INSTITUT FÜR INFORMATIK
Algorithmische Bioinformatik

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine
Universität
Düsseldorf

# Contig-Assembly der MHC-Region mittels kräftebasierter Verfahren

**Jan Höckesfeld**

Bachelorarbeit

| | |
|---|---|
| Beginn der Arbeit: | 02. September 2019 |
| Abgabe der Arbeit: | 02. Dezember 2019 |
| Gutachter: | Univ.-Prof. Dr. G. Klau |
| | Dr. phil. A. 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 02. Dezember 2019

Jan Höckesfeld

## Zusammenfassung

Der Haupthistokompatibilitätskomplex (MHC) spielt bei der Immunreaktion bei Wirbeltieren eine große Rolle. Deshalb ist das Entschlüsseln der DNA in dieser Region besonders wichtig. Die hohe Variabilität dieser Region erschwert den Weg zu diesem Ziel. Ausgehend von Contigs, welche aus der DNA-Sequenz assembliert wurden, muss nun ein Scaffold zusammengefügt werden. Dazu sollen die Contigs mittels ermittelter Abstände zwischen ihnen, angeordnet werden. Zunächst wird dafür ein Prozess entwickelt, bei dem sich wiederholende Sequenzen im Genom (sogenannte Repeats) erkannt und verstanden werden. Dabei werden einige Eigenschaften dieser Repeats ausgenutzt, die das eigentliche Problem der Contig-Anordnung vereinfachen. Beruhend auf diesen Daten werden daraufhin Methoden vorgestellt, die eine Anordnung mittels kräftebasierter Verfahren aufbaut. Zudem wird sich mit der Evaluation solcher Lösungsansätze befasst.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Das Immunsystem muss fortlaufend seine Beschaffenheit anpassen, da ständig neue Viren und Bakterien abgewehrt werden müssen. Dies gilt auch für den Haupthistokompatibilitätskomplex (Major Histocompatibility Complex (MHC)), welcher durch den evolutionären Druck zu den variabelsten Genomregionen der Wirbeltiere gehört [1]. Die Hauptaufgabe der MHC-Proteine ist das Erkennen von Fremdproteinen, das für die Aktivierung einer Immunreaktion durch andere Proteine essenziell ist [1]. Daraus ergibt sich auch die Motivation einer Sequenzierung: Je höher die Übereinstimmung der MHC-Regionen beispielsweise bei einer Knochenmarktransplantation zwischen Spender und Empfänger ist, desto unwahrscheinlicher ist eine Abstoßungsreaktion. Infolge dessen ist abzuleiten, dass die MHC-Merkmale oftmals wichtiger als die Blutgruppenkompatibilität sind [2]. Aufgrund physikalischen Einschränkungen ist das Assemblieren des Genoms eine große Herausforderung. Dabei gibt es viele Probleme, wie zum Beispiel sich wiederholende DNA-Sequenzen. Dadurch können Genomregionen nicht am Stück, sondern nur in kleineren Contigs (contiguous sequences) assembliert werden. Es ist jedoch möglich, die Abstände zwischen diesen Contigs zu approximieren. Die Aufgabe dieser Bachelorarbeit ist nun das Anordnen der beschriebenen MHC-Contigs zu einem sogenannten Scaffold.

## 1.2 Probleme

Scaffolding wurde schon häufig in der Forschung aufgegriffen, sodass es dazu bereits einige Ansätze gibt. Allerdings ist die oben beschriebene Genom-Region besonders divers, sodass diese Scaffolder nicht länger ausreichen. Die Sequenz ist beispielsweise so variabel, dass ein Vergleich mit einer verwandten Sequenz nicht ausreicht. Dies liegt unter anderem an repetitiven Basenpaar-Sequenzen, welche das Zusammensetzen der einzelnen Contigs verhindern. Zudem gibt es einige Sequenzen innerhalb des MHC, welche gleich an mehreren Stellen des Genoms vorkommen, sogenannte Repeats. Aufgrund der technischen Assemblierung der Contigs kann nicht zwischen diesen Repeat-Vorkommen unterschieden werden.

## 1.3 Ziel

Da der Bereich eines potenziellen Scaffolding-Prozesses mittels eines kräftebasierten Verfahrens (Spring-Embedder) weitgehend unerforscht ist, ist dies als Ziel der Arbeit festzulegen. Spring-Embedder werden herkömmlich dazu verwendet, Graphen zu visualisieren. Nun soll eine Contiganordnung geschaffen werden, die von den visuellen Anforderungen der Embedder befreit ist. Möglicherweise ergibt sich so eine bessere oder ergänzende Methode, komplizierte Genom-Regionen schneller und genauer zu analysieren. Zudem muss eine Methodik entwickelt werden, welche die Lösung hinsichtlich ihrer Qualität überprüft. Letztlich soll ermittelt werden, ob kräftebasierte Verfahren ein gutes Ergebnis produzieren können.

### 1.4 Struktur

Wir beginnen zunächst im ersten Teil mit der Entstehung der Daten sowie der Erläuterung des Problems. In Teil zwei geht es dann um die beste Einbettung der rohen Daten in unsere Instanz. Anschließend versuchen wir das Kernproblem zu vereinfachen, indem wir uns mit dem Lösen von Repeats beschäftigen. Der darauffolgende Abschnitt erläutert Möglichkeiten, dieses Scaffold-Problem mittels kräftebasierter Verfahren zu lösen. Abschließend werden die Ergebnisse resümiert und das Vorgehen reflektiert.

## 2 Grundlagen

### 2.1 Sequenzierung und Assemblierung

Zunächst schauen wir uns die Entstehung der Contigs an. Dabei wird eine Sequenz häufig kopiert, zufällig in Fragmente geteilt und zu Reads sequenziert. Anschließend werden aus diesen Reads Contigs assembliert. Im Folgenden werden diese einzelnen Schritte genauer erläutert.

**Repeat** Ein Repeat ist eine sich an verschiedenen Stellen des Genoms wiederholende Sequenz. Sie können sehr kurz, aber auch mehrere hundert Basenpaare lang sein. Man bezeichnet ein Repeat-Vorkommen als eine bestimmte Stelle im Genom, an der diese Sequenz auftaucht. Zu unterscheiden ist dies von einer repetitiven Sequenz. Diese zeichnet sich durch ständig wiederholende Basenpaar-Strukturen innerhalb einer Sequenz aus (z.B. ATATATAT).

**Fragmente** Das zu analysierende DNA-Stück wird vielmals kopiert und anschließend zufällig in Fragmente geteilt. Es kann festgelegt werden, wie lang diese Fragmente ungefähr sein sollen. Dabei ist die tatsächliche Länge nicht bestimmbar. Je weiter diese von der gewollten Länge abweicht, desto seltener kommt sie jedoch vor.

**Short-Read: Illumina [3]** Typischerweise werden kurze einzelsträngige Fragmente (100 bis 600 bp) zu Doppelsträngen synthetisiert, wobei die Sequenz durch fluoreszierende Basen verfolgt wird. Beide Stränge des Doppelstrangs werden gelesen, indes dies immer vom 5′ zum 3′ Ende geschieht. Dadurch wird von beiden Enden des Fragment-Doppelstrangs sequenziert. Dieser Prozess bricht jedoch früher ab, sodass die Mitte des Strangs meist unbekannt bleibt. Man bezeichnet dies auch als Paired-End-Sequencing. Nun werden die dadurch entstehenden Reads zu einem Short-Read zusammengefasst. Die Vorteile von Illumina sind die hohe Genauigkeit und Schnelligkeit, wodurch viele Fragmente in kurzer Zeit analysiert werden können.

Bei hoch repetitiven Sequenzen lässt sich so außerdem der Abstand zwischen zwei Contigs feststellen. Passen beide Enden des Reads zu jeweils einem Contig, lässt sich der Abstand dieser beiden Contigs durch die Länge der Fragmente abschätzen. Wenn nun viele Reads zu ähnlichen Abständen kommen, ist es wahrscheinlich, dass dies der echte Abstand ist [3].

**Long-Read: Nanopore [4]** Bei dieser Methode werden die beiden Stränge der DNA an einem Ende mittels einer „hairpin“ verbunden. Danach wird die DNA denaturiert, sodass ein langer Strang entsteht. Dieser wird im Anschluss durch ein Transport-Molekül (Nanopore) einer künstlichen Membran geleitet. Die unterschiedliche Masse der Basen resultiert in unterschiedlich starken elektrischen Spannungen, sodass Sequenzen gelesen werden können. Dadurch, dass auch der komplementäre Strang eingelesen wird, kann die Fehler-Rate vermindert werden. Sie liegt dennoch zwischen 5-15% und ist der größte Nachteil der Nanopore. Da diese Technologie sehr lange Reads von über 10kb produzieren kann, werden diese Reads auch Long-Reads genannt.

Die Länge der Reads ermöglicht es, zwei bereits gefundene Contigs innerhalb des Reads zu finden und so approximierte Abstände zwischen ihnen zu ermitteln [4].

**Contig** Überlappende Reads werden nun übereinandergelegt, sodass eine längere contiguous Sequence (Contig) entsteht. Überlappungen sind identische Sequenzen vom Ende des einen Reads zum Anfang des Nächsten. Durch die hohe Anzahl von kurzen Reads und den anspruchsvollen Rechenaufwand der Überlappungen besteht lediglich die Möglichkeit, Contigs zu erstellen, jedoch keine komplette Sequenz. Zudem stellen fehlerhafte Basen der Reads ein Problem dar, da sie Überschneidungen behindern. Repeats finden hingegen sehr viele Überlappungen, auch an Stellen, an denen keine vorkommen. So werden zwei Reads aus zwei Repeat-Vorkommen auf das selbe Contig gesetzt, obwohl diese weit auseinander liegen. Daraus entstehen Repeat-Contigs, die zwei oder mehr Stellen im Genom repräsentieren. Repetitive Basensequenzen tragen zudem zu einem getrennten Vorkommen zweier Contigs bei. Dies ist damit zu begründen, dass potenziell alle Basen der Enden überlappen oder weitere Wiederholungen zwischen ihnen liegen könnten. Repetitive Basensequenzen lassen sich deshalb nicht aus mehreren Reads assemblieren. Somit spielen lange Reads eine wichtige Rolle, da diese, wenn sie repetitive Sequenzen komplett beinhalten, die Contig-Abstände bestimmen können.

**Scaffold** Die ungefähren Distanzen zwischen den Contigs durch die Analyse der Reads ermöglichen es nun, die Sequenzen in einen Gesamtkontext zu stellen (siehe Abbildung 1). Man projiziert die Contigs anhand der Distanzen in ein eindimensionales Koordinatensystem und bestimmt so die ungefähre Position der Sequenz-Stücke (Contigs) in der Genomregion. Zu einem späteren Zeitpunkt können dann die fehlenden Stücke zwischen den Contigs aufgefüllt werden.

## 2.2 Abstraktion

Im Folgenden werden wir einige Begriffe definieren, die das spätere Vorgehen vereinfachen sollen.

**Gerichteter Multigraph** Ein Multigraph ist ein Graph, bei dem es mehr als eine Kante zwischen zwei Knoten geben kann. Ein gerichteter Multigraph $MG = (V, E)$ besteht aus einer endlichen Menge Knoten $V$ und einer endlichen Menge Kanten $E$. Jedes Tripel $e = (u, v, k) \in E$ besteht aus einem Startknoten $u \in V$, einem Endknoten $v \in V$ und

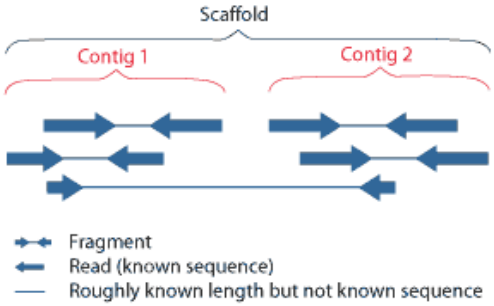


Abbildung 1: Überblick: Vom Read zum Scaffold
*Quelle: https://mycocosm.jgi.doe.gov/help/scaffolds.jsf*

einem Schlüssel $k \in \mathbb{N}$. Es gilt $E \subset V \times V \times \mathbb{N}$. Als Nachbarn von $u \in V$ bezeichnet man alle Knoten $v \in V$, die mindestens eine Kante $(u, v, x)$ oder $(v, u, y)$ $x, y \in \mathbb{N}$ in $E$ besitzen.

Eine Abbildung $\pi : E \to V \times V$ mit $\pi((v, u, k)) = (v, u)$ für alle $(u, v, k) \in E$ soll nun das Vergleichen von Knotenpaaren bei Multikanten vereinfachen. Mit anderen Worten zeigt $\pi(e_1) = \pi(e_2)$, dass die Kanten $e_1, e_2$ die gleichen Start- und Endknoten teilen.

**Scaffold-Graph** Gegeben sei eine Menge Contigs $V$ und eine Multimenge von Distanzen zwischen diesen Contigs $E \subset V \times V \times \mathbb{N}$. Sei $MG = (V, E)$ ein Scaffold-Graph (gerichteten Multigraph). Contigs werden als Knoten und deren Beziehungen als Multikanten $e \in E$ mit den Distanzen als Gewicht $w_e$ repräsentiert. Da es Multikanten mit der gleichen Länge geben kann, ist diese Länge nicht als Schlüssel des Multigraphen wählbar. Beruhend auf dem Kontext der Daten ist von einem zusammenhängenden Graphen auszugehen. Ein **vereinfachter Scaffold-Graph** ist ein gerichteter Graph $G = (V, E)$, der keine Repeats enthält. Da es keine Repeats gibt, sollte es auch keine Multikanten mit großen Distanzunterschieden geben. Die übrigen Contig-Beziehungen mit ähnlichen Distanzen lassen sich nun als eine Kante mit gemittelter Länge darstellen. Deswegen gibt es nur einfache gerichtete Kanten $(v, u) \in E$.

**Scaffold** Ein Scaffold $c$ beschreibt eine Einbettung der Knoten $v \in V$ eines vereinfachten Scaffold-Graphen $G$ in einem eindimensionalen reellen Raum. $c_v \in [0, N]$ sind dabei die Koordinaten der Knoten, wobei $N$ die approximierte Genom-Region-Länge beschreibt.

**Graph-Distanz** Die Graph-Distanz beschreibt die Länge (Gewicht) $w_e$ der Kante $e \in E$, sprich den Abstand der Contigs. Durch die Präsenz von Repeats kann es Multikanten mit sehr verschiedenen Abständen geben.

**Euklidische Distanz** Die euklidische Distanz ist der berechnete Abstand der Knoten $v, u \in V$, also der Abstand der Contigs im Scaffold $|c_v - c_u|$.

**(Vereinfachtes-) Scaffolding-Problem** Bei dem (vereinfachten) Scaffolding-Problem soll ein Scaffold $c$ auf einem (vereinfachten) Scaffold Graphen $G$ gefunden werden, sodass $\sum_{(v,u)\in E} ||c_v - c_u| - w_{(v,u)}|$ minimiert wird. Es ist anzustreben, die euklidische Distanz an die Graph-Distanz anzupassen.

## 2.3 Einführung in kräftebasierte Verfahren

Dieser Abschnitt ist hauptsächlich aus einer Beschreibung in*„Spring Embedders and Force Directed Graph Drawing Algorithms“* [5] entstanden.

Kärftebasierte Verfahren stellen eine Methode dar, das Scaffolding-Problem zu lösen. Sie werden meistens zum zeichnen von mehrdimensionalen Graphen eingesetzt, da sie zusammenhängende Knoten nah zueinander ziehen. Man kann sich die Kanten wie Federn vorstellen, die ein Kräfteequilibrium suchen, welches dem Gewicht der Kante entspricht. Wird dieses Gleichgewicht beispielsweise durch das Auseinanderziehen ebenjener Feder gestört, wirkt eine entgegengerichtete Kraft. Dabei werden physikalische Kräfte eingesetzt, die die Knoten beeinflussen. Diese Kräfte können mechanisch, elektrisch oder magnetisch inspiriert sein, aber auch frei gewählt werden. Die vorgestellten Algorithmen basieren auf der Idee des Hookesches Gesetzes, welche die Federdehnung beschreibt. Man berechnet dieses System nun iterativ, sodass sich die Kräfte des Gesamtsystems mit jedem Schritt verkleinern. Schlussendlich wird jedem Knoten eine entsprechende Koordinate zugeordnet. Ziel des kräftebasierten Verfahrens ist, dass sich jedes Kantengewicht als weitestgehend proportional zu den Abständen der Knoten darstellt.

Der Algorithmus von **Fruchterman und Reingold** (1991) setzt auf anziehende Kräfte zwischen adjazenten Knoten:

$$f_a(d) = \frac{c_{vu}^2}{k}$$

Des Weiteren werden abstoßende Kräfte zwischen allen möglichen Knotenpaaren definiert:

$$f_r(d) = -\frac{k^2}{c_{vu}}$$

Dabei stellt $c_{vu}$ die tatsächliche Distanz zwischen den Knoten im Raum dar. Konstante $k$ skaliert diese Distanzen in die Zeichenfläche (Canvas):

$$k = C\sqrt{\frac{Flaeche}{|N|}}$$

Zusätzlich wird die Technik „simulated annealing“ angewendet, bei der eine Temperatur linear gesenkt wird, was zum „abkühlen“ der Kräfte mit jeder Iteration führt. Diese Temperatur beschränkt die Bewegungsfreiheit der Knoten, wodurch Knoten nicht endlos durch den „Raum“ oszillieren können. Durch die Kühlung konvergieren die Knoten schließlich. Der herkömmliche Algorithmus beinhaltet keinen Weg, die Gewichte der Kanten mit in die Kräfte einzubeziehen. Es waren dazu in der Literatur keine Anhaltspunkte zu finden, allerdings bieten einige Python-Pakete wie NetworkX [6] eine Möglichkeit, Gewichte einzubinden. Die anziehenden Kräfte wurden folgendermaßen verändert:

$$f_a(d) = \frac{\frac{1}{w_{uv}} * c_{uv}}{k}$$

Diese Modifikation lässt sich damit begründen, dass auf Graph-Ebene nähere, aber im Koordinatensystem gleich weit entfernte Knoten stärker angezogen werden sollen.

Der Algorithmus benötigt $\mathcal{O}(I * (|E| + \mathcal{O}(|V^2|)))$, wobei $I$ die Anzahl der Iterationen sei. Mittels einer Variante, bei welcher weit entfernte Kanten ignoriert werden, kann die Laufzeit auf $\mathcal{O}(I * (|E| + \mathcal{O}(|V|)))$ reduziert werden. Für die Größenordnung der MHC-Daten stellt sich der Fruchterman-Reingold-Algorithmus als eine Methode dar, die sich durch ihre Schnelligkeit als praktikabel erweist. Die lokalen Optima charakterisieren ein generelles Problem. Durch ebenjene wird jeder Schritt aus diesem lokalen optimalen „Tal" heraus als schlechter gewertet. Zudem ist ursprünglich eine Gewichtung der Kanten nicht vorgesehen, es kann also auch deshalb zu falschen Anordnungen kommen.

Die Methode von **Kamada-Kawai** stützt sich auf einen anderen Ansatz: Hier werden nicht etwa adjazente Knoten angezogen sowie die übrigen abgestoßen, sondern es wird versucht, die Graph-Distanzen einzuhalten. Dieses Verfahren basiert nun nicht länger auf Kräften, sondern vielmehr auf Energie. Dabei wird durch eine Berechnung aller paarweise kürzesten Pfade zwischen Knoten eine Distanzmatrix $(d_{ij})_{i,j\in[1,|N|]} \in \mathbb{R}^2$aufgebaut. Daraus leitet sich dann folgende Feder-Energie $k_{ij}$ mit Konstante $C$ ab:

$$k_{vu} = \frac{C}{d_{uv}}$$

Durch die Minimierung der Energiefunktion $E$ optimieren wir die Positionen $p_1 ... p_n$ der Knoten aus $V$:

$$E = \sum_{i\in[1,|V|]} \sum_{j\in[i+1,|V|]} \frac{k_{ij}}{2}(|p_i - p_j| - l_{ij})^2$$

Hier soll $l_{ij}$ proportional zu $d_{ij}$ den gewünschten Abstand im Koordinatensystem darstellen. Minimiert wird hierbei über den zweidimensionalen Newton-Raphson-Prozess.

Diese Methode ist allerdings sehr zeitaufwändig. Sie benötigt alleine $\mathcal{O}(|V^3|)$ für das berechnen aller kürzesten Pfade. Deswegen ist dieser Algorithmus für kleinere Graphen ausgelegt, da er zudem $\mathcal{O}(|V^2|)$ Speicher für die Distanzmatrix benötigt. Zudem ist das Berechnen der kürzesten Pfade für das Scaffolding nicht optimal, da Repeats die Wege deutlich verkürzen, obwohl sie theoretisch deutlich länger sein sollten.

Eine weitere Möglichkeit ist die **multidimensionale Skalierung**, abgekürzt MDS, die nicht nur für das Zeichnen von Graphen, sondern unter anderem auch zum Reduzieren von Dimensionen verwendet wird. Wie bei Kamada-Kawai wird hier eine Distanzmatrix als Eingabe verlangt. Auch wird das gleiche Energiemodell angewendet, nur wird dabei $k_{ij}$ auf 1 und $l_{ij}$ auf die Graph-Distanz $d_{ij}$ gesetzt. Anders als bei Kamada-Kawai wird die Funktion nun *majorisiert* und ist im Gegensatz zur lokalen Newton-Raphson-Methode global anwendbar. Dadurch verbessert sich insbesondere bei großen Graphen das Resultat und ist außerdem im eindimensionalen Raum anwendbar. Diesen Prozess nennt man auch Stress-Majorization. Indem nur die Distanzmatrix eine Rolle spielt und auf eine beliebige Dimension projiziert werden kann, wird diese Methode auch für die Dimensionenreduktion verwendet. Multidimensionale Skalierung präsentiert sich als eine Methode, die vergleichbar mit dem vorherig definierten vereinfachten Scaffolding-Problem ist:

$$stress = \sum_{e\in E', r(e)=(v,u)} (|c_v - c_u| - w_e)^2$$

Multidimensionale Skalierung benutzt hierbei eine Quadrierung der Summanden. Das Scaffolding-Problem wiederum benutzt Betragsstiche. Dieser Unterschied beeinflusst jedoch nicht die spätere Lösung.

# 3 Phase 0: Pre-Processing

## 3.1 Ausgangssituation

Wir beginnen mit einem Scaffold-Graphen, mit Contigs als Knoten und deren Verbindungen als Kanten. Zur Verfügung stehen etwa 2.200 Contigs der MHC-Region mit einer Länge zwischen 100 und 29.000 Basenpaaren. Zudem gibt es circa 144.400 Kanten zwischen jenen Contigs $c \in V$. Die Längen der Contigs werden als $l_c$ bezeichnet. Diese Verbindungen beschreiben den Abstand vom Ende des ersten Contigs bis zum Anfang des Zweiten. Die Distanz einer Kante, die durch zwei Contigs definiert wird, kann auch negative Werte annehmen, falls sich Contig-Bereiche überschneiden. Es existieren einige Multikanten mit verschiedenen Distanzen. Außerdem kommen Contigs vor, die Kanten zurück zu sich selbst besitzen, was als Schleife bezeichnet wird. Es handelt sich also um einen gerichteten Multigraphen.
Die Kanten sind nicht zufällig zwischen den Knoten verteilt, sondern bilden eine sichtbare Richtung. Dies liegt daran, dass ein Contig aufgrund der technischen Herkunft nur Nachbarn hat, welche im DNA-Strang in ihrer Nähe liegen. Mit einer Graph-Dichte von 3% bleibt der Großteil des Graphen zusammenhängend, was gegen eine zufällige Anordnung der Contigs spricht. Je kürzer eine Distanz ist, desto wahrscheinlicher ist sie im Graphen enthalten. Dementsprechend folgt auch die Dichte der Distanzen einer steil abfallenden Kurve (Abbildung 12). Es gibt dennoch Knotengruppierungen mit hoher Kantendichte, wie Abbildung 2 zeigt. Hier ist zudem erkennbar, dass es eindeutige Pfade durch den Graphen gibt. Diese Pfade durchlaufen einige Zyklen. Häufig verursachen einige wenige Knoten das Bilden von Kreis-Strukturen, an anderen Stellen hält ein einzelner Knoten den Graph zusammen. Diese Abschnitte sind problematisch, da aus ihnen Fehler entspringen können. Die sichtbaren Pfad-Strukturen in Abbildung 2 sind größtenteils gleichsinnig ausgerichtet. Die große Anzahl an Kanten erschweren jedoch die Darstellung von Kantenrichtungen. Aufgrund der Instanzgröße ist $\mathcal{O}(V^2)$ mit circa $10^6$ Operationen noch praktikabel. $\mathcal{O}(E^2)$ mit circa $10^{10}$ Operationen sollte jedoch vermieden werden.

**Anmerkung.** *Da es in einem zusammenhängenden Scaffold-Graphen grundsätzlich mindestens $|N|$-1 Kanten geben muss (einfacher Pfad), kann man davon ausgehen, dass $\mathcal{O}(|V|) \in \mathcal{O}(|E|)$ liegt. Grundsätzliche Operationen wie die Tiefensuche ($\mathcal{O}(|V| + |E|)$) sind in $\mathcal{O}(|E|)$ machbar ($\mathcal{O}(|V| + |E|) \in \mathcal{O}(2|E|) = \mathcal{O}(|E|)$ ).*

## 3.2 Anforderungen

Folgende Beobachtungen sollten möglichst gut erfüllt werden:

- Die Abstände des Ergebnisses sollten vorzugsweise denen der errechneten Contig-

Abstände entsprechen. Messfehler können jedoch diesen Vergleich trüben, dementsprechend können Abstandsunterschiede von bis zu 500bp übergangen werden.

- Wegen der oben beschriebenen Entstehung der Daten sollte es bei kleineren Distanzen eine deutlich höhere Konnektivität geben. Gibt es also Datenpunkte (Contigs), die keine bekannte Beziehung zueinander haben (Kanten), liegen diese wahrscheinlich weit auseinander. Dies resultiert aus der limitierten Länge der Reads, welche die Abstände zwischen Contigs festlegt.
- Repeats müssen identifiziert und eingefügt werden. Dabei stellt der Zusammenfall der Repeat-Vorkommen in ein einzelnes Contig ein Problem dar, da viele widersprüchliche Abstände entstehen. Hier muss von widersprüchlichen Distanzen durch Messfehler differenziert werden.

## 3.3 Pre-Processing

Zunächst sollen diese Daten in ein Format gebracht werden, welches die Arbeit auf einem gerichteten Graphen ermöglicht. Wie beschrieben existieren auch negative Distanzen bezüglich der Kanten. Diese müssen nun so definiert werden, dass alle Abstände positiv sind. Wenn nun $w_{(a,b)} + l_a$ negativ ist, bedeutet dies, dass $b$ vor $a$ liegt. In diesem Fall tauschen wir Start- und Zielknoten:

$$\forall (u,v,k) \in E : e' := \begin{cases} (u,v,k) & \text{wenn } w_{(u,v)} + l_u \geq 0 \\ (v,u,k') & \text{sonst} \end{cases}$$

$e'$ ersetzt dabei $(u,v,k)$ in E ($k'$ ist ein bisher noch unbenutzter Schlüssel).

Bisher befinden sich die Distanzen der Kanten in einem „Contig-$a$-Ende zu Contig-$b$-Anfang"-Format. Eine „$a$-Anfang zu $b$-Anfang"-Distanz ist jedoch viel hilfreicher, da sich dieses Format deutlich besser als Graph darstellen lässt. Die Länge eines Knotens ist nicht länger wichtig. So wird die Entfernung nur über das Kantengewicht gesteuert und komplizierte Zusammenhänge, wie das Überlappen der Knoten-Positionen, werden verhindert. Die neue Länge entspricht:

$$w_{(a,b)} := w_{(a,b)} + l_a$$

Infolge vorhergehenden Filterns sowie von Messfehlern ist ein „Rauschen " in den Daten auszumachen. Einige Knoten bilden abgekapselte Komponenten, sodass der Graph initial nicht zusammenhängend ist. Wir reduzieren also die Daten auf die größte schwach zusammenhängende Komponente. Diese Komponente besitzt 96% aller Knoten.

Doppelkanten $e_1, e_2 \in E\ \pi(e_1) = \pi(e_2)$ mit ähnlichen Distanzen von circa 400 bp Abweichungen $|w_{e_1} - w_{e_2}| < 400$ sollen nun zusammengefasst werden, da sie auf Messungenauigkeiten zurückgeführt werden können. Nun kann man einen Fall modellieren, in dem gilt:

$$|w_{e_1} - w_{e_2}| < 400 \text{ und } |w_{e_2} - w_{e_3}| < 400 \quad \text{aber: } |w_{e_1} - w_{e_3}| > 400$$

wobei $e_1, e_2, e_3 \in E\ \pi(e_1) = \pi(e_2) = \pi(e_3)$. $e_1$ und $e_3$ liegen weit auseinander, sollten jedoch trotzdem zusammengefasst werden, da eine weitere Kante $e_2$ dazwischen liegt.

Wir bestimmen also alle Distanzabweichungen der Doppelkanten durch eine Ganzzahldivision:

$$div_e = \left\lfloor \frac{w_e}{400} \right\rfloor$$

Anschließend werden die Kanten nach ihrer Distanz $[e_1...e_m]$ sortiert. Nun gruppieren wir alle benachbarten $e_j e_{j+1}$ zu einer Gruppe $g_k = \{e_i...e_j\}$, wenn $|div_{e_j} - div_{e_{j+1}}| \leq 1$. Daraufhin bestimmen wir den Durchschnitt aller Distanzen einer Gruppe $g_k$ und speichern den Wert als Distanz $w_{e'_k}$ in einer neuen Kante $e'_k$. $e'_k$ fasst dabei die gemeinsamen Start- und Endknoten der Gruppe zusammen. Die Anzahl der Kanten von $g_k$ werden als Signifikanz $s_{e'_k}$ festgehalten. Dieser Wert gibt Auskunft über die Irrtumswahrscheinlichkeit. Eine Kante mit einer hohen Signifikanz ist eine Kante mit viel Evidenz, die wahrscheinlich keinem Fehler entspringt. Alle Kanten dieser Gruppe $g_k$ werden danach gelöscht.

Den daraus folgenden gerichteten Multigraphen nennen wir im folgenden $MG = (V, E)$. Damit auch Algorithmen, z.B. zur Suche des kürzesten Pfades, angewendet werden können, erstellen wir zudem einen gerichteten Graphen $G = (V, E')$ mit $E' = \{(u, v)|(u, v, k) \in E\}$. Dieser beinhaltet die jeweils signifikantesten Multikanten aus $MG$. Daraus folgt:

$$\forall (u, v) \in E' : w_{(u,v)} = \max_{\substack{e \in E \\ \pi(e)=(u,v)}} w_e$$

| Startknoten $u$ | Zielknoten $v$ | $w_{(u,v,x)}$ (Abstand) |
|---|---|---|
| contig A | contig B | 1.290 |
| contig A | contig B | 1.300 |
| contig A | contig B | 1.500 |
| contig A | contig B | 40.000 |
| contig A | contig B | 39.800 |
| contig A | contig C | 6.000 |
| contig B | contig C | 4.000 |

Tabelle 1: Beispielhafter Ausschnitt der Rohdaten
Die Zeilen sind bereits durch Trennstriche in Gruppen geteilt. Kantenschlüssel x ist künstlich und wird daher nicht gelistet.

| Startknoten $u$ | Zielknoten $v$ | $w_{(u,v,x)}$ (Abstand) | $s_{(u,v,x)}$ (Signifikanz) | $\in G$ |
|---|---|---|---|---|
| contig A | contig B | 1.364 | 3 | ✓ |
| contig A | contig B | 39.900 | 2 | × |
| contig A | contig C | 6.000 | 1 | ✓ |
| contig B | contig C | 4.000 | 1 | ✓ |

Tabelle 2: Daten nach der Gruppierung
Dies entspricht der Liste in Tabelle 1 nach der Gruppierung der Elemente.

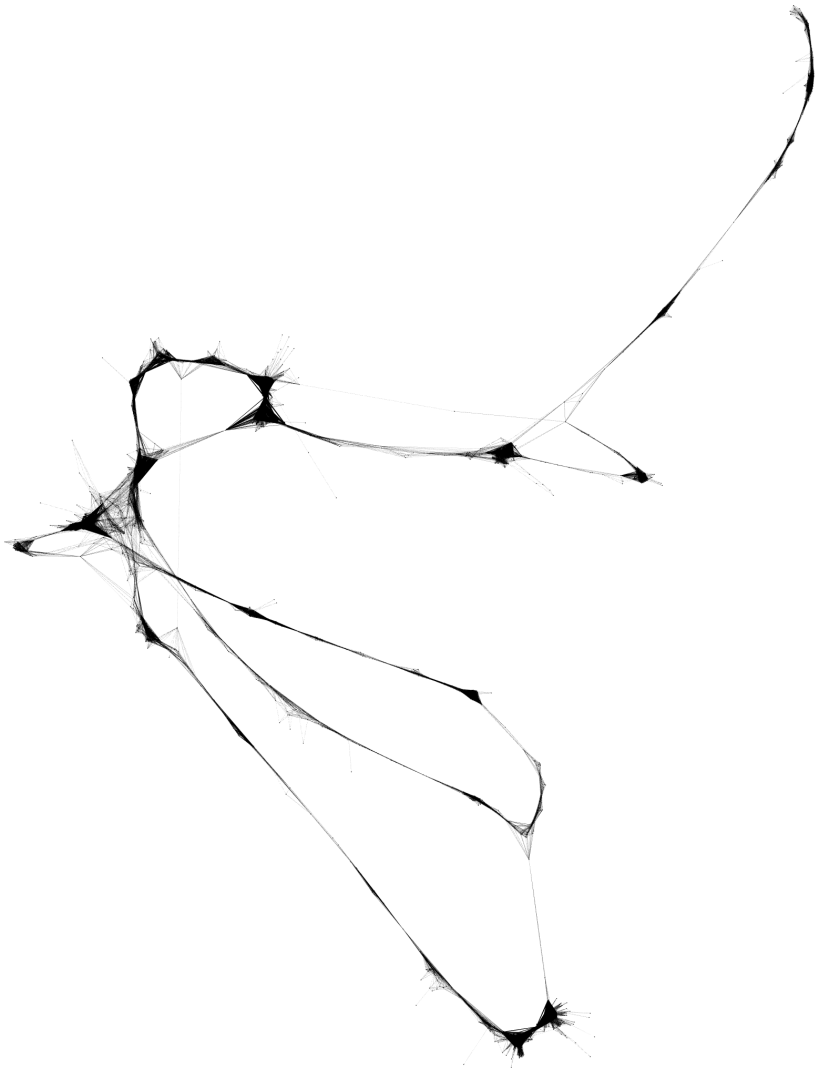

Abbildung 2: Ausgangssituation
Hier zu sehen ist der Ausgangsgraph mit dem Fruchterman-Reingold-Algorithmus ohne Berücksichtigung gewichteter Kanten. Diese Darstellung soll die Verteilung der Kanten veranschaulichen.

# 4 Phase 1: Repeats auflösen

Im Folgenden versuchen wir eine Heuristik zu entwickeln, um Repeats in dem Graphen zu identifizieren. Anschließend werden wir diese Repeat-Knoten duplizieren und ihre Kanten auf diese Duplikate aufteilen. Dies ist notwendig, um einen vereinfachten Scaffold-Graphen aufbauen zu können, damit sich kräftebasierte Verfahren wirkungsvoll anwenden lassen. Um Repeats aufzulösen, muss zunächst verstanden werden, wie sich dessen Eigenschaften von anderen Stellen unterscheiden. Eine Repeat-Sequenz aus den Contigs $c_1..c_n$ tritt in mehreren verschiedenen Regionen des Genoms auf. Dementsprechend besitzen diese Contigs Kanten zu mehreren verschiedenen Regionen. Wir konstruieren einen Beispielpfad von Contig $c_1$ nach Contig $c_n$:

$$c_1 \rightarrow c_2 \rightarrow .. \rightarrow \mathbf{r_1} \rightarrow .. \rightarrow \mathbf{r_m} \rightarrow c_i \rightarrow .. \rightarrow c_j \rightarrow \mathbf{r_1} \rightarrow .. \rightarrow \mathbf{r_m} \rightarrow c_k \rightarrow .. \rightarrow c_n$$

mit $2 < i < j < k < n$. Die Contigs $r_1..r_m$ gehören in diesem Fall zu einem Repeat, sie kommen also in zwei Regionen vor. $c_1..c_n$ hingegen sind reguläre Contigs. Wird nun der kürzesten Pfad von $c_1$ nach $c_n$ abgelaufen, werden die Contigs $c_i..c_j$ ignoriert. Das liegt daran, dass die Repeat-Contigs $r_1..r_m$ Kanten zu allen, teils weit auseinanderliegenden Knoten $c_i..c_j$ besitzen. So ist

$$c_1 \rightarrow c_2 \rightarrow .. \rightarrow \mathbf{r_1} \rightarrow .. \rightarrow \mathbf{r_m} \rightarrow c_k \rightarrow .. \rightarrow c_n$$

der deutlich kürzere Pfad. Dabei muss man zwischen nah und weit entfernten Regionen differenzieren. Wenn diese Regionen nah beieinander liegen, gibt es Kanten zwischen den Repeat-Regionen. Wie man in Abbildung 3 erkennt, versuchen wir hier einen kreisfreien Graphen zu erzeugen, da jedes Repeat eine Schleife im Pfad verursacht.

## 4.1 Grundsätzliche Idee

Bei herkömmlichen Scaffoldern werden die Repeats entfernt, sodass die problematischen Kanten nicht berücksichtigt werden müssen. Dies birgt das Problem, dass Informationen abhandenkommen können. Infolgedessen kann der Zusammenhang des Graphen wegfallen. Daher behalten wir die problematischen Stellen und versuchen sie zu lösen. Ist man sich sicher, ein Repeat mit den Contigs $r_1..r_m$ gefunden zu haben, kann man es nun auflösen. Zunächst dupliziert man alle Repeat-Contigs zu $r0_1..r0_m$ und $r1_1..r1_m$. Alle Kanten, welche $r_1..r_m$ als Start oder Ziel hatten, müssen nun auf die beiden Duplikate aufgeteilt werden. Die Herausforderung dabei stellt die Tatsache dar, dass für jede Kante einzeln entschieden werden muss, aus welcher Region sie stammt.

Ein reibungsloser Verlauf ist dabei essenziell, damit sich das Problem löst. Wird eine Kante falsch zugeordnet, gibt es einen Pfad, der alle Contigs zwischen den Repeat-Vorkommen überspringt.

Werden eine Reihe von Contigs fälschlicherweise als Repeat identifiziert, ist dies grundsätzlich kein Problem, da sich falsche Repeats beim Ausführen eines kräftebasierten Verfahrens wieder nah zueinander ordnen werden. Voraussetzung dafür ist, dass der Graph an dieser Stelle ausreichend viele Kanten besitzt, damit er weiterhin zusammenhängt.

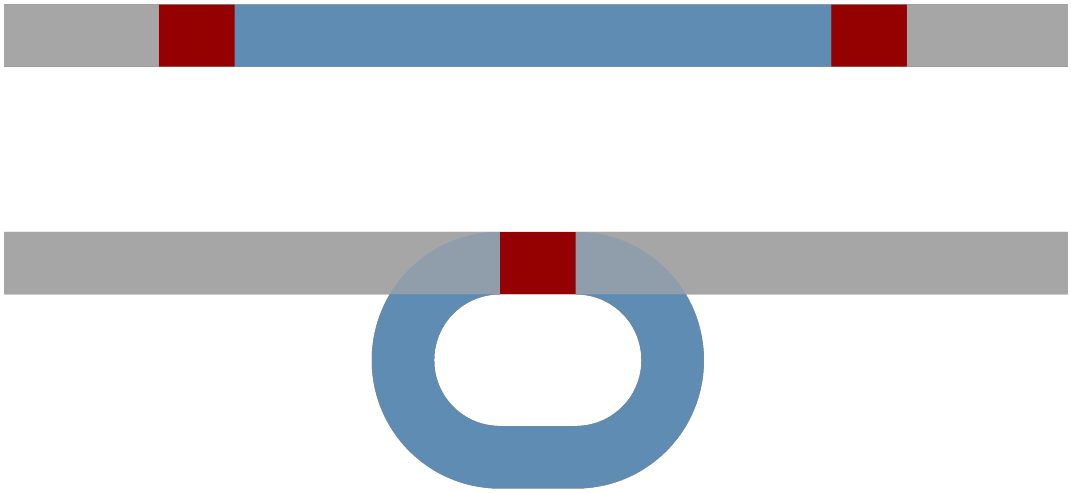

Abbildung 3: Repeat als Sequenz
Im oberen Strang ist ein Repeat (rot), wie er in der DNA vorkommt, dargestellt. Der untere Strang visualisiert eine Schleife (blau) am Repeat, nach dem Konstruieren der Contigs. Da die rot gekennzeichneten Bereiche aufgrund der Ähnlichkeit zusammengefasst werden, entsteht ein Kreis. Beim Berechnen eines kürzesten Pfades würde der blaue Teil übersprungen werden.

## 4.2 Eigenschaften zum Identifizieren von Repeats

Um nun diese Repeats aufzulösen, müssen erst Contigs aus den Repeats identifiziert werden. Es gibt einige Eigenschaften, die Repeat-Contigs von normalen Contigs unterscheiden:

- **Multikanten**, das heißt Kanten $(u, v, k), (u, v, l) \in E$, sind ein Indikator für ein Repeat. Sie weisen darauf hin, dass es verschiedene Entfernungen zwischen $u$ und $v$ gibt. Eines der beiden Contigs $u, v$ kommt dementsprechend zweimal in der Sequenz vor. Es bleibt zu entscheiden, welches der Contigs nun Teil des Repeats ist.
- **Schleifen** weisen auf ein doppeltes Vorkommen in der Sequenz hin. Dies liegt daran, dass $u$ von sich selbst entfernt liegt.
- **ungerichtete Kanten**, das bedeutet, es gibt Kanten $(u, v, k), (v, u, l) \in E$, die sowohl vorwärts als auch rückwärts vorhanden sind. Diese Kanten befinden sich zwischen dem blauen und dem roten Bereich in Abbildung 3. Ist der Abstand der Repeat-Vorkommen gering genug, kann es Contigs geben, die zu beiden Vorkommen Kanten haben. Auch hier bleibt , wie bei den Multikanten auch, zu entscheiden, welches der beiden Contigs Teil des Repeats ist und welches im blauen Bereich liegt (Abbildung 3).

- **Durchschnittlicher Grad** Theoretisch hat ein Repeat-Contig mit zwei Vorkommen einen doppelt so hohen Kantengrad wie seine Nachbarn. Jedoch variiert die Kantenanzahl der Knoten je nach Region stark, sodass kein gutes Ergebnis erzielt werden kann. Die ungleiche Verteilung der Kanten ist in Abbildung 2 deutlich erkennbar.

- **Starke Kräfte** Auf Knoten, die an mehreren Stellen auftauchen sollten, wirken höhere Kräfte, da widersprüchliche Kanten eine hohe Spannung aufbauen. Nach dem Erstellen einer ersten Lösung mittels kräftebasierter Verfahren ließen sich besonders starke Kräfte ermitteln. Allerdings bauen auch alle Contigs in der Nähe zum Repeat eine hohe Spannung auf. Repeats können dennoch geringe Kräfte besitzen, wenn sie aus einer Region mit geringer Kantendichte stammen. Zudem lässt sich nicht bestimmen, ab welchem Kräfte-Level ein Repeat vorliegt. Es muss bereits eine vorläufige Lösung existieren, um die Kräfte berechnen zu können.

- **Kanten zu mehreren Vorkommen** Hat ein Contig $u$ viele Kanten zu Contigs, die sein direkter Nachbar $v$ nicht besitzt, liegt dies an dem mehrfachen Vorkommen der Sequenz. $u$ besitzt Kanten zu unterschiedlichen Regionen, was auf ein Repeat schließen lässt. Dies wird in Unterabschnitt 4.3 näher erläutert.

Bei den Multikanten und den ungerichteten Kanten ist zu entscheiden, welcher der beiden Knoten Teil des Repeats ist. Zunächst wird dazu ein ungerichteter Teilgraph $MG' = (V', E')$ erzeugt, wobei $E'$ aus allen Multikanten beziehungsweise ungerichteten Kanten aus $E$ besteht. Jetzt lässt sich jeder Knoten $u$ durch alle Kanten $\{u, v\}$ in $E'$ bewerten ($N(u)$ sind die Nachbarn von $u$):

$$score_u = |\{v \mid v \in N(u) \wedge Grad(v) \leq Grad(u)\}|$$

So kristallisiert sich die Menge der Knoten mit hohem $score_u$ heraus, die besonders viele solcher Kanten besitzt. Dies reicht jedoch nicht, um Repeats eindeutig zu identifizieren. Als Zusatzinformation ist gegeben, dass die Daten etwa 50 Repeat-Contigs enthalten sollten. Bei einem minimalen Score von vier gibt es allerdings noch 880 Knoten durch Multikanten und 131 durch die ungerichteten Kanten. Zudem gibt es 223 Knoten mit Schleifen. Dies sind deutlich zu viele mögliche Repeats: Der Lösungsraum muss weiter eingeschränkt werden. Dazu kann man den Teilgraphen $MG'$ weiter begrenzen, sodass er nur noch Kanten einer höheren Signifikanz enthält. Zudem kann man auch verlangen, dass ein Repeat alle Eigenschaften erfüllt: Es sollen sowohl Multikanten als auch ungerichtete Kanten und Schleifen enthalten sein. Dies kann über die Schnittmenge dieser drei Mengen erzielt werden. Übrig bleiben dann noch etwa 81 mögliche Repeat-Contigs.

## 4.3 Zusammenhängende Komponenten

**Definition.** *Ein Graph ist $k$-fach zusammenhängend, wenn es zwischen jedem Knotenpaar $k$ knotendisjunkte Wege gibt.*

**Definition.** *Eine $k$-fache Zusammenhangskomponente ist ein maximaler $k$-fach zusammenhängender induzierter Teilgraph eines ungerichteten Graphen. Der Spezialfall $k = 1$ heißt Zusammenhangskomponente.*

Wie sich aus Abbildung 3 ableiten lässt, befinden sich DNA-Abschnitte zwischen zwei Repeat-Vorkommen in einem Kreis. Dabei ist dieser Strang zum Rest durch ein Repeat, den roten Bereich, verbunden. In einem idealen einfachen Scaffold-Graphen, in dem es demnach keine Repeats gibt, sollten alle Contigs, die räumlich nah aneinander liegen, auch eine Distanzkante besitzen. Angenommen $r$ ist ein Contig, welches alleine ein Repeat bildet: Dieses Repeat soll aus zwei entfernten Sequenz-Abschnitten stammen. Diese Abschnitte beinhalten jeweils die Contigs $r, a_1, .., a_n$ beziehungsweise $r, b_1, .., b_m$. Wir definieren $A = a_1, .., a_n$ und $B = b_1, .., b_m$ als die Mengen ohne das gemeinsame Repeat-Contig $r$. Nun besitzen diese Contigs aufgrund ihrer räumlichen Nähe Kanten zu anderen Contigs aus ihrem Abschnitt, also auch Kanten zu dem Repeat $r$. Da aber $A$ in der Sequenz weit entfernt von $B$ liegen soll, gibt es keine Kanten zwischen Contigs aus $A$ und $B$. Wenn nun ein induzierter Teilgraph mit allen Contigs aus $A$ und $B$ erstellt wird, zerfällt dieser Teilgraph in zwei Zusammenhangskomponenten. Die Knoten dieser Komponenten entsprechen den Contig-Mengen $A$ und $B$. Wenn $r$ nicht Teil eines Repeats wäre, würde dieser Teilgraph nicht zerfallen. So lässt sich jedes Contig $r \in V$ überprüfen, ob es aus einem einelementigen Repeat stammt.

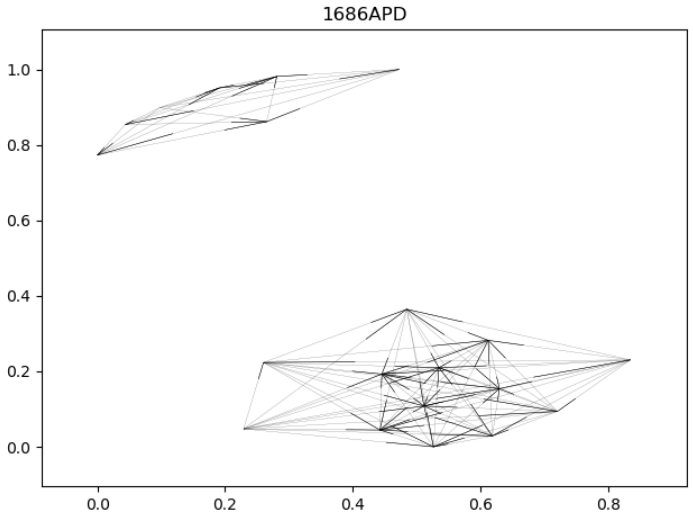


Abbildung 4: Nachbarschafts-Graph von Contig 1686APD
Dieser Graph teilt sich in zwei zusammenhängende Komponenten $A$ und $B$, da nur Contig 1686APD $A$ und $B$ direkt verbindet.

$A$ und $B$ sind allerdings noch nicht klar definiert, sie sollen nur grob in der Umgebung von $r$ liegen. Dazu definieren wir ein $C$ als die Menge aller Nachbar-Knoten zu $r$, dass heißt $C = N(r)$. Wenn man einen induzierten Teilgraphen über die Knoten in $C$ spannt und anschließend die Zusammenhangskomponenten bestimmt, bekommt man eben diese Komponenten $A$ und $B$, falls $r$ einem einelementigen Repeat entstammt (siehe Abbildung 4). Da der Algorithmus für zusammenhängende Komponenten und insbesondere der für $k$-fach zusammenhängende Komponenten auf ungerichtete Graphen ausgelegt ist, interpretieren wir vorübergehend jede gerichtete Kante in $MG$ als eine ungerichtete Kante.

Wenn nun Komponenten $A$ und $B$ gefunden werden können, lässt sich $r$ zu $r_A$ und $r_B$ auflösen. Jede Kante $(r, a_i, k)$ beziehungsweise $(a_i, r, k)$ mit $a_i \in A$ und $k \in \mathbb{N}$ kann nun $r_A$ zugewiesen werden. Folglich werden diese Kanten zu $(r_A, a_i, k)$ beziehungsweise $(a_i, r_A, k)$ umbenannt. Genauso kann $B$ zu $r_B$ zugewiesen werden.

Da das Bestimmen des Nachbarschafts-Graphen $(V', E')$ sich auf das Berechnen der Zusammenhangs-Komponenten in $\mathcal{O}(|E'|)$ beschränkt und $E'$ deutlich kleiner als $E$ ist, lässt sich dies effizient für jeden Knoten in $V$ durchführen. So bestimmen wir alle Nachbarn eines Knoten in $V$ und berechnen dessen Komponenten $C_1..C_m$. Da es auch Fehler in den Daten gibt, betrachten wir nur die Komponenten $C_i..C_j$ ($i \leq j$), welche mit ihrer Knotenanzahl einen Schwellenwert $t$ überschreiten.
Gibt es nur eine solche Komponente ($i = j$), kann abgebrochen werden. Dieser Knoten kann nicht als Repeat-Contig identifiziert werden und wir fahren mit dem nächsten Knoten in $V$ fort.
Wenn jedoch mindestens zwei Komponenten vorhanden sind, wird $r$ nach $r_1...r_{(j-i)}$ aufgeteilt und die jeweiligen $C_i..C_j$ zugewiesen. Dies ist der allgemeine Fall mit $j-i$ Repeat-Vorkommen. Ist $j - i = 2$, so haben wir den speziellen Fall mit $C_i = A$ und $C_j = B$. Die restlichen aussortierten Komponenten $C_l$ mit $|C_l| < t$ werden zusammengefasst und einem beliebigen Repeat-Vorkommen (z.B. $r_1$) zugeordnet. Dies ist notwendig, da sonst die Kanteninformation zum Repeat verloren geht. Es kann passieren, dass der Graph durch diese Aufteilung in Komponenten zerfällt. Hierauf muss diese Auftrennung abgebrochen werden. Anschließend wird mit dem nächsten Knoten in $V$ fortgefahren.

Dadurch, dass eindeutige Zusammenhangskomponenten nur entstehen, wenn die Repeat-Vorkommen genügend weit entfernt liegen, lassen sich weniger entfernte Repeats nicht auflösen. Ein weiteres Problem stellt die Contiganzahl in den Repeats dar. Jedes hier aufgelöste Repeat hat genau ein Contig und jedes weitere Repeat-Contig würde den Zerfall in einzelne Komponenten verhindern. Die Überprüfung aller Paare/Triple usw. von Repeat-Contigs und die Einordnung deren Nachbarn kann nicht mehr effizient geschehen. Deshalb suchen wir im Folgenden einen besseren Weg, Repeats mit multiplen Contigs aufzulösen.

## 4.4 k-fach zusammenhängende Komponenten

Dieses Problem lässt sich nun hinsichtlich der Anzahl an Contigs im Repeat generalisieren. Sei $r$ wieder ein Contig, welches aus einem Repeat mit mehreren Contigs stammt. Dieses Repeat muss wieder der Anforderung genügen, dass seine Vorkommen weit auseinander liegen. Wenn nun wie im vorherigen Abschnitt den Nachbarschafts-Graph von $r$ betrachtet, zerfällt dieser nicht. Dies liegt daran, dass es neben $r$ weitere Contigs gibt, welche die weit entfernten Sequenzabschnitte zusammenhalten. Um nun eine Auftrennung in Komponenten zu gewährleisten, wählen wir ein strikteres Kriterium, den k-fachen Zusammenhang. Die Sequenzabschnitte sind nur über Repeat-Contigs verbunden, demnach sind sie in sich stärker zusammenhängend als untereinander. Folglich lassen sich Repeat-Contigs identifizieren und auflösen.

Nun bildet $T = (V', E')$ einen induzierten Teilgraphen über alle Nachbarn $N(r)$ von $r$. Es gibt zwei oder mehr Mengen $C_1..C_m$, welche in sich $k$-fach zusammenhängend sind. Die Menge $R = (\bigcap_{i \in [1,m]} C_i) \cup \{r\}$ beinhaltet nun die Contigs des gesuchten Repeats. Es

ist zu beachten, dass diese Menge auch zusammenhängend sein sollte, da sie aus dem gleichen wiederholenden Abschnitt stammen.

**Anmerkung.** *Wenn es eine $j$-zusammenhängende Komponente gibt, so gibt es auch eine $(j-1)$-fache Komponente. Da es keine $j-1$ große Knotenmenge geben kann, die die Komponente trennt, kann sie nicht durch $j-2$ Knoten getrennt werden. Ist also die Komponente mit dem größten Zusammenhalt $n$ gefunden und $k \in [1,n]$, existieren auch $k$-fache Zusammenhänge mit mindestens einer Komponente.*

Nun muss für eine Aufteilung in diese Mengen ein geeignetes $k$ gefunden werden. Dazu berechnen wir zunächst alle Komponenten für jedes $k \in [1,n]$, wobei $n$ der maximal mögliche Zusammenhalt einer Komponenten von $T$ ist. Da es auch kleinere Komponenten geben kann, die durch Fehler entstehen könnten, definieren wir zudem einen Schwellenwert $t$, welcher die Mindestgröße einer Komponente spezifiziert:

$$customkComponents(T) = \{c \mid c \in kComponents(T), |c| \geq t\}$$

**Anmerkung.** *Für $k \geq t$ ist dieser Schwellenwert irrelevant, da für einen $k$-fachen Zusammenhalt mindestens $k$ Knoten benötigt werden.*

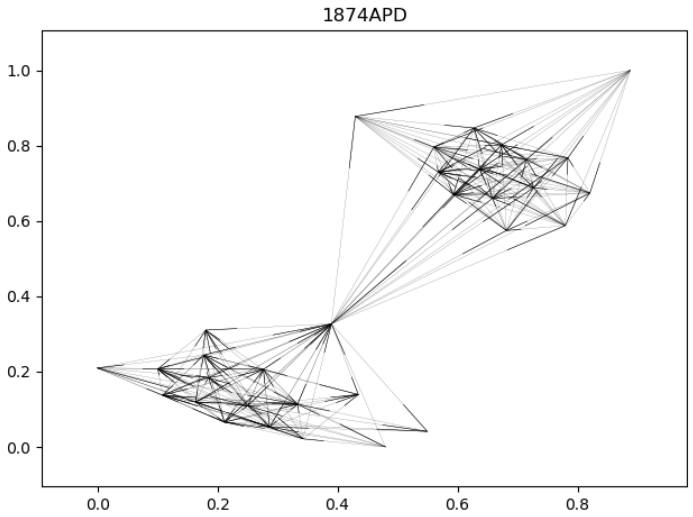


Abbildung 5: Nachbarschafts-Graph von 1874APD
Dieser Graph teilt sich durch zwei 2-fach zusammenhängende Komponenten mit einem Bindeglied, welches wahrscheinlich ein Repeat-Contig ist.

Letztendlich suchen wir ein $k$, das die $k$-fach zusammenhängende Komponenten $C_1..C_m$ über $\left|(\bigcup_{i\in[1,m]} C_i)\right|$ maximiert (im Standardfall entspricht $m = 2$). Gleichzeitig soll die gefundene Repeat-Menge $R = (\bigcap_{i\in[1,m]} C_i)$ maximal halb so groß wie $k$ sein:

$$\max\{\left|(\bigcup_{i\in[1,m]} C_i)\right| \mid k \in [1,n], customkComponents(T) = \{C_1..C_m\}, m \geq 2, \left|(\bigcap_{i\in[1,m]} C_i)\right| \leq \frac{k}{2}\}$$

**Anmerkung.** *Es gilt* $\left|(\bigcap_{i\in[1,m]} C_i)\right| < k$ *wenn* $m \geq 2$. *Dies ist damit zu begründen, dass die Komponenten* $C_1..C_m$ *sonst* $k$ *gemeinsame Knoten hätten, sie also* $k$*-fach zusammenhängend wären. Mit* $\left|(\bigcap_{i\in[1,m]} C_i)\right| \leq \frac{k}{2}$ *legen wir fest, dass sich die Komponenten tatsächlich deutlich unterscheiden (Abbildung 6).*

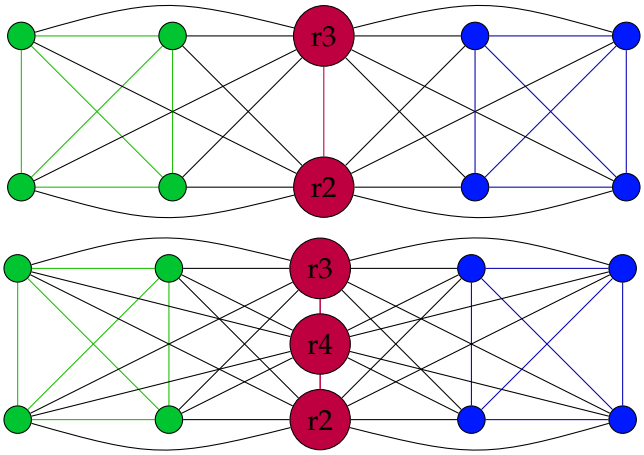


Abbildung 6: Beispiel für die $\frac{k}{2}$ Beschränkung
Zu sehen sind zwei $r_1$-Nachbarschafts-Graphen mit zwei, beziehungsweise drei weiteren Repeat-Contigs (Rot). Der obere Graph wird durch 4- und 3-fachen Zusammenhang in eine grüne und eine blaue Komponente (+ r3 r4) geteilt. Bei dem Graphen darunter ist der 4-fache Zusammenhang genauso gegeben, 3-facher Zusammenhang unterscheidet aber nicht mehr zwischen blau und grün. Deswegen können die roten Knoten nicht als Repeat identifiziert werden.

Seien $C_1..C_m$ jetzt die Komponenten eben dieser maximierten $customkComponents(T)$. Nun lösen wir $r$ wie bei den normalen zusammenhängenden Komponenten in $r_1, .., r_m$ auf, sodass $C_1..C_m$ ihren Repeat-Vorkommen zugewiesen werden können. Abgesehen von den Knoten in den Komponenten $C_1..C_m$ existieren weitere Knoten in der Nachbarschafts-Knotenmenge $V'$. Diese restlichen Contigs werden in der Menge $F$ dargestellt, sodass $(\bigcup_{i\in[1,m]} C_i) \cup F = V'$. Jeder dieser Knoten $u$ in dieser Menge $F$ wird allen Repeat-Vorkommen $r_i$ zugewiesen, bei denen $u$ eine Kante zu mindestens einem Knoten in $C_i$ hat. Wenn Knoten $u$ zu keiner dieser Komponenten Kanten hat, wird $u$ $r_1$ zugewiesen. So wird sichergestellt, dass keine Kanteninformationen verloren gehen. Die restlichen Repeat-Contigs $R$ lassen sich dann in den nächsten Iterationen als $k-1$-fache Komponenten finden.

Der Algorithmus von Moody und White [7] für $k$-fach zusammenhängende Komponenten benötigt eine Laufzeit von $O(|V^4|)$ und ist in der praktischen Anwendung zu langsam, um dies auf alle 2000 Knoten anwenden zu können. Für den Fall $k = 2$, also dem doppelten Zusammenhang, ist dies noch effizient mit einer Tiefensuche machbar. Wir können nun also alle Repeats mit maximal zwei Contigs identifizieren und auflösen, solange diese weit genug auseinander liegen. Zudem lassen sich Repeats mit $k$ Contigs auflösen oder auch validieren, ob es sich dabei tatsächlich um ein Repeat handelt. Da dies

sehr aufwändig ist, ist es nicht möglich, die gesamten Contigs dahingehend zu überprüfen. In der Implementation werden zunächst die mutmaßlichen Repeats aus Abschnitt 4.2 entdeckt. Diese werden anschließend für die Repeat-Identifizierung und folgende Repeat-Auflösung in den hier beschriebenen Algorithmus gegeben. Eine spezielle Version mit $k = 2$ (doppelter Zusammenhang, siehe Abbildung 5) kann anders als der allgemeine k-fache Zusammenhang anschließend auf alle $v \in V$ angewandt werden. Schlussendlich wird die beschriebene Methode aus Abschnitt 4.3 ausgeführt. Alle Methoden verwenden den Schwellenwert $t = 5$, welcher sich in der Praxis als sinnvoll erwies.

## 4.5 Stark zusammenhängende Komponenten

**Definition.** *Ein gerichteter Graph ist genau dann stark zusammenhängend, wenn es für alle $u, v \in V$ Wege $(u, v)$ und $(v, u)$ gibt.*

**Definition.** *Ein Teilgraph $G'$ von $G$ ist genau dann eine starke Zusammenhangskomponente, wenn $G'$ ein maximler induzierter Teilgraph ist, welcher stark zusammenhängt.*

Ob es weitere Repeats gibt, lässt sich leicht feststellen. So sollte ein repeatfreier Scaffold-Graph keine Kreise enthalten. Kreise sind Teil von stark zusammenhängenden Komponenten, welche ab einer bestimmten Anzahl an Knoten auf ein nicht gelöstes Repeat hinweisen. Knotenmengen unterhalb dieser Anzahl können aufgrund von Fehlern nicht sicher eingeordnet werden.

Ein kürzester Pfad von einem vermuteten Start-Contig zu einem End-Contig überspringt alle Repeat-Kreise, da diese den Pfad immer verlängern. Allerdings enthält der kürzeste Pfad zumindest teilweise Repeats, da diese den Pfad verkürzen. Wenn man nun also das erste und letzte Contig des Scaffolds ermitteln kann, lässt sich ein kürzester Pfad mit den besuchten Knoten als Menge $S$ ermitteln. Auch können die Knotenmengen $K$ der Kreise durch stark zusammenhängende Komponenten entdeckt werden. Deshalb sollte der Durchschnitt der Mengen Teil eines Repeats sein: $S \cap K$. Da ein Repeat auch aus mehreren Contigs bestehen kann, ist es allerdings wahrscheinlich, dass der kürzeste Pfad nicht alle Repeat-Contigs enthält. Da wir zudem nicht unbedingt Start- und Endknoten kennen, bleibt diese Methode eine Theorie, die nicht implementiert wurde.

Zusätzlich ist anzumerken, dass es sehr wohl auch Zyklen gibt, welche nicht durch Repeats verursacht wurden. So gibt es auch Kanten, die zwei Sequenz-Regionen miteinander verknüpfen, welche aber aufgrund ihrer hohen Distanz zueinander keine Kanten besitzen können. Diese Kanten müssen Messfehlern entspringen und haben deswegen keine anderen Kanten, die diese Tatsache stützen (Signifikanz). Für eine brauchbare Lösung sollten auch diese Kanten entfernt werden. Hier besteht allerdings das Problem, dass sich diese Zyklen nicht von Repeat-Kreisen unterscheiden lassen. So werden nicht nur Repeats gelöscht, sondern auch Contigs mit fehlerhaften Kanten entfernt.

# 5 Phase 2: Kräftebasierte Anordnung

Im nächsten Schritt soll nun das vereinfachte Scaffold-Problem gelöst werden. Wir gehen davon aus, dass alle Repeats aufgelöst wurden. Da es sich bei diesem Problem vermutlich um ein (NP-)schweres Problem handelt, benötigen wir eine Heuristik, um dieses zu lösen. So ähnelt die früher beschriebene Definition des Scaffold-Problems $\min \sum_{(v,u)\in E} ||c_v - c_u| - w_{(v,u)}|$ stark der einer linearen Anordnung, wessen Problemstellung NP-schwer ist[8]:

**Definition** (Linear Arrangement)**.** *Sei $G = (V, E)$ ein zusammenhängender ungerichteter Graph mit der Knotenmenge $V = \{1..n\}$ und E die Menge aller Kanten. Kanten $(i, j) \in E$ haben dabei ein Gewicht $w_{ij}$. Ein lineares Arrangement ist eine bijektive Abbildung $\pi : V \to \{1..n\}$ der Knoten in V. Sei $v \in V$, dann ist $\pi(v)$ die Positionierung des Knotens.*

**Definition** (Minimum Linear Arrangement Problem (MinLA))**.** *Wir definieren eine Kostenfunktion*

$$LA_\pi(G) = \sum_{(i,j)\in E} w_{ij} * |\pi(i) - \pi(j)|$$

*und minimieren diese: $min_{\pi\in Permutationen}\{LA_\pi(G)\}$. Wenn wir $w_{ij}$ auf 1 setzen, bekommen wir ein ungewichtetes Minimum Linear Arrangement Problem. [8]*

Wenn man nun von den Repeats absieht, ähnelt das MinLA dem generellen Scaffolding-Problem. MinLA berechnet jedoch diskrete natürliche Zahlen nur als Koordinaten und nicht euklidische Abstände. Zudem wird hier die Graph-Distanz multipliziert statt subtrahiert. Anders als bei kräftebasierten Verfahren können hier keine initialen Koordinaten übergeben werden. Diese Arbeit bezieht sich jedoch auf kräftebasierte Verfahren, welche das Problem durch Minimierung der Kantenkräfte löst.

## 5.1 Contig-Anordnung mittels Fruchterman-Reingold-Algorithmus

Die Idee ist zunächst klar: Löse den Scaffold-Graph mittels einer eindimensionalen Variante des Fruchterman-Reingold-Algorithmus und nutze die Reihenfolge der Koordinaten als Anordnung der Contigs. Nun nimmt man die Graph-Distanzen und erhält so die Koordinaten des Scaffolds. Da naheliegende Knoten eine hohe Konnektivität besitzen, verfälschen Kanten mit fehlerhaften Distanz-Gewichten das Gesamtergebnis nicht.

Da der Algorithmus vor allem für das Zeichnen von Graphen entwickelt wurde, erhalten wir nur Koordinaten aus einem vorher definierten Intervall $[0, b]$. So verliert die Abbildung den direkten Bezug zu den Graph-Distanzen. Es ist möglich, $b$ auf die vermutete Sequenzlänge $5 * 10^6$ zu setzen. Dadurch, dass es aber unvermeidliche „Ausreißer"-Knoten gibt, befinden sich die meisten Knoten nur in einem Bruchteil dieses Intervalls. Diese „Ausreißer" entstehen durch besonders widersprüchliche Kanten, welche die Kräfte deutlich erhöhen. Folglich muss noch ein Weg gefunden werden, diese Anordnung auf Genomkoordinaten abzubilden.

Ein entscheidendes Problem liegt hier in der niedrigen Dimension. Koren und Harel beschreiben in ihrer Publikation [9] das sogenannte Tausch-Problem. Dabei bilden die abstoßenden Kräfte eine Barriere, sodass Knoten ihre Plätze nicht tauschen können. So sei

beispielsweise eine Knotenanordnung A B C am energetisch günstigsten. Diese wird allerdings nie von B A C erreicht, da die abstoßenden Kräfte stärker als die anziehenden Kräfte sind (siehe Abbildung 7). Es muss einen Übergangsstatus geben, bei dem A und B sehr nah aneinander liegen. In diesem Moment steigen die abstoßenden Kräfte quadratisch an. Bei der mehrdimensionalen Variante ist dies anders: Die Knoten können auf andere Dimensionen ausweichen, sodass A nicht von B durchschritten, sondern umrundet werden kann. So beschreiben es auch Koren und Harel[10]: „*Interestingly, 2-D drawing is much easier for such methods. Probably, the reason is that there is less space for maneuver in one dimension when seeking a nice layout, which prevents convergence to an optimum. Furthermore, in several works even a 3-D layout is used to avoid local minima* “.

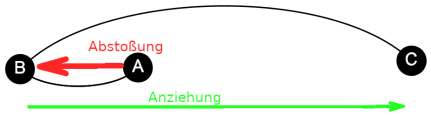


Abbildung 7: Das Tausch-Problem
Um das globale Optimum A B C zu erreichen, müsste B A passieren. Die abstoßenden Kräfte zwischen A und B sind jedoch größer als die anziehende Kraft zu C. So ist B A C ein schlechtes lokales Optimum.

Dementsprechend kann keine eindimensionale Variante verwendet werden. Ein zusätzliches Problem ist, dass die eindimensionalen Koordinaten mit diesem System auf ein theoretisches Zeichenbrett (Canvas) gezeichnet werden. Dadurch werden die tatsächliche Distanzen im Graphen so skaliert, dass sie sich in diesen theoretischen Canvas einfügen. Diese Koordinaten müssen auf die gewollten Genomkoordinaten abgebildet werden. Zudem ist nicht gezeigt, ob die gewichtete Version des Fruchterman-Reingold-Algorithmus tatsächlich Gewichte korrekt einbeziehen kann.

Abgesehen von diesen Schwierigkeiten ist das Tausch-Problem mit zusätzlichem Aufwand lösbar. Dazu wird zunächst eine n-dimensionale Version des Problems betrachtet, das auch als *hoch-dimensionale Einbettung* beschrieben wird. Je größer $n$ ist, umso besser können sich die Knoten anordnen, da mit jeder Dimension die Ausweichmöglichkeiten steigen. Harel und Koren schlagen in *High-Dimensional Embedding* [11] ein $n$ von 50 vor. So wird dieses Problem nicht nur deutlich einfacher, es konvergiert auch deutlich schneller. Fraglich ist jedoch, bis zu welchem $n$ von diesem Vorgehen profitiert wird, wenn der durchschnittliche Knotengrad einen ähnlichen Wert aufweist. Wenn jeder Nachbar $v$ eines Knoten $u$ orthogonal zu jedem anderen Nachbarn $k$ liegen kann, erhalten wir keine Datenreduktion. Wenn wir genügend Dimensionen einbeziehen, erzielen wir somit keine Vereinfachung des Problems. Ein vollständiger Graph mit fünf Knoten kann zum Beispiel in einem fünfdimensionalen kräftebasierten Verfahren perfekt eingeordnet werden. Infolgedessen entsprechen alle Kantengewichte exakt denen der euklidischen Distanzen. Auch dadurch vereinfacht sich das Problem nicht.

Ein weiterer potentieller Vorteil der hohen Dimensionalität ist die Vermeidung von lokalen Minima. So sieht man zum Beispiel in Abbildung 2 eine Ausführung des unge-

richteten Fruchterman-Reingold-Algorithmus auf dem Scaffold-Graphen. Dort sind sich kreuzende Wege erkennbar, die aber keine verbindenden Kanten haben. Dies ist ein klassischer Fall eines lokalen Optimums. Eine andere Anordnung wäre energetisch vorteilhaft, dafür müsste aber ein Tal mit stark abstoßenden Kräften durchschritten werden. Dies passiert bei höheren Dimensionen deutlich seltener.

Aus dem oben beschriebenen Prozess gehen mehrdimensionale Koordinaten $x$ in $\mathbb{R}^n$ hervor (siehe Abbildung 8). Um nun Koordinaten $x'$ in $\mathbb{R}$ zu erhalten, müssen wir eine Dimensionsreduktion anstreben:

- **PCA** Ein Beispiel stellte eine *principal components analysis* (PCA) dar. PCA ermöglicht es, die Dimensionen zu reduzieren und dabei die Variation der Koordinaten möglichst genau beizubehalten. Vereinfacht ausgedrückt werden dabei alle Koordinaten auf eine Gerade projiziert. Jeder Punkt bekommt einen Score, je nachdem wie weit er von der Geraden entfernt ist. Im Allgemeinen entstehen solche Komponenten für alle gewünschten Dimensionen, in unserem Fall nur für eine. Die Summe dieser Scores wird nun minimiert, um eine möglichst gute Projektion zu bekommen. Ein großer Vorteil ist die Schnelligkeit. Diese Methode benötigt $\mathcal{O}(n*|E|+|V|)$ [11].
- **MDS** Eine weitere Möglichkeit ist die metrische multidimensionale Skalierung (MDS). Diese, wie in einem früheren Abschnitt bereits beschrieben, benötigt eine Distanzmatrix als Eingabe. Dadurch entsteht ein hoher Speicheraufwand von $\mathcal{O}(|V|^2)$, welcher bei der Größe der Knotenmenge noch praktikabel ist. Die Distanzmatrix lässt sich leicht durch die Koordinaten $x$ im $\mathbb{R}^n$ berechnen. Anders als bei der PCA werden hier die Distanzen hinsichtlich des Stresses optimiert.

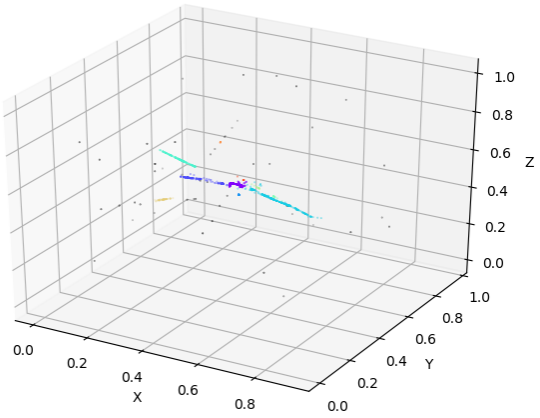


Abbildung 8: 3D-Fruchterman-Reingold-Algorithmus Ergebnis
Für die bessere Sichtbarkeit sind die dreidimensionalen Contig-Positionen farblich hervorgehoben. Die Contigs ordnen sich größtenteils in eine Reihe an. Es existieren jedoch auch einige „Ausreißer“.

Vergleicht man bezüglich der Summe aller Kräfte im System MDS und PCA mit dieser Instanz, so schneidet MDS meist besser ab. Dies ist aber mit Vorsicht zu betrachten, da der Fruchterman-Reingold-Algorithmus von einer zufälligen Anordnung als Start ausgeht. Dadurch ist der gesamte Prozess nicht deterministisch und nicht reproduzierbar. Zunächst wird ein dreidimensionaler Fruchterman-Reingold-Algorithmus auf die Instanz angewendet. Dazu wird das Intervall von 0 bis 1 betrachtet (das heißt $x \in [0,1]^3$), da die Koordinaten aus einem theoretischen Canvas stammen. Das Ergebnis wird dann in einen Algorithmus eingespeist, der die Distanzmatrix berechnet. Folgend wird diese wieder dem MDS-Algorithmus übergeben. Die daraus resultierenden Koordinaten orientieren sich um den Nullpunkt anstatt um $0,5$ (die Mitte des Intervalls $[0,1]$). Zudem sind die Koordinaten nicht länger an Intervalle gebunden. Sie können sowohl weiter als $0,5$ vom Nullpunkt entfernt liegen, als auch negative Werte annehmen.

Nun müssen die Koordinaten aus dem MDS-Ergebnis auf Genomkoordinaten abgebildet werden. Dazu wird angenommen, dass die meisten Knoten etwa zwischen $-0,4$ und $0,4$ liegen. Da davon ausgegangen werden kann, dass die MHC-Region etwa 5 Millionen Basenpaare lang ist, werden alle Koordinaten mit $(5 * 10^6) * (|0,4 - (-0,4)|)$ multipliziert. Dadurch wird die Ausbreitung auf $-0,4$ bis $0,4$ beschränkt. Alle weiter außen liegenden Knoten stellen sich als „Ausreißer “ dar, die wegen fehlender Kanten abgestoßen wurden. Dabei sind die Werte der Begrenzungen durch das Testen verschiedener Eingaben entstanden.

Anschließend werden alle Koordinaten soweit nach rechts verschoben, sprich in positive Bereiche, bis sich die kleinste Koordinate auf dem Nullpunkt befindet. Diese Streckung der Koordinaten stellt keine besonders gute Heuristik dar. Aufgrund aller anderen Nachteile der Fruchterman-Reingold-Methode wird diese Idee verworfen.

## 5.2 Contig-Anordnung mittels multidimensionaler Skalierung

MDS kann auch direkt angewendet werden, da dies auf einer Distanzmatrix basiert. Dadurch entsteht der Vorteil, dass nicht länger Positionen auf einem virtuellen Canvas verwendet werden müssen. Die Distanzen werden so direkt in die richtigen Koordinaten übersetzt.

Da die MHC-Contigs nur gewichtete Kanten zu nahen Nachbarn haben, ist der Scaffold-Graph kein vollständiger Graph. Somit lässt sich aus diesen Kanten nur eine dünnbesetzte Distanzmatrix aufbauen. In solchen Fällen wird bei MDS häufig der *all-pairs shortest path* Algorithmus von Floyd / Warshall [12][13] angewandt. Diese Methode zeichnet sich durch ihre Schnelligkeit aus. Allerdings ist sie aufgrund von Messfehlern und vor allem Repeats sehr fehleranfällig. Da ein Repeat an unterschiedlichen Stellen vorkommt, sich aber nur als ein Contig exprimiert, werden große Abschnitte zwischen Repeat-Vorkommen komplett ignoriert (Abbildungen 9 und 10 illustrieren dies). Deswegen ist es unabdingbar, dass alle Repeats vorher gelöst wurden.

Prinzipiell wird nach einem Algorithmus gesucht, der möglichst viele Knoten auf einem Pfad umfasst, demnach auch Kreise (Zyklen) innerhalb des Graphen. Dabei unterscheidet sich dieses Problem nur dadurch vom Längsten-Pfad-Problem, dass Knoten innerhalb des Pfades auch mehrmals besucht werden dürfen (Repeats). Leider ist dies NP-

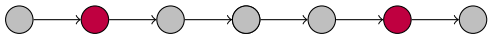

Abbildung 9: Gesuchter Pfad
Angenommen jede Kante hat die Distanz 1, dann sollte der echte Pfad 6 lang sein.

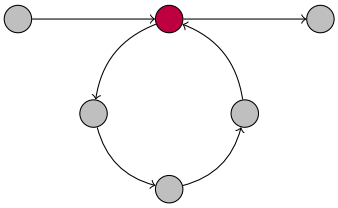

Abbildung 10: Distanz Repeat Problem
Dadurch, dass das Contig-Repeat (Rot) nicht aufgelöst ist, ist der kürzeste Pfad fälschlicherweise 2 lang.

vollständig und somit keine effiziente Lösung. Es gibt einen effizienten Algorithmus für azyklische gerichtete Graphen. Sollte es möglich sein, alle Repeats sowie alle übrigen Kreise zu entfernen, kann ein längster Pfad effizient gefunden werden. Daraufhin ist zu lösen, wie eine Distanzmatrix mittels dieses Pfades aufgebaut werden kann.

Ist es nicht möglich, einen Kreis aufzulösen, wird sich dieser Kreis in der Lösung über die eigentlichen Contigs „legen“. So verkürzt sich die Länge der Lösung um die Länge des Kreises. Stellt man sich die Lösung als Strang vor, teilt sich dieser Strang auf der Höhe desjenigen Contigs auf, an dem der Kreis Anschluss an den Rest des Graphen erhält. So gibt es dann zwei parallele Stränge, die keine Kanten zueinander haben, dennoch zueinander geordnet werden. Es entstehen Regionen in der Contig-Anordnung, in denen sich gehäuft Contigs befinden. Dies stellt einen weiteren Grund dar, weshalb Repeatfindungen mittels des Identifizierens von hohen Kräften in Knoten nicht funktioniert. Die beiden parallel laufenden Stränge haben untereinander keine gemeinsamen Kanten. Das bedeutet, auf sie wirken stark abstoßende Kräfte. Repeats und andere Knoten, die jedoch den Kreis zum restlichen Graphen verbinden, haben schlecht eingebettete Kanten mit hohen Kräften. Allerdings besitzen sie weniger abstoßende Kräfte als andere Knoten des Kreises.

Es lässt sich daraus schließen, dass multidimensionale Skalierung deutlich bessere Ergebnisse produziert, wenn es sich um einen azyklischen Graphen handelt. Wenn ein Graph Zyklen (Kreise) enthält, werden zwei Contig-Sequenzen auf dem gleichen Strang-Abschnitt abgebildet. Dies ließe sich mit einer besseren Distanzmatrix lösen, welche auch Zyklen berücksichtigen kann.

# 6 Evaluation

Die erstellte Lösung muss nun auf ihre Güte geprüft werden. Dazu gibt es eine Referenzlösung, welche per Hand erstellt wurde. Es wird von der Korrektheit der Referenzlösung ausgegangen. Dies lässt sich in einem Szenario wie diesem nicht vollständig feststellen, die Lösungen sind jedoch nun vergleichbar.

**Länge** Ein erster Ansatz ist die Länge des Scaffolds. Die Referenzlösung erreicht eine Länge von fünf Millionen Basenpaaren, während die Lösung mit MDS nur auf 1,7 Millionen Basenpaare kommt. Dies ist dem kürzesten Pfad für die Distanzmatrix geschuldet. Der kürzeste Pfad vom ersten bis zum letzten Knoten muss also 1,7 Millionen Basenpaare lang sein. Dies kann nur auf verkürzende Kanten und Knoten zurückzuführen sein, so zum Beispiel nicht gefundene Repeat-Contigs oder fehlerhafte Kanten. Die Methode mit dem Fruchterman-Reingold-Algorithmus lässt sich mit dieser Evaluationsmethode nicht vergleichen. Dort wurden die Canvas-Positionen auf Basenpaar-Koordinaten gestreckt, sodass sie insgesamt fünf Millionen Basenpaare lang sind.

**Visueller Vergleich** Darüber hinaus lässt sich die Lösung auch visuell mit dem Graphen vergleichen. In Abbildung 2 lässt sich der ungefähre Verlauf des Stranges nachvollziehen. Wäre eine visuelle Einbettung des Verlaufs der Lösung in diesen Graphen möglich, könnte die Korrektheit annähernd gezeigt werden. Dazu wird nun die Einbettung des Ausgangsgraphen aus Abbildung 2 betrachtet. Es wird ein Farbverlauf von weiß nach violett über die Koordinaten der Lösung sowie die der Referenzlösung erstellt. Jede Kante und jeder Knoten wird dann, je nach Koordinate, eingefärbt. Eine optimale Lösung würde nun einen eindeutigen Pfad aufweisen, der diesem Farbverlauf folgt. Dies trifft bei

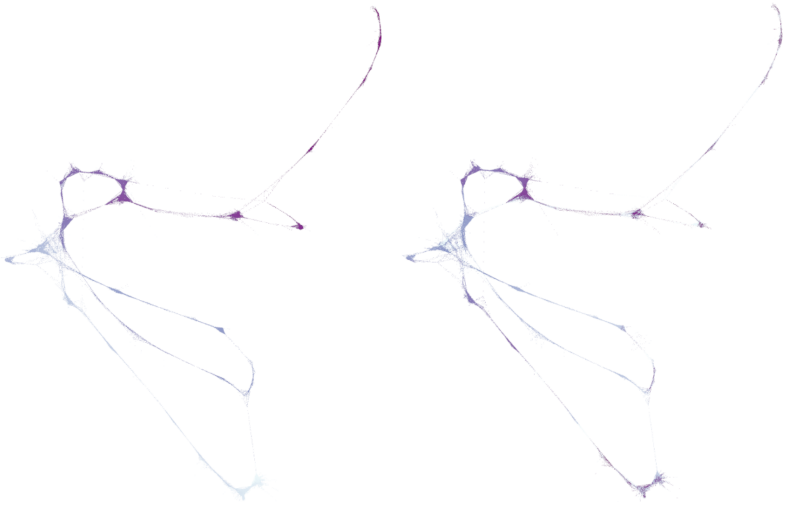

Abbildung 11: Evaluation mittels Farbgradient
Links der Farbverlauf über die Koordinaten für die Referenzlösung. Rechts das gleiche Vorgehen für die berechnete Lösung.

der Referenzlösung in Abbildung 11 zu. Man kann dem Farbverlauf von weiß nach violett eindeutig folgen. So beginnt der Pfad unten und endet oben. Anders als bei der mit MDS ermittelten Lösung: Sie beginnt mittig und besitzt zwei verschiedene Enden oben und ebenfalls mittig. Dies ist darauf zurückzuführen, dass es einen kürzesten Pfad zum eigentlichen Start (unten) gibt, der aus der Mitte entspringt. So konnte der Start nicht als solcher identifiziert werden. Positiv ist zu bemerken, dass lokal gesehen alle Knoten ähnlich gefärbt sind.

Folglich findet die Berechnung lokal eine zufriedenstellende Anordnung. In Bezug auf den gesamten Graphen wurden diese lokalen Contig-Anordnungen falsch zusammengesetzt.

**Konnektivität** Die Konnektivität der Lösung wird überprüft, um zu sehen, ob Contigs genau dann nahe beieinander liegen, wenn dies auch im Scaffold der Fall ist. Im Scaffold direkt aufeinanderfolgende Contigs sollten im Graphen benachbart sein. Wie man bei Abbildung 12 gut erkennen kann, befinden sich die Kanten aufgrund der Arbeitsweise der Nanopore- und Illuminatechnologie zwischen naheliegenden Contigs. Nun wird der Anteil der direkt nebeneinander liegenden Contigs (im Scaffold) betrachtet, die auch Kanten zueinander haben. Da die Kantendichte im Graph lediglich 3% beträgt, ist zu vernachlässigen, dass auch manche Kanten weit entfernte Contigs verknüpfen. Dies dürfte eine Ausnahme bleiben. Die Referenzlösung kommt so auf 90%, während die Lösung mit MDS nur 50% erreicht. Dies ist ebenfalls darauf zurückzuführen, dass sich mehrere korrekt zusammengefügte Sequenzen aufgrund der Zyklen fälschlicherweise übereinander legen. Wenn sich zwei unabhängige Contig-Sequenzen in der gleichen Region abbilden, ist zu erwarten, dass sich die Konnektivität halbiert. Grund dafür ist, dass immer eine 50-prozentige Chance besteht, das nächste Mal auf ein Contig einer anderen Sequenzregion zu treffen. Treffen zwei Contigs verschiedener Regionen aufeinander, besitzen diese keine Kanten. Wenn man nun nicht nur die direkten Koordinaten-Nachbarn anschaut, sondern auch Nachbarn, die maximal fünf Contigs entfernt angesiedelt sind, bessert sich dieses Ergebnis. Dies wird mit einer 80-prozentigen Abdeckung der Lösung untermauert. Die Referenzlösung bleibt stabil bei 90%.

**Stress-Score** Eine naheliegende Möglichkeit, diese Lösung zu bewerten, ist die Berechnung des Stresses von MDS:

$$\sum_{(u,v)\in E} (|c_u - c_v| - w_{(u,v)})^2$$

Problematisch wird dies nun bei gefundenen Repeat-Contigs $u$. Diese besitzen mehrere Vorkommen in der Sequenz und dementsprechend mehrere Koordinaten $c_u = c_{u_1}...c_{u_n}$. Nun lassen sich die Koordinaten nicht in einzelne Knoten auftrennen. Dies ist darin zu begründen, dass unbekannt ist, welche Kante zu welchem Knotenvorkommen gehört. Somit ist vom besten Fall (best case) auszugehen: In solch einem Fall wäre eine Aufteilung der Kanten in Knotengruppen vorteilhaft, bei welcher der Stress minimiert wird. So entsteht folgender Stress-Score:

$$\sum_{(u,v)\in E} \min_{c_{u_i}\in c_u, c_{v_j}\in c_v} (|c_{u_i} - c_{v_j}| - w_{(u,v)})^2$$

Mit diesem Ansatz kommt man bei der Referenzlösung auf einen Stress von etwa $0.5*10^9$ gegenüber $12 * 10^9$ bei der MDS-Lösung. Das bedeutet, dass die Lösung mittels MDS im globalen Fall 24 mal schlechter ist als die Referenzlösung. Dabei ist zu berücksichtigen, dass jedes zusätzliche Repeat belohnt wird. Ursächlich dafür ist die Tatsache, dass bei jeder Contig-Verdoppelung auch die Möglichkeiten steigen, den Stress zu senken. Zudem besitzt die Referenzlösung gegenüber der MDS-Lösung nur 1966 gegenüber 2127 Contigs. Mit jedem zusätzlichen Contig sollte der Stress steigen. Dennoch wäre zu erwarten, dass die Lösung deutlich besser ausfiele.

Nun können wir ermitteln, ob dieses Ergebnis an Phase eins oder Phase zwei scheitert. Dazu wird Phase 2 auf die Instanz der Referenzlösung angewendet und der Stress-Score überprüft. Zunächst muss der Ausgangsgraph so transformiert werden, dass die Repeat-Vorkommen ihre passenden Kanten erhalten. Es wird die gleiche Methode wie beim Stress-Score verwendet: Die Kante soll demjenigen Repeat-Vorkommen zugeordnet werden, welches den geringsten Stress aufweist. Durch diesen Graph lässt sich nun über die kürzesten Wege die Distanzmatrix aufbauen und anschließend MDS anwenden. Heraus kommt eine ein 1,2 Millionen Basenpaare lange Scaffold-Sequenz mit einem Stress-Score von $12 * 10^9$. Dies beweist, dass das Problem entweder in der Erstellung der Distanzmatrix begründet ist, oder gar in der MDS-Methode selbst. Das Problem kann also, davon ausgehend, dass die Repeats der Referenzlösung korrekt sind, nicht im Lösen der Repeats liegen. Es könnten weitere widersprüchliche Kanten und Knoten existieren, die die Erstellung der Distanzmatrix verzerren.

## 6.1 Implementation

Der beschriebene Prozess ist in Python[14] implementiert worden. Für generelle Algorithmen mit Graphen wurden networkx 1.11 [6] zusammen mit matplotlib 2.2.4[15] verwendet. Zunächst wurden die Daten, wie im Abschnitt Pre-Processing beschrieben, geformt. Anschließend wurden die Repeats mittels der zusammenhängenden Komponenten gelöst, wie im darauf folgenden Abschnitt beschrieben wurde. Abschließend wurde die Lösung durch einen im scipy-Paket[16] realisierten MDS-Algorithmus berechnet. Zudem gibt es eine weitere Phase, die die Lösung zuletzt auf ihre Güte prüft.

Der gesamte Prozess benötigt etwa 70 Minuten auf einem herkömmlichen Computer. Mit jeder Eingabe scheinen während des Prozesses leicht abgeänderte Ergebnisse ermittelt zu werden, sodass zum Beispiel in einem weiteren Durchlauf andere Repeats gefunden werden. Ein größeres Problem ist jedoch, dass MDS auf zufälliger Positionierung basiert. All dies resultiert in einem nicht reproduzierbaren Ergebnis. Dieses nichtdeterministische Verhalten macht das Evaluieren komplizierter. So sind kleine Änderungen an der Implementierung schwer in ihrer Auswirkung einzuschätzen.

Der Ansatz mittels des Fruchterman-Reingold-Algorithmus wurde sowohl als eindimensionale als auch als mehrdimensionale Variante implementiert. Da die Koordinaten der Lösung mittels MDS vielversprechender als die der Lösung mittels des Fruchterman-Reingold-Algorithmus sind, wurde dieser Lösungsweg nicht weiter verfolgt. Die eindimensionale Version des Fruchterman-Reingold-Algorithmus ist in networkX nicht implementiert worden, weswegen gleich zwei eigene Versionen eingebaut wurden. Die Erste löst das Problem iterativ, so wie es auch im Abschnitt „Grundlagen“ vorgestellt wurde.

Da diese Methode aber durch die Größe der Instanz ineffizient läuft, gibt es eine weitere Variante, welche mit einigen Matrix-Optimierungen deutlich schneller läuft.

## 6.2 Fazit

Aufgrund der gerade beschriebenen Schwierigkeiten ist diese Methode noch nicht ganz ausgereift. Prinzipiell sind kräftebasierte Verfahren anwendbar, jedoch sind die jetzigen Lösungen noch unbrauchbar. Kann eine Möglichkeit gefunden werden, eine Distanzmatrix aufzubauen ohne dabei Contigs zu überspringen, wäre dies eine valide Herangehensweise. Solange diese Hürden bestehen bleiben, kann ein kräftebasiertes Verfahren nur lokal zur Verfeinerung einer globalen Lösung verwendet werden. Wenn ein Pfad durch alle Knoten berechnet werden kann, können diese Knoten anschließend mit diesen Verfahren besser anordnet werden. Ein direktes Lösen des Scaffold-Problems, wie es das Ziel dieser Arbeit ist, ist jedoch nicht erfolgreich. Die Methoden für das Lösen von Repeats sind jedenfalls auch mit Ansätzen jenseits kräftebasierter Verfahren kombinierbar. Beispielsweise mit einem MinLA-Problem, welches mit linearer Programmierung gelöst werden kann.

# A Weitere Abbildungen

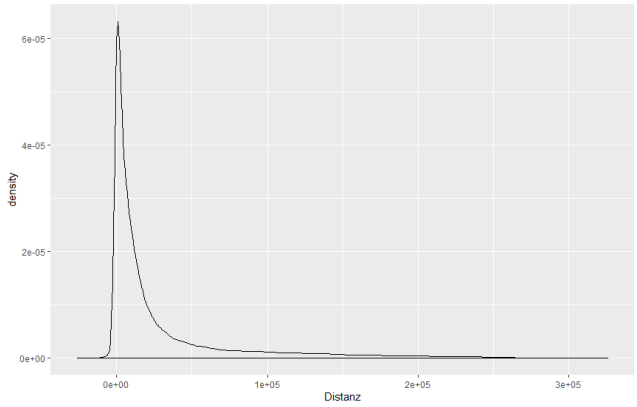


Abbildung 12: Dichte gegen Distanz
Die Distanz der Contigs in den Daten und deren Dichte.

# B Quellcode

Repository mit dem Quellcode:

`https://gitlab.cs.uni-duesseldorf.de/klau/bsc-thesis-contig_placement_force.git`

Zuletzt aufgerufen am 02.12.2019

## Literatur

[1] S. Piertney and M. Oliver, "The evolutionary ecology of the major histocompatibility complex," *Heredity*, vol. 96, no. 1, p. 7, 2006.

[2] V. Kiefel, "Hla und transplantation," *Online publiziert: www. tmed. med. uni-rostock. de/hla. pdf (Stand 18.10. 2013)*, 2017.

[3] Illumina, Inc. `https://www.illumina.com/`, letzter Zugriff am 02.12.2019.

[4] Oxford Nanopore Technologies. `https://www.nanoporetech.com`, letzter Zugriff am 02.12.2019.

[5] S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," *arXiv preprint arXiv:1201.3011*, 2012.

[6] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[7] J. Moody and D. R. White, "Structural cohesion and embeddedness: A hierarchical concept of social groups," *American sociological review*, pp. 103–127, 2003.

[8] Y. Koren and D. Harel, "A multi-scale algorithm for the linear arrangement problem," in *International Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 296–309, Springer, 2002.

[9] L. Carmel, D. Harel, and Y. Koren, "Drawing directed graphs using one-dimensional optimization," in *International Symposium on Graph Drawing*, pp. 193–206, Springer, 2002.

[10] Y. Koren and D. Harel, "One-dimensional graph drawing: Part i—drawing graphs by axis separation," *submitted. Available at: www. wisdom. weizmann. ac. il/yehuda/pubs/1d gd. pdf*, 2003.

[11] D. Harel and Y. Koren, "Graph drawing by high-dimensional embedding," in *International symposium on graph drawing*, pp. 207–219, Springer, 2002.

[12] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[13] S. Warshall, "A theorem on boolean matrices," in *Journal of the ACM*, Citeseer, 1962.

[14] Python Core Team, *Python: A dynamic, open source programming language*. Python Software Foundation, 2019. Python version 3.7.

[15] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

# Abbildungsverzeichnis