Genome Rearrangement

nach Pavel A. Pevzner

Konrad Rieck, Konrad Kretschmer,
Manuel Beetz, Marcus C. Gottwald

Herbst 2001

1 Biologischer Hintergrund

Seit der Entdeckung der Struktur der DNA durch Watson und Crick 1953 stellt diese einen zentralen Aspekt der biologischen Forschung dar. Die Analyse genetischer Informationen findet in nahezu allen Teilbereichen der Biologie Anklang, so unter anderem in der vergleichenden Anatomie, der Physiologie, der Taxonomie und natürlich der Genetik.

Hierbei unterscheidet man das *Gen* als Teilsequenz der DNA und das *Genom* als Gesamtheit aller Gene eines Individuums. Eine Sequenz der DNA kann in zwei Richtungen gelesen werden, es ist also grundlegend festzustellen, dass es sich bei Genen um *gerichtete* Teilsequenzen der DNA handelt.

Die Forschung beschränkte sich lange Zeit auf den Vergleich von Genen, das heisst auf den Vergleich der DNA-Sequenzen einzelner Gene. Erst in den späten 80er Jahren erkannte Jeffrey Palmer als einer der ersten Genetiker die Bedeutung der *Gen-Ordnung* innerhalb des Genoms. Besitzen zwei Individuen dieselben oder ähnliche Gene, so sind diese nicht zwingend an der gleichen Stelle innerhalb des Genoms wiederzufinden.

Wechselt ein Gen oder eine DNA-Sequenz die Position innerhalb des Genoms, so wird dies als *Translokation* bezeichnet, ändert sich die Ausrichtung, bezeichnet man diesen Vorgang als *Reversal*. Interchromosomale Ordnungsänderungen werden als *Fission* (Bruch) und *Fusion* (Verschmelzung) bezeichnet und treten bei multichromosomalen Organismen auf.

Sowohl Translokationen also auch Fissionen und Fusionen lassen sich mittels Reversals erzeugen [Pevzner, S.186], es ist also durchaus genügend, in diesem Rahmen die Analyse der Gen-Ordnung auf Reversals zu beschränken.

Genome, die die gleichen Gene enthalten, lassen sich mittels Reversals ineinander überführen. Hierbei wird pragmatischerweise angenommen, dass die kürzeste Sequenz von Reverals einer Überführung die natürlichste ist. Das Gebiet des

Genome Rearrangement beschäftigt sich mit genau diesen Ordnungsänderungen des Genoms und den nötigen Reversals, die ein Genom in ein anderes überführen. Kern des Genome Rearrangements ist die zentrale Frage nach dem genetischen Unterschied zweier Arten auf Ebene des Genomes.

Mittels dieser Technik sind völlig neue Vergleiche innerhalb der Biologie möglich geworden, so konnte unter anderem der Unterschied zwischen Mensch und Schimpanse verfeinert [Pevzner, S.186] werden und auch die evolutive Entwicklung innerhalb des X-Chromosoms von der Maus bis hin zum Menschen untersucht werden [Pevzner, S.185].

2 Modulierung des Genoms

Die Analyse von Reversals eines Genoms stellt eine nicht triviale mathematische Aufgabe dar. Neben mehreren Modellen zur Darstellung des Genoms ist es nötig, Begriffe zur Klassifizierung spezieller Strukturen innerhalb der Darstellung einzuführen.

Im folgenden ist ein Überblick über die eingeführten Darstellungen und Strukturen gegeben, die in diesem Papier besprochen werden:

- Darstellung als signierte Permutation
  Strukturen:
  - Sortieren durch Reversals
  - Breakpoints
- Darstellung als Breakpoint-Graph
  Strukturen:
  - Good und Bad Cycles.
  - Good und Bad Components
  - Non, Simple und Super Hurdles
  - Fortress-Bedingung

2.1 Darstellung als Permutation

Um die Gene innerhalb eines Genoms angemessen zu repräsentieren, verwendet man eine *signierte Permutation*. Bei einer Permutation handelt es sich um eine endliche Folge von Objekten, deren Reihenfolge festgelegt ist. Bei einer signierten Permutation wird zusätzlich die Ausrichtung der Objekte mittels eines Vorzeichens gekennzeichnet.

Die folgende Gen-Ordnung der Gene A, B, C, D kann also wie folgt als Permutation π dargestellt werden. Die Ausrichtung der Gene ist jeweils durch Pfeile gekennzeichnet, die Elemente der Permutation sind natürliche Zahlen:

$$\begin{array}{rcccc} \text{Genom 1:} & C \leftarrow, & D \rightarrow, & B \leftarrow, & A \rightarrow \\ \sigma = & (-3, & 4, & -2, & 1) \end{array}$$

$$\begin{array}{rcccc} \text{Genom 2:} & B \leftarrow, & C \rightarrow, & A \rightarrow, & D \leftarrow \\ \pi = & (-2, & 3, & 1, & -4) \end{array}$$

Ziel ist es, eine Aussage darüber treffen zu können, wie weit zwei gegebene signierte Permutationen (Genome 1 und 2) voneinander entfernt sind. Gemessen wird diese Entfernung in der notwendigen und minimalen Anzahl von Reversals am Genom, die nötig sind, um die gewünschte Reihenfolge herstellen zu können.

Zur weiteren Vereinfachung wird eines der beiden zu untersuchenden Genome auf die *Identitätspermutation* abgebildet und das andere Genom entsprechend in eine signierte Permutation transformiert, wie das folgende Beispiel demonstriert:

$$\begin{array}{rcccc} \text{Genom 1:} & C \leftarrow, & D \rightarrow, & B \leftarrow, & A \rightarrow \\ id = & (1, & 2, & 3, & 4) \end{array}$$

$$\begin{array}{rcccc} \text{Genom 2:} & B \leftarrow, & C \rightarrow, & A \rightarrow, & D \leftarrow \\ \pi = & (3, & -1, & 4, & -2) \end{array}$$

Man kann also das Problem auf die Transformierung einer signierten Permutation in die Identitätpermutation zurückführen. Hierbei bezeichnet man die minimale Anzahl der nötigen Reversals einer signierten Permutation π hin zu der Identitätspermutation als *reverse Distanz* $d(\pi)$. Tatsächlich führt man nun keine Transformation mehr durch, sondern *Sortieren durch Reversals* [Winter].

Bei dem Reversal $r(i, j)$ einer signierten Permutation π ändern sich die Reihenfolge und das Vorzeichen der gedrehten Elemente[Ananth, S.3] im spezifizierten Intervall (i, j). Das folgende Beispiel demonstriert ein Reversal vom zweiten bis zum vierten Element:

$$\begin{array}{rcccc} \text{Ursprungs-Genom:} & B \leftarrow, & C \rightarrow, & A \rightarrow, & D \leftarrow \\ \pi = & (3, & -1, & 4, & -2) \end{array}$$

$$\begin{array}{rcccc} \text{Genom nach Reversal(2,4):} & B \leftarrow, & D \rightarrow, & A \leftarrow, & C \leftarrow \\ \pi \cdot r(2,4) = & (3, & 2, & -4, & 1) \end{array}$$

In der Ausgangspermutation π kann man nun die aufeinanderfolgenden Elemente untersuchen. Hierbei unterscheidet man Elemente, die in der korrekten Reihenfolge vorliegen, und solche, die noch sortiert werden müssen. So sind zum Beispiele die Elemente $3, 4, 5$ oder $-4, -3, -2$ korrekt sortiert (im zweiten Fall würde ein einfaches Reversal die Sequenz in $2, 3, 4$ umwandeln) und die Elemente $3, -2, 4$ oder $-4, 2, -3$ nicht sortiert.

Man bezeichnet die Räume zwischen nicht sortierten Elementen als *Breakpoints*. Formal sind Breakpoints definiert als eine Position innerhalb der Permutation π zwischen zwei aufeinanderfolgenden Objekten i und j, für die nicht gilt: $\mid i - j \mid = 1$ [Pevzner, S.179].

Die Permutation, die keine Breakpoints enthält, ist die Identitätspermutation oder die komplett gedrehte Identitätspermutation. Eine signierte Permutation mit Reversals zu sortieren ist also äquivalent zur Entfernung aller Breakpoints [Pevzner, S.180].

2.2 Darstellung als Breakpoint-Graph

Neben der Darstellung als signierte Permutation verwendet man auch die Darstellung als Graph, die es ermöglicht, weitere Strukturen zu definieren, die die Sortierung erleichtern.

Jedes Gen des Genoms ist wie oben bemerkt *gerichtet*. Pevzner modelliert diese Eigenschaft, indem er für jedes Gen zwei Knoten in dem Graphen erzeugt, jeweils den Knoten Ia für den Kopf und den Knoten Ib für das Ende des Gens I [Pevzner, S.192], mit $I \in \mathbb{N}$. Der Übersicht halber werden zwei Hilfsknoten eingefügt, die Anfang und Ende markieren.

Die Gene werden in der vorliegenden Reihenfolge im Graphen verbunden, aber in der letztendlich gewünschten Reihenfolge nummeriert, d.h. sie werden entsprechend ihrer signierten Permutation nummeriert. Nach diesem ersten Schritt erhält man einen Graph in Form einer Kette.

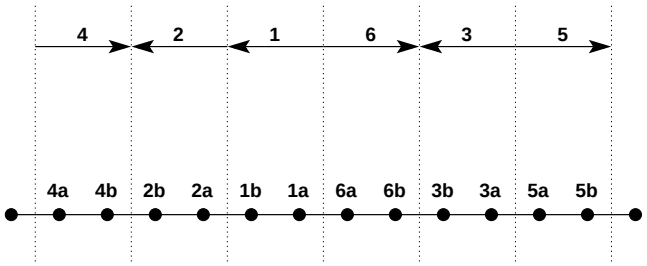

Abbildung 1: Kette, jedes Gen ist repräsentiert durch zwei Knoten

Als *Breakpoints* bezeichnet man in einem Graphen all die Stellen, an denen zwei Knoten aufeinander folgen, die dies in der gewünschten Reihenfolge nicht tun, d.h. auch in der signierten Permutation des Genoms durch einen Breakpoint getrennt sind, wie in der Abbildung 2 ersichtlich.

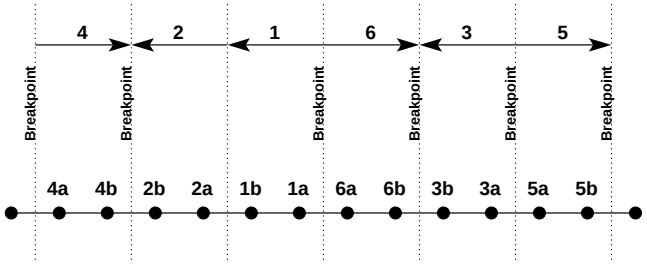

Abbildung 2: Breakpoints

In einem weiteren Schritt werden die Kanten eingezeichnet, die existierten, wenn die Gene in der gewünschten Reihenfolge wären und man sie der Reihe nach verbinden würde.

Pevzner nennt die Kanten der Kette *schwarz*, die anderen *grau*. Eine andere Darstellung, von Anantharaman, stellt die Kette der schwarzen Kanten als

Kreis dar [Ananth, S.5], graue Kanten sind dort Sehnen im Kreis. Er nennt den Graph „Reality-Desire-Diagram“, da es sich bei den schwarzen Kanten um reale Verbindungen innerhalb des Genoms handelt und die grauen Kanten den zu Erreichenden, dem Verlangen, entsprechen.

Kanten zwischen benachbarten Knoten Ia und Ib (also jede zweite schwarze Kante) können entfernt werden. Die Knoten sind untrennbar, da sie Kopf und Ende *eines* Gens darstellen. Übrig bleiben Kreise, die aus alternierend grauen und schwarzen Kanten bestehen. Dieser Graph heißt *Breakpoint Graph.*

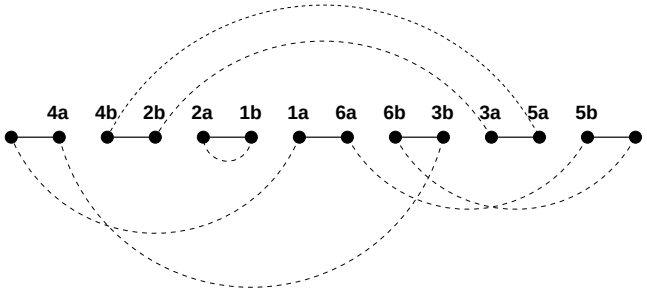

Abbildung 3: Breakpoint Graph

Ähnlich wie die Breakpoints in der signierten Permutation stehen auch die Kreise in einem Breakpoint Graph in direkter Verbindung zum Sortieren durch Reversals. Beim Entfernen eines Breakpoints entsteht immer ein neuer Kreis [Pevzner, S.180]. Die Anzahl der Kreise in einer Permutation π wird als $c(\pi)$ definiert. Daraus folgt, dass die Identitätspermutation genau $c(id) = n + 1$ Kreise in ihrer Repräsentation als Graph enthält, wobei n die Anzahl der zu untersuchenden Gene ist.

2.2.1 Arten von Kreisen

Als *Good Cycle* wird ein Kreis bezeichnet, in dem es möglich ist, durch ein geeignetes Reversal die Anzahl der Breakpoints um 1 zu verringern. Dies ist dann der Fall, wenn sich innerhalb des Kreises zwei schwarze Kanten finden lassen, die in unterschiedlicher Richtung durchlaufen werden. Im Graph nach Pevzner erkennt man diesen Zustand daran, daß sich eine ungerade Anzahl Überschneidungen grauer Kanten findet. Im Kreis nach Anantharaman erkennt man einen Good Cycle daran, daß sich Sehnen des Cycles überschneiden.

Kreise, in denen die Anzahl der Breakpoints nicht durch ein Reversal vermindert werden kann, werden entsprechend als *Bad Cycle* bezeichnet.

Eine Ausnahme bilden Kreise, die genau eine graue (und somit genau eine schwarze) Kante besitzen. Sie lassen sich nicht auflösen und repräsentieren bereits korrekt sortierte Elemente der Identitätspermutation.

Die Abbildung 4 zeigt, dass nur ein Reversal (b) zwischen zwei gegenläufigen Kanten die Anzahl der Kreise $c(\pi)$ erhöht und die Anzahl der Breakpoints damit erniedrigt; jedes andere Reversal (a) erhöht die Anzahl der Kreise oder lässt sie unverändert.

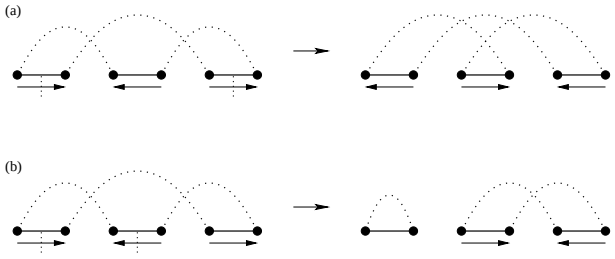

Abbildung 4: Auswirkung eines Reversals auf Anzahl der Cycles

2.2.2 Komponenten

Sich überschneidende Kreise im Breakpoint-Graph werden zu *Komponenten* zusammengefaßt. Eine Komponente, die mindestens einen Good Cycle enthält, wird als *Good Component* bezeichnet, alle anderen Komponenten heißen entsprechend *Bad Component* und lassen sich nicht direkt auflösen. Bad Components stellen ein Problem dar, da sie keinen Good Cycle enthalten und so durch kein Reversal ein Breakpoint eliminiert werden kann.

Eine Komponente *trennt* zwei andere Komponenten genau dann, wenn sich in der trennenden Komponente (mindestens) eine graue Kante findet, die in der Darstellung nach Pevzner eine der getrennten Komponenten überspannt, die andere nicht. (Die Komponenten werden entweder vollständig überspannt oder gar nicht, da eine Überschneidung die beiden Komponenten verschmelzen würde.) Eine andere Definition besagt, daß jede Kante, die von einer der getrennten Komponenten zu der anderen führen würde, die trennende Komponente schneiden müßte.

Eine Bad Component, die nicht zwei andere Bad Components trennt, heißt *Hurdle*. Eine Bad Component, die zwei Bad Components voneinander trennt, heißt *Non-Hurdle*.

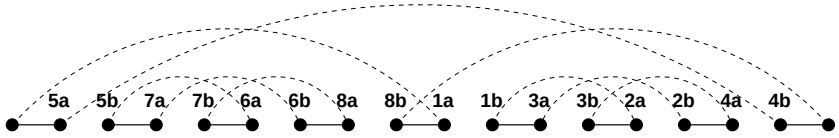

Abbildung 5: Die mittlere Komponente ist eine Non-Hurdle: Die Kante vom Anfangsknoten zum Knoten 1a überspannt die linke Komponente, nicht aber die rechte.

Eine Hurdle, deren Entfernung die Umwandlung einer Non-Hurdle in eine Hurdle bewirken würde, wird *Super Hurdle* genannt.

Alle anderen Hurdles werden als *Simple Hurdle* bezeichnet.

Wenn es eine ungerade Anzahl von Super Hurdles und keine Simple Hurdles gibt, dann hat man eine sogenannte *Fortress Condition*.

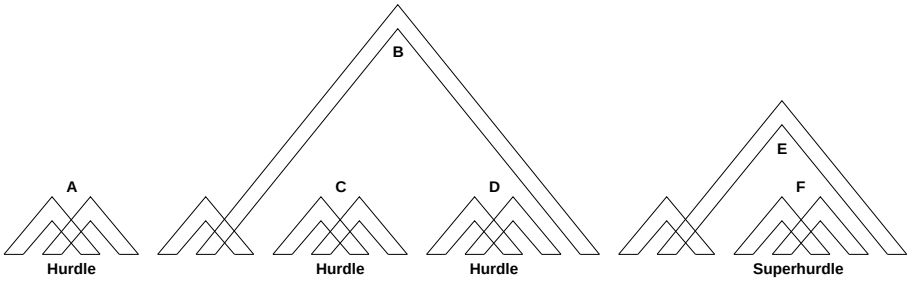

Abbildung 6: B und E sind Non-Hurdles. Es gibt keine Good Cycles.

3 Resultate

3.1 Reverse Distanz

Um eine gegebene signierte Permutation π in die Identitätspermutation umzuformen, ist es notwendig, alle Breakpoints zu eliminieren. Die Permutation wird also sortiert.

Breakpoints werden jedoch nur durch Reversals von Good Cycles (siehe oben) eliminiert. Es müssen also zuerst Bad Cycles in Good Cycles aufgelöst werden. Ingesamt müssen daher $n + 1 - c(\pi)$ Kreise durch einfache Reversals erzeugt werden, um die $n + 1$ Kreise der Identitätspermutation zu erhalten.

Wie durch den Breakpoint-Graphen deutlich geworden, müssen vorher die Bad Components aufgelöst werden. Hierzu müssen alle Hurdles mittels $h(\pi)$ Reversals vereinigt werden.

Im ungünstigen Fall ist die Fortress-Bedingung für die Permutation π erfüllt, d.h. aus einer der Super Hurdles entsteht erst eine Simple Hurdle. Dieser Fall muss speziell berücksichtigt werden. Definiert wird daher $f(\pi) = 1$, falls eine Fortress-Bedingung erfüllt ist, $f(\pi) = 0$ andernfalls.

Es ergibt sich nun folgende reverse Distanz einer Permutationen π:

$$d(\pi) = n + 1 - c(\pi) + h(\pi) + f(\pi)$$

wobei n die Anzahl der Knoten, $c(\pi)$ die Anzahl der alternierenden Kreise in π, $h(\pi)$ die Anzahl der Hurdles in π und $f(\pi)$ die Fortess Condition, also 0 oder 1, sind.

3.2 Algorithmisches Resultat

Die obige Herleitung der reversen Distanz zeigt bereits deutlich algorithmische Strukturen. Ein Algorithmus zur Transformation eines Genoms, modelliert durch eine signierte Permutation, in ein anderes Genom, modelliert durch die Identitätspermutation, lässt sich also wie folgt vereinfacht formulieren:

- Eliminiere die Fortress-Bedingung, d.h. füge die Super Hurdles so zusammen, so dass nur noch Simple Hurdles verbleiben.

- Eliminiere die Hurdles, d.h. füge die Hurdles so zusammen, so dass sich die Bad Components zu Good Components auflösen.
- Eliminiere alle Bad Cycles, d.h. zerlege diese so, so dass aus ihnen Good Cycles entstehen.
- Eliminiere alle Breakpoints, d.h. zerlege alle Good Cycles, so dass nur noch einfache Kreise mit einer grauen und einer schwarzen Kante verbleiben.

Alle Operationen des obigen Algorithmus' beziehen sich auf das Ausführen von geeigneten Reversals. Pevzner bestimmt für eine etwas komplexere Version dieses Algorithmus' eine Laufzeit von $O(n^4)$ [Pevzner, S.213].

Glenn Tesler und Yang Yu erstellten unter Anleitung Pevzners eine Web-Applikation namens *GRIMM* [GRIMM] die Genome-Rearrangement-Algorithmen veranschaulicht. Die Anwendung demonstriert die in diesem Skript vorgstellten Techniken, reicht jedoch im Detail weitaus tiefer in das Gebiet des Genome-Rearrangement von multichromosomalen Genomen hinein.

Literatur

[Pevzner] Pavel A. Pevzner. *Computational Molecular Biology.* MIT Press, 2000. 1, 2, 3, 4, 5, 8

[Ananth] Thomas Anantharaman. *Genome Rearrangements, Skript.* http://www.cs.wisc.edu/ tsa/cs638/lec13.pdf, October 2000. 3, 5

[GRIMM] Glenn Tesler und Yang Yu aus Pavel Pevzner's Kurs CSE 291 *GRIMM – Genome Rearrangements In Man and Mouse.* http://www-cse.ucsd.edu/groups/bioinformatics/GRIMM, Frühling 2001. 8

[Winter] Pawel Winter. *Computational Biology – Sorting by Reversals, Skript.* http://www.diku.dk/ pawel/comp-bio/sort-rev.ps, Herbst 2001. 3

Biological Supervision:	Konrad R.
Graphics Director:	M. Beetz
Style & Didactics:	M. Gottwald
Computational Advisor:	Konrad K.