „Robustness of DNA Data Storage Algorithms under Variable Conditions," in *Proc. IEEE Symp. Future BioData Storage*, 2024.

[37] „SST-sequence-designer," 2023.

[38] „Fast Algorithm for Predicting the Secondary Structure of Single-stranded RNA," *Proc. Natl. Acad. Sci. USA,* Bd. 77, p. 6309–6313, 1980.

[39] „Accurate Prediction of RNA Secondary Structure," *Methods Enzymol.,* Bd. 468, p. 79–98, 2009.

[40] „Simultaneous Solution of the RNA Folding, Alignment and Protosequence Problems," *SIAM J. Comput.,* Bd. 10, p. 303–312, 1981.

[41] „Heuristic Approaches for Designing RNA Sequences to Avoid Secondary Structures," *Bioinformatics,* Bd. 22, p. 3170–3176, 2006.

[42] „Challenges in DNA data storage: Synthesis, sequencing, and error correction," *Biosystems,* Bd. 96, p. 200–210, 2009.

[43] „Error Correction for Genomic Digital Data," *NAR Genom. Bioinform.,* 2021.

[44] „Deep Learning for RNA Secondary Structure Prediction," *Bioinformatics,* Bd. 35, p. 328–336, 2019.

[45] „Advances in Nanopore Error Correction," *IEEE Trans. Nanobiosci.,* 2020.

[46] „Combinatorial Methods in DNA Sequence Design: A Fractal Approach," *IEEE Trans. Nanobioscience,* Bd. 12, p. 45–53, 2013.

[47] „DNA Sequence Analysis Methods," *Nucleic Acids Res.,* 2011.

[48] „Thermodynamic Analysis of DNA Hairpins," *Nucleic Acids Res.,* Bd. 32, p. 2962–2974, 2004.

[49] „Multiobjective Optimization in Genetic Algorithms for DNA Sequence Design," in *Proc. IEEE Int. Conf. on Evolutionary Computation*, 2015.

[50] „Simulation and Complexity Analysis of DNA Secondary Structure Avoidance," in *Proc. IEEE Int. Conf. Bioinformatics*, 2022.

[51] D. Retkowitz, Datenbasierte Algorithmen zur Unterstützung von Entscheidungen mittels künstlicher neuronaler Netze, wiesbaden: Springer Vieweg, Wiesbaden, 2021.

[52] T. T. e. a. Nguyen, „On the design of codes for DNA computing: Secondary structure avoidance codes," *International Symposium on Information Theory,* 2023.

[53] H. W. R Zhang, „On secondary structure avoidance of codes for DNA storage," *Computational and Structural Biotechnology Journal, ,* pp. 140-147., 2024.

[54] „ « http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi »".

[55] „Exploring DNA-secondary-structures-with-Python," 2023.

[56] „Improved Data Storage using DNA," *Nat. Biotechnol.,* Bd. 36, p. 242–248, 2018.

[57] „Nucleic acid memory," *Nature Materials,* pp. 15(4), 366–370, 2016.

[58] Principles of Nucleic Acid Structure, Springer, 1984.

[59] „Enzymatic DNA Synthesis White Paper," 2022.

[60] „DNA Mechanical Properties," *Science,* Bd. 265, p. 1599–1600, 1994.

[61] „DNA Helix Parameters," *J. Mol. Biol.,* Bd. 166, p. 419–441, 1983.

[62] „DNA Base Stacking Energies," *Nucleic Acids Res.,* Bd. 40, p. 2987–2994, 2012.

[63] „Cytosine Dipole Moments," *J. Am. Chem. Soc.,* Bd. 120, p. 327–338, 1998.

[64] „Accurate Prediction of RNA Secondary Structure," *Nucleic Acids Res.,* Bd. 31, p. 3406–3415, 2003.

[65] „Stem-loop (Hairpin loop): Properties, Types, Examples, Uses," 2024.

Abbildungsverzeichnis

Anhang

DATENTABELLE

1. Sekundärstrukturindex

länge	original seque	SSA encoder	permutation coder secondary structure i	
115	0.011953134	0.011848509	0.0098037874538879 98	
132	0.067244267	0.010665762	0.029763001570630096	
132	0.070349074	0.020691803	0.02961418656277694	
232	0.075063865	0.017495492	0.034157434537697096	
368	0.358368744	0.010617682	0.011388184342463952	
372	0.095193440	0.019513568	0.044749890841184956	
492	0.104100862	0.021373674	0.05073408250441743	
512	0.011598983	0.011574837	0.011046025 24490858	
552	0.099327665	0.019979154	0.0453644484008422	
612	0.023428906	0.023336137	0.02354772871407339	
652	0.104686628	0.020286177	0.048980227831762854	
800	0.011501953	0.011491946	0.011326099 115887852	
932	0.118069843	0.023290820	0.05886124143990104	
1023	0.012126332	0.012111555	0.012294783404042785	
1132	0.120365315	0.021820166	0.05971347320139355	
1132	0.115931359	0.021311151	0.058447982831644735	
1532	0.123718151	0.020938107	0.06055799899688726	
1692	0.129716063	0.022759443	0.06447045249908923	
1800	0.009594250	0.009591252	0.00961746612173076	
1932	0.125004962	0.020677254	0.06190818880034232	
2048	0.011770476	0.011764720	0.011243374 658100299	

2. GC Gehalt

länge	original seque	original seque	ssa coder gc	permutation coder gcCount		
115	61	53%	61	50		
132	106	80%	77	106		
132	106	80%	89	106		
232	186	80%	152	186		
368	184	50%	176	184		
372	314	84%	249	314		
492	426	87%	339	426		
512	257	50%	257	251		
552	468	85%	376	468		
612	438	72%	438	438		
652	562	86%	448	562		
800	401	50%	401	401		
932	830	89%	671	830		
1023	548	54%	548	548		
1132	1010	89%	780	1010		
1132	1002	89%	784	1002		
1532	1364	89%	1030	1364		
1692	1530	90%	1160	1530		
1800	687	38%	687	687		
1932	1736	90%	1294	1736		
2048	1059	52%	1059	1023		

3. Freie Energie

länge	freie energie (	freie energie :	freie energie permutation C	
115	-41,10	-41,10	-32,3	
132	-142,20	-26,8	-72,2	
132	-141,40	-48,8	-74,6	
232	-282,20	-86,3	-134,8	
368	-351,60	-76,50	-116,5	
372	-486,00	-167,7	-249,3	
492	-666,10	-238,3	-375	
512	-154,20	-154,20	-145,5	
552	-738,50	-262,6	-360,9	
612	-312,10	-312,10	-310,5	
652	-892,40	-304	-456,7	
800	-251,20	-251,20	-250,5	
932	-1311,20	-446,2	-702,5	
1023	-336,60	-336,60	-328,7	
1132	-1605,70	-574,2	-881,8	
1132	-1598,00	-544,9	-852,4	
1532	-2177,50	-733,54	-1191	
1692	-2438,20	-886,67	-1533,9	
1800	-421,60	-421,60	-444,8	
1932	-2782,50	-970	-1533,9	
2048	-659,70	-659,70	-632,9	