FACHBEREICH INFORMATIK

Seminararbeit

Algorithmische Aspekte der DNA-Sequenzierung

Christoph Altrock

Student der Informatik, 2. Semester

Wintersemester 2018/2019

Seminarreihe *Mathematische Anwendungen*

Seminarleitung:

Prof. Dr. Sebastian Iwanowski

Fachhochschule Wedel

Feldstraße 143, 22880 Wedel

28. Februar 2019

Inhaltsverzeichnis

Abkürzungsverzeichnis ... III
Symbolverzeichnis ... III
Abstract ... IV
1 Einleitung ... 1
1.1 Struktur und Funktion der DNA ... 1
1.2 Abriss der Bioinformatik ... 2
1.3 Motivation der Problemstellung ... 3
1.3.1 Potenziale der Gentechnik ... 3
1.3.2 Biotechnische Sequenzierungsverfahren ... 4
2 Formale Problemdefinition ... 6
2.1 Shortest Common Supersequence Problem ... 6
2.2 Sequencing by Hybridization ... 7
2.2.1 DNA-Arrays ... 8
2.2.2 SBH Problemdefinition ... 8
3 Algorithmen-Entwicklung ... 9
3.1 Natürliche Modellierung ... 10
3.2 Zweite graphentheoretische Modellierung ... 12
3.3 Grenzen der Problemdefinition ... 14
4 Fortgeschrittene Konzepte ... 17
4.1 Klassische Vorgehensweise ... 17
4.2 Superpfade ... 19
4.2.1 Ablaufschema in ESP ... 20
4.2.2 Transformationen auf Graphen ... 22
5 Zusammenfassung und Ausblick ... 26
Literaturverzeichnis ... 27

Abkürzungsverzeichnis

BSP	–	de Bruijn Superpfad Problem
CPP	–	Chinese Postman Problem
DBG	–	de Bruijn Graph Paradigma
ESP	–	Eulerian Superpath Problem
HGP	–	Human Genome Project
OLC	–	Overlap-Layout-Consensus Paradigma
SBH	–	Sequencing By Hybridization
SCS	–	Shortest Common Supersequence
TSP	–	Traveling Salesman Problem

Symbolverzeichnis

$\boldsymbol{\varepsilon}$	–	kleiner griechischer Buchstabe Epsilon (verwendet für das leere Wort)
$\boldsymbol{\Sigma}$	–	großer griechischer Buchstabe Sigma (verwendet für Alphabete)
$\boldsymbol{\varphi}$	–	kleiner griechischer Buchstabe Phi (verwendet für den gesuchten Superpfad, der der DNA-Sequenz entspricht)
$\boldsymbol{\omega}$	–	kleiner griechischer Buchstabe Omega (verwendet für das gesuchte Superwort, das der DNA-Sequenz entspricht)
$\subseteq$	–	Teilmengenbeziehung
$\subset$	–	echte Teilmengenbeziehung
$\cap$	–	Schnittmenge
$\in$	–	Elementzeichen
$\times$	–	Kreuzprodukt
$\emptyset$	–	Leere Menge
$\exists$	–	Existenzquantor
$\wedge$	–	Logische Konjunktion

Abstract

Diese Ausarbeitung erörtert Algorithmen, welche für das Problem der DNA-Sequenzierung entwickelt wurden. Der Fokus liegt dabei auf Algorithmen für graphentheoretische Modellierungen des Problems.

Die Einleitung motiviert die Problemstellung und erläutert den biologischen Hintergrund. Zum Kern der Arbeit sind die Ideen von Pavel Pevzner et al. geworden. Die Gliederung in den Kapiteln zwei und drei ist an den Spannungsbogen des Lehrbuches [JoPe04] angelehnt. Sie diskutieren nach dem trial-and-error-Prinzip verschiedene Algorithmen. Das vierte Kapitel erörtert komplexere Konzepte, die die praktischen Beschränkungen der einfachen Algorithmen aus den vorigen Kapiteln lösen sollen. Im fünften Kapitel wird überdies ein Ausblick auf hier nicht behandelte Algorithmen und Modellierungen gegeben.

Ziel dieser Arbeit ist es die Gründe für die Anforderungen an und das Scheitern von Algorithmen gut verständlich darzustellen. Es wurde Wert darauf gelegt (mathematisch) präzise zu sein, gerade weil es die verwendete Literatur in einigen Fällen nicht ist. Insbesondere durch diese Genauigkeit konnten sogar einige Unstimmigkeiten gefunden werden. Darüber hinaus werden an einigen Stellen Verweise auf weiterführende Unterlagen gegeben. Diese sind für den Zweck dieser Arbeit nicht unmittelbar erforderlich. Sie berühren jedoch die Thematik und bieten interessierten Lesern einen tieferen Einstieg bzw. ein breiteres Hintergrundwissen.

Die Arbeit enthält auch eigene Überlegungen des Autors, welche entsprechend gekennzeichnet sind. Im Vergleich zum gehaltenen Seminarvortrag ist diese Arbeit detaillierter und umfangreicher. Im Vortrag konnten Aspekte aus dem vierten Kapitel noch nicht geklärt werden, diese Aspekte sind nun soweit wie möglich beantwortet worden.

1 Einleitung

In diesem Kapitel werden Grundlagen erläutert. In einigen Fällen sind diese bereits in Seminaren früherer Semester behandelt worden. Da sie für das Verständnis jedoch notwendig sind, werden sie an dieser Stelle kurz beleuchtet. Hauptsächlich wird jedoch um weitere Aspekte ergänzt oder vertieft.

1.1 Struktur und Funktion der DNA

DNA ist die Kurzform für Desoxyribonukleinsäure (engl. Deoxyribonucleic acid). Der Name leitet sich aus der **Struktur des Moleküls** ab[1]: DNA besteht aus einer Kette von *Nukleotiden*. Die Nukleotide bestehen wiederum aus einem *Phosphat-Desoxyribose-Rückgrat*, welches die Verkettung ermöglicht, und einer der vier Basen *Adenin* (A), *Cytosin* (C), *Guanin* (G) und *Thymin* (T). Die Basen der Nukleotide sind paarweise zueinander komplementär[2]. Das bedeutet, sie besitzen die Fähigkeit eine chemische Bindung untereinander aufzubauen. Indem sich A mit T und C mit G zu Basenpaaren verbinden, entsteht die bekannte Form der Doppelhelix.

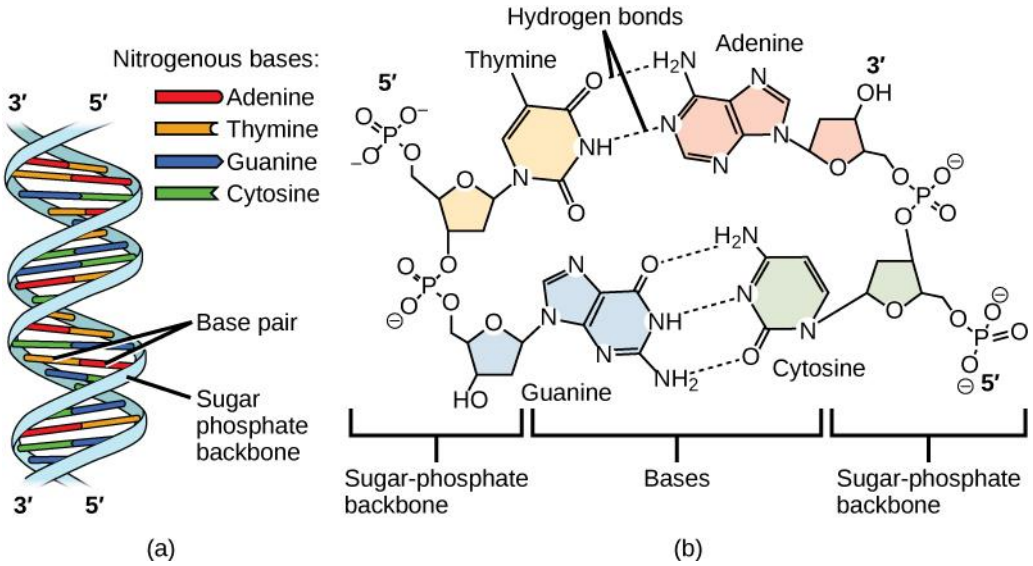

Abb. 1.1 – Struktur der DNA und Nukleotide[3]

Die Abfolge der Basen in den Nukleotiden ermöglicht es der DNA **Informationen zu speichern** (prinzipiell ähnlich der binären Kodierung von Bytes). Die gespeicherten Informationen dienen dazu einen Organismus aufzubauen und zu erhalten. Um diese Aufgabe zu erfüllen, besteht die menschliche DNA aus etwa 3 Milliarden Basen.

Gene sind Abschnitte in der DNA mit einer Länge zwischen einigen Hundert bis über 2 Millionen Nukleotiden. Sie dienen dazu Proteine herzustellen und die Sequenzen der DNA, auf denen sie liegen, werden auch als *codierende DNA* bezeichnet. Proteine dienen als Signalstoffe für die Körperzellen und betreiben alle biochemischen Prozesse wie das Wachstum, Bekämpfung von Infektionen, den Hormonhaushalt, Blutgerinnung und nahezu alle anderen lebenswichtigen Körperfunktionen.

---

[1] Die Ausführungen dieses Kapitels gehen zurück auf: NLM18
[2] vgl. Stein13, S.3f
[3] Abb. entnommen aus MoGa13, Kapitel 9.1

Menschen unterscheiden sich in ihrer körperlichen Gestalt nur aufgrund von verschiedenen Ausprägungen in den Genen (sogenannten *Allelen*).

Der Anteil **nicht-codierender DNA** liegt beim Menschen bei über 90%. Auch nicht-codierende DNA erfüllt mindestens teilweise einen Zweck. Dazu gehört es zum Beispiel Ansatzpunkte für Proteine zu bilden, welche die DNA replizieren. Oder es werden bestimmte Gene „aktiviert" (diese erzeugen dann Proteine: *Genexpression*)[4]. Welche Gene in einer Zelle aktiv sind, bestimmt die Funktion der Zelle. Das führt dazu, dass obwohl nahezu alle Zellen die gleiche DNA in sich tragen (Ausnahmen bilden zum Beispiel Eizellen oder Spermien) eine Gehirnzelle andere Aufgaben ausführen kann als eine Hautzelle. Somit sind auch in der nicht-codierenden DNA wichtige Informationen enthalten, weshalb es wichtig ist auch diese zu sequenzieren. Wie groß der Anteil an echt funktionsloser DNA (*Junk DNA*) ist, darüber streiten sich die Wissenschaftler.

1.2 Abriss der Bioinformatik

Das vorige Kapitel hat kurz die umfassende Bedeutung der DNA für das Funktionieren eines Organismus dargelegt. Wegen dieser Schlüsselrolle ist es nicht weiter verwunderlich, dass ein Forschungsprojekt gestartet worden ist, welches das Ziel hatte die Gesamtheit der menschlichen Gene (das *Genom*) zu finden und die Funktion der durch sie herstellbaren Proteine zu bestimmen[5]. Um genau zu sein war zum Start des **Human Genome Project** (HGP) in 1990 die exakte Funktionsweise der DNA noch nicht bekannt, allerdings wusste man aus einigen Experimenten um die generelle Rolle der DNA[6].

Man war sich im Rahmen des HGP um die Bedeutung von **computerbasierten Verfahren** bewusst. Denn die enorme Anzahl von ca. 3 Milliarden Nukleotiden und 20.000 Genen[7] ist manuell nicht analysierbar. Aus diesem Grund ist im HGP eine *Joint Informatics Task Force* aufgestellt worden, die die Aufgabe hatte informationstechnische Probleme bei der Entschlüsselung des Genoms zu lösen. Dieses Forschungsfeld nennt man auch *Bioinformatik*.

Im Folgenden sind einige zentrale Problemstellungen der Bioinformatik aufgeführt:

- interpretieren des Outputs von Sequenzierungsmaschinen als Basensequenz (*base calling*)
- zusammenfügen von Teilsequenzen zur gesamten DNA (*fragment assembly*)
- abgleichen von Basensequenzen (*sequence aligning*)[8]
- auffinden von codierenden Regionen in der DNA
- unterscheiden von *Exons* und *Introns* in Genen[9]
- identifizieren von *Homologien* zwischen Genen
- ermitteln von Bindungsstellen an der DNA für Proteine und RNA-Polymerase

Während des HGP bestand die Hauptrolle der Bioinformatik darin die DNA zu sequenzieren, d.h. die Reihenfolge der Basen zu ermitteln. Das HGP veröffentlichte in 2001 den bis dato sequenzierten Teil des Genoms (ca. 90%). In 2003 war die Sequenzierung des gesamten menschlichen Genoms abgeschlossen. Seitdem ist der Forschungsschwerpunkt auf die Ermittlung der Funktionsweise der nicht-codierenden DNA geschwenkt. Weiterhin wird die DNA anderer Spezies sequenziert[10].

---

[4] vgl. NIGMS10, S. 12 und MoGa13, S. 340
[5] Die Ausführungen dieses Kapitels gehen zurück auf: Allex99, S.1-3
[6] Avery-Macleod-McCarty Experiment, Hershey-Chase Experimente.
Siehe auch: https://www.yourgenome.org/stories/revealing-dna-as-the-molecule-of-life
[7] vgl. NIGMS10, S. 16
[8] siehe auch Sem2 und Sem3
[9] siehe auch Sem5, S. 5
[10] vgl. NIGMS10, S.62f und NHGRI

1.3 Motivation der Problemstellung

1.3.1 Potenziale der Gentechnik

Der Anreiz zum Start des HGR war, dass nach der „Entschlüsselung“ des Genoms – also der Sequenzierung der DNA, Bestimmung der Gene und deren Funktionen – tiefgreifende **Erkenntnisse** über die biologische Funktionsweise des menschlichen Körpers entstehen. 15 Jahre nach dieser Entschlüsselung ist klar, dass diese Hoffnungen erfüllt wurden. Die Genetik hat eine ganze Reihe von Potenzialen:

Das Forschungsfeld **Pharmakogenetik** beschäftigt sich damit, warum Menschen aufgrund ihrer Gene verschieden auf Medizin reagieren[11]. Dass es unterschiedliche Reaktionen gibt liegt darin begründet, dass Medikamente von Proteinen verarbeitet werden, welche wiederum bei jedem Menschen durch leicht unterschiedliche Gene codiert werden. Auf dieser Informationsgrundlage können Ärzte Entscheidungen über die Behandlungsmethode von Patienten treffen. Ein Beispiel ist das Medikament Herceptin zur Behandlung von Brustkrebs[12].
Eine andere Anwendungsmöglichkeit ist das Entwickeln von Antibiotika, die Gene von Bakterien deaktivieren können ohne dass die Bakterien eine Resistenz dagegen aufbauen können[13].
Des Weiteren sind Krankheiten häufig genetisch bedingt, d.h. sie werden vererbt, weil es sich um einen Gendefekt handelt. Durch seine Methode *genome-wide location analysis* konnte Richard Young bspw. feststellen, dass einige Gene in den Leberzellen von Diabeteskranken nicht „aktiv“ waren. Das weist auf einen Einfluss von defekten Genen auf die Krankheit hin. Mit wirkungsvollen Methoden, die die „inaktiven“ Gene wieder „aktivieren“ könnten Krankheitsverläufe verändert oder der Ausbruch einer Krankheit ganz verhindert werden.

Das Forschungsfeld **Sociogenomics** untersucht welchen Einfluss die Umwelt auf das „Aktivsein“ von Genen hat[14]. Mit Forschungsergebnissen die darlegen, wie man das „Aktivsein“ steuern kann, wäre es möglich durch beeinflussbare Faktoren wie sportliche Betätigung das Risiko für Krankheiten zu senken.

Molekulare Biotechnologie verändert schon jetzt Pflanzen genetisch dazu resistent gegen bestimmte Seuchen zu sein oder mehr Erträge zu liefern[15]. Das kann den Hunger in der Welt bekämpfen, aber Langzeitfolgen sind noch nicht erforscht. Mithilfe von *Stammzellen*, dies sind Zellen mit der Fähigkeit jede möglichen Körperzelle zu werden, erhofft man sich Körperteile zu erschaffen, um Verletzte zu heilen[16]. Genetisch veränderte Bakterien sind in der Lage Medikamente wie Insulin zu erzeugen[17].

Die DNA eines Menschen ist – außer bei eineiigen Zwillingen – **einzigartig** daher kann sie ähnlich wie Fingerabdrücke zur Verbrechensaufklärung verwendet werden. Auch ist es möglich die Opfer einer Naturkatastrophe zu identifizieren, Familienverhältnisse oder passende Organspender zu ermitteln[18].
Weiterhin ist es möglich geworden den Werdegang einer Spezies zu rekonstruieren, indem man deren Genome vergleicht und nach Unterschieden sucht. Phylogenetische Bäume sind eine Möglichkeit zur Visualisierung von Beziehungen zwischen Spezies[19].

---

[11] vgl. NIGMS10, S.62-66
[12] siehe NIGMS10, S. 62f
[13] vgl. NIGMS10, S. 16, 18, 20
[14] vgl. NIGMS10, S.66f
[15] vgl. NIGMS10, S.82f
[16] vgl. NIGMS10, S.52f
[17] https://www.pharmazeutische-zeitung.de/ausgabe-372010/hormon-aus-bakterien-hefen-und-pflanzen/
[18] vgl. NIGMS10, S.81
[19] siehe auch Sem1, Sem4 und Sem6

Trotz aller utopischen Möglichkeiten ist das Verwenden von Genomen sozial und ethisch ein schwieriges Thema[20]. Denn das Genom ist eine sehr **persönliche Information** und daher auch besonders schützenswert. Es enthält die Vergangenheit (z.B. die Abstammung), beschreibt die Gegenwart und lässt Aussagen über die Zukunft zu (z.B. Risiken für Krankheiten). Diese Informationen bergen auch das Risiko von Diskriminierung.

1.3.2 Biotechnische Sequenzierungsverfahren

Die im vorigen Kapitel angesprochenen Potenziale begründen die Relevanz der DNA Sequenzierung. Dieses Kapitel stellt nun einige biotechnische Verfahren dar und motiviert so die in Kapitel 2 besprochenen algorithmischen Problemstellungen.

Die frühen Verfahren zur DNA Sequenzierung zerlegen einen DNA-Strang in kurze Fragmente[21]. Auch später entwickelte Verfahren können eine DNA-Sequenz nicht vom Anfang bis zum Ende lesen. Nur aus diesem Grund sind Algorithmen zur DNA-Sequenzierung überhaupt notwendig.

Die **Starvation-Methode** nach Frederik Sanger unterbricht bei der Replikation der DNA – dies ist ein chemischer Prozess – den Nachschub an Basen. In der Folge ergeben sich Teilsequenzen der DNA, die vor der nicht mehr verfügbaren Base stoppen. Zum Beispiel produziert die Sequenz ACGTAAGCTA, welche mit nicht genügend T versorgt wird, die Fragmente ACG und ACGTAAGC.
Dieser Prozess wird für alle vier Basen durchgeführt. Die in der Reaktion entstandenen Fragmente werden nun der Länge nach geordnet (Abb. 1.2). Dazu platziert man sie auf einer Platte mit Gel und setzt diese unter eine elektrische Spannung. Die Spannung bewirkt, dass sich die Fragmente durch das Gel bewegen, wobei kürzere Fragmente schneller sind. Dieser Prozess nennt sich *Elektrophorese*. Die Länge der Fragmente verrät, an welchen Positionen die in der Reaktion fehlenden Basen in der DNA vorkommen. Um die Positionen sichtbar zu machen, markiert man die Primer, welche die Reaktion in Gang setzen, radioaktiv und erstellt Röntgenaufnahmen[22].

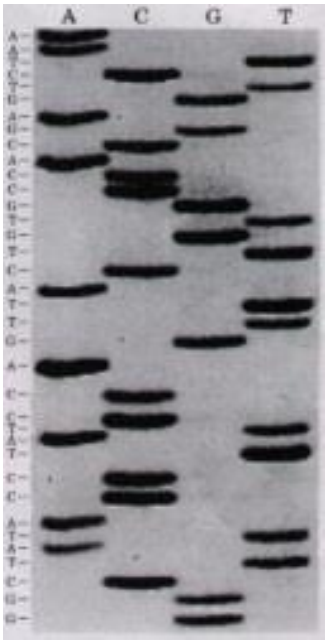

Abb. 1.2 – Aufnahme radioaktiv markierter Fragmente (*Sanger ladder*)[23]

[20] vgl. NIGMS10, S.78
[21] vgl. JoPe04, S. 263f
[22] vgl. NYCCT
[23] Abb. entnommen aus LeNeCo93

Die **Dideoxy-Methode** nach Sanger funktioniert nach dem gleichen Prinzip wie die *Starvation-Methode*, allerdings wird nicht der Nachschub an Basen unterbrochen, sondern man gibt terminierende Basen hinzu. Diese werden statt A, C, G oder T in die kopierte Teilsequenz eingesetzt, und verhindern dort ein weiteres Anhängen von Basen.
Statt radioaktiv markierter Primer werden nun die terminierenden Basen (*Dideoxynukleotide*) jeweils mit einer fluoreszierenden Farbe versehen[24]. Anschließend platziert man sie wie bisher auf einer Platte mit Gel und setzt diese unter Spannung. Die Fragmente passieren nun ein optisches Lesegerät, welches anhand der fluoreszierenden Farbe die Basenabfolge bestimmt (Abb. 1.3). Dieser Prozess entspricht mehr einer *Chromatographie*.

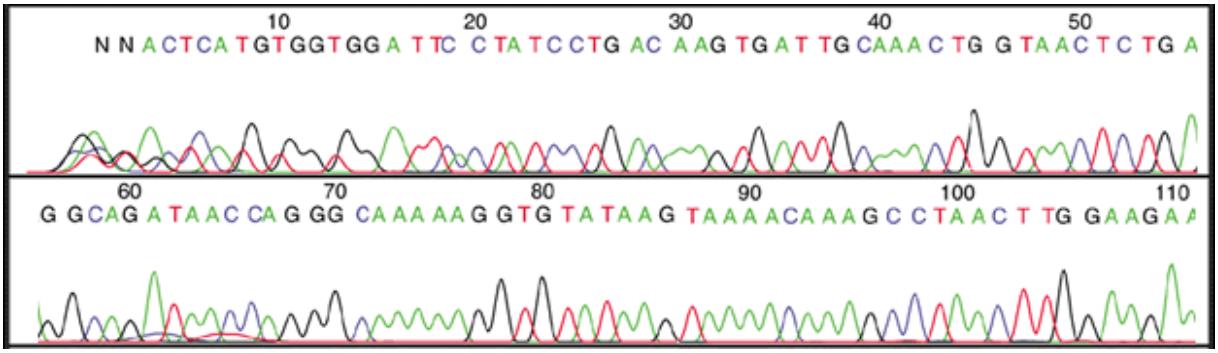

Abb. 1.3 – Ausgabe eines Lesegeräts für fluoreszierend markierte Fragmente[25]

Beide Methoden benötigen eine fein eingestellte Dosierung der chemischen Elemente. Es müssen genug Basen vorhanden sein um auch längere Sequenzen zu erlauben, aber nicht zu viele damit sich eine gute Verteilung über alle Fragmente ergibt. Generell können mit diesen Methoden nur relativ **kurze Fragmente** sequenziert werden (ca. 500-700 Basen). Die Basenabfolgen der sequenzierten Fragmente nennt man auch *reads*.

Um eine DNA-Sequenz von mehreren Milliarden Nukleotiden sequenzieren zu können, muss diese zunächst in kürzere Fragmente zerlegt werden, welche sich dann mit den obigen Methoden sequenzieren lassen. Dies erfolgt indem die lange DNA-Sequenz z.B. in einem Ultraschallbad **fragmentiert** wird (*Shotgun Sequencing*).Die Fragmente werden anschließend durch Bakterien vervielfältigt und sequenziert.

Nach der Fragmentierung der DNA-Sequenz und der Bestimmung der Basenabfolge in den Fragmenten, müssen nun die *reads* wieder zu jener DNA-Sequenz zusammengesetzt werden, aus der sie stammen. Dies lässt sich **modellieren** als das Zusammenfügen von Zeichenketten zu einem sogenannten Superstring.

Durch diese Modellierung wird von den biologischen Details abstrahiert und die Bioinformatik kann sich auf die Lösung eines mit einfachen Datentypen darstellbaren Problems konzentrieren. Allerdings sind die biotechnischen Verfahren maßgeblich für den Input des Problems und damit dessen Art.

Insgesamt kommt man also zu einem **zweistufigen Prozess**: Der erste Schritt ist experimenteller Natur und beinhaltet das Sequenzieren von Fragmenten aus der DNA. Der zweite Schritt ist kalkulatorisch (*computational*) und behandelt das Zusammensetzen der Teilsequenzen zu der einzelnen langen Sequenz von der sie stammen[26].

---

[24] vgl. Allex99, S. 10-13
[25] Abb. entnommen aus *https://www.genome.gov/25020000/online-education-kit-bioinformatics-introduction*, zugegriffen am 13.11.2018
[26] Für weiterführende Informationen siehe auch *https://www.genome.gov/25019885/online-education-kit-how-to-sequence-a-human-genome/*

2 Formale Problemdefinition

Zu Beginn sind die grundlegenden Konstrukte **mathematisch präzise** zu definieren. Dazu wird sich an dieser Stelle kurz dem Teilgebiet der **Formalen Sprachen** aus der Theoretischen Informatik bedient[27]:

Sei die Menge $\boldsymbol{\Sigma}$ = {A, C, G, T} ein *Alphabet*. Die Elemente von $\boldsymbol{\Sigma}$ heißen *Buchstaben*. Eine endliche Folge $\boldsymbol{w}$ von Buchstaben nennt man ein *Wort*. Das leere Wort enthält keinen Buchstaben, es wird mit $\boldsymbol{\varepsilon}$ bezeichnet. Die Länge eines Wortes $\boldsymbol{w}$ wird angegeben durch $|\boldsymbol{w}|$, wobei $|\boldsymbol{\varepsilon}| = 0$. $\boldsymbol{\Sigma}^{\boldsymbol{k}}$ bezeichnet die Menge aller Wörter mit einer Länge von $\boldsymbol{k}$. $\boldsymbol{\Sigma}^{+}$ bezeichnet die Menge aller endlichen, nichtleeren Wörter. Die Menge aller Wörter ist $\boldsymbol{\Sigma}^{*} = \{\boldsymbol{\varepsilon}\} \cap \boldsymbol{\Sigma}^{+}$.
Die *Konkatenation* von Wörtern (bzw. Buchstaben) werde mit der Funktion $\circ : \boldsymbol{\Sigma}^{*} \times \boldsymbol{\Sigma}^{*} \rightarrow \boldsymbol{\Sigma}^{*}$ ausgedrückt. Ein Wort $\boldsymbol{t} \in \boldsymbol{\Sigma}^{*}$ heißt *Teilwort* eines Wortes $\boldsymbol{w} \in \boldsymbol{\Sigma}^{*}$genau dann, wenn es Wörter $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{\Sigma}^{*}$ gibt mit $\boldsymbol{w} = \boldsymbol{u} \circ \boldsymbol{t} \circ \boldsymbol{v}$. Üblicherweise lässt man die explizite Konkatenation weg und schreibt $\boldsymbol{w} = \boldsymbol{utv}$. $\boldsymbol{u}$ heißt auch ein *Präfix* von $\boldsymbol{w}$, $\boldsymbol{v}$ nennt man ein *Suffix* von $\boldsymbol{w}$.
In diesem Zusammenhang sei die Funktion $\texttt{overlap}(\boldsymbol{s}, \boldsymbol{t}) : \boldsymbol{\Sigma}^{*} \times \boldsymbol{\Sigma}^{*} \rightarrow \boldsymbol{\Sigma}^{*}$ definiert. Deren Ergebnis ist das längste Teilwort, welches sowohl Suffix des Wortes $\boldsymbol{s}$ als auch Präfix des Wortes $\boldsymbol{t}$ ist.

Die von den biotechnischen verfahren generierten *reads* entsprechen einer endlichen Menge $\boldsymbol{W} \subseteq \boldsymbol{\Sigma}^{+}$ von Wörtern über dem Alphabet $\boldsymbol{\Sigma}$. Die Länge der Wörter ist beliebig, aber endlich, Je zwei Wörter aus $\boldsymbol{W}$ dürfen ein gemeinsames Teilwort haben, d.h. sie können sich überlappen.

Im Folgenden wird nun eine geeignete Problemdefinition entwickelt[28].

2.1 Shortest Common Supersequence Problem

Das Ziel ist es die gesamte DNA zu rekonstruieren, in dieser müssen also alle *reads* vorkommen. Gesucht ist demnach ein Wort $\boldsymbol{\omega}$, von dem alle anderen Wörter stammen. Das kann formal ausgedrückt werden indem man verlangt, dass jedes Wort aus $\boldsymbol{W}$ ein Teilwort von $\boldsymbol{\omega}$ ist.

Diese Problemstellung ist jedoch **zu ungenau**, denn sie erlaubt als Lösung die Konkatenation aller Wörter aus $\boldsymbol{W}$. Zum Beispiel ergäbe sich für $\boldsymbol{W}_1$ = {GGCTAAGTC, ACGGG, GGGCTAA} das Wort $\boldsymbol{w}$ = GGCTAAGTCACGGGGGGCTAA. Die unerwünschte Eigenschaft dieser Lösung ist, dass aus ihr andere Wörter abgeleitet werden können als in $\boldsymbol{W}$ vorhanden sind. Um die Menge der aus $\boldsymbol{\omega}$ ableitbaren Wörter möglichst identisch mit der Menge $\boldsymbol{W}$ zu halten, fasst man die sich überlappenden Teilwörter zusammen. So ergäbe sich zum Beispiel aus <u>GGCTAA</u>GTC und G<u>GGCTAA</u> das Wort G<u>GGCTAA</u>GTC.

Diese Überlegung kann in die Problemdefinition einbezogen werden, indem man nach dem **kürzesten** $\boldsymbol{\omega}$ sucht. Diese Problemdefinition ist auch als das *Shortest Common Supersequence* Problem (SCS) bekannt (analog heißt in dieser Arbeit $\boldsymbol{\omega}$ auch *Superwort*):

Gegeben sei eine Menge $\boldsymbol{W} \subseteq \boldsymbol{\Sigma}^{+}$ von Wörtern über dem Alphabet $\boldsymbol{\Sigma}$ = {A, C, G, T}.
Gesucht ist das kürzeste Wort $\boldsymbol{\omega} \in \boldsymbol{\Sigma}^{+}$, von dem alle Wörter aus $\boldsymbol{W}$ Teilwörter sind.

---

[27] vgl. VoWi11, S. 13-15
[28] Die Ausführungen dieses Kapitels gehen zurück auf: JoPe04, S. 264-271

Diese Problemdefinition führt dann beim obigen Beispiel $\boldsymbol{W}_1$ zur Lösung $\boldsymbol{\omega}$ = ACGGGCTAAGTC. Im Hinblick auf die Lösung des Problems der DNA-Sequenzierung ergeben sich leider trotzdem **zwei entscheidende Schwierigkeiten**:

1) Das Wort $\boldsymbol{\omega}$ muss **nicht eindeutig** sein.

Zum Beispiel sind für die Menge $\boldsymbol{W}_2$ = {AAA, AAT, ATA, ATT, TAA, TAT, TTA, TTT} sowohl das Wort AAATTTATAA als auch das Wort AAATATTTAA kürzeste Superworte. Die Reihenfolge der Basen in der DNA muss allerdings eindeutig bestimmt werden.

2) Das Problem SCS ist **NP-vollständig**.

Im Folgenden wird die Grundidee einer Reduktion von SCS auf das *Traveling Salesman Problem* (TSP) erläutert. Daraus folgt nicht, dass SCS ein NP-vollständiges Problem ist, denn dafür müsste TSP in polynomieller Zeit auf SCS reduziert werden. Allerdings plausibilisiert die Existenz der Transformation dass auch SCS ein NP-vollständiges Problem ist[29]. Das bedeutet, dass es (vermutlich) keinen effizienten Algorithmus zur Lösung dieser Probleme gibt.
Man behandle jedes Wort aus $\boldsymbol{W}$ als einen Knoten in einem gerichteten Graphen. Alle Knoten werden untereinander verbunden, d.h. der Graph ist *vollständig*. Auf diese Weise lassen sich alle Wörter miteinander vergleichen und kombinieren. Die Länge einer Kante von einem Knoten $\boldsymbol{s}$ zu einem Knoten $\boldsymbol{t}$ sei gegeben als $|\mathtt{overlap}(\boldsymbol{s},\ \boldsymbol{t})|$ * –1. Das Wort $\boldsymbol{\omega}$ zu finden bedeutet die Wörter so miteinander zu verschmelzen, dass sich die Wörter größtmöglich überlappen. Denn dann produziert die Verschmelzung möglichst kleine Wörter. $\boldsymbol{\omega}$ entspricht also einem Pfad, der jeden Knoten des Graphen enthält und eine minimale Gesamtlänge hat.

Im Endeffekt ist die DNA-Sequenzierung mithilfe der Modellierung als SCS also **weder eindeutig noch effizient lösbar**.

Der Vollständigkeit halber sei angeführt, dass es einen Greedy-**Algorithmus** gibt, der in polynomieller Zeit eine angenäherte Lösung des SCS produziert. In diesem werden wiederholt die Wörter zusammengefasst, die sich am stärksten überlappen.

`while` $\boldsymbol{W}$ enthält mehr als ein Wort:
- berechne für alle Wörter $\boldsymbol{s}$ und $\boldsymbol{t}$ in $\boldsymbol{W}$ $|\mathtt{overlap}(\boldsymbol{s},\boldsymbol{t})|$
- ersetzte in $\boldsymbol{W}$ die Wörter $\boldsymbol{u}$ und $\boldsymbol{v}$, bei denen $|\mathtt{overlap}(\boldsymbol{u},\boldsymbol{v})|$ den maximalen Wert annimmt, durch ihr kürzestes Superwort

Dieses Kapitel hat gezeigt, dass eine Modellierung der DNA-Sequenzierung als *Shortest Common Supersequence* Problem eine **zu allgemeine Problemdefinition** darstellt. Diese Modellierung zwingt sich für die bisher beschriebenen Sequenzierungsverfahren aber auf.
Wie das nächste Kapitel darlegt, besteht aufgrund eines anderen experimentellen Verfahrens die Möglichkeit das Problem auf eine andere Art zu definieren. Bei dieser zweiten Problemdefinition handelt es sich um eine Spezialisierung des SCS, wie im Kapitel 2.2.2 ausgeführt wird.

2.2 Sequencing by Hybridization

Die **Sequenzierung durch Hybridisierung** (engl. SBH) ist eine alternative Sequenzierungsmethode. Sie beruht im Wesentlichen auf dem Einsatz von *DNA-Arrays* (auch *DNA-Chips*), welche im Kapitel 2.2.1 erläutert werden. Das Kapitel 2.2.2 formalisiert den zweiten Prozessschritt als Problem des Zusammensetzens der durch die DNA-Arrays identifizierten Fragmente.

[29] ergänzend beweist RäUk80, dass SCS NP-vollständig ist für jedes Alphabet mit mehr als einem Buchstaben

2.2.1 DNA-Arrays

Die Idee der DNA-Arrays stammt aus dem Jahr 1988 und ist, dass in einer zu untersuchenden DNA-Sequenz (*Ziel-DNA*, engl. *target DNA*) **bekannte Teilsequenzen identifiziert** werden. Das Verfahren funktioniert folgendermaßen:

Auf einer Oberfläche wird eine Menge von Nukleotid-Sequenzen angebracht, welche allen möglichen Permutationen einer festen Länge $\boldsymbol{l}$ der vier Basen entspricht. Die **Positionen der Permutationen** sind bekannt und folgen einem Matrix-Muster. Die Die Ziel-DNA wird nun mit einem fluoreszierenden Stoff verbunden und mit dem Array in Kontakt gebracht. Aufgrund der Komplementarität der Basen (vgl. Kapitel 1) kann die Ziel-DNA mit den Permutationen hybridisieren. Das heißt sie geht mit jenen Permutationen eine chemische Verbindung ein, die ein *Watson-Crick-Komplement* einer Teilsequenz in der Ziel-DNA sind. Man kann anhand der Markierung feststellen an welchen Stellen des Array eine Hybridisierung stattfand. Weil die Basenabfolge der Permutation an den einzelnen Positionen bekannt ist, kann daraus geschlossen werden welche Teilsequenzen in der Ziel-DNA vorkommen.

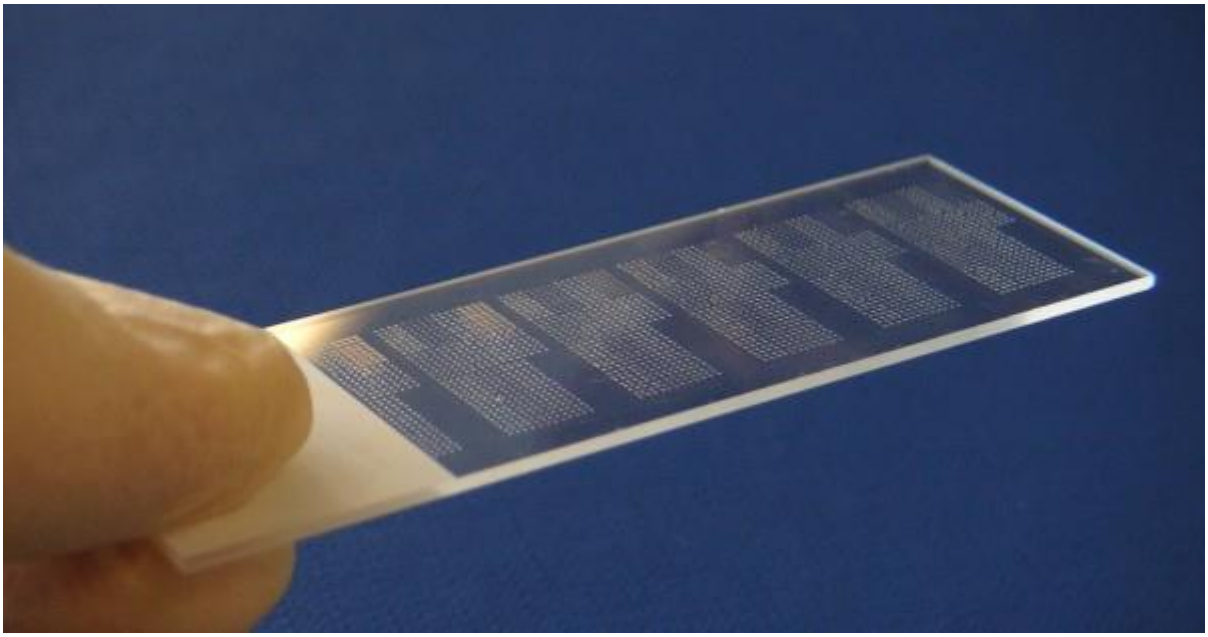

Abb. 2.1 – Foto eines DNA-Arrays[30]

Das **Generieren der Permutationen** erschien zunächst praktisch nicht durchführbar, weil dafür $4^{\boldsymbol{l}}$ Reaktionen benötigt worden wären. Dank einer Weiterentwicklung der Biotechnologien im Jahr 1991, der *light-directed polymer synthesis* von Steve Fodor, konnte die Anzahl der benötigten Reaktionen auf $4\boldsymbol{l}$ reduziert werden. Dadurch wurden DNA-Arrays praktisch einsetzbar.

2.2.2 SBH Problemdefinition

An dieser Stelle wird erneut die Brücke zur Informatik geschlagen und über die Ergebnisse des DNA-Array Verfahrens abstrahiert: Ein DNA-Array ermittelt zu einem Wort $\boldsymbol{w}$ alle Teilwörter einer festen Länge $\boldsymbol{l}$. Die **Menge aller Teilwörter** der Länge $\boldsymbol{l}$ wird angegeben als $\texttt{spectrum}(\boldsymbol{w}, \boldsymbol{l})$.
Mithilfe dieser Funktion lässt sich die formale Problemdefinition für die Sequenzierung durch Hybridisierung ableiten:

[30] Abb. entnommen aus *https://www.cancerquest.org/index.php/node/6241*, zugegriffen am 13.11.2018

Gegeben sei eine Menge $\boldsymbol{W} \subseteq \boldsymbol{\Sigma}^{l}$ von Wörtern über dem Alphabet $\boldsymbol{\Sigma}$ = {A, C, G, T}.
Gesucht ist das Wort $\boldsymbol{\omega} \in \boldsymbol{\Sigma}^{+}$, für das gilt: $\texttt{spectrum}(\boldsymbol{\omega}, \boldsymbol{l}) = \boldsymbol{W}$.

An dieser Definition ist erkennbar, dass das Problem SBH dem SCS sehr ähnlich ist. Tatsächlich verlangt das SBH zusätzlich, dass alle Teilwörter die gleiche Länge $\boldsymbol{l}$ haben, und dass alle Teilwörter in $\boldsymbol{W}$ enthalten sind. Bei SBH handelt es sich also um eine **Spezialisierung** des allgemeineren SCS Problems.

Die zusätzlichen Einschränkungen, die SBH stellt, mögen auf den ersten Blick marginal erscheinen. Tatsächlich führen sie aber dazu, dass **SBH nicht NP-vollständig** ist, sondern sich effizient lösen lässt. Ein solcher Algorithmus wird im Kapitel 3 entwickelt.

Zur Veranschaulichung wird das **Beispiel** $\boldsymbol{W}_1$ (vgl. Kapitel 2.1) nochmal aufgegriffen und als SBH Problem formuliert. Sei also $\boldsymbol{W}_3$ = {ACGGGC, CGGGCT, CTAAGT, GCTAAG, GGCTAA, GGGCTA, TAAGTC}. Daraus ergibt sich das Wort $\boldsymbol{\omega}$ = ACGGGCTAAGTC.
Dieses Beispiel lässt sich wegen den prägnanten Teilwörtern GGG und AA noch durch scharfes Hinsehen lösen. Einen systematischen Ansatz stellt das Kapitel 3 vor.

Hinweis: Ursprünglich wurden DNA-Arrays entwickelt als eine Alternative zu bisherigen Sequenzierungsmethoden. In der Praxis gibt es aber ein großes Problem, weshalb sie für die DNA-Sequenzierung **nicht eingesetzt** werden[31]:
Es lässt sich keine Unterscheidung treffen zwischen *perfekten Übereinstimmungen*, d.h. Teilwörter die im Spektrum vorkommen, und *sehr stabilen Fehlzuordnungen*, d.h. Teilwörter die nicht im Spektrum vorkommen aber deren Nukleotidsequenzen genug chemisches Bindungspotenzial haben um eine Hybridisierung einzugehen (obwohl die Basenpaare kein Komplement sind). Die Fehlzuordnungen weichen meist nur um wenige Basen ab, aber da die Reihenfolge eindeutig und exakt zu bestimmen ist, sind diese kleinen Ungenauigkeiten nicht akzeptabel.
Nichtsdestotrotz finden DNA-Arrays in der Erforschung genetischer Variationen Verwendung. Dort wird angenommen dass die Sequenz der Ziel-DNA bekannt ist, bzw. nur wenige Unterschiede zwischen der Ziel-DNA und einer bekannten Sequenz bestehen. Auf diese Weise lassen sich Fehlzuordnungen durch einen Vergleich mit der erwarteten Sequenz relativ einfach erkennen.

Sequenzierungsmethoden der „nächsten Generation“ wie die *Nanoball-Sequenzierung* verwenden Ansätze, die mit dem der DNA-Arrays vergleichbar sind. Letztlich wurde die Idee von SBH damit doch noch praktikabel für den Einsatz zur DNA-Sequenzierung[32].

3 Algorithmen-Entwicklung

Dieses Kapitel entwickelt einen effizienten Algorithmus zur Lösung des SBH Problems[33]. Dazu verwendet es das mathematische Gebiet der Graphentheorie. Das Kapitel 3.1 entwickelt auf natürliche Weise einen Ansatz zur Problemlösung. Das Kapitel 3.2 führt diesen dann weiter.

[31] vgl. JoPe04, S. 270
[32] vgl. CoPe10, S. 23
[33] Die Ausführungen dieses Kapitels gehen zurück auf: JoPe04, S. 271-275

Im Kapitel 2 wurde das SCS Problem mit Hilfe der Graphentheorie gelöst. Da sich das allgemeinere SCS mit der Graphentheorie modellieren lässt, liegt es nahe auch die Lösung des SBH Problems mit Graphen zu modellieren. Dies bestätigt insbesondere der natürliche Ansatz.

Zunächst einmal werden **Graphen mathematisch definiert**[34]: Ein *Graph* ist ein Paar $(\boldsymbol{V}, \boldsymbol{E})$. $\boldsymbol{V}$ ist eine endliche Menge von Knoten, wobei in dieser Arbeit die Knoten durch ein Wort identifiziert werden. Es gilt also $\boldsymbol{V} \subseteq \boldsymbol{\Sigma}^+$ mit $\boldsymbol{\Sigma}$ = {A, C, G, T}. $\boldsymbol{E}$ ist eine Menge von Kanten zwischen je zwei Knoten des Graphen, es gilt $\boldsymbol{E} \subseteq \{ (\boldsymbol{u}, \boldsymbol{v}) \mid \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}\}$. In dieser Arbeit werden nur *gerichtete* Graphen betrachtet. Eine Kante $(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E}$ bedeutet also, dass ein Weg von $\boldsymbol{u}$ nach $\boldsymbol{v}$ verläuft aber nicht von $\boldsymbol{v}$ nach $\boldsymbol{u}$. Zwei Knoten, die durch eine Kante verbunden sind, heißen *adjazent*. Die Anzahl eingehender Kanten eines Knoten $\boldsymbol{u}$ wird durch die Funktion $\texttt{indegree}(\boldsymbol{u}) = |\{(\boldsymbol{v}_0, \boldsymbol{v}_1) \subseteq \boldsymbol{E} \mid \boldsymbol{v}_1 = \boldsymbol{u}\}|$ angegeben. Entsprechend gibt $\texttt{outdegree}(\boldsymbol{u}) = |\{(\boldsymbol{v}_0, \boldsymbol{v}_1) \subseteq \boldsymbol{E} \mid \boldsymbol{v}_0 = \boldsymbol{u}\}|$ die Anzahl ausgehender Kanten an.
Ein *Pfad* von Knoten $\boldsymbol{u}$ zu Knoten $\boldsymbol{w}$ ist eine Abfolge von Knoten, beginnend mit $\boldsymbol{u}$ und endend mit $\boldsymbol{w}$, bei der je zwei aufeinanderfolgende Knoten adjazent sind. Ein Pfad muss nicht eindeutig sein, er ist ein Element der Menge $\texttt{path}(\boldsymbol{u}, \boldsymbol{w}) = \{ (\boldsymbol{u} = \boldsymbol{v}_0, \boldsymbol{v}_1, \dots \boldsymbol{v}_j = \boldsymbol{w}) \mid (\boldsymbol{v}_i, \boldsymbol{v}_{i+1}) \in \boldsymbol{E}$ für $0 \leq \boldsymbol{i} < \boldsymbol{j}\}$.
An den Kanten können Label stehen, auch bei diesen handelt es sich in dieser Arbeit in der Regel um Wörter, d.h. $\texttt{label} : \boldsymbol{V} \rightarrow \boldsymbol{\Sigma}^+$. Mit der Definition als Funktion wird ausgedrückt, dass jede Kante genau ein Label hat. Die Funktion `label` wird üblicherweise auch *injektiv* sein (es gibt keine zwei Kanten mit gleichem Label) muss sie aber nicht.

3.1 Natürliche Modellierung

Betrachtet man die Konstruktion der Teilwörter in $\texttt{spectrum}(\boldsymbol{\omega}, \boldsymbol{l})$, so fällt auf, dass sich bestimmte Teilwörter von $\boldsymbol{\omega}$ immer um einen Buchstaben am Anfang und am Ende unterscheiden (Abb. 3.1). Diese Eigenschaft kann genutzt werden, um festzustellen ob zwei **Teilwörter aufeinander folgen**.

```
ACGGGCTAAGTC
ACGGGC
 CGGGCT
  GGGCTA
   GGCTAA
    GCTAAG
     CTAAGT
      TAAGTC
```

Abb. 3.1 – das Spektrum der Teilwörter in $\boldsymbol{W}_3$

Für Teilwörter der Länge $\boldsymbol{l}$ folgen zwei Teilwörter $\boldsymbol{s}$ und $\boldsymbol{t}$ aus dem Spektrum genau dann direkt aufeinander, wenn $|\texttt{overlap}(\boldsymbol{s}, \boldsymbol{t})| = \boldsymbol{l} - 1$ gilt. Das bedeutet also, dass zwei Teilwörter genau dann zusammengefasst werden dürfen, wenn die letzten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{s}$ mit den ersten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{t}$ übereinstimmen. Diese Bedingung ist die stärkste, weil garantiert ist, dass alle Teilwörter im Spektrum vorkommen (vgl. Kapitel 2.2.2). Es ist durch die Spezialisierung also einfach zu bestimmen welcher Teilwörter kombiniert werden müssen. $\boldsymbol{\omega}$ entspricht nun einer Abfolge von aufeinander folgenden Worten.

Die Vorgehensweise wird erneut am **Beispiel** $\boldsymbol{W}_3$ illustriert. Natürlicherweise wird man bei der Lösung mit einer übersichtlichen Darstellung wie in Abb. 3.2 arbeiten. In dieser entsprechen die Teilwörter Knoten eines Graphen, und wenn die obige Bedingung erfüllt ist, dann werden zwei Knoten durch eine gerichtete Kante verbunden.

[34] vgl. MaPo17, S 75

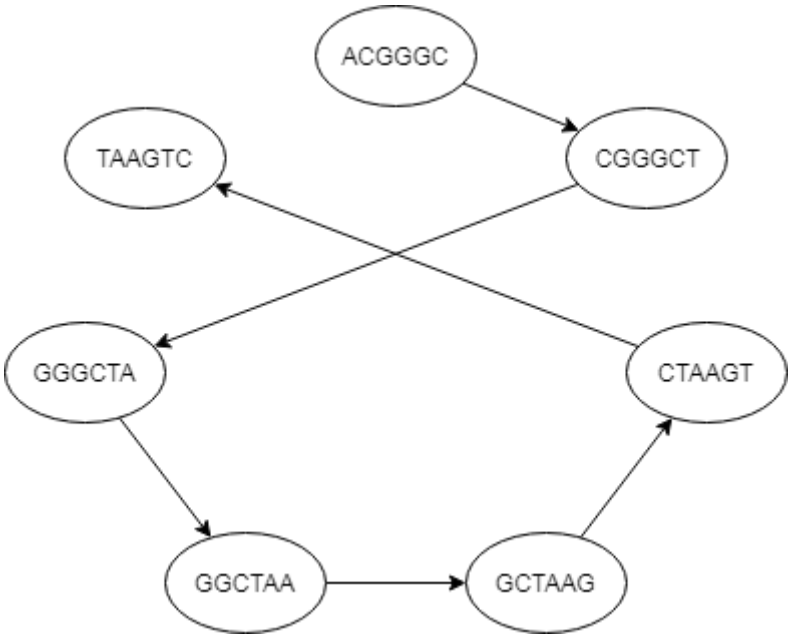

Abb. 3.2 –Graph zur Lösung des Beispiels $\boldsymbol{W}_3$

Es ergibt sich im Beispiel $\boldsymbol{W}_3$ für die Teilwörter ein Pfad, welcher einem $\boldsymbol{\omega}$ = ACGGGCTAAGTC entspricht. Die Ermittlung von $\boldsymbol{\omega}$ geschieht so, dass die ersten $\boldsymbol{l}$ Buchstaben dem ersten Teilwort entsprechen und in jedem nachfolgenden Teilwort des Pfades der letzte Buchstabe des Teilworts an den Zwischenstand von $\boldsymbol{\omega}$ angehängt wird[35].

Leider kann es bei dieser Vorgehensweise, wie auch schon bei SCS, zu **Uneindeutigkeiten** kommen. Das illustriert das folgende Beispiel mit $\boldsymbol{W}_4$ = {ATG, CGT, GCA, GCG, GGC, GTG, TGC, TGG}:

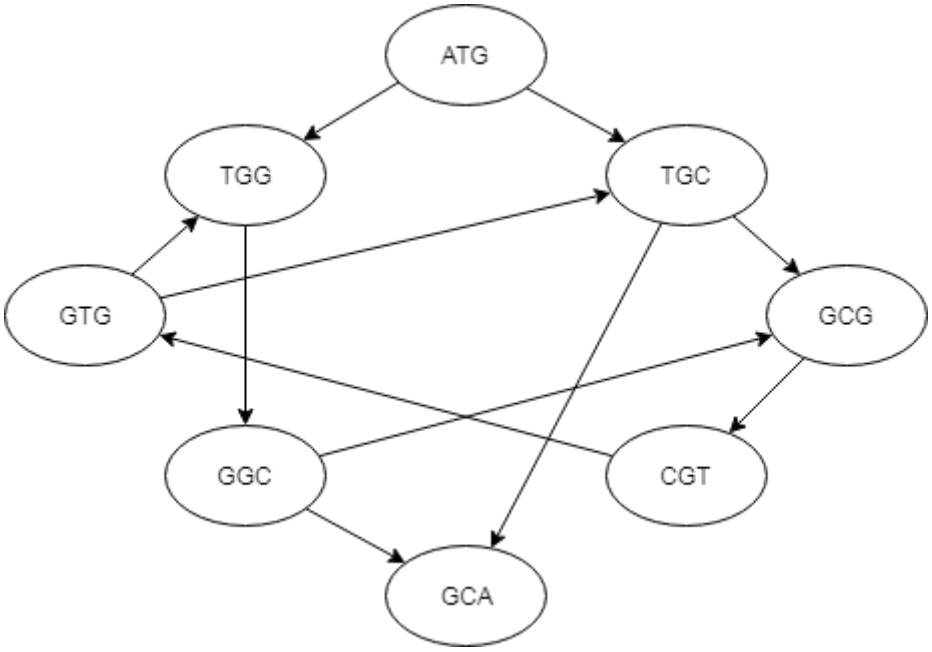

Abb. 3.3 – Graph zu $\boldsymbol{W}_4$ mit zwei möglichen Lösungen

In $\boldsymbol{W}_4$ können sich für die Teilwörter zwei Pfade ergeben. Zum einen $\boldsymbol{p}_1$ = (ATG, TGC, GCG, CGT, GTG, TGG, GGC, GCA), dies entspricht einem $\boldsymbol{\omega}$ = ATG<u>CGTG</u>GCA, und zum anderen der Pfad $\boldsymbol{p}_2$ = (ATG, TGG,

[35] vgl. CoPe10, S. 14

GGC, GCG, CGT, GTG, TGC, GCA) welche einem $\boldsymbol{\omega}$ = ATGGCGTGCA entspricht. Wie auch schon bei SCS ist solch eine Uneindeutigkeit unerwünscht, in Kürze wird aber noch ein größeres Problem offenbar.

Im obigen Graph wären auch die Pfade $\boldsymbol{p}_3$ = (ATG, TGC, GCA) oder $\boldsymbol{p}_4$ = (ATG, TGG, GGC, GCA) möglich. Denn beides sind gültige Pfade vom *Startknoten* ATG (dieser hat ein `indegree` von 0) zum *Endknoten* GCA (`outdegree` von 0) des obigen Graphen. Die Pfade $\boldsymbol{p}_3$ und $\boldsymbol{p}_4$ sind allerdings keine Lösungen für das SBH Problem, denn die Spektren der durch diese Pfade angegebenen $\boldsymbol{\omega}$ (ATGCA bzw. ATGGCA) enthalten nicht **alle Teilwörter**, die in $\boldsymbol{W}_4$ vorkommen. Das ist bei den Pfaden $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ anders, denn in diesen Pfaden sind alle Knoten des Graphen in Abb. 3.3 enthalten. Weil die Knoten genau den im Spektrum vorkommenden Teilwörtern entsprechen, sind die $\boldsymbol{\omega}$ der Pfade $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ valide Lösungen.

Ausgehend von dieser Beobachtung wird nun der **Algorithmus** zur Berechnung von $\boldsymbol{\omega}$ formuliert:

- erstelle einen Graphen $\boldsymbol{G}$ mit den Teilwörtern als Knoten und verbinde zwei Knoten $\boldsymbol{s}$ und $\boldsymbol{t}$ mit einer gerichteten Kante, wenn sie die Bedingung $|\texttt{overlap}(\boldsymbol{s},\boldsymbol{t})| = \boldsymbol{l} - 1$ erfüllen
- finde in $\boldsymbol{G}$ einen Pfad, in dem alle Knoten genau einmal vorkommen

Bei der letzten Aufgabe handelt es sich um das Finden eines **Hamiltonpfades**. Dieses ist ebenfalls ein NP-vollständiges Problem, sodass der Algorithmus, und damit auch die vorgenommene Modellierung, nicht praxistauglich sind.
Um genau zu sein schränkt das Suchen eines Hamiltonpfades die Problemstellung SBH noch zusätzlich ein. Denn SBH sucht ein Wort $\boldsymbol{\omega}$, in dem jedes Wort aus $\boldsymbol{W}$ mindestens einmal und nicht genau einmal vorkommt. Das Suchen nach einem Hamiltonpfad impliziert also, dass $\boldsymbol{\omega}$ keine Wiederholungen enthält. Dieser Sachverhalt wird im Kapitel 3.3 noch genauer erläutert.

3.2 Zweite graphentheoretische Modellierung

Im vorigen Kapitel sind die Teilwörter des Spektrums als Knoten modelliert worden. In diesem Kapitel wird daher nun eine Modellierung mit den **Teilwörtern als Kanten** versucht.

Wenn die Teilwörter als Label der Kanten auftauchen, stellt sich die Frage was der **Inhalt der Knoten** ist. Der Grundgedanke ist es die Teilwörter noch weiter zu zerlegen. Dann kann über die gemeinsamen Präfixe und Suffixe die Reihenfolge der Teilwörter bestimmt werden:
Es ist bekannt, dass ein Teilwort $\boldsymbol{t}$ dann als Nachfolger eines Teilwortes $\boldsymbol{s}$ gilt, wenn die letzten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{s}$ mit den ersten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{t}$ übereinstimmen (vgl. Kapitel 3.1). Diese Eigenschaft lässt sich umdrehen: ein Teilwort $\boldsymbol{w}$ der Länge $\boldsymbol{l}$ ist dann eine gerichtete Kante $(\boldsymbol{u}, \boldsymbol{v})$ zwischen den Knoten $\boldsymbol{u}$ und $\boldsymbol{v}$, wenn die ersten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{w}$ mit $\boldsymbol{u}$ übereinstimmen und die letzten $\boldsymbol{l}$ - 1 Buchstaben mit $\boldsymbol{v}$. Demnach entsprechen die Knoten Teilwörtern der Länge $\boldsymbol{l} - 1$.

Formal gilt also: die Wertmenge der Funktion `label` ist die Menge $\boldsymbol{W}$ der Teilwörter aus $\texttt{spectrum}(\boldsymbol{\omega}, \boldsymbol{l})$. Die Menge der Knoten $\boldsymbol{V}$ entspricht der Vereinigungsmenge aller Präfixe und Suffixe der Länge $\boldsymbol{l} - 1$ von den Wörtern in $\boldsymbol{W}$ ($\boldsymbol{V} = \{\boldsymbol{v} \mid \boldsymbol{v} \in \texttt{spectrum}(\boldsymbol{w}, \boldsymbol{l} - 1) \wedge \boldsymbol{w} \in \boldsymbol{W}\}$). Die Menge der Kanten ist $\boldsymbol{E} = \{ (\boldsymbol{v}_1, \boldsymbol{v}_2) \in \boldsymbol{V} \times \boldsymbol{V} \mid \exists \boldsymbol{w} \in \boldsymbol{W}: |\texttt{overlap}(\boldsymbol{v}_1, \boldsymbol{w})| = |\texttt{overlap}(\boldsymbol{w}, \boldsymbol{v}_2)| = \boldsymbol{l} - 1\}$.

Das Wort $\boldsymbol{\omega}$ muss alle Teilwörter enthalten. Da die Teilwörter nun den Kanten entsprechen, gilt es einen Pfad zu ermitteln, der alle Kanten genau einmal (vgl. Kapitel 3.3) beinhaltet. Dieses Problem ist bekannt als das Finden eines **Eulerpfades**. Im Gegensatz zum Problem einen Hamiltonpfad zu finden, ist das Finden eines Eulerpfades kein NP-vollständiges Problem, sondern es lässt sich in polynomieller Zeit lösen. Weil den Seminarteilnehmern die Ermittlung eines Eulerpfades bereits bekannt ist, wird hier auf eine eingehendere Erklärung verzichtet. Interessierte seien verwiesen auf die Erläuterungen in [JoPe07] Kapitel 8.8 oder [CoPe10] Kapitel 2 und 5.

Um SBH dadurch lösen zu können, dass man in dem Graphen, welcher sich aus der obigen Konstruktionsbeschreibung ergibt einen Eulerpfad findet, muss sichergestellt sein dass der Graph immer einen Eulerpfad enthält. Dazu eignen sich die Erkenntnisse des niederländischen Mathematikers Nicolaas de Bruijn[36]. De Bruijn analysierte eine spezielle Klasse von Graphen. Für ein beliebiges Alphabet $\boldsymbol{\Sigma}$ mit $\boldsymbol{n}$ Buchstaben und eine natürliche Zahl $\boldsymbol{l}$ bestehen die Knoten des **de Bruijn Graph** $\boldsymbol{\mathcal{B}(n, \Sigma)}$ aus der Menge $\boldsymbol{V}$ aller möglichen Wörter der Länge $\boldsymbol{l} - 1$ über dem Alphabet $\boldsymbol{\Sigma}$. Zwei Knoten $\boldsymbol{u}$ und $\boldsymbol{v}$ werden genau dann durch eine gerichtete Kante $\boldsymbol{k}$ verbunden, wenn $\boldsymbol{u}$ ein Suffix und $\boldsymbol{v}$ ein Präfix eines Wortes $\boldsymbol{k}$ der Länge $\boldsymbol{l}$ ist. Diese Definition beschreibt (mit $\boldsymbol{n}$ = 4 und entsprechendem $\boldsymbol{\Sigma}$) dieselbe Art von Graph wie die obige Konstruktionsbeschreibung.

Es wird nun **bewiesen**, dass jeder de Bruijn Graph einen Eulerpfad enthält. Betrachtet wird ein beliebiger Knoten $\boldsymbol{v}$, welcher einem Wort der Länge $\boldsymbol{l} - 1$ entspricht. Es existieren $\boldsymbol{n}$ Knoten zu denen man von $\boldsymbol{v}$ gelangt (löschen des ersten Buchstabens und anfügen eines der $\boldsymbol{n}$ Buchstaben an $\boldsymbol{v}$), daher ist $\texttt{outdegree}(\boldsymbol{v}) = \boldsymbol{n}$. Ebenso gibt es genau $\boldsymbol{n}$ Kanten über die man zu $\boldsymbol{v}$ gelangt (letzten Buchstaben löschen und einen der $\boldsymbol{n}$ möglichen vorne anhängen). Es gilt also $\texttt{indegree}(\boldsymbol{v}) = \texttt{outdegree}(\boldsymbol{v}) = \boldsymbol{n}$. Nach dem zweiten Theorem von Euler[37] besitzt somit jeder de Bruijn Graph einen Eulerkreis. Da es sich bei einem Eulerkreis um einen Spezialfall eines Eulerpfades handelt (Start- und Endknoten des Pfades sind identisch) besitzt somit jeder de Bruijn Graph einen Eulerpfad.

Aber Vorsicht ist geboten. Denn die obige Konstruktionsvorschrift für den Graphen ist abhängig vom Wert von $\boldsymbol{\omega}$. Es wird nur die spezielle Sequenz von $\boldsymbol{\omega}$ in das Spektrum der Teilwörter mit der Länge $\boldsymbol{l} - 1$ zerlegt. In diesem Spektrum müssen nicht alle möglichen Teilwörter der Länge $\boldsymbol{l} - 1$ über dem Alphabet vorkommen. Zum Beispiel fehlen im Graphen des Beispiels $\boldsymbol{W}_4$ (vgl. Abb. 3.4) die Knoten AA, AC, AG, CC, CT, GA, TA, TC und TT um aus ihm einen de Bruijn Graphen zu machen. Das heißt konkret, dass die obige Konstruktionsvorschrift einen Graphen erzeugt, der ein **Untergraph** des zugehörigen de Bruijn Graph ist. Die Literatur nimmt diese Unterscheidung nicht immer vor, bzw. definiert einen de Bruijn Graph entsprechend obiger Konstruktion. Diese Arbeit nennt den Graphen, der sich aus der Konstruktionsvorschrift ergibt, *de Bruijn-artig*.

Auch für de Bruijn-artige Graphen lässt sich **beweisen**, dass sie immer einen Eulerpfad enthalten[38]. Begründet liegt dies in der Konstruktionsvorschrift. Abgesehen vom ersten und letzten Teilwort in $\boldsymbol{\omega}$ hat jedes Teilwort $\boldsymbol{u}$ genau einen Vorgänger und einen Nachfolger (vgl. Abb. 3.1). Es gilt also $\texttt{indegree}(\boldsymbol{u}) = \texttt{outdegree}(\boldsymbol{u})$. Dabei gibt es eine Ausnahme, und zwar wenn ein Teilwort mehrfach vorkommt. In diesem Fall hat es nicht genau einen Vorgänger und einen Nachfolger, sondern für jedes Vorkommen einen Vorgänger und einen Nachfolger. Daher bleibt auch bei mehrfachem Auftreten eines Teilworts das Verhältnis gewahrt (vgl. Abb. 3.4). Zusammenfassend haben also nur der Start- und der Endknoten einen Vorgänger- bzw. Nachfolgerknoten „zu wenig". Indem man eine zusätzliche Kante vom End- zum Startknoten einfügt, ist Eulers zweites Theorem erfüllt. Insofern muss ein Eulerpfad vom Start- zum Endknoten existieren.

Im Folgenden wird der **Algorithmus** zur Berechnung von $\boldsymbol{\omega}$ mittels eines Eulerpfades formuliert:

- berechne aus $\boldsymbol{W}$ die Menge der Knoten $\boldsymbol{V} = \{ \boldsymbol{v} \mid \boldsymbol{v} \in \texttt{spectrum}(\boldsymbol{w}, \boldsymbol{l} - 1) \wedge \boldsymbol{w} \in \boldsymbol{W} \}$ und der Kanten $\boldsymbol{E} = \{ (\boldsymbol{v}_1, \boldsymbol{v}_2) \in \boldsymbol{V} \times \boldsymbol{V} \mid \exists \boldsymbol{w} \in \boldsymbol{W}: |\texttt{overlap}(\boldsymbol{v}_1, \boldsymbol{w})| = |\texttt{overlap}(\boldsymbol{w}, \boldsymbol{v}_2)| = \boldsymbol{l} - 1 \}$
- finde im de Bruijn-artigen Graphen $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ einen Eulerpfad
- leite das Wort $\boldsymbol{\omega}$ aus dem gefundenen Pfad ab: die ersten $\boldsymbol{l} - 1$ Buchstaben von $\boldsymbol{\omega}$ entsprechen dem Startknoten und für jeden weiteren Knoten wir der letzte Buchstabe an $\boldsymbol{\omega}$ angehängt

[36] zu de Bruijn Graphen vgl. CoPe10, S. 17-19

[37] vgl. CoPe10, S. 12. Das Theorem besagt ein zusammenhängender, gerichteter Graph $\boldsymbol{G}$ enthält einen Eulerkreis genau dann, wenn für alle Knoten $\boldsymbol{v}$ in $\boldsymbol{G}$ indegree($\boldsymbol{v}$) = outdegree($\boldsymbol{v}$) gilt.

[38] dieser eher informelle Beweis entstammt den Überlegungen des Autors

3.3 Grenzen der Problemdefinition

Der Abschnitt über de Bruijn Graphen hat ein Detail angesprochen, welches in diesem Kapitel bisher nicht näher thematisiert wurde: Wiederholungen von Teilwörtern. Diese haben keinen Einfluss auf das Vorhandensein eines Eulerpfades in den de Bruijn-artigen Graphen. Aber sie führen dazu, dass es auch bei Eulerpfaden zu **Uneindeutigkeiten** kommen kann. Dazu wird das schon bei Hamiltonpfaden verwendete Beispiel $\boldsymbol{W}_4$ herangezogen. Für $\boldsymbol{W}_4$ = {ATG, CGT, GCA, GCG, GGC, GTG, TGC, TGG} ergeben sich auch im de Bruijn-artigen Graph die gleichen Pfade und Lösungswörter $\boldsymbol{\omega}$ = ATGCGTGGCA bzw. $\boldsymbol{\omega}$ = ATGGCGTGCA wie schon zuvor bei der ersten Modellierung.

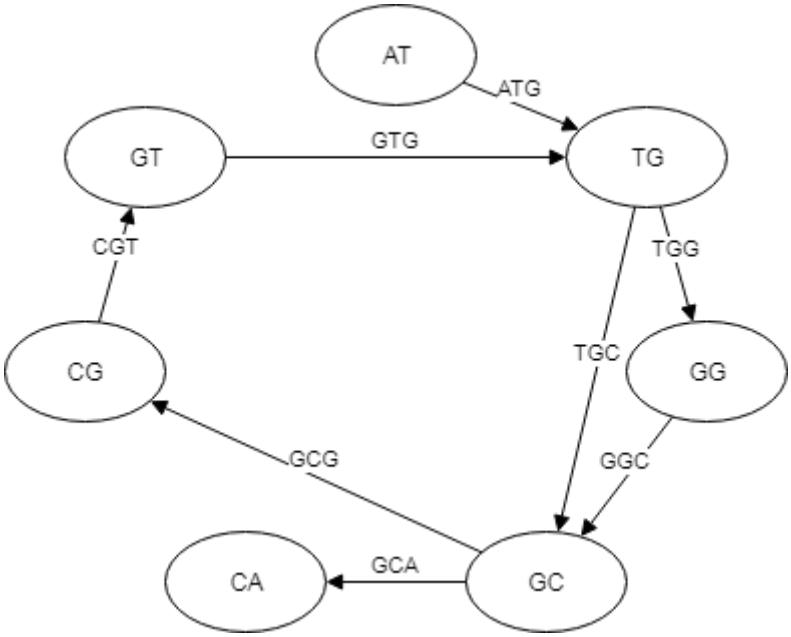

Abb. 3.4 – de Bruijn-artiger Graph zu $\boldsymbol{W}_4$

Für das Problem, dass es mehrere Eulerpfade in einem Graphen gibt (Uneindeutigkeit) gibt es mit den bisherigen Mitteln keine Lösung. [Kap08] führt daher eine **Wahrscheinlichkeit** ein, dass ein gefundener Eulerpfad „der Richtige“ ist. Diese entspricht genau 1 geteilt durch die Anzahl aller möglichen Eulerpfade im Graphen. Zur Ermittlung der Anzahl kann das *BEST-Theorem* herangezogen werden, benannt nach seinen Entdeckern de **B**ruijn, van Aardenne-**E**hrenfest, **S**mith und **T**utte. Das Theorem führt die Anzahl der Eulerkreise eines Graphen $\boldsymbol{G}$ auf die Anzahl der spannenden Bäume in $\boldsymbol{G}$ zurück. Diese wiederum lässt sich mit dem *matrix tree Theorem* ermitteln. Für den Zweck dieser Arbeit genügt es zu wissen, dass die Lösung nur unter einer bestimmten Wahrscheinlichkeit korrekt ist und dass diese Wahrscheinlichkeit berechnet werden kann. Geneigte Leser finden die Details und Beweise der Theoreme in [Kap08] S. 4-12.

Das Beispiel $\boldsymbol{W}_4$ ist insofern interessant, als es auch ohne Wiederholungen von Teilwörtern in $\boldsymbol{W}_4$ eine Uneindeutigkeit hervorruft. Diese entsteht erst in der Knotenmenge $\boldsymbol{V}$ für TG und GC. Tatsächlich stellen **Wiederholungen** das zentrale Problem in der DNA-Sequenzierung dar.

In der Theorie lassen sich Wiederholungen von Teilwörtern durch Multimengen abbilden. Zu diesem Zweck wird die Funktion `spectrum` erweitert, sodass sie nicht mehr eine Menge, sondern eine **Multimenge** von Teilwörtern erzeugt. In einer Multimenge dürfen Elemente mehrfach vorkommen, sie wird häufig mit eckigen Klammern notiert. Als Datenstruktur kann eine (ungeordnete) Liste verwendet werden, welche meistens auch mit eckigen Klammern notiert wird. In der Graphentheorie werden Multimengen eingesetzt, um Mehrfachkanten zu erlauben. Eine Mehrfachkante tritt auf, wenn

zwei Knoten durch mehrere Kanten verbunden sind. In diesem Fall ist die Menge der Kanten $\boldsymbol{E}$ eine Multimenge. Mehrfachkanten im de Bruijn-artigen Graph zeigen also eine Wiederholung an.

Als Wiederholungen noch nicht betrachtet wurden, konnte verlangt werden, dass $\boldsymbol{\omega}$ einem Hamilton- bzw. einem Eulerpfad entspricht. Wenn das Problem SBH von wiederholungsfreien Wörtern auf Wörter, die Wiederholungen enthalten, **verallgemeinert** werden soll, dann sind Multimengen bzw. Mehrfachkanten ein hilfreiches Mittel.

Der Vorteil von Mehrfachkanten ist, dass, wenn die Anzahl der Wiederholungen eines Teilworts in $\boldsymbol{\omega}$ bekannt ist, $\boldsymbol{\omega}$ weiterhin durch einen Eulerpfad ermittelt werden kann. Mehrfachkanten sind in diesem Fall sogar notwendig, denn das folgende Beispiel zeigt, dass ohne diese unter Umständen keine Lösung gefunden werden kann: Sei $\boldsymbol{\omega}$ = AGCAGGCA, dann ist `spectrum'(`$\boldsymbol{\omega}$`,3)` = [AGC, AGG, CAC, CAG, GCA, GCA, GGC]. Aus dieser Multimenge ergibt sich ein eindeutiger Eulerpfad, wenn sich die Multiplizität der Teilwörter in einer entsprechenden Anzahl von Kanten im de Bruijn-artigen Graphen niederschlägt (Abb. 3.5 links). Mit einer einfachen Menge $\boldsymbol{W}_5$ = { AGC, AGG, CAC, CAG, GCA, GGC } ist in dem daraus resultierenden Graphen (Abb. 3.5 rechts) kein Eulerpfad bestimmbar. Denn die Kante GCA, die einer Wiederholung entspricht, muss mehrfach verwendet werden.

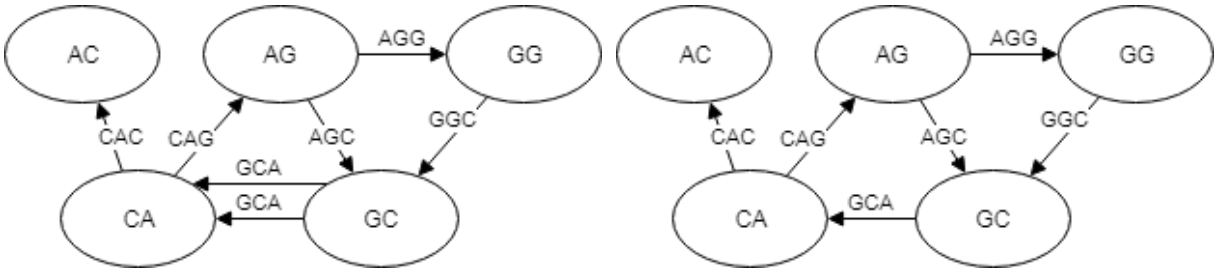

Abb. 3.5 – Unterschied zwischen Multimenge und Menge

Das Problem in der Praxis ist, dass sich die Multiplizitäten der Teilwörter durch das experimentelle SBH-Verfahren **nicht bestimmen lassen**.
Wenn darauf verzichtet wird das Problem mit einem Eulerpfad zu lösen, aber man an der Modellierung als de Bruijn-artiger Graph festhält, dann führt dies dazu, dass ein Pfad bestimmt werden muss in dem jede Kante mindestens einmal vorkommt. Dieser Pfad sollte eine minimale Länge haben, wobei minimal bedeutet, dass möglichst wenige Kanten mehrfach verwendet werden. Das führt dazu, dass das sich aus dem Pfad ergebende $\boldsymbol{\omega}$ so kurz wie möglich ist (vgl. Kapitel 2.1). Dieses Problem ist in der Literatur als **Chinese Postman Problem** (CPP) bekannt und polynomiell lösbar[39]. Genau genommen sucht CPP einen kürzesten Kreis. Gibt man aber jeder Kante die gleiche Länge und verbindet End- und Startknoten, so entspricht dies obiger Definition. CPP ist eine Generalisierung des Eulerpfad Problems, denn wenn keine Kante mehrfach verwendet wird, enthält der minimale CPP-Pfad jede Kante genau einmal, er ist also ein Eulerpfad. Insofern lassen sich die unbekannten Multiplizitäten durch das Lösen von CPP statt dem Eulerpfad Problem abbilden. Eine Lösung des CPP ist entsprechend in eine Lösung des Problems Eulerpfad überführbar, wenn eine im CPP-Pfad $\boldsymbol{n}$-mal benutzte Kante zu einer $\boldsymbol{n}$-fachen Mehrfachkante wird. CPP ist außerdem das Äquivalent zum TSP für Kanten statt Knoten und mehrere Besuche einer Kante.

Wenn Wiederholungen existieren, dann kann es beim Modellieren des Problems als SCS dazu kommen, dass das Superwort $\boldsymbol{\omega}$ eine Wiederholung nur einmal enthält (***over-collapsing***) [40]. Sucht man einen minimalen (Chinese Postman oder Euler-) Pfad im de Bruijn-artigen Graph, kann das Problem nicht auftreten. Denn je Vorkommen einer Wiederholung gibt es eine Kante, die zur Wiederholung führt, und von diesen müssen alle benutzt werden (vgl. Kapitel 4.2.2).

[39] vgl. EdJo73, S. 88f
[40] vgl. MGMB07, S. 1f

Das Kapitel 4.2 stellt einen Lösungsansatz auf der Basis des Problems CPP vor. Des Weiteren gibt es verschiedene weitere Lösungsansätze mit dem Problem von Wiederholungen umzugehen. Einige davon werden im Folgenden kurz vorgestellt[41]:

Ein Lösungsansatz ist die sogenannte **BAC-by-BAC-Strategie**. In dieser wird zunächst das Genom in eine Reihe von überlappenden Sequenzen mit einer Länge von etwa 150.000 Nukleotiden aufgeteilt. Dann ermittelt man die Reihenfolge dieser langen Sequenzen. Anschließend werden diese Sequenzen in BACs (*Bacterial Artificial Chromosomes*) geklont und können separat sequenziert werden. Diese hierarchische Strategie hat den Vorteil, dass die Anzahl der Wiederholungen drastisch sinkt (um den Faktor 30.000 bei Menschen, da dort etwa so viele BACs benötigt werden). Allerdings wird die Sequenzierung insgesamt langwierig und mühseliger. Dieser Ansatz wurde im HGP verwendet.

Die Idee des zweiten Ansatzes ist es virtuell die Länge der *reads* zu erhöhen, indem *reads* verpaart werden, zwischen denen eine Lücke fester Länge liegt. Dieser Ansatz resultierte in der **mate-pair reads** Sequenzierungstechnik. In dieser werden Sequenzen einer Länge $\boldsymbol{l}$ ausgewählt und beide Enden sequenziert. Das ergibt dann ein Paar von *reads* die ungefähr die Lücke $\boldsymbol{l}$ haben. $\boldsymbol{l}$ wird so gewählt, dass es länger ist als die meisten Wiederholungen im Genom. Daher ist es unwahrscheinlich, dass beide *reads* des Paares in einer größeren Wiederholungssequenz liegen. Auf diese Weise bestimmt der Partner, der in einer eindeutigen Sequenz liegt, in welcher Wiederholung der andere Partner liegt.

Der dritte Ansatz nennt sich **whole-genome Shotgun Sequencing**[42]. In diesem wird das gesamte Genom fragmentiert, die Fragmente einzeln sequenziert und die *reads* dann zu der Basensequenz des Genoms zusammengesetzt. Dieser Ansatz ist für Genome mit bis zu einigen Millionen Nukleotiden erfolgreich, das sind zum Beispiel Genome von Bakterien.

Es ist wichtig **Wiederholungen nach ihrer Länge** zu **unterscheiden**. Es lässt sich jede Wiederholung $\boldsymbol{w}$, mit $|\boldsymbol{w}| + 2 \leq \boldsymbol{l} < |\boldsymbol{\omega}|$, durch Anwendung der erweiterten Funktion $\texttt{spectrum'}(\boldsymbol{\omega}, \boldsymbol{l})$ lösen (vgl. Kapitel 4.2.2 *deckender Wortpfad*)[43]:
Das Spektrum enthält wegen $\boldsymbol{l} < |\boldsymbol{\omega}|$ mindestens zwei Wörter $\boldsymbol{a} = \boldsymbol{p}_1\boldsymbol{w}\boldsymbol{s}_1$ und $\boldsymbol{b} = \boldsymbol{p}_2\boldsymbol{w}\boldsymbol{s}_2$ bei denen die Buchstaben $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ sowie die Buchstaben $\boldsymbol{s}_1$ und $\boldsymbol{s}_2$ unterschiedlich sind. Über diese Buchstaben kann die Wiederholung eindeutig in die Wörter $\boldsymbol{a}$ und $\boldsymbol{b}$ eingeordnet werden. Für $\boldsymbol{l} = |\boldsymbol{\omega}|$ hat $\texttt{spectrum'}(\boldsymbol{\omega}, \boldsymbol{l})$ nur das eine Wort $\boldsymbol{\omega}$ und es muss keine Wiederholung zugeordnet werden.

Sei dagegen $\boldsymbol{\omega}$ = ATTTTTGTTTTTA und $\boldsymbol{l} = 7 \geq |\text{TTTTT}| + 2$. Wäre $\boldsymbol{l}$ kleiner als 7 (z.B. 6), so würde je Vorkommen von TTTTT entweder nur der Buchstabe vor oder nach der Wiederholung bestimmt werden. Dadurch lässt sich aber nicht die Reihenfolge der Teilwörter zwischen den Wiederholungen bestimmen. Die Wiederholung selbst ist erkennbar, aber nicht lösbar.
Wenn keine Multimenge verwendet wird, etwa wie beim *Shotgun Sequencing*, dann lassen sich Wiederholungen, die länger sind als die maximale *read*-Länge, nicht mal mehr erkennen.

Insgesamt ergibt sich noch ein weiteres grundlegendes Problem: Die Fragmente werden **nicht fehlerfrei** gelesen, sondern mit einer Fehlerrate von 1-3%. Das führt dazu, dass einige *reads* (Teilwörter im Spektrum) nicht oder zu Unrecht als aufeinander folgend erkannt werden, weil eines der beiden Teilwörter an mindestens einer Stelle durch eine falsch gelesene Base abweicht.
Des Weiteren vermischen sich in den biotechnischen Verfahren die **Fragmente der beiden komplementären DNA-Stränge**. Es lässt sich daher nicht bestimmen, von welchem Strang ein *read* stammt. Für dieses Problem gibt es unterschiedliche Herangehensweisen[44].

---

[41] vgl. JoPe04, S. 275-279
[42] vgl. Allex99, S. 83
[43] diese Formel und die Ausführungen dazu entstammen den Überlegungen des Autors
[44] siehe MGMB07, S. 2

4 Fortgeschrittene Konzepte

Das Kapitel 3 hat einen Algorithmus für die Lösung des SBH-Problems entwickelt. Im letzten Abschnitt wurde erläutert, dass die Lösung einigen Einschränkungen unterliegt. Zwar ist auf der Seite der Informatik theoretisch alles in Ordnung, aber durch die experimentellen Verfahren der biologischen Komponente muss der entwickelte **Algorithmus** für die Praxis **weiterentwickelt** werden, um mit den fehlerhaften bzw. unvollständigen Inputdaten umgehen zu können.

Der Fokus dieses Kapitels liegt auf einem Sequenzierungsalgorithmus, der im Kern die Idee **ω** durch einen Pfad zu finden beibehält, allerdings in der Praxis mit den erwähnten Schwierigkeiten der Wiederholungen und fehlerhaften *reads* umgehen kann.

Es existieren nur **zwei Paradigmen** das Problem der Zusammensetzung von Teilstücken (*fragment assembly*) zu lösen: OLC (Overlap-Layout-Consensus) bildet die Grundlage des ersten Unterkapitels, DBG (De Bruijn Graph) die des zweiten.

4.1 Klassische Vorgehensweise

Generell ist DNA-Sequenzierung ein zweistufiger Prozess (vgl. Kapitel 1.3.2). Der kombinatorische zweite Schritt behandelt die Zusammensetzung jener Teilstücke, deren Basenabfolgen im experimentellen ersten Schritt bestimmt wurden (*base calling*)[45].

Klassischerweise folgen Algorithmen für *fragment assembly* einem dreistufigen Schema:

1. overlap – (Überlappung) finde Teilwörter die sich möglicherweise überlappen
2. layout – (Anordnung) finde eine Reihenfolge der Teilwörter auf der DNA
3. consensus – (Einigung) leite die DNA-Sequenz aus der Reihenfolge der Teilwörter ab

Der **erste Schritt** beschäftigt sich damit die Überlappungen von Teilwörtern festzustellen. Er kämpft dabei mit der Schwierigkeit, dass die Bestimmung der Basenabfolgen (*reads*) in den Fragmenten nicht fehlerfrei ist. Daher kann es zu falschen Zuordnungen, zu Einfügungen oder zu Löschungen von Basen kommen. Wegen dieser Fehler werden in diesem Schritt oft Varianten der Algorithmen für das Abgleichen von Basensequenzen (*sequence aligning*, vgl. Kapitel 1.2) verwendet, da diese keine exakte Übereinstimmung fordern.

Der **zweite Schritt** ordnet die Teilwörter anhand ihrer Überlappungen an. Es ist der schwierigste Schritt, weil entschieden werden muss ob sich zwei Teilwörter tatsächlich überlappen und die Unterschiede auf Sequenzierungsfehler zurückzuführen sind, oder ob die Unterschiede echt sind und die Teilwörter aus zwei voneinander abweichenden Wiederholungen stammen. Bei Wiederholungen kann weiterhin die Position in der Anordnung uneindeutig sein. Die Algorithmen für *sequence aligning* führen üblicherweise den ersten und zweiten Schritt zusammen aus.

Im **dritten Schritt** wird die endgültige Sequenz festgelegt, indem fehlerhafte *reads* korrigiert werden. Die einfachste Strategie ist den am häufigsten auftauchenden Buchstaben zu wählen. Wenn sichergestellt ist, dass jede Position im Genom häufig genug vorkommt, kann mit dieser Vorgehensweise der *Experimentelle Fehler* minimiert werden. Alternativ können den nicht eindeutig

[45] Die Ausführungen dieses Kapitels gehen zurück auf: JoPe04, S. 279f; Allex99, S. 3-5, 14-19

bestimmten Basen Buchstabencodes für die dort vorhandene Mehrdeutigkeit gegeben werden (siehe Allex99, S. 16). Diese werden dann manuell korrigiert, indem man den Output der Sequenzierungsmaschinen (vgl. Abb. 1.3) inspiziert.

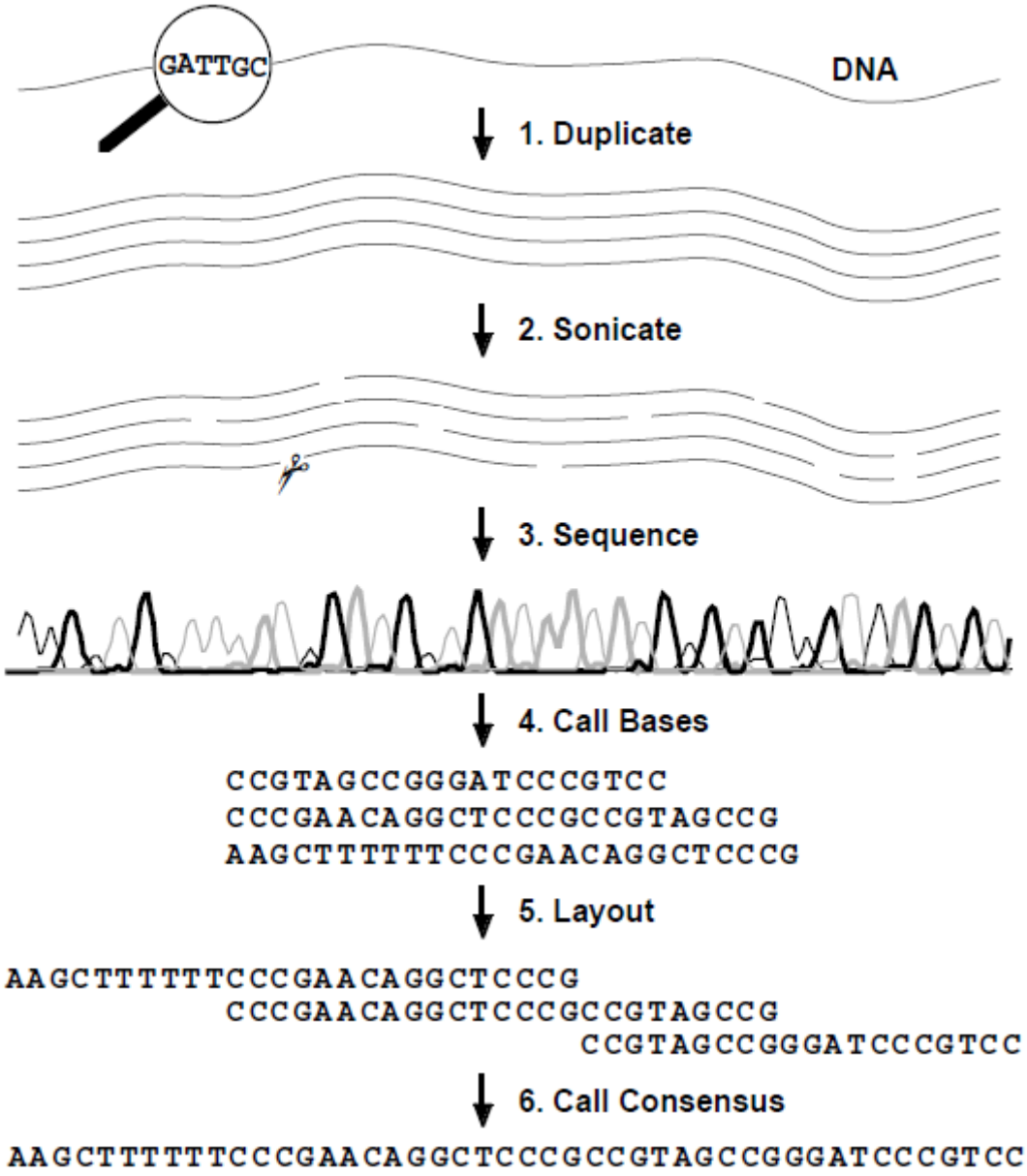

Abb. 4.1 – Überblick über den Prozess der DNA-Sequenzierung[46]

Die Abbildung 4.1 fasst den gesamten Prozess der DNA-Sequenzierung zusammen. Die ersten drei Schritte sind dabei experimenteller Natur, während die Schritte vier bis sechs algorithmisch sind. Der *overlap*-Schritt wird hier mit dem *layout*-Schritt zusammen durchgeführt.

Genauer soll auf die Vorgehensweise bei OLC nicht eingegangen werden. Stattdessen liegt der Schwerpunkt auf dem folgenden Kapitel über DBG, da es die Idee der de Bruijn-artigen Graphen fortführt.

[46] Abb. entnommen aus Allex99, S. 4

4.2 Superpfade

Dieses Kapitel diskutiert einen Ansatz, der die Idee des *Shotgun Sequencing* weiterentwickelt. Der Grundgedanke ist die generierten *reads* mit der Funktion `spectrum` weiter zu zerlegen. Denn auf diese Weise ergibt sich die gutartige Eigenschaft der de Bruijn-artigen Graphen, dass aufeinander folgende Teilwörter einfach zu bestimmen sind. Man hofft mit der **weiteren Zerlegung** die Überlappungen zwischen den Teilwörtern ebenfalls einfach bestimmen zu können. Der Preis dafür ist, dass auch die Teilwörter selbst wiederhergestellt werden müssen.

Die ursprüngliche Idee des Ansatzes stammt von **Idury und Waterman**[47]. Zunächst wird wie im bisherigen S*hotgun Sequencing* das Genom fragmentiert und die Basensequenzen der Fragmente gelesen. Das ergibt die Menge $\boldsymbol{W}$ der *reads*. Für jedes Wort $\boldsymbol{w} \in \boldsymbol{W}$ wird nun die Funktion $\texttt{spectrum}(\boldsymbol{w}, \boldsymbol{l})$ aufgerufen. An dieser Stelle ist der große Vorteil zu DNA-Arrays, dass die Länge $\boldsymbol{l}$ der Teilwörter frei wählbar ist und nicht durch den Stand der Technik limitiert wird. So kann wegen der höheren Länge der von den *Shotgun Sequencing* Methoden generierten Fragmente das $\boldsymbol{l}$ eventuell so hoch gewählt werden, dass $\boldsymbol{l}$ größer ist als die Länge kürzerer Wiederholungen. Sinnvoll ist es $\boldsymbol{l}$ auf die Länge des kürzesten Wortes in $\boldsymbol{W}$ zu setzen.

Mit den Elementen (Wörtern) der Spektren wird nun der **de Bruijn-artige Graph erzeugt**. Die den Elementen entsprechenden Kanten können des Weiteren mit Informationen zu dem spezifischen $\boldsymbol{w}$, von dem sie stammen, und der Position innerhalb des $\boldsymbol{w}$ angereichert werden. Auf diese Weise gehen die Informationen nicht verloren und beim Finden des Pfades kann darauf geachtet werden, dass die Kanten zum selben $\boldsymbol{w}$ gehören. So lässt sich bis zu einem gewissen Grad mit Wiederholungen (innerhalb eines Teilwortes oder zwischen verschiedenen Teilwörtern) umgehen.
Wenn die zugrundeliegenden **Daten ideal** sind, d.h. (1) es gibt keine Sequenzierungsfehler, (2) die Teilwörter überdecken die gesamte Ziel-DNA und (3) aufeinander folgende Teilwörter überlappen sich mit mindestens $\boldsymbol{l}$ Buchstaben, dann entspricht die zu bestimmende Ziel-DNA einem Chinese Postman Pfad im de Bruijn-artigen Graphen.

In ihrem Ansatz versuchten Idury und Waterman die **Vorteile** von *Shotgun Sequencing* und *Sequencing by Hybridization* zu **kombinieren**. Tatsächlich gelang es ihnen auch, denn das „Zerkleinern" der Fragmente in regelmäßige Teilwörter überführt den (im klassischen Ablauf) komplizierten *layout-Schritt* in das mit polynomiellen Algorithmen lösbare Problem einen Chinese Postman Pfad zu finden. Ihr Ansatz zur Verarbeitung des generierten Graphen – sie führten bereits das Ausführen von Reduktionen auf dem Graphen ein, um diesen zu vereinfachen – habe sich aber in der Praxis als schlecht skalierend herausgestellt (der Grund dafür seien die fehlerhaften Teilwörter). Daher griffen Pevzner et al. das Konzept auf und entwickelten es weiter[48].

Pevzner et al. gaben ihrem Ansatz den Namen ESP, nach dem **Eulerian Superpath Problem** welches im Graphen gelöst wird. Nach der Meinung des Autors ist diese Bezeichnung allerdings falsch, denn der Pfad, welcher im ESP gesucht wird, ist kein Eulerpfad, sondern ein Chinese Postman Pfad. Später[49] erwähnen Pevzner et al. diesen Sachverhalt und führen Mehrfachkanten ein, um CPP auf das Problem Eulerpfad zu transformieren. In der Konstruktion des Graphen, werden dennoch keine Mehrfachkanten erzeugt und es wird weiter nach einem Eulerpfad gesucht. Der folgende Abschnitt beleuchtet diesen Aspekt etwas näher.

[47] vgl. IdWa95, S.295f, 301f
[48] vgl. PeTaWa01-a, S. 257
[49] vgl. PeTaWa01-b, S. 9751

Der Ansatz ESP hat das Potenzial einen Teil der **Multiplizitäten** zu **bestimmen**[50]. Denn für die Zerlegung der Teilwörter könnte auch die Funktion `spectrum'` genutzt werden. Es können aber nicht alle Multiplizitäten bestimmt werden, weil an den Enden der Teilwörter einige Wörter des Spektrums abgeschnitten und so nicht als Wiederholung erkannt werden. Die Schwierigkeit ist, dass auch wenn die Multiplizitäten vieler Teilwörter bekannt sind, dennoch $\boldsymbol{\omega}$ nicht durch einen Eulerpfad ermittelbar ist. Denn bei ausreichend großen Überlappungen zwischen den Teilwörtern finden sich diese als Mehrfachkanten im de Bruijn-artigen Graph. Entsprechend müssten dann die Überlappungen mehrfach in $\boldsymbol{\omega}$ vorkommen, und das ist nicht korrekt (vgl. Kapitel2.1). Dies verdeutlicht auch das nächste Beispiel für $\boldsymbol{\omega}$ = AACTTCTAGT und die gelesenen *reads* $\boldsymbol{W}_6$ = { AACTTCT, TCTAGT }.

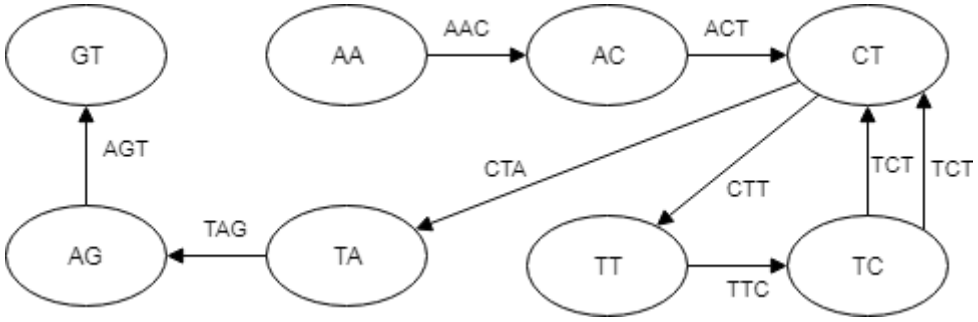

Abb. 4.2 – Graph zu $\boldsymbol{W}_6$ mit Mehrfachkanten

Das Bestimmen der Multiplizitäten hat für den Ansatz also keinen Mehrwert, weil trotzdem nicht nach einem Eulerpfad gesucht werden kann. Der Mehrwert von ESP liegt nach Meinung des Autors eher im Umgang mit fehlerhaften Teilwörtern. Darauf geht das nächste Kapitel ein.

4.2.1 Ablaufschema in ESP

In ESP wird das klassische Vorgehensmodell aufgehoben, weil es einige **Schwierigkeiten nicht lösen** könne[51]. Kritikpunkte sind zum einen, dass im *overlap*-Schritt nur paarweise Ähnlichkeiten festgestellt werden. Ein besserer Weg wäre es Ähnlichkeiten zwischen mehreren Teilwörtern festzustellen. Denn fehlerhaft gelesene Basen tauchen an zufälligen Stellen auf, während Unterschiede zwischen Teilwörtern immer an derselben Position auftauchen. Allerdings haben die Algorithmen zum *multiple sequence alignment* (vgl. Sem3) eine zu hohe Laufzeitkomplexität, sodass sie nicht sinnvoll eingesetzt werden können. Des Weiteren ist das Finden eines korrekten Pfades in einem Graphen mit vielen fehlerhaften Kanten sehr schwierig. Allerdings würden selbst auf fehlerfreien Daten die etablierten Tools zur *fragment assembly* Fehler im *layout*-Schritt begehen, weil sie Wiederholungen falsch einordnen.
Anmerkung: Dies war der Stand als Pevzner et al. den DBG-Ansatz proklamierten. Mit dem **String-Graph** Ansatz von Myers[52] und dem darauf basierenden String Graph Assembler[53] haben sich die dem OLC-Paradigma folgenden Algorithmen dieser Schwächen entledigt. String-Graph soll hier nicht näher erläutert werden, [MGMB07] beweist, dass auch dieser Ansatz NP-vollständig ist.

Als **erster Schritt** in ESP wird der *consensus*-Schritt durchgeführt. Dies dient dem einen Zweck die Komplexität des Graphen zu reduzieren, der im folgenden Schritt aufgebaut wird. Um diese Reduzierung zu erreichen, gilt es die Konflikte aufzulösen, die entstehen, wenn bei sonst identischen Teilwörtern an einer Position mehrere Basen in Frage kommen. Es ist bei diesem Vorgehen weniger wichtig, ob der korrekte Buchstabe ausgewählt wird, weil sich dieser später im erzeugten Genom

[50] dieser Abschnitt entstammt den Überlegungen des Autors
[51] Die Ausführungen dieses Kapitels gehen zurück auf: PeTaWa01-a, S.256-260
[52] siehe Myers05
[53] siehe SiDu12

einfach korrigieren lässt (z.B. über ein Mehrheits-Verfahren). Entscheidend ist es den Konflikt aufzulösen, damit sich überlappende Teilwörter auch als solche erkannt und falsche Kanten im Graphen entfernt werden. [PeTaWa01-a] formuliert den *consensus*-Schritt als ein Optimierungsproblem[54], in dem je Teilwort nur eine bestimmte Anzahl an Korrekturen durchgeführt werden darf. Dies liegt daran, dass durch die Konfliktauflösung nur Fehler der Sequenzierungsmaschinen korrigiert werden sollen. Bei einer Fehlerwahrscheinlichkeit von 2% sind bei einer durchschnittlichen *read*-Länge von 100 Nukleotiden also nur zwei Buchstaben falsch. Je mehr korrigiert werden, desto höher ist die Gefahr, dass sich eigentlich nicht überlappende Teilwörter so stark aneinander annähern, dass sie sich nach der „Fehlerkorrektur" doch überlappen. Implizit nutzt der erste Schritt des ESP dabei Vergleiche zwischen mehreren Teilwörtern (*multiple alignment*).

Der **zweite Schritt** in ESP ist das Konstruieren des de Bruijn-artigen Graphen $\boldsymbol{G_{W'}}$ =($\boldsymbol{V}$, $\boldsymbol{E}$) aus der Menge der korrigierten Wörter $\boldsymbol{W'}$. Es wird dabei grundsätzlich so vorgegangen wie in Kapitel 3.2 beschrieben. Das heißt die Menge der Knoten $\boldsymbol{V}$ ist die Vereinigungsmenge aller Teilwörter mit Länge $\boldsymbol{l} - 1$ von korrigierten Wörtern $\boldsymbol{w} \in \boldsymbol{W'}$. Eine Kante $\boldsymbol{e} \in \boldsymbol{E}$ entspricht einem Teilwort $\boldsymbol{w}$ der Länge $\boldsymbol{l}$ von einem Wort $\boldsymbol{pws}$ aus $\boldsymbol{W'}$. Sie verbindet zwei Knoten genau dann, wenn der erste ein Präfix der Länge $\boldsymbol{l} - 1$ und der zweite ein Suffix der Länge $\boldsymbol{l} - 1$ von $\boldsymbol{w}$ ist. Formell gilt demnach:

$$\boldsymbol{V} = \{\, \boldsymbol{v} \mid \boldsymbol{v} \in \texttt{spectrum}(\boldsymbol{w}, \boldsymbol{l} - 1) \wedge \boldsymbol{w} \in \boldsymbol{W'} \,\}$$
$$\boldsymbol{E} = \{\, (\boldsymbol{v}_1, \boldsymbol{v}_2) \mid \exists \boldsymbol{pws} \in \boldsymbol{W'} : |\boldsymbol{w}| = \boldsymbol{l} \wedge |\texttt{overlap}(\boldsymbol{v}_1, \boldsymbol{w})| = |\texttt{overlap}(\boldsymbol{w}, \boldsymbol{v}_2)| = \boldsymbol{l} - 1 \,\}$$

Es sei darauf hingewiesen, dass dieser de Bruijn-artige Graph sich von den de Bruijn-artigen Graphen in Kapitel 3.2 unterscheidet. Denn hier wird eine Menge von Wörtern zur Konstruktion verwendet statt nur einem einzelnen Wort. Idury und Waterman führen daher unterschiedliche Bezeichnungen für die beiden Graphen ein. Weil die zugrunde liegenden Konzepte identisch sind, wird hier darauf verzichtet.

Ein korrigiertes Wort $\boldsymbol{w} \in \boldsymbol{W'}$entspricht im Graphen $\boldsymbol{G_{W'}}$ einem Pfad (*Wortpfad*) und das Problem der *fragment assembly* entspricht dem Finden eines (Super-) Pfades, der jeden Wortpfad enthält. Die Wortpfade werden zusätzlich zu $\boldsymbol{G_{W'}}$ in einer Menge $\boldsymbol{P}$ gespeichert.

ESP ist formuliert als eine **Generalisierung des Eulerpfad Problems**, in diesem entspricht jeder Pfad nur einer einzelnen Kante. Weil der gesuchte Pfad ein Pfad über Pfade ist, heißt er auch *Superpfad*. Die Aussage, dass es eine Generalisierung ist, kann deshalb getroffen werden, weil angenommen wird, dass die gesuchte DNA-Sequenz keine Wiederholungen enthält.

Eine **verbesserte Formulierung** von ESP geben Medvedev et al[55]. Das *de Bruijn Superpfad Problem* (BSP) ist eine Generalisierung des *Chinese Postman Problems*. Es ist keine Generalisierung des Eulerpfad Problems, weil nicht verlangt werden kann, dass jeder Pfad genau einmal im Superpfad vorkommt. Denn sonst wären einerseits Graphen mit Wiederholungen, die länger sind als die Wortpfade, nicht durch das Problem lösbar. Andererseits würde dadurch verlangt, dass Überlappungen mehrfach vorkommen müssen. Die formale Definition von BSP lautet:

Gegeben sei ein de Bruijn-artiger Graph $\boldsymbol{G_{W'}} = (\boldsymbol{V}, \boldsymbol{E})$ und eine Menge $\boldsymbol{P}$ von Pfaden in $\boldsymbol{G_{W'}}$.
Gesucht ist ein minimaler Superpfad $\boldsymbol{\varphi}$, der jeden Pfad $\boldsymbol{p} \in \boldsymbol{P}$ als Unterpfad enthält.

Wobei ein Pfad $\boldsymbol{p} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots \boldsymbol{v}_j)$ genau dann ein Unterpfad eines Pfades $\boldsymbol{q} = (\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots \boldsymbol{v}_n)$ ist, wenn es einen Index $\boldsymbol{i}$ gibt mit $0 \leq \boldsymbol{i} \leq \boldsymbol{n}\text{-}\boldsymbol{j}$ für den die Knoten $\boldsymbol{v}_0$ bis $\boldsymbol{v}_j$ in $\boldsymbol{p}$ mit den Knoten $\boldsymbol{v}_i$ bis $\boldsymbol{v}_{i+j}$ in $\boldsymbol{q}$ übereinstimmen.

[54] vgl. PeTaWa01-a, S. 258
[55] vgl. MGMB07, S. 6f

Zusätzlich zum ESP verlangt BSP, dass der Superpfad *minimal* ist, also möglichst kurz. Vergleicht man die Definition des *de Bruijn Superpfad Problems* mit der von *Shortest Common Supersequence* (Kapitel 2.1), so ist der einzige Unterschied, dass die Wörter nun Pfaden entsprechen. Statt einem Superwort $\boldsymbol{\omega}$ wird nach einem Superpfad $\boldsymbol{\varphi}$ gesucht, wobei sich $\boldsymbol{\omega}$ aus $\boldsymbol{\varphi}$ ableiten lässt (analog zu Kapitel 3.1). Pevzner et al. merken an, dass eine theoretische Analyse des ESP noch aussteht[56]. Diese führten Medvedev et al. durch: Sie beweisen, dass BSP im Kontext der *fragment assembly* ein **NP-vollständiges Problem** ist. Dazu reduzieren sie das (NP-vollständige) Problem SCS auf BSP[57].

Im folgenden Kapitel wird die **generelle Lösungsstrategie für ESP** dargestellt. Das geschieht, um ein prinzipielles Verständnis für die Abläufe herzustellen. Die angegebenen Transformationen werden heuristisch angewandt. Dadurch haben sie in der Praxis zwar ein gutes Laufzeitverhalten, führen aber nicht immer zur korrekten Lösung[58].

4.2.2 Transformationen auf Graphen

Wiederholungen in $\boldsymbol{\omega}$ (und auch Überlappungen von Wörtern aus $\boldsymbol{W}$) fallen im de Bruijn-artigen Graphen $\boldsymbol{G_{W'}}$ zu einem einzelnen Pfad zusammen[59]:
Ein Pfad ($\boldsymbol{v_0}$, …, $\boldsymbol{v_n}$) in $\boldsymbol{G_{W'}}$ heiße **Wiederholungspfad**, wenn $\texttt{indegree}(\boldsymbol{v_0}) > 1$, $\texttt{outdegree}(\boldsymbol{v_n}) > 1$ und für alle $\boldsymbol{i}$ mit $0 \leq \boldsymbol{i} < \boldsymbol{n}$ $\texttt{outdegree}(\boldsymbol{v_i}) = 1$ gilt. Ein Wiederholungspfad hat also mehrere *Zugänge* und mehrere *Ausgänge*. An dieser Stelle sei explizit darauf hingewiesen, dass die Definition deutlich macht, dass $\boldsymbol{E}$ eine Menge und keine Multimenge ist. Deswegen haben die inneren Knoten des Wiederholungspfades die Valenz 1.

Wiederholungspfade schaffen beim Finden des Superpfades die Schwierigkeit, dass nicht zugeordnet werden kann welcher Zugang zu welchem Ausgang gehört. Diese Information lässt sich wiederherstellen mittels Wortpfaden, die sowohl einen Eingang als auch einen Ausgang des Wiederholungspfades enthalten. Solche Wortpfade heißen *deckende Wortpfade*, sie liefern Informationen über die **korrekte Paarbildung zwischen Zugängen und Ausgängen**.

Diese Eigenschaft bildet die Grundlage für die Idee der Transformationen. Wenn ein deckender Wortpfad existiert, dann kann der Wiederholungspfad aufgespalten werden in den bekannten Teil (Zu- und Ausgang im deckenden Wortpfad) und den unbekannten Teil. Dies gilt zum Beispiel für die Abbildung 4.2. Der oberste Wortpfad erlaubt in diesem Wiederholungspfad das **Zuordnen** des oberen Zugangs zum oberen Ausgang. Das Auflösen von Wiederholungspfaden ist die zentrale Herausforderung bei der Ermittlung des Superpfades $\boldsymbol{\varphi}$.

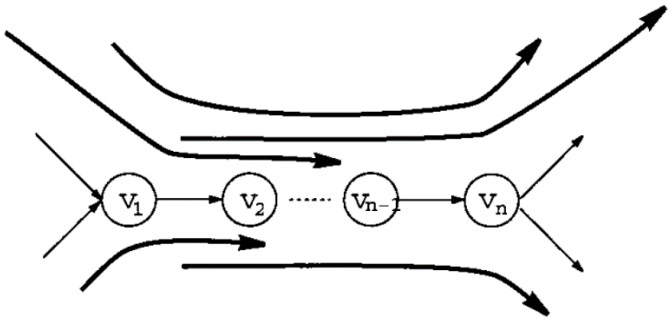

Abb. 4.2 – Ein auflösbarer Wiederholungspfad

[56] vgl. PeTaWa01-a, S. 262
[57] vgl. MGMB07, S.7f
[58] vgl. MGMB07, S.7
[59] Die Ausführungen dieses Kapitels gehen zurück auf: PeTaWa01-a, S. 260-262

Die zentrale Idee beim Vorgehen $\boldsymbol{\varphi}$ zu finden ist, das Paar $(\boldsymbol{G}, \boldsymbol{P})$ zu vereinfachen. Dazu wird eine **Reihe von äquivalenten Transformationen** angewendet. Eine Transformation von einem Graphen $\boldsymbol{G}_1$, der die Menge an Pfaden $\boldsymbol{P}_1$ enthält, in einen Graphen $\boldsymbol{G}_2$, der die Pfadmenge $\boldsymbol{P}_2$ enthält, heißt *äquivalent*, wenn es eine eindeutige Beziehung zwischen den Superpfaden in $\boldsymbol{G}_1$ und $\boldsymbol{G}_2$ gibt. Das heißt es existiert eine bijektive Funktion $\mathtt{f}$ für die gilt: $\boldsymbol{\varphi}$ ist ein Superpfad in $(\boldsymbol{G}_1, \boldsymbol{P}_1)$ genau dann, wenn $\mathtt{f}(\boldsymbol{\varphi})$ ein Superpfad in $(\boldsymbol{G}_2, \boldsymbol{P}_2)$ ist. Die Transformation verändert dabei nicht das Wort $\boldsymbol{\omega}$, welches aus $\boldsymbol{\varphi}$ abgeleitet wird[60].

Jede äquivalente Transformation **vereinfacht** das Paar $(\boldsymbol{G}, \boldsymbol{P})$. Das Ziel ist es zu einem $\boldsymbol{P_k}$ zu gelangen, in dem jeder Pfad nur noch einer Kante entspricht. Dann muss im Paar $(\boldsymbol{G_k}, \boldsymbol{P_k})$ nur noch ein Chinese Postman Pfad gefunden werden. Denn weil jede Transformation äquivalent ist und ein Chinese Postman Pfad ein Spezialfall des Superpfades ist, entspricht ein Chinese Postman Pfad in $(\boldsymbol{G_k}, \boldsymbol{P_k})$ einem Superpfad in $(\boldsymbol{G}, \boldsymbol{P})$.

Pevzner et al. fanden drei mögliche äquivalente Transformationen, welche im Folgenden vorgestellt werden[61]:

Die **erste Transformation** heißt $\boldsymbol{x,y}$*-Abtrennung* (engl. detachment). Sie wird angewandt, um zwei adjazente Kanten $\boldsymbol{x}$ und $\boldsymbol{y}$ durch eine Kante $\boldsymbol{z}$ zu ersetzen. Pevzner et al. schränken die Verwendung dieser Transformation auf den Fall ein, dass es in $\boldsymbol{G}$ keine Mehrfachkanten gibt. Nach der Meinung des Autors genügt es zu verlangen, dass $\boldsymbol{x}$ und $\boldsymbol{y}$ die gleiche Multiplizität $\boldsymbol{m}$ haben.

Sei $\boldsymbol{x}$ eine gerichtete Kante von $\boldsymbol{v}_{\text{in}}$ zu $\boldsymbol{v}_{\text{mid}}$ und $\boldsymbol{y}$ eine gerichtete Kante von $\boldsymbol{v}_{\text{mid}}$ zu $\boldsymbol{v}_{\text{out}}$ (Abb. 4.3). Aus der Menge $\boldsymbol{P}$ werden nun verschiedene **Pfadklassen** gebildet:

- Sei $\boldsymbol{P}_{\rightarrow \boldsymbol{x}} \subseteq \boldsymbol{P}$ die Menge aller Pfade, die $\boldsymbol{x}$ als letzte Kante enthalten[62].
- Sei $\boldsymbol{P}_{\boldsymbol{y}\rightarrow} \subseteq \boldsymbol{P}$ die Menge aller Pfade, die $\boldsymbol{y}$ als erste Kante enthalten.
- Sei $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{y}} \subseteq \boldsymbol{P}$ die Menge aller Pfade welche die Kante $\boldsymbol{x}$ direkt vor der Kante $\boldsymbol{y}$ verwenden.
- Sei $\boldsymbol{P}_{\text{not}} \subseteq \boldsymbol{P}$ die Menge aller Pfade, welche weder $\boldsymbol{x}$ noch $\boldsymbol{y}$ enthalten.

Die $\boldsymbol{x,y}$-Abtrennung ersetzt in $\boldsymbol{G}$ die beiden Kanten $\boldsymbol{x}$ und $\boldsymbol{y}$ durch eine neue gerichtete Kante $\boldsymbol{z}$ von $\boldsymbol{v}_{\text{in}}$ zu $\boldsymbol{v}_{\text{out}}$. Die Kante $\boldsymbol{z}$ erhält als Multiplizität $\boldsymbol{m}$ und als Label die Verschmelzung der Label (aufeinander folgende Teilwörter) von $\boldsymbol{x}$ und $\boldsymbol{y}$. Dadurch verändert sich das Wort $\boldsymbol{\omega}$ nicht. Wenn $\boldsymbol{P}_{\text{not}} = \emptyset$, dann kann der Knoten $\boldsymbol{v}_{\text{mid}}$ aus $\boldsymbol{G}$ entfernt werden, ansonsten reduziert sich dessen Valenz um 1. Die Elemente der Pfadklassen verändern sich folgendermaßen: In den Pfaden der Mengen $\boldsymbol{P}_{\rightarrow \boldsymbol{x}}$, und $\boldsymbol{P}_{\boldsymbol{y}\rightarrow}$ werden die Kanten $\boldsymbol{x}$ bzw. $\boldsymbol{y}$ durch die Kante $\boldsymbol{z}$ ersetzt. In den Pfaden von $\boldsymbol{P}_{\boldsymbol{x},\boldsymbol{y}}$ ersetzt $\boldsymbol{z}$ die aufeinander folgenden Kanten $\boldsymbol{x}$ und $\boldsymbol{y}$. Die Pfade in $\boldsymbol{P}_{\text{not}}$ verändern sich nicht.

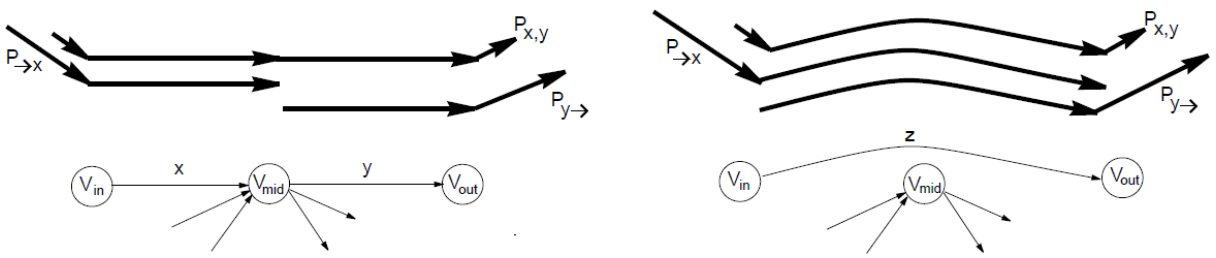

Abb. 4.3 – $\boldsymbol{x,y}$*-Abtrennung*

[60] Hierzu äußern sich Pevzner et al. nicht direkt, die Aussage folgt aber aus der Definition der Transformationen

[61] Der Autor folgt in diesem Abschnitt unterschiedlich stark den Ausführungen von Pevzner et al., weil er der Meinung ist dass diese Unstimmigkeiten enthalten und nicht optimal erklärt sind

[62] Pfade sind Tupel von Knoten, die Formulierung dass Pfade Kanten enthalten meint, dass es einen Index $\boldsymbol{i}$ gibt, so dass die mit der Kante $\boldsymbol{e} = (\boldsymbol{u},\boldsymbol{v})$ inzidenten Knoten $\boldsymbol{u}$ und $\boldsymbol{v}$ im Tupel an der Stelle $\boldsymbol{i}$ und $\boldsymbol{i}$ +1 vorkommen

Die $\boldsymbol{x}$,$\boldsymbol{y}$-Abtrennung ist nur dann eine **äquivalente Transformation**, wenn jeder Pfad $\boldsymbol{p} \in \boldsymbol{P}$ in genau einer dieser vier Mengen vorkommt, das heißt die vier Mengen sind eine Einteilung von $\boldsymbol{P}$ in Äquivalenzklassen[63]. Ist dies nicht der Fall muss es mindestens einen Pfad geben, der nach $\boldsymbol{x}$ eine von $\boldsymbol{y}$ verschiedene Kante verwendet, oder vor $\boldsymbol{y}$ eine von $\boldsymbol{x}$ verschiedene Kante nimmt. Wenn solch ein Pfad existiert, dann würde das Ersetzen von $\boldsymbol{x}$ bzw. $\boldsymbol{y}$ durch die Kante $\boldsymbol{z}$ das Wort $\boldsymbol{\omega}$ verändern.

Die **zweite Transformation** behandelt, angelehnt an die erste Transformation, den Fall, dass es Pfade gibt, die nach der Kante $\boldsymbol{x}$ verschiedene Kanten wählen (statt nur $\boldsymbol{y}$). Die Kante $\boldsymbol{x}$ muss dann eine Mehrfachkante sein, weil der Superpfad sie durch die verschiedenen Pfade mehrfach verwenden muss.

Die relevanten Pfadklassen werden nun etwas anders definiert:

- Sei $\boldsymbol{P}_{\rightarrow x} \subseteq \boldsymbol{P}$ die Menge aller Pfade, die $\boldsymbol{x}$ als letzte Kante enthalten.
- Sei $\boldsymbol{P}_{y1\rightarrow} \subseteq \boldsymbol{P}$ die Menge aller Pfade, die $\boldsymbol{y}_1$ als erste Kante enthalten.
- Sei $\boldsymbol{P}_{x,y1} \subseteq \boldsymbol{P}$ die Menge aller Pfade welche die Kante $\boldsymbol{x}$ direkt vor der Kante $\boldsymbol{y}_1$ verwenden.
- Sei $\boldsymbol{P}_{x,y2} \subseteq \boldsymbol{P}$ die Menge aller Pfade welche die Kante $\boldsymbol{x}$ direkt vor einer von $\boldsymbol{y}_1$ verschiedenen Kante $\boldsymbol{y}_2$ verwenden. Von $\boldsymbol{v}_{\text{mid}}$ dürfen also auch mehr als zwei Kanten ausgehen. $\boldsymbol{y}_2$ ist dann irgendeine davon (außer $\boldsymbol{y}_1$)
- (Die weiteren Mengen $\boldsymbol{P}_{\text{not}}$ und $\boldsymbol{P}_{y2\rightarrow}$ spielen keine Rolle)

In der vereinfachten Betrachtung (Abb. 4.4 links) ist $\boldsymbol{x}$ eine Mehrfachkante mit Multiplizität 2 und der Knoten $\boldsymbol{v}_{\text{mid}}$ hat zwei einfache Kanten $\boldsymbol{y}_1$ und $\boldsymbol{y}_2$ zu den Knoten $\boldsymbol{v}_{\text{out1}}$ und $\boldsymbol{v}_{\text{out2}}$. Die auszuführende Transformation ist eine $\boldsymbol{x}$,$\boldsymbol{y}_1$-Abtrennung. In dieser wird in $\boldsymbol{G}$ eine der beiden $\boldsymbol{x}$ Kanten und die Kante $\boldsymbol{y}_1$ durch $\boldsymbol{z}$ ersetzt. Entsprechend werden in den Pfaden der Menge $\boldsymbol{P}_{x,y1}$ die Kanten $\boldsymbol{x}$ und $\boldsymbol{y}_1$ durch $\boldsymbol{z}$ ersetzt sowie in den Pfaden der Menge $\boldsymbol{P}_{y1\rightarrow}$ die Kante $\boldsymbol{y}_1$ durch $\boldsymbol{z}$.

Ob diese Transformation **äquivalent** ist hängt von der Beschaffenheit der Menge $\boldsymbol{P}_{\rightarrow x}$ ab. Im trivialen Fall ist $\boldsymbol{P}_{\rightarrow x}$ leer, dann müssen darin keine Pfade angepasst werden und die Transformation ist äquivalent. Wenn die Menge $\boldsymbol{P}_{\rightarrow x}$ nichtleer ist, dann lässt sich für einen Pfad aus $\boldsymbol{P}_{\rightarrow x}$ nicht unbedingt entscheiden wie die Kante $\boldsymbol{x}$ verändert werden soll (Abb. 4.4 rechts unten).

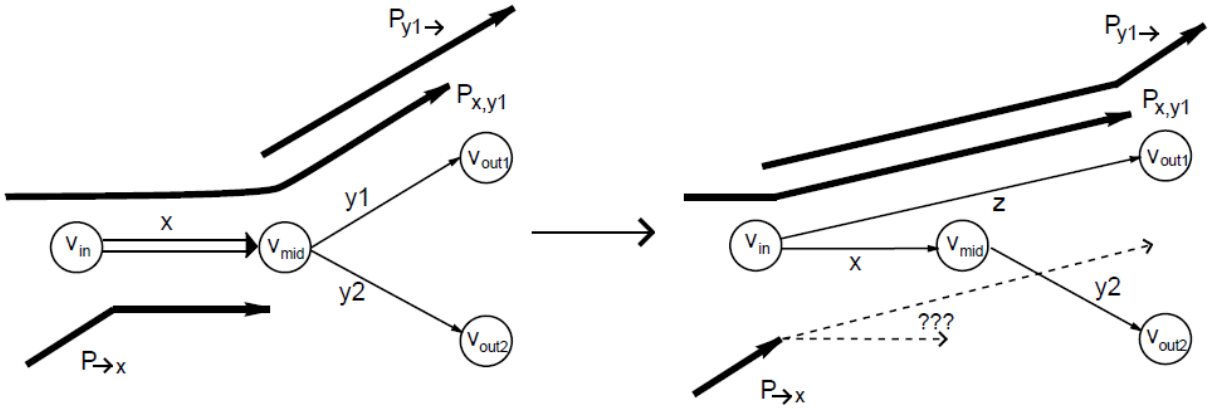

Abb. 4.4 – $\boldsymbol{x}$,$\boldsymbol{y}_1$-*Abtrennung* bei Mehrfachkanten

Um Aussagen über die Entscheidbarkeit, ob ein Pfad $\boldsymbol{p} \in \boldsymbol{P}_{\rightarrow x}$ zur Menge $\boldsymbol{P}_{x,y1}$ gehört ($\boldsymbol{x}$ und $\boldsymbol{y}_1$ werden durch $\boldsymbol{z}$ ersetzt) oder zur Menge $\boldsymbol{P}_{x,y2}$ ($\boldsymbol{x}$ bleibt), formulieren zu können, wird ein neues Konzept eingeführt:
Ein Pfad heißt ***konsistent*** zu einem anderen Pfad, wenn ihre Vereinigung wieder ein Pfad ist (d.h. die Pfade lassen sich wie sich überlappende Wörter verschmelzen). Ein Pfad heißt konsistent mit einer Menge von Pfaden, wenn er mit jedem Pfad der Menge konsistent ist. Bei der $\boldsymbol{x}$, $\boldsymbol{y}_1$-Abtrennung muss

[63] Zu diesem Sachverhalt äußern sich Pevzner et al. nicht, er entstammt den Überlegungen des Autors

für jeden Pfad $\boldsymbol{p} \in \boldsymbol{P}_{\rightarrow x}$ die Konsistenz mit den Mengen $\boldsymbol{P}_{x,\,y1}$ und $\boldsymbol{P}_{x,\,y2}$ geprüft werden. Es gibt drei Möglichkeiten:

1. Wenn $\boldsymbol{p}$ mit genau einer der Mengen konsistent ist, dann wird er der Menge entsprechend angepasst.
2. Wenn $\boldsymbol{p}$ mit keiner der Mengen konsistent ist, dann ist die Datengrundlage des Graphen inkonsistent und es kann kein Superpfad gefunden werden.
3. Wenn $\boldsymbol{p}$ mit beiden Mengen konsistent ist, dann gibt es zwei Möglichkeiten. Entweder wird durch andere äquivalente Transformationen $\boldsymbol{p}$ zu einem späteren Zeitpunkt mit nur noch einer Menge konsistent sein, oder nicht. Im ersten Fall wird gewartet, im zweiten Fall wendet man die dritte Transformation an.

Nur wenn jeder Pfad $\boldsymbol{p} \in \boldsymbol{P}_{\rightarrow x}$ einer der beiden Mengen zugeordnet werden kann, dann ist die $\boldsymbol{x},\boldsymbol{y}_1$-Abtrennung eine äquivalente Transformation.

Die **dritte Transformation** behandelt den Fall, dass eine Kante nicht *aufgelöst* werden kann. Zum Beispiel in der Situation der Abbildung 4.5. In dieser gibt es keine Informationen welche der Zugänge zu welchem Ausgang gehören. Um die Möglichkeit zu geben andere äquivalente Transformationen durchzuführen, wird in solch einer Situation nur die Menge $\boldsymbol{P}$ verändert und nicht $\boldsymbol{G}$. Wenn $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{v})$ eine Kante ist, die in jedem Pfad aus $\boldsymbol{P}$ entweder als erste Kante, als letzte Kante oder gar nicht vorkommt, und $\texttt{indegree}(\boldsymbol{u}) = \texttt{outdegree}(\boldsymbol{v}) = 1$ gilt, dann ist ein $\boldsymbol{x}$-*Schnitt* (engl. cut) eine äquivalente Transformation. In diesem wird die Kante $\boldsymbol{x}$ aus allen Pfaden $\boldsymbol{P}_{\rightarrow x}$ und $\boldsymbol{P}_{x\rightarrow}$ entfernt.

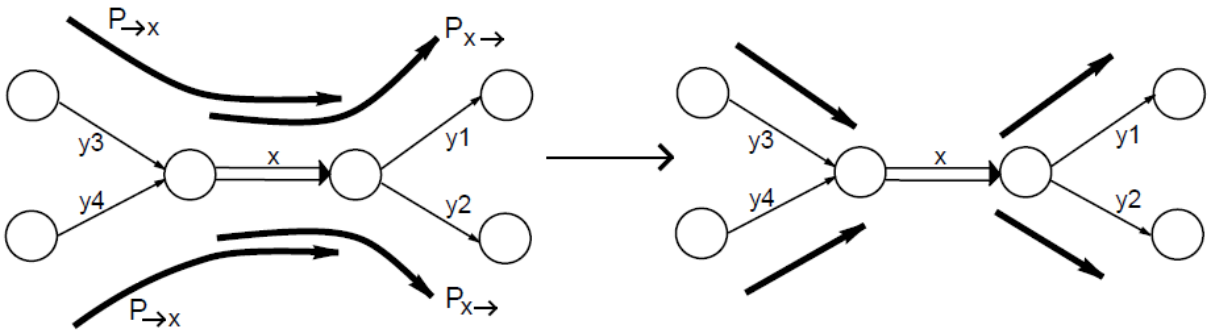

Abb. 4.5 – $\boldsymbol{x}$-*Schnitt*

Pevzner et al. stellten fest, dass **Abtrennungen und Schnitte ausreichend** sind um das ESP auf das CPP zu reduzieren. Es wird allerdings darauf hingewiesen, dass dies nur praktische Erfahrungen sind.

Ergänzend sei auf die Arbeit von **Kingsford, Schatz und Mihai** hingewiesen[64]. Sie formulierten mehrere Transformationen im de Bruijn-artigen Graph ohne dabei die Menge der Pfade zu nutzen.
Des Weiteren geben sie einen alternativen Ansatz zur Bestimmung der Wahrscheinlichkeit, dass die im de Bruijn-artigen Graph ermittelte Lösung die korrekte Lösung ist (vgl. Kapitel 3.3). Ihr Ansatz bestimmt die Anzahl der aus einem de Bruijn-artigen Graphen ableitbaren Wörter statt die vorhandenen Eulerpfade. Denn auch wenn es mehrere Pfade im Graphen gibt, so müssen sich die daraus abgeleiteten Wörter nicht unterscheiden. Zum Beispiel gibt es für $\boldsymbol{\omega}$ = TACACACACAC und $\boldsymbol{l}$ = 5 im de Bruijn-artigen Graph jeweils eine Mehrfachkante der Multiplizität 3 zwischen den Knoten ACAC und CACA. Unabhängig von der Reihenfolge, in der die Kanten gewählt werden, bleibt das erzeugte Wort immer gleich.

[64] vgl. KiScPo10, S. 3-6

5 Zusammenfassung und Ausblick

DNA-Sequenzierung ist ein **zweistufiger Prozess**. Die erste Stufe behandelt das experimentelle Generieren von Fragmenten der zu sequenzierenden DNA sowie das Bestimmen der Basenabfolgen der Fragmente (*reads*). Die zweite Stufe beschäftigt sich mit dem algorithmischen Problem des Zusammensetzens der Basenabfolgen (*fragment assembly*) und hängt von der ersten ab.

Das zunächst betrachtete experimentelle Verfahren *Shotgun Sequencing* führt zur Problemstellung *Shortest Common Supersequence*. Dieses Problem ist NP-vollständig, also nicht effizient lösbar. Das zweite experimentelle Verfahren, *Sequencing by Hybridization*, ist durch zwei **Problemstellungen** aus der Graphentheorie modelliert worden. Wenn die Basenabfolgen Knoten entsprechen, muss ein Hamiltonpfad gefunden werden. Auch dieses Problem ist NP-vollständig. Werden die Basenabfolgen als Kanten modelliert, so führt dies zum polynomiell lösbaren Problem einen Eulerpfad zu finden.

Die wesentliche Herausforderung beim Zusammensetzen ist das Behandeln von **Wiederholungen** in der DNA. Für die Modellierungsansätze der Graphentheorie bedeutet dies, dass einige Knoten bzw. Kanten mehrfach im zu findenden Pfad vorkommen müssen. Dadurch wird das Eulerpfad Problem zum ebenfalls polynomiellen Chinese Postman Problem.

Weitere Probleme in der *fragment assembly* sind:

- das kleine Alphabet aus nur vier Basen, welches dazu führt dass viele Fragmente allein aus statistischen Gründen gleiche Teilsequenzen haben. Dies erschwert das Erkennen von echten Überlappungen, die von aufeinander folgenden Fragmenten herrühren.
- die Fehler der Sequenzierungsmaschinen, welche den Einsatz von Algorithmen aus dem *sequence alignment* nötig machen. Die Fehler würden sonst dazu führen, dass sich eigentlich überlappende Fragmente nicht als solche erkannt werden oder umgekehrt.
- der Aufbau der DNA als Kombination zweier komplementärer Stränge, wodurch es nicht möglich ist zu bestimmen von welchem Strang eine Basensequenz stammt.

In der *fragment assembly* gibt es zwei generelle Vorgehensweisen. Das *overlap-layout-consensus* Paradigma und die Verwendung von de Bruijn-artigen Graphen. In letzteren wird ein Superpfad über Pfade, die *reads* entsprechen, gesucht und durch ein schrittweises Reduktionen des Graphen ermittelt. Auch dieser Ansatz ist NP-vollständig.

Nach den grundlegenden Ideen wurde in dieser Arbeit bei den fortgeschrittenen Konzepten der Schwerpunkt auf die de Bruijn-artigen Graphen gelegt. Daher sind die Details des *overlap-layout-consensus* Paradigmas und des *String-Graph* Ansatzes ausgeklammert worden.

Des Weiteren stellt [MGMB07] einen polynomiellen Algorithmus zur Lösung von *fragment assembly* vor. Auch dieser soll in dieser Arbeit zwar erwähnt aber nicht erschöpfend behandelt werden.

Literaturverzeichnis

Allex99 Allex, Carolyn: Computational methods for fast and accurate DNA fragment assembly, Wisconsin USA, 1999
Im Internet unter *http://ftp.cs.wisc.edu/machine-learning/shavlik-group/allex.thesis99.pdf*, abgerufen am 12.10.2018

CoPe10 Compeau, Phillip; Pevzner, Pavel: Genome Reconstruction: A Puzzle with a Billion Pieces, University of California at San Diego, 2010
Im Internet unter *http://cseweb.ucsd.edu/~ppevzner/B4B/excerpts/Compeau-Pevzner.pdf*, abgerufen am 10.10.2018

EdJo73 Edmonds, Jack; Johnson, Ellis: Matching, Euler Tours and the Chinese Postman, Mathematical Programming Vol. 5, Springer, 1973
Im Internet unter *http://web.eecs.umich.edu/~pettie/matching/Edmonds-Johnson-chinese-postman.pdf*, abgerufen am 12.12.2018

IdWa95 Idury, Ramana; Waterman, Michael: A New Algorithm for DNA Sequence Assembly, Journal of Computational Biology Volume 2 Number 2, 1995
Im Internet unter *https://cs.brown.edu/courses/csci1820/resources/papers/idurywaterman.pdf*, abgerufen am 28.11.2018

JoPe04 Jones, Neil; Pevzner, Pavel: An Introduction to Bioinformatics Algorithms, 1. Auflage, Cambridge, The MIT Press, 2004
Im Internet unter *http://bioinformaticsinstitute.ru/sites/default/files/an_introduction_to_bioinformatics_algorithms_-_jones_pevzner.pdf*, abgerufen am 08.07.2018

Kap08 Kaptcianos, Jonathan: A Graph Theoretical Approach to DNA Fragment Assembly, American Journal of Undergraduate Research, 2008
Im Internet unter *http://www.ajuronline.org/uploads/Volume%207/Issue%201/71B-KaptcianosArt.pdf*, abgerufen am 12.10.2018

KiScPo10 Kingsford, Carl; Schatz, Michael; Pop, Mihai: Assembly complexity of prokaryotic genomes using short reads, BMC Bioinformatics 11.1, 2010
Im Internet unter *https://bmcbioinformatics.biomedcentral.com/articles/10.1186/ 1471- 2105-11-21*, abgerufen am 15.12.2018

MaPo17 Mackworth, Alan; Poole, David: Artificial Intelligence Foundations of Computational Agents, Second Edition, Cambridge University Press, 2017

MGMB07 Medvedev, Paul; Georgiou, Konstantinos; Myers, Gene; Brudno, Michael: Computability of Models for Sequence Assembly, Algorithms in Bioinformatics, Springer Berlin Heidelberg, 2007
Im Internet unter *https://pdfs.semanticscholar.org/930a/e9575da5a6943be5e5eaf40d606fdb28c9fa.pdf*, abgerufen am 15.12.2018

MoGa13 Molnar, Charles; Gair, Jane: Concepts of Biology - 1st Canadian Edition
Im Internet unter *https://opentextbc.ca/biology/*, abgerufen am 18.10.2018

Myers05 Myers, Eugene: The fragment assembly string graph, Bioinformatics Vol. 21, 2005
Im Internet unter *https://academic.oup.com/bioinformatics/article-pdf/21/suppl_2/ii79/6686032/bti1114.pdf*, abgerufen am 01.01.2019

NIGMS10 National Institute of General Medical Sciences: The New Genetics, 2010
Im Internet unter *https://publications.nigms.nih.gov/thenewgenetics/index.html*, abgerufen am 09.10.2018

PeTaWa01-a Pevzner, Pavel; Tang, Haixu; Waterman, Michael: A New Approach to Fragment Assembly in DNA Sequencing, RECOMB 01, 2001
Im Internet unter *http://sirius.cs.put.poznan.pl/~inf84789/acm/p256-pevzner.pdf*, abgerufen am 12.11.2018

PeTaWa01-b Pevzner, Pavel; Tang, Haixu; Waterman, Michael: An Eulerian path approach to DNA fragment assembly, Proceedings of the National Academy of Sciences Volume 98 Number 17, 2001
Im Internet unter *http://www.pnas.org/content/pnas/98/17/9748.full.pdf*, abgerufen am 12.10.2018

RäUk80 Räihä, Kari-Jouko; Ukkoven, Esko: The shortest common supersequence problem over binary alphabet is NP-complete, Helsinki, Theoretical Computer Science, 1980
Im Internet unter *https://www.sciencedirect.com/science/article/pii/030439758190075X*, abgerufen am 02.11.2018

SiDu12 Simpson, Jared; Durbin, Richard: Efficient de novo assembly of large genomes using compressed data structures, Genome Research 22.3, 2012
Im Internet unter *https://genome.cshlp.org/content/early/2012/01/22/gr.126953.111.full.pdf+html*, abgerufen am 04.01.2019

VoWi11 Vossen, Gottfried; Witt, Kurt-Ulrich: Grundkurs Theoretische Informatik, 5. Auflage, Vieweg + Teubner Verlag, 2011

weitere Quellen:

LeNeCo93 Lehninger, Albert; Nelson, David; Cox, Michael: Principles of Biochemistry, Second Edition, 1993
Im Internet unter *http://www.bioinfo.org.cn/book/biochemistry/chapt12/bio6.htm*, abgerufen am 13.11.2018

NHGRI National Human Genome Research Institute: An Overview of the Human Genome Project, 2016
Im Internet unter *https://www.genome.gov/12011238/an-overview-of-the-human-genome-project/*, abgerufen am 18.10.2018

NLM18 U.S. National Library of Medicine: Genetics Home Reference –Cells and DNA
Im Internet unter *https://ghr.nlm.nih.gov/primer/basics/dna*, abgerufen am 09.10.2018

NYCCT New York City College of Technology: Sanger Sequencing of DNA
Im Internet unter *https://openlab.citytech.cuny.edu/bio-oer/analyzing-dna/sanger-sequencing-of-dna/*, abgerufen am 13.11.2018

frühere Seminarvorträge

abrufbar unter *http://intern.fh-wedel.de/mitarbeiter/iw/lv/ws2012/seminar/*

Sem1 Kröger, Christoph: Bioinformatics
Sem2 Eichhorn, Cordula: Pairwise Sequence Alignment
Sem3 Schmidt, Timo: Multiple Sequence Alignment
Sem4 Reichardt, Nikolas: Phylogenese
Sem5 Weißer, Kim: Einführung in die Genomanalyse
Sem6 Jacobs, Timo: Konstruktion Phylogenetischer Bäume

Stein13 Von Stein, Niels: Persistente Methoden für DNA basierte Speichermedien, Seminarvortrag FH Wedel, 2013
Im Internet unter *http://intern.fh-wedel.de/fileadmin/mitarbeiter/iw/Lehrveranstaltungen/2013SS/Seminar/Ausarbeitung1DNA-basiertStein.pdf*, abgerufen am 09.10.2018