INSTITUT FÜR INFORMATIK
Algorithmische Bioinformatik

Universitätsstr. 1 D–40225 Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Contig-Assembly der MHC-Region mittels Linearer Programmierung

Marvin Lindemann

Bachelorarbeit

Beginn der Arbeit:	09. Juni 2019
Abgabe der Arbeit:	09. September 2019
Gutachter:	Prof. Dr. Gunnar W. Klau
	Dr. Alexander Dilthey

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 09. September 2019

_____

Marvin Lindemann

Zusammenfassung

Hier kommt eine ca. einseitige Zusammenfassung der Arbeit rein.

Inhaltsverzeichnis

1 Einleitung

Der Major Histocompatibility Complex, kurz MHC, ist ein Teilstück der DNA von Wirbeltieren, welches unter anderem eine tragende Rolle bei Vorgängen des Immunsystems besitzt. Durch seine große Variabilität dient es als Ausweis der körpereigenen Zellen, um sich vor dem Immunsystem von Fremdgewebe zu unterscheiden. Unter anderem ergibt sich daraus ein großes Interesse für Organtransplantationen, die konkrete Sequenz der MHC-Region zu kennen. Ferner ist der MHC der Teil des Genoms, mit der größten Relevanz für Erbkrankheiten. Leider resultiert aus der großen Variabilität ein eben so großes Problem bei der Analyse dieses Aufbaus. Um dieses Problem zu verstehen, müssen wir erst verstehen, wie bei der Assemblierung, also der Bestimmung der DNA-Sequenz, vorgegangen wird.

Etablierte DNA-Sequenzierungstechnologien ermöglichen keine direkte Ermittlung der Basenpaare bei langen DNA-Sequenzen. Daher werden sie in kürzere Stücke zerlegt, für die dann die Basenpaare bestimmt werden können. Diese kürzere Stücke heißen *Reads*. In einem *Assembly* werden die Reads zu größeren zusammenhängenden DNA-Sequenzen (genannt *Contigs*) zusammengefügt, welche so mit hoher Sicherheit in der originalen DNA-Sequenz vorliegen. Für das Erstellen solcher Reads gibt es verschiedene Verfahren, die alle verschiedene Vor- und Nachteile haben. So werden bei der sogenannten Illumina-Sequenzierung sehr kleine Reads mit einer Länge von ungefähr 200 Basenpaaren erzeugt.Diese weisen große Überlappungen untereinander auf, wodurch Reads, die große Übereinstimmungen haben, oftmals zusammgefügt werden können. Ein Fall, bei dem dies nicht möglich ist, zeigt diese Situation:

Trotz der Überlappung von Contig a und Contig b, kann nicht genau bestimmt werden, wie diese zueinander stehen. Ein solch repetitiver Aufbau über mehrere hundert Basenpaare tritt immer mal wieder in einer DNA-Sequenz auf.

Dies kann durch ein *Scaffolding* gehandhabt werden. Dabei werden Contigs nicht lückenlos zu größere Contigs zusammengefügt, sondern nur relativ zueinander Positioniert und können dabei auch Lücken zueinander aufweisen.

Für kurze Distanzen kann eine Paired-End-Sequenzierun durchgeführt werden. Dabei werden auch längere DNA-Stücke mit einer Länge von 400 bis 800 Basenpaaren von beiden Seiten gelesen. Das Illumina-Verfahren kann nur die 150 ersten und letzten Basenpaare bestimmen. Wenn die ersten 150 Basenpaare nun zu Contig a passen und die letzten 150 Basenpaare zu Contig b, so kann aus deren Position in Contig a und Contig b die Entfernung der beiden Contigs eingegrenzt werden. Die Länge der 400-800 Basenpaare langen Reads folgt einer bekannten Verteilung. Dadurch kann die Entfernung der Contigs gut geschätzt werden, wenn viele Reads Contig a mit Contig b verbinden.

Noch größere Schwierigkeiten machen eine andere Form von repetitiven Regionen in der

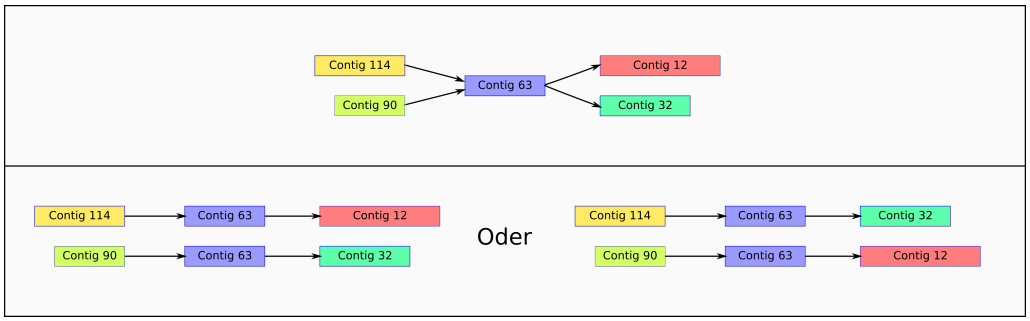


Abbildung 1: Repeat in einer Sequenz

DNA, bei denen sich zwei Regionen gleichen, welche in unterschiedlichen Bereichen in dem Strang befinden. Das Problem dieser Repeats wird in Abbildung 1 verdeutlicht. Zu sehen sind fünf Contigs, wobei ein Pfeil von Contig a zu Contig b bedeutet, dass in der DNA Contig a direkt vor Contig b kommt. Da Contig 63 (blau) zwei mal in der DNA vorkommt, ist es nicht trivial zu bestimmen, wie der Strang verläuft. Und MHC ist sehr repetitiv, somit werden viele Contigs mehrfach in der Sequenz auftauchen. Daher reichen die Daten aus Illumina nicht aus um MHC zu assemblieren.

Eine weitere Möglichkeit der Sequenzierung ist die Nanopore-Sequenzierung. Hierbei können sehr lange Reads von 10 000 bis über 1 000 000 Basenpaaren gelesen werden. Aufgrund des Längenunterschieds werden die Reads aus der Illuminar Sequenzierung auch *Short-Reads* genannt, und die Reads aus der Nanopore-Sequenzierung *Long-Reads*. Durch ihre Länge umfassen die Long-Reads oftmals bereits vollständige Repeats plus Umgebung. In unserer Abbildung 1 entspräche dies einem Read, in dem Contig 114, Contig 63 und Contig 32 (gelb, blau und grün) Teil von einem Read sind und es keine Schwierigkeiten diesbezüglich gibt. Leider ist die Fehlerrate bei dieser Methode sehr hoch. So werden sehr viele Long-Reads aus der selben Region benötigt, um die richtigen Basenpaare einigermaßen verlässlich zu bestimmen.

Daher hat sich die Manchot-Forschungsgruppe vom Institut für Medizinische Mikrobiologie und Krankenhaushygiene der HHU, mit deren Zusammenarbeit diese Arbeit entstand, eine Kombination der beiden Methoden entwickelt. Die verlässlichen Contigs aus der Illuminar-Sequenzierung wurden auf die Long-Reads gemappt und so deren Abstände zueinander bestimmt. Betrachten wir zur Anschauung Abbildung 2 eines Long-Reads. Die kleinen rot eingefärbten Bereiche stellen Fehler dar, die bunten Bereiche stellen Gebiete dar, in denen die Basenpaarsequenzen mit denen eines Contigs aus den Illuminar-Daten übereinstimmen. In der selben Situation wie in Abbildung 1 hätten wir nun die Information erhalten, dass Contig 114 und Contig 32 zusammengehören, also die rechte Auflösung der Abbildung richtig ist. Die Manchot-Forschungsgruppe hat diese Daten ge-

Abbildung 2: Ein Long-Read über mehrere Contigs

sammelt und daraus eine Liste aus paarweisen Daten extrahiert. Diese besitzt die Form: Contig a hat zu Contig b die Entfernung d (in Anzahl von Basenpaaren zwischen a und b). Dieses Dreiertuple aus zwei Contigs und einer Distanz nennen wir einen *Constraint*.

Es bleiben noch einige Schwierigkeiten für die Assemblierung zu beachten. Die Distanzwerte zwischen den Contigs sind meistens durch Fehler in den Long-Reads verfälscht. Dadurch sind erst mehrere Constraints, die eine ähnliche Distanz zwischen zwei Contigs prognostizieren, wirklich belastend. Bis zu welchen Distanz sich Constraints noch bestätigen und ab wann sie sich widersprechen ist hierbei ein entscheidender Aspekt, der betrachtet werden muss. Die Repeats lassen sich nicht immer so eindeutig auflösen wie in unserem Beispiel. In manchen Fällen kann erst im Gesamtzusammenhang erkannt werden, wie der Strang verläuft. Letztlich bleibt noch die große Datenfülle als Herausforderung zu nennen: Bei rund 122 000 auftretenden Distanzen zwischen 2 124 Contigs ist eine manuelle Zusammenfügung nicht zielführend und mindestens eine Teilautomatisierung der Prozesse obligatorisch. Auf der anderen Seite sind die Constraints nicht gleichmäßg verteilt, sodass es Regionen gibt, bei denen mit sehr wenig Informationen ausgekommen werden muss.

Hier stellt sich die Frage, mit welchen Methoden diese Probleme bewältigt werden können. Denkbar wäre eine Umsetzung mittels Linearer Programmierung. Das Ziel dieser Arbeit wird sein, zu untersuchen, ob lineare Programmierung hierbei anwendbar ist, und welche Vor- und Nachteile lineare Programme mit sich bringen.

2 Formalisierung

Die vorliegenden Daten der Manchot-Forschungsgruppe bestehen aus zwei Dateien:

1. einer Liste von allen Contigs und deren Länge gemessen in der Anzahl an Basenpaaren, die im Contig auftreten
2. einer Datei mit den eingangserwähnten Constraints.

Letztere besitzt folgenden dreispaltigen Aufbau:

$$\begin{array}{ccc} a & b & 1000 \\ a & b & 1100 \\ b & c & 3000 \\ a & c & 6000 \end{array}$$

Dabei entspricht jede Zeile einem Constraint. Die ersten beiden Zeilen geben jeweils die beiden involvierten Contigs an. In der dritten Spalte sind die gemessenen Distanzen angegeben. Diese sind als Entfernung vom rechten Rand des ersten Contigs zum linken Rand des zweiten Contigs in Basenpaaren zu interpretieren. Negative Distanzen sind auf Überlagerungen einzelner Contigs zurückzuführen. Durch Hinzunahme der Längen der Contigs lassen sich die Distanzen auch so modifizieren, dass sie den Abstand der ersten Basenpaare der jeweiligen Contigs angeben. Im Folgendem werden wir nur noch die

so modifizierten Distanzwerte verwenden. Zusätzlich zu den Constraints aus den Long-Reads stammen etwa 2% der Constraints direkt aus der Illumina Sequenzierung. Hier wurde mehrmals die Distanz zwischen benachbarte Contigs gemessen und der Durchschnitt daraus berechnet. Dadurch können auch nicht ganzzahlige Werte als Distanzen auftreten.

Wir werden im folgenden oft eine graphentheoretische Darstellung der Daten verwenden. Dabei sind die Contigs Knoten in einem Multigraphen und ein Constraint (a, b, d) eine gerichtete Kante von a nach b mit Länge d als Attribut. Dabei werden die Begriffe Contig und Knoten sowie Constraint und Kante synonym verwendet: „Nachfolger eines Contigs“, „ausgehender Constraint“.

Zu einer gegebenen Positionierung ist der Fehler eines Constraints gegeben durch den Unterschied von der Distanz in der Positionierung zu der Distanz im Constraint. Wenn also Contig a 300 Basenpaare vor Contig b positioniert wurde, dann haben die beiden Constraints $(a, b, 200)$ und $(a, b, 400)$ beide einen Fehler von 100. Ein Constraint ist durch eine Positionierung erfüllt, bzw eine Positionierung von einem Constraint bestätigt, wenn der Fehler des Constraints kleiner als ein Schwellenwert ist, der im Laufe des folgeden Abschnittes festgelegt wird.

2.1 Anforderungen an die Lösung

Nun wollen wir einige Gütekriterien für mögliche Lösungen festlegen, um während der Bearbeitung Orientierungspunkte für die weitere Optimierung des Algorithmus zu haben und um diesen nach Abschluss anhand dieser Kriterien zu bewerten. Folgende Punkte sollen beachtet werden:

1. Wenn mehr als ein Constraint eine ähnliche Distanz zwischen zwei Contigs sieht, sollten die Lösung diese Distanz möglichst gut erfüllen.

2. Ein Großteil der restlichen Constraints sollte auch zu der Positionierung passen.

3. Es sollten möglichst wenig Contigs nahe zueinander positioniert werden, die keine gemeinsamen Constraints aufweisen. Diese Situation wird durch Abbildung 3 illustriert. Der obere Teilabschnitt zeigt eine Lösung, bei der die Bedingung nicht erfüllt wurde. Realistischer ist jedoch, dass hierbei ein Teilblock bestehend aus Contig 12 und Contig 32 doppelt in dem DNA-Strang vorkommt und bei der Lösung des Problems zwei separate Stränge ineinander verflochten wurden. Dies ist in der unteren Bildhälfte dargestellt.

4. Die Entfernung vom ersten zum letzten Contig sollte zwischen 4,8 Millionen und 5 Millionen Basenpaare lang sein, da dies die ungefähre Gesamtlänge des Strangs ist.

5. Bei einer Lösung wäre es zudem gut, wenn man Informationen darüber besitzen würde, wie wahrscheinlich es ist, ob verschiedene Teilgebiete tatsächlich so im Strang auftreten. Dabei wäre zum Beispiel eine Unterteilung in „sichere“ und „unsichere“ Gebiete interessant.

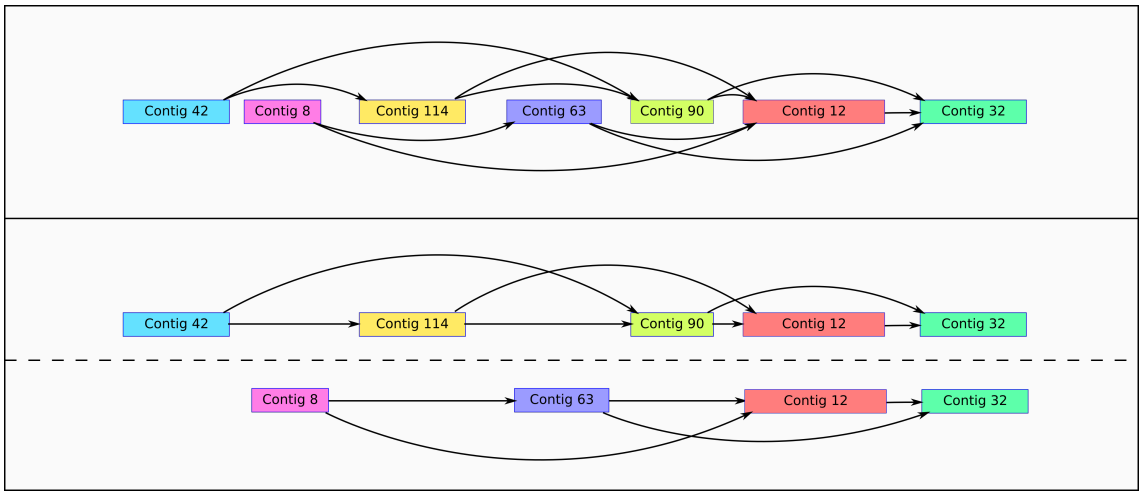


Abbildung 3: verflochtene Stränge

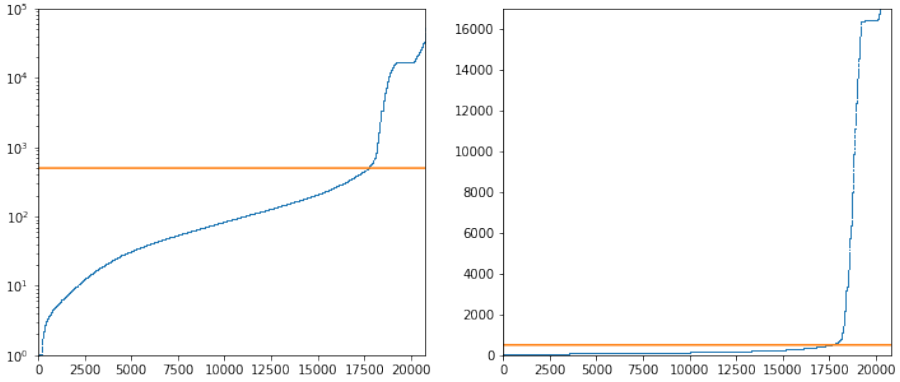


Abbildung 4: Standardabweichung der Distanzwerte

Nun wollen wir konkretisieren, bis zu welchem Abstand Constraints ähnliche Distanzen haben. Dazu betrachten wir, wie die Standardabweichung der Distanzen von Constraints verteilt ist. In der Abbildung 4 werden die zu einem Basenpaar zugehörigen Constraints gegen ihre Standardabweichung geplottet. Dabei wurden pro Basenpaar die jeweiligen Distanzen der Constraints zu einer (Multi-)Menge zusammengefasst. Es wurde symmetrisch vorgegangen, das heißt Paare der Form (a,b) und (b,a) werden in der gleichen Menge behandelt. Ferner wurden Mengen mit einem Element nicht berücksichtigt, da es hier keine Abweichung gibt. Für die jeweiligen Mengen wurden dann die Standardabweichungen berechnet, der Größe nach geordnet und dann mit Berücksichtigung dieser Ordnung geplottet. Der Plot wird mit zwei Skalierungen angegeben: Auf der linken Seite sieht man eine logarithmische Skalierung, während der rechte Plot eine lineare Skalierung verwendet.

Eine optische Betrachtung der Plots legt folgende Interpretation nahe: Es gibt einen Bereich der natürlichen Abweichung in der Datenmenge. Dies entspricht dem relativ fla-

chen Anfangsbereich des Graphen. Ab einem gewissen Punkt ëxplodieren"die Werte. Hier ist die Standardabweichung innerhalb der Constraints so hoch, dass man nicht mehr von natürlicher Abweichung innerhalb der Daten ausgehen kann. Die orangene Linie grenzt diese Bereiche intuitiv voneinander ab. Diese liegt bei einer Standardabweichung von 500 Basenpaaren. Somit unterstützen sich Constraints, deren Distanzwerte sich nicht um mehr als 500 Basenpaare unterscheiden.

Um die oben genannten Forderungen an eine Lösung zu erfüllen, ist es notwendig die Repeats auszumachen und die Constraints auf diese Repeats aufzuteilen. Dazu halten wir uns an das Prinzip: „so viel wie nötig, so wenig wie möglich“. Um dies sicherzustellen, sollte ein Contig, der mehrfach in der DNA vorkommen soll, zwei Punkte für jede seiner Versionen erfüllen:

1. Es sollte mindestens ein Contig in der Nachbarschaft liegen, zu welchem es zwei oder mehr Constraints gibt.
2. Sowohl unter den Vorgängern als auch unter den Nachfolgern des Contigs, sollte es je einen Contig geben, der einen gemeinsamen Constraint aufweist.
3. Sowohl zu einem der Vorgängern als auch zu einem der Nachfolgern des Contigs, sollte es ein Constraint mit dem Contig geben.

Der erste Punkt soll sicherstellen, dass es nicht einfach ein Fehler in den Daten ist. Der zweite Punkt stellt sicher, dass der richtige Contig als Repeat markiert wurde. Wenn der Distanzwert eines Constraints nicht erfüllt ist, ist erstmal nicht klar, welcher der beiden beteiligten Contigs eine Repeat-Version haben soll.

3 Das lineare Programm

3.1 Eine kurze Einführung

Die Problemstellung in dieser Arbeit lässt sich als lineares Programm (kurz LP) formulieren. In der linearen Programmierung möchten wir, unter Berücksichtigung von linearen Nebenbedingungen an die Funktionsparameter, eine lineare Funktion maximieren oder minimieren. Formal mathematisch lässt sich der Minimierungsfall so fassen:

$$\begin{aligned} Gegeben: &\quad c \in \mathbb{R}^n,\ A \in \mathbb{R}^{n \times m},\ b \in \mathbb{R}^m \\ Gesucht: &\quad \arg\min_{x \in \mathbb{R}^n} \{c^T x \mid Ax \leq b\} \end{aligned}$$

Dabei wird folgende Notation verwendet:

Variablen $x_1, ..., x_n$

Zielfunktion $c^T x = \sum_{i=1}^{n} c_i x_i$

Nebenbedingungen $Ax \leq b \ \Leftrightarrow \ \sum_{i=1}^{n} a_{ji} x_i \leq b_j,\ j = 1, ..., n$

Lineare Programme lassen sich in Polynomialzeit berechnen. Daher eignen sie sich oft als Werkzeug für verschiedenste Probleme. Es ist auch möglich, den Definitionsraum der Variablen (teilweise) auf ganze Zahlen zu beschränken. Dies bezeichnet man dann als ein ganzahliges lineares Programm (kurz ILP). ILPs bieten einige Möglichkeiten die mit LPs nicht umzusetzen wären. Sie sind dafür aber NP-schwer und somit wahrscheinlich nicht polynomialzeitberechenbar.

3.2 Assembly als LP

Im Folgenden bezeichnet C die Menge aller Contigs und D die Multimenge aller Constraints. Dabei wurde der Distanzwert jedes Constraints in D so umgerechnet, dass er die Entfernung von linken Rand vom ersten Contig bis zum linken Rand des zweiten Contigs angibt. Wir möchten die Contigs so positionieren, dass der durchschnittliche Fehler aller Constraints möglich klein ist. Als Formel:

$$\mathrm{pos} = \underset{\mathrm{pos}\,:\,C \to \mathbb{N}}{\arg\min} \sum_{(a,b,\delta) \in D} |\,\mathrm{pos}(b) - \mathrm{pos}(a) - \delta|$$

Um daraus ein lineares Programm zu machen, führen wir für jeden Constraint (a, b, δ) aus D eine Hilfsvariable ε ein:

$$\varepsilon = |\,\mathrm{pos}(b) - \mathrm{pos}(a) - \delta|$$

Mit Hilfe dieser Variablen können wir die zu minimierende Zielfunktion wie folgt darstellen:

$$\sum_{d \in D} \varepsilon_d$$

Nun müssen wir noch die Informationen über die Fehler, also $\varepsilon = |\,\mathrm{pos}(b) - \mathrm{pos}(a) - \delta|$, einbauen. Da wir ohnehin die Summe der Fehler minimieren wollen, ist es ausreichend, folgende Ungleichung zu fordern:

$$\varepsilon \geq |\,\mathrm{pos}(b) - \mathrm{pos}(a) - \delta|$$

Dies liegt daran, dass bei Minimierung immer die untere Schranke angenommen wird, welche in diesem Fall die Gleichheit ist. Nun ist die Betragsfunktion aber nicht linear. Sie lässt sich aber äquivalent durch die folgenden beiden linearen Ungleichung darstellen:

$$\begin{aligned} \mathrm{pos}(b) - \mathrm{pos}(a) - \delta &\leq \varepsilon \\ -\mathrm{pos}(b) + \mathrm{pos}(a) + \delta &\leq \varepsilon \end{aligned}$$

Zusammengefasst erhalten wir also folgendes lineares Programm für die Berechnung der optimalen Positionierung:

$$\begin{aligned} \text{Variablen:} \quad & \mathrm{pos}(c)\ \forall c \in C \ \text{ und } \ \varepsilon_d\ \forall\, d \in D \\ \text{Zielfunktion:} \quad & \sum_{d \in D} \varepsilon_d \\ \text{Bedingungen:} \quad & \begin{aligned} \mathrm{pos}(b) - \mathrm{pos}(a) - \delta &\leq \varepsilon_d \\ -\mathrm{pos}(b) + \mathrm{pos}(a) + \delta &\leq \varepsilon_d \end{aligned} \quad \forall\ (a, b, \delta) = d \in D \end{aligned}$$

Distanzwerte weisen zu große Schwankungen auf, um auf ein Basenpaar genau zu sein. Daher bietet es sich an, die Relaxierung des LPs zu betrachten, also auch reelle Positionen zuzulassen. Durch diese Lockerung der Bedingungen lässt sich das Programm wesentlich schneller lösen.

Dieses LP sorgt nur dafür, dass die Constraints möglichst gut erfüllt sind. Weder verhindert es, dass Contigs, welche keine Constraints vorweisen, nahe positioniert werden, noch erkennt es Repeats. Um das weitere Vorgehen planen zu können, bietet es sich trotzdem an, diesen Ansatz auszuführen und anhand der Lösung konkrete Problemstellen zu lokalisieren.

Die Implementierung erfolgt in der Jupyter-Umgebung der Programmiersprache Python. Dabei wird Gurobi, einem Programm das auf mathematische Optimierung spezialisiert ist, verwendet. Das Programm liefert uns als Rückgabe eine konkrete Positionierung der Contigs. Nun wäre es gut, wenn wir eine Möglichkeit hätten, festzustellen, inwiefern die von uns festgelegten Kriterien erfüllt sind. Im nächsten Kapitel werden dazu verschiedene optische Verfahren diskutiert

4 Wie man eine Lösung grafisch darstellt

Ein wichtiger Aspekt für die Arbeit mit großen Datenmengen ist die Visualisierung. Sie sollte in der Lage sein, die wichtigsten Informationen auf einen Blick sichtbar zu machen. Wir beschäftigen uns nun mit Möglichkeiten, konkrete Positionierungen der Contigs, also Abbildungen $pos : C \to \mathbb{R}$, zu visualisieren. Eine erste, naheliegende Option ist eine lineare Darstellungsweise. Hierbei werden die einzelnen Contigs gemäß ihrer Positionierung und ihrer Länge auf der reellen Zahlengerade eingezeichnet. Um Überlappungen zu beachten, bietet es sich an, mehrschichtig zu arbeiten. Mehrschichtig bedeutet hier, dass bei Überlappung zweier Contigs der hintere Contig in einer anderen Höhe geplottet wird. Diese Methode wird durch Abbildung **??** illustriert. Vorteile dieser Darstellungsweise sind unter anderem die folgenden Aspekte:

1. Der gesamte Strang inklusive seiner Länge ist auf einen Blick sichtbar. Dies liefert dem Betrachter einen groben Überblick. Ferner lässt sich der fünfte Unterpunkt der Anforderungen an Lösungen, die Länge des Stranges, so auch optisch leicht überprüfen.

2. Regionen, die hauptsächlich aus großen oder aus kleinen Contigs bestehen, lassen sich anhand des Plots leicht lokalisieren. Für den fünften Unterpunkt unsere Anforderungen an eine Lösung lässt sich dies als Indiz für die Sicherheit dieser Umgebungen verwenden. In Gebieten, in denen nur wenige, aber große Contigs vorkommen, sollten im Schnitt weniger Fehler hinsichtlich der geringeren Datenmenge auftreten als in mit vielen kleinen Contigs überfüllten Regionen.

3. Durch Hinzunahme von Färbungen ist es möglich, einige Kerninformationen leicht zugänglich zu machen. Dazu zählen etwa die Anzahl der Repeats eines einzelnen Contigs oder der Gesamtanteil an sich wiederholenden Contigs.

Jedoch gibt es einen zentralen Nachteil bei der Wahl dieser linearen Darstellungsform: Die Information, die wir durch die Constraints erhalten, werden in der Visualisierung nicht berücksichtigt. Damit lässt sich die Einhaltung der ersten drei Kriterien an eine Lösung optisch nicht analysieren. Beispielsweise sieht man nicht, ob benachbarte Contigs auch tatsächlich gemeinsame Constraints aufweisen. Um auch die Constraints mit einzubeziehen, bedarf es einer weiteren Darstellungsweise.

Um auch die Constraints mit einzubeziehen, bedarf es einer weiteren Darstellungsweise. Ähnlich wie in Abbildungen **??** und **??** lassen sich dafür gerichtete Graphen verwenden. Dabei bildet die Menge der Contigs die Knoten des Graphen. Constraints zwischen zwei Knoten lassen sich als (gerichtete) Kanten darstellen. Um eine gewisse Übersichtlichkeit zu erhalten, ist es aber sinnvoll, nicht alle Constraints zu plotten. Stattdessen nutzen wir die gegebene Positionierung aus und zeichnen nur Kanten für nah beieinander positionierte Contigs mit gemeinsamen Constraints.Dabei gehen wir für jeden auftretenden Contig a wie folgt vor:

- Zunächst werden alle Constraints gelöscht, deren Fehler in der vorgegebenen Positionierung größer als eine fixierte Konstante ist. Oftmals fällt die Wahl hier auf eine Zahl in der Größenordnung 500 gemäß der Überlegungen bezüglich der Standardabweichungen in dem Kapitel der Formalisierung.
- Ferner werden mehrfach auftretende Constraints zusammengefasst und alle Schleifen, also alle Constraints der Form (a, a, d) gelöscht.
- Sei nun N_a die Menge aller Contigs, die zusammen mit a in einem verbliebenden Constraints enthalten sind. Nun werden die Contigs in N_a und a selbst bezüglich ihrer Positionierung sortiert.
- Schließlich wird je eine Kante von a zu den ersten beiden Contigs, die hinter a positioniert wurden, gezeichnet. Umgekehrt wird je eine Kante von den ersten beiden Contigs, die vor a positioniert wurden, zu a gezeichnet.
- Zusätzlich werden noch grün gefärbte Kanten gezeichnet für alle Constraints, die nicht verlängerbar sind. Als verlängerbar gelten alle Constraints von a nach b und von b nach c für die es ein Constraint von a nach c gibt.

Im Allgemeinen tritt hierbei auch der Fall auf, dass eine Kante (a, b) während des Prozesses doppelt eingezeichnet werden soll: Einmal im Durchgang von a als auslaufende Kante und einmal im Durchgang von b als einlaufende Kante. Sie werden trotzdem nur einmal gezeichnet.

Die Wahl, dass jeder Knoten zwei Vorgänger und zwei Nachfolger bekommt, ist nicht fest, hat sich aber als besonders geeignet herausgestellt. Bei nur einem Vorgänger und einem Nachfolger kann es schnell passieren, dass ein eigentlich zusammenhängender Strang unzusammenhängend dargestellt wird. Abbildungen **??** und **??** illustrieren diese Situation. Dabei entspricht die erste Abbildung der tatsächlich im Programm auftretenden Darstellungsform. In der zweiten Abbildung wurde die Situation noch einmal etwas schematischer skizziert und optisch mit Farben unterlegt.

Auch können mehrere Kanten interessante Einblicke über die Struktur geben. So gibt es Contigs die nur eine einlaufende Kante besitzen, oder deren Kanten mehrere Contigs überspringen. Alles ein Zeichen von Unstimmigkeiten, die näher untersucht werden könnten.

5 Algorithmus Phase 1: Grobe Lösung aufbauen

Am Ende des vorherigen Abschnittes haben wir festgestellt, dass eine einfache Ausführung des lineare Programmes nicht ausreicht, um eine Lösung zu konstruieren, die unsere Gütekriterien erfüllt. In dem Graphen, der zu der Lösung korrespondiert, erkennt man dies anhand der starken Verflechtung/Verknotung verschiedener Teilstränge Um dies möglichst zu verhindern, gehen wir nun wie folgt vor:

Unser Ziel ist es, ein "Grundgerüst"für eine fertige Lösung aufzubauen. Dieses soll die Form eines Pfades von dem ersten bis zum letzten Contig des Stranges besitzen. Dabei sollte der Pfad möglichst viele Contigs enthalten, welche zueinander richtig positioniert sind. Ist ist nicht notwendig, dass alle Contigs korrekt positioniert sind, da das lineare Programme vereinzelte Fehlpositionierungen der Contigs gut beheben kann.

Wir speichern die Constraints als einen gerichteten Multigraphen G aus der NetworkX Bibliothek **??**.

Für zeitintensivere Berechnungen, erstellen wir eine reduzierte Version G_r, in der ähnliche Kanten zusammengefasst werden zu einer Kante. Dabei werden für je zwei Contigs alle Distanzwerte zwischen diesen beiden Contigs sortiert (wobei für eine Richtung die Distanzwerte als negativ betrachtet werden). Dann werden die Distanzwerte aufgeteilt, sobald sie sich mehr als 500 Basenpaare zum nächsten Wert unterscheiden. Die so gruppierten Werte werden durch ihr Median ersetzt und mit der Gruppengröße gewichtet.

Gehen wir dies an einen kleinen Beispiel durch. Gegeben sind folgende Constraints zwischen zwei Contigs a und b:

a	b	100
a	b	11 000
b	a	50
b	a	60 000
a	b	70
a	b	300
a	b	10 950
a	b	11 050
a	b	20
a	b	600
b	a	200

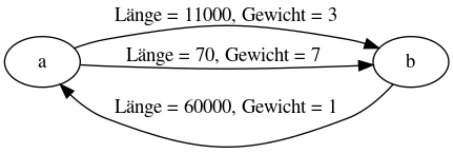
Die sortierten Distanzwerte:

$$-60000 \quad -200 \quad -50 \quad 20 \quad 70 \quad 100 \quad 300 \quad 600 \quad 10950 \quad 11000 \quad 11050$$

Gruppen bilden bei Abständen über 500:

$$-60000\,|\,-200\;\;-50\;\;20\;\;70\;\;100\;\;300\;\;600\,|\,10950\;\;11000\;\;11050$$

Es resultieren diese drei Kanten in G_r:



Unser Ziel ist es, ein Pfad durch G zu finden vom ersten bis zum letzten Contig, der möglichst viele Contigs beinhaltet. Dafür werden wir beim ersten Contig anfangen und uns Schrittweise vorarbeiten. Sollte sich der Graph aufteilen, so versuchen wir zuerst mithilfe von Heuristiken zu bestimmen, welcher Pfad der richtige ist. Sollte dies nicht möglich sein, werden alle Pfade ausprobiert.

Wir beginnen mit Contig 2345APD an Position 0. Für jede auslaufende Kante e aus 2345APD wird zu dem Endknoten von e die Länge der Kante in einer Liste gespeichert. Die ersten Listen der ersten 10 Contigs sehen so aus:

1483APD:6632 6662 6662
1395APD:6877 6943 6977 6979 6980 6982 6985 6993 6998 6999 7002
1596APD:9809 9867 9903 9931 9939 9942 9951 9978
2235APD:14013 14070 14179 14219 14263 14290 14304
1534APD:14687 14732 14841 14930 14957 14972
546APD:16242 16254 16372 16470 16550
577APD:18659 18782 18967 19040 19081
998APD:32244 32453
209APD:34061 34257
1635APD:43666

Es folgen noch 101 weitere Contigs mit jeweils nur einen Eintrag. Diese Werte werden genau wie bei der Erstellung des reduzierten Graphen gruppiert und Position der größten Gruppe den Contigs vorläufig zugeordnet.

Die Positionierung des Contigs mit der niedrigsten vorläufigen Position wird fest gesetzt.

Dies ist in diesem Fall Contig 1483APD an Position 6662. Auch von 1483APD werden die Distanzwerte der auslaufenden Kannten genommen. Diese werden aber zuerst zu seiner Position addiert und dann zu den schon bestehenden Werten von 2345APD hinzugefügt:

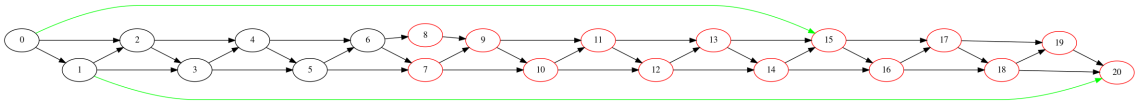


Abbildung 5:

1395APD:	6877	6943	6977	6979	6980	6982	6985	6993	6998	6999	7002		6982	6989	6994
1596APD:	9809	9867	9903	9931	9939	9942	9951	9978		9906	9946				
2235APD:	14013	14070	14179	14219	14263	14290	14304			14109	14308				
1534APD:	14687	14732	14841	14930	14957	14972		14771		14976					
546APD:	16242	16254	16372	16470	16550		16293	16554							
577APD:	18659	18782	18967	19040	19081		18698	19044							
998APD:	32244	32453		32248											
209APD:	34061	34257		34065											
1635APD:	43666														

So wird der Strang Contig für Contig aufgebaut. Wir sammeln die bereits fest Positionierten Contigs in einer Menge P und die vorläufig positionierten Contigs in einer Menge P_V.Den zuletzt in P hinzugefügten Contig bezeichnen wir mit a^*. Sollte ein Nachfolger von a^* bereits in P sein, so wird dieser nur erneut aufgenommen, wenn die neue Position mehr als 20 000 Basenpaare von der alten entfernt ist und die zugehörige Kante im reduzierten Graphen mindestens ein Gewicht von drei aufweist. Weiterhin darf a^* nicht selbst ein Repeat sein. Damit werden Endlosschleifen vermieden.

Doch wie wird erkannt, dass der Pfad sich aufspaltet? Dafür wird P_V näher untersucht. Es wird ein Teilgraph H von G_r aus den vorläufig positionierten Contigs genommen. Es werden nur Kanten übernommen, deren Fehler kleiner sind als 500. Sobald es in H mehr als ein Contig keine einlaufenden Kanten besitzt, wird genauer untersucht, mit welchem dieser Contigs weitergearbeitet wird.

Sei S_0 die Menge der Contigs ohne einlaufenden Kanten. Sollte es zu einem Contig s aus S_0 keine zur Vorpositionierung passende Kante von dem aktuell positionierten Contig aus geben, wird s aus S_0 entfernt. Damit wird sichergestellt, dass die Lösung ein Pfad darstellt.

Als erstes behandeln wir Fälle, wie den in Abbilung 5 dargestellten Fall. Der abgebildete Graph ist aus der im letzten Kapitel erläuterten Methode entstanden. Die roten Knoten sind die Contigs aus P_V, die schwarzen Knoten sind die Contigs P, welche Constraints nach P_V aufweisen. Die Knoten wurden nach ihrer Position durchnummeriert und so benannt. Die Zahl an den Kanten gibt die Anzahl an Constraints an, welche die Kante unterstützen.

S_0 besteht aus den beiden Contigs 7 und 8. Für ein Grundgerüst der DNA ist es egal, mit welchem der beiden Contigs weitergearbeitet wird. Daher werden in einem Fall, bei denen von zwei Contigs aus S_0 die gleichen Contigs erreicht werden können, immer der Contig aus S_0 entfernt, welcher mit den meisten fest positionierten Contigs erfüllte Cons-

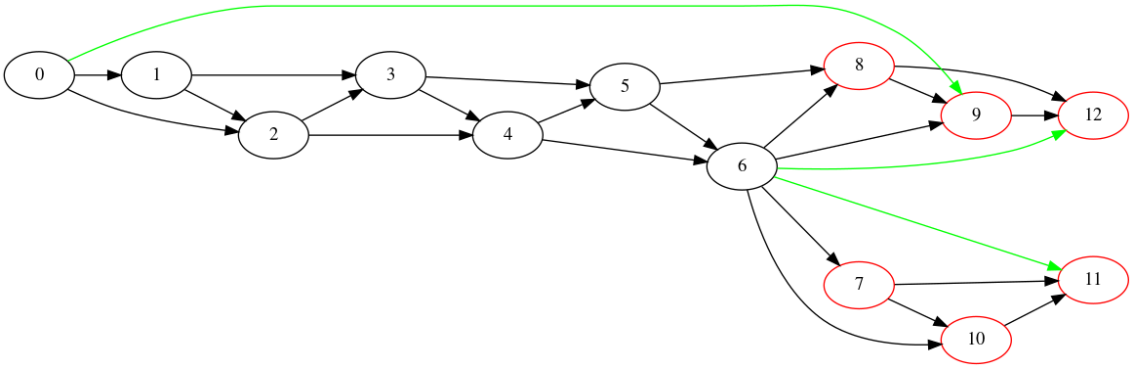


Abbildung 6:

traints aufweist. In diesem Fall wäre es Contig 7, da Contig 8 nur eine einlaufende Kante von 6 aus hat, wohingegen Contig 7 mindestens zwei Einlaufende Kanten hat (wahrscheinlich hat 7 aber auch von den restlichen vorherigen Contigs einlaufende Kanten).

Nun nehmen wir uns die restlichen Fälle vor, bei denen eine falsche Entscheidung gravierende Auswirkungen haben können. Ein Standardbeispiel ist in Abbildung 5 dargestellt. Die Menge aus Contig 7, 10 und 11, haben ausschließlich eine Kante zu Contig 6, wärend 8,9 und 12 zu sämtlichen vorhergehnden Contigs Kanten haben (Die sind zwar nicht alle eingezeichnet, aber es werden nur Contigs aus P dargestellt, die eine Kante in V haben und da Contig 6 der eizige Contig mit eine grünen Kante in die Menge 8,9 und 12 ist, können die Contigs von der 6 keine weiteren Kanten zu 8,9 oder 12 haben).

Die Typische Situation bei einem Repeat ist, dass aus der Repeat-Region (aus einem oder mehrere Contigs) es Constraints zu allen Abzweigungen gibt, aber nur zu einer Abzweigung gibt es Constraints die von vor der Repeat-Region kommen. Dies ist, was wir als erstes Untersuchen.

Wir ordnen jedem Element s aus S_0 jene Knoten aus H zu, welche ausschließlich von s erreicht werden können und bezeichnen die Menge dieser Knoten mit V_s (es gilt stets $s \in V_s$). Nun erstellen wir für jedes s aus S_0 eine Menge P_s aus Contigs in P welche Constraints in V_s aufweisen. Die P_s Mengen schneiden wir (der Schnitt ist nicht leer da a^* immer enthalten ist) und ermitteln den Contig l mit der niedrigsten Position aus dem Schnitt. Schließlich berechnen wir für jedes s die Anzahl k_s der Contigs aus P_s welche vor l liegt. Sollte für ein $\hat{s}$ aus S_0 gelten, dass $k_{\hat{s}} \geq 5k_s + 2$ für alle anderen $s \in S_0$ gilt, so ist dies ein deutliches Zeichen, dass $\hat{s}$ der nächste Contig ist.

Sollte dies nicht der Fall sein, so wird noch ein zweiter Aspekt betrachet. Wie in Abbildung 5 angedeutet, erzeugt ein Repeat in Daten ein Kreis in den dazugehörigen Graphen. Befinden wir uns gerade im roten Abschnitt, und haben den Kreis noch nicht durchlaufen, so sollten wir zuerst den Kreis gehen. Wenn wir den Kreisdurchlaufen haben, können wir immer noch den anderen Abzweig nehmen.

Leider sind diese Kreise nicht so eindeutig zu identifizieren. Es würde reichen, dass ein Constraint von einem der letzen Contigs zu einem der ersten Contigs geht, damit jeder Contig von jedem erreichbar ist. Somit kann man nicht einfach nach einem Kreis in G

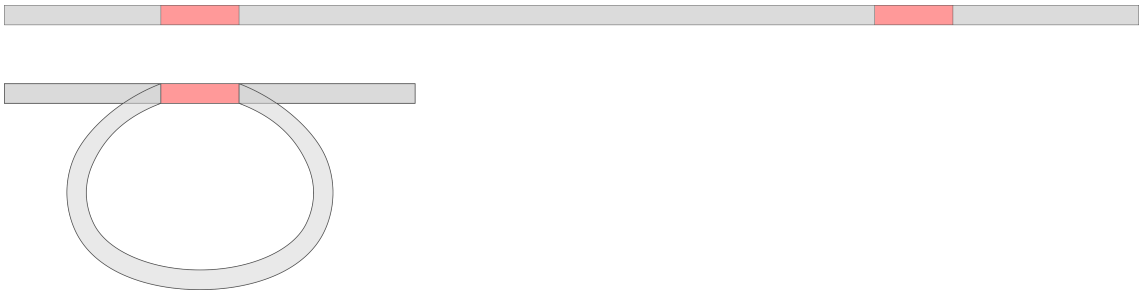


Abbildung 7: repeat in den Daten

suchen. Daher schränken wir G auf die Knoten ein, die noch nicht besucht wurden und suchen dann $s_1, s_2 \in S_0$, sodass es einen Weg von s_1 nach s_2 gibt, aber keinen Weg von s_2 nach s_1. In so einem Fall werfen wir s_2 aus S_0 raus.

Sollten wir es geschafft haben uns auf ein Pfad festzulegen, so wird dieser beschritten, ansonsten werden alle Pfade die noch nicht ausgeschlossen werden konnten, nacheinander ausprobiert und deren Lösungen gesammelt zurückgegeben.

Wenn es keine weiteren Contigs gibt und die Länge vom Strang im erlaubten Bereich ist, so wird die Lösung zurückgegeben.

Gibt es mehr als eine Lösung, so werden die Lösungen anhand der Anzahl der verwendeten Contigs und der Anzahl an Repeats verglichen (viele Contigs und wenig Repeats sind gut).

6 Algorithmus Phase 2: Lösung mithilfe vom LP verfeinern

7 Auswertung der Ergebnisse

8 Diskussion

Abbildungsverzeichnis

Tabellenverzeichnis