

Ausarbeitung zum Seminar

„Aktuelle Themen der BioInformatik“, SS04

„MCMC genome rearrangement“
István Miklós (2003)

André Brück
(2579542)

Inhalt

1. Einführung
2. Grundlagen
 - a. Permutationen
 - b. „Reality – Desire – Diagram“
 - c. „breakpoint graph“
 - d. Hürden und Festungen, Hannenhalli & Pevzner
 - e. Algorithmus I mit Laufzeitverhalten
3. MCMC genome rearrangement, István Miklós
 - a. Methoden
 - b. MCMC Strategie
 - c. Ergebnisse / DERANGE II
4. Ausblick / Diskussion

A Quellenverzeichnis

1. Einführung

In dem vorliegenden Essay führt István Miklós eine neue Methode ein, die es ermöglicht, evolutionäre Abstände anhand der Anzahl der benötigten Mutationen abzuschätzen. Er verweist auf einige bereits publizierte Arbeiten, die sich allerdings in einem wichtigen Punkt von seiner Methode unterscheiden:

Alle bisherigen Methoden zur Abschätzung evolutionärer Abstände beruhen entweder nur auf einem Mutationstypus oder gewichten alle vorkommenden Mutationstypen gleich (Inversionen: Bader et al, 2001 ; Bergeron, 2001; Transpositionen: Hannenhalli, 1996). Es gibt nur eine Ausnahme (DERANGE II, Blanchette et al, 1996), auf die im Verlauf dieser Ausarbeitung noch eingegangen wird.

Miklós beschreibt nun eine stochastische Methode, in der den zwei wichtigsten Mutationsarten (Inversion und Transposition) unterschiedliche Gewichte zugewiesen werden können. Er nutzt die Möglichkeiten einer Markoff-Ketten-Monte-Carlo-Methode (MCMC), die die Gewichte (oder hier: Raten) der Mutationen adaptieren kann (schätzen kann). Im Wesentlichen wird dies durch sinnvolle Wahl der Wahrscheinlichkeiten für die verschiedenen Mutationstypen erreicht, wobei Miklós die Erkenntnisse von Hannenhalli & Pevzner nutzt, die in früheren Arbeiten wichtige Voraussetzungen geschaffen haben (siehe Grundlagen: Hürden und Festungen).

2. Grundlagen

a. Permutationen

Definition: Jede mögliche Anordnung von Elementen, wobei alle Elemente der Ursprungsmenge verwendet werden, heißt Permutation dieser Elemente.

$A = (1, 2, 3, 4, 5)$, so ist $B = (2, 4, 3, 1, 5)$ eine Permutation von A.

Im vorliegenden Fall betrachten wir vorzeichen-behaftete Permutationen („signed permutations“), da wir jede Zahl als *eine - bei beiden zu vergleichenden Individuen - gleiche Abfolge von Nukleotiden* behandeln, und die Orientierung der Ursprungssequenz zu der Zielsequenz eine entscheidende Rolle bezüglich der Anzahl der benötigten Mutationen spielt.

Erläuterung:

Bei Individuum A wird folgende Sequenz gefunden: $S_A = \text{,ATTGCGA'}$. Bei Individuum B wird die gleiche Sequenz gefunden, allerdings mit der umgekehrten Orientierung, folglich $S_B = \text{,AGCGTTA'}$. Der Sequenz S_A wird nun die Zahl 1 zugeordnet, der Sequenz S_B -1. Um aus der Sequenz S_B nun die Sequenz S_A herzuleiten, kann man eine Inversion dieses Genabschnitts annehmen, womit sich die Anzahl der benötigten Mutationen um 1 erhöht.

b. „Reality – Desire – Diagramm“

Die Erstellung des Reality-Desire-Diagramms (im folgenden: RD) und die daraus ableitbaren Erkenntnisse wurden erstmals 1996 von Hannenhalli und Pevzner in dieser Form veröffentlicht.

Die Grundidee sagt aus, dass bestimmte Muster von sich schneidenden Graphen auf bestimmte Problemstellungen zurückzuführen sind. Das darauf beruhende „Hürden und Festungen“ - Modell stelle ich in Abschnitt 2d vor.

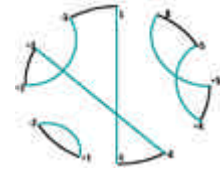


Abb 1

An dieser Stelle möchte ich kurz darauf eingehen, wie ein RD erstellt wird:

Eine vorzeichen-behaftete Permutation der Art $P = \{+3, -2, -1, +4, -5\}$ kann als Graph mit entsprechend orientierten Pfeilen dargestellt werden (siehe Abb. 2)

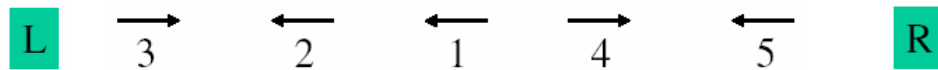


Abb 2

Damit die Orientierung als solche beibehalten wird, werden links und rechts weitere Platzhalter oder Anker eingefügt, üblicherweise L und R. Entsprechend der allgemein akzeptierten Konvention, werden nun die Elemente auf die in Abbildung 3 erkennbare Art zusammengeführt.

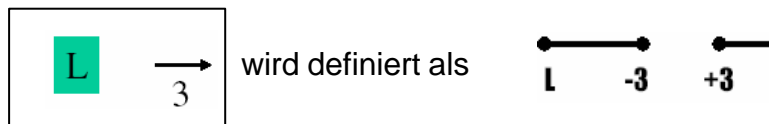


Abb 3

Dies führt zu einem Graphen, der jedes Element mit seinem Nachbarn verknüpft:



Abb 4

Dies entspricht dem in der Realität gefunden Graphen, und wird deshalb als „reality-graph“ bezeichnet. Der Vorteil, den man aus dieser Schreibweise gewinnt, wird deutlich, wenn man in den selben Graphen die „desire-edges“ einzeichnet.

Üblicherweise wird das „desire diagram“ als sortierte Permutation betrachtet, woraus dann der folgende Graph konstruiert wird:

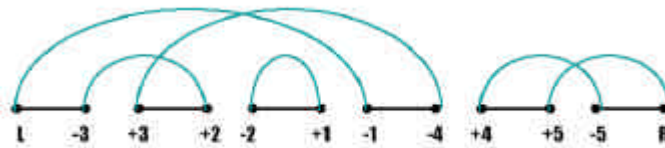


Abb 5

Diese sequentielle Schreibweise dient eigentlich nur der Herleitung, üblich und dem Modell von Hannenhalli und Pevzner entsprechend, wird dieses Diagramm als Kreis gezeichnet.

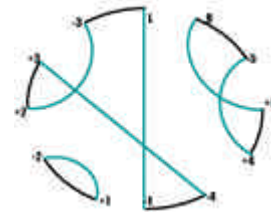


Abb 6

c. „breakpoint-graph“

Miklós verwendet in seinem Essay eine Aussage, die ich nun mittels des „breakpoint Graphen“ veranschaulichen möchte, namentlich, dass durch ein reversal (eine Inversion) maximal zwei Kreise entstehen können, bzw. maximal zwei breakpoints gelöscht werden können. Dies dient der groben Abschätzung, wie viele Inversionen mindestens benötigt werden, um einen Graphen zu sortieren.

Die Definition eines breakpoints:

Wenn (x,y) in einem Graphen α vorliegt, in einem Graphen β aber weder (x,y) noch $(-y,-x)$ vorkommen, so bezeichnet man (x,y) als „breakpoint“ von α in Bezug auf β .

Als Beispiel nehmen wir folgenden Graphen an:

Die reality-edges des Graphen werden immer dann mit einem Bogen verbunden, wenn es keinen direkten

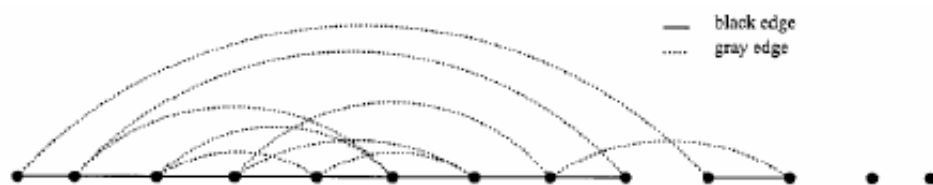


Abb 7

Weg von x nach y gibt. In dem vorliegenden Graphen wird $(3,5)$ als breakpoint betrachtet, $(10,11)$ nicht.

Im folgenden Schema wird die Aussage von Hannenhalli und Pevzner verdeutlicht:

$$(0, \underset{|}{2}, \underset{|}{3}, \underset{|}{1}, \underset{|}{4}, \underset{|}{6}, \underset{|}{5}, \underset{|}{7}, \underset{|}{8}) \text{ --- } (0, 1, 2, 3, 4, 5, 6, 7, 8)$$

Ein reversal um (6,5) löscht genau zwei breakpoints. Wie aus der Grafik zu ersehen ist, kann es keine Möglichkeit geben, mehr als zwei breakpoints zu löschen, nur zwei [(6,5)], einen [(4,6)] oder keinen [(1,4)], oder im schlimmsten Fall sogar welche hinzufügen [(5,7)] (auch nicht mehr als zwei, aus identischen Gründen). Da ein reversal nur auf zwei Kanten arbeitet (vergl. Abb. 7), können nur die benachbarten breakpoints gelöscht werden, interne breakpoints werden beibehalten.

$$(0, \underset{|}{2}, \underset{|}{3}, \underset{|}{1}, \underset{|}{4}, \underset{|}{5}, \underset{|}{6}, \underset{|}{7}, \underset{|}{8}) \text{ --- } (0, 1, 2, 3, 4, 5, 6, 7, 8)$$

Somit ergibt sich als grobe untere Schranke : $d_b(\mathbf{a}) \geq \frac{b_b(\mathbf{a})}{2}$

Die Distanz von a zu β , ist größer gleich der Anzahl der breakpoints von a in Bezug auf β geteilt durch 2, oder $d_\beta(\mathbf{a})$ ist die minimale Anzahl an Mutationen, so dass aus β a wird.

d. Hürden und Festungen, Hannenhalli & Pevzner (H&P)

Das Hürden und Festungen – Modell von H&P führt, wie eingangs erwähnt, bestimmte Konstellationen (Strukturen) der überschneidenden reality-edges des Reality-Desire-Diagramms auf bestimmte Problemstellungen zurück. Nach diesem Modell werden Komponenten des RDs nach gut und schlecht klassifiziert (Hürden und nicht Hürden). Nicht-Hürden erzeugen bei einem reversal nur Kreise, wohingegen man für das Auflösen einer Hürde mindestens zwei reversals benötigt. Hürden sind weiterhin unterteilt in „Simple“- und „Super“-Hürden. Die Festung besteht einzig aus einer ungeraden Anzahl an „Super“-Hürden [vergl. Abb. 8].

Um die Klassifizierung besser nachvollziehen zu können, gehe ich zuerst auf die Erstellung des Interleaving-Diagramms ein, aus dem man die Komponenten sofort erkennen kann.

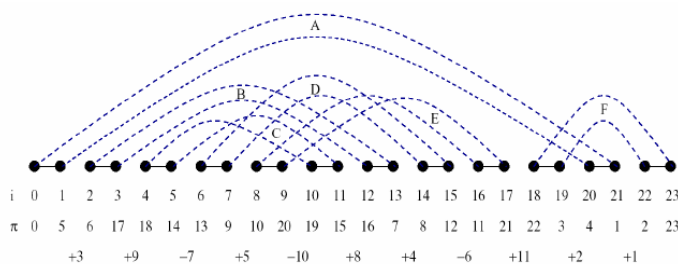


Abb 9

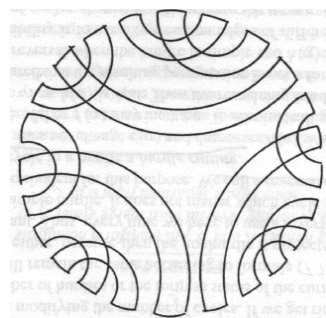


Abb 8

In Abbildung 9 wird die Permutation $P = \{+3, +9, -7, +5, -10, +8, +4, -6, +11, +2, +1\}$ als lineares RD-Derivat dargestellt. Dieses RD wird entgegen den bisher bekannten RDs durch eine Verdopplung der Sequenzlänge ermöglicht, wobei bei positiven Vorzeichen, das Doppelte der Zahl oben rechts notiert wird, bei negativem Vorzeichen entsprechend oben links. Jeweils (n-1) füllt die Nachbarposition auf. Dies führt erst zu der Berechenbarkeit, da das Sortieren von vorzeichen-behafteten Permutationen NP-hart ist. Die Orientierung des entstehenden Graphen p ist weiterhin

durch die spezielle Konstruktion erhalten. Die Bögen werden nun entsprechend des neuen Graphen p gezogen. Man fängt bei 0 an und verbindet die 1. Nun folgt der wichtige Schritt, in dem der durch die Verdopplung entstandene Partner mit dem Partner der gerade verbundenen Index ebenfalls konnektiert wird. Da, wie unter 2b erklärt, ein Anker eingeführt wurde (0 und 23) entsteht ein Diagramm, wo die reality-Verbindung um 1 verschoben ist. Die nun entstandenen Strukturen haben Hannenhalli & Pevzner auf die o.a. Problemstellungen zurückgeführt. Der Vollständigkeit halber noch eine Abbildung des aus Abb. 9 entstandenen Interleaving-Graphen (Abb.10).

Grund für die Erstellung des Interleaving-Graphen ist u.A., dass man mit einem reversal nur Abhängigkeiten innerhalb einer Komponente (=zusammenhängende Graphen) ändern kann, und damit das Wissen über die Zusammenhänge eminent wichtig für die Erstellung eines Algorithmus ist.

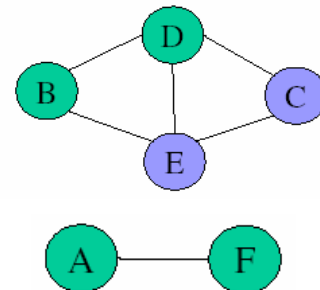


Abb 10

Im folgenden gehe ich auf die Klassifizierung nach Hannenhalli & Pevzner anhand eines anderen Beispiel-Graphen genauer ein:

In Abbildung 11 sind die Komponenten gut zu erkennen: $\{F\}$, $\{A,E\}$ und $\{B,C,D\}$.

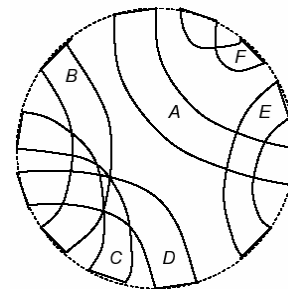


Abb 11

H&P erkannten, dass divergente (in sich kreuzende) Graphen immer mit einem einzigen reversal aufzulösen sind (F). Entsprechend klassifizierten sie diese als gut (Nicht-Hürde). Dreht man die durch F markierten inneren Kanten, so ergeben sich zwei Kreise, d.h. reality-edge und desire-edge sind identisch.

Da in der Komponente $\{B,C,D\}$ ebenfalls ein divergenter Graph (C) vorhanden ist, wird die ganze Komponente als gut klassifiziert.

Da in der Komponente $\{A,E\}$ keine divergenten Kanten vorliegen, wird diese als schlecht eingeordnet.

Gute Komponenten beinhalten immer eine Nicht-Hürde, wodurch ein „safe-reversal“ entsteht, in dem immer nur gute Kreise entstehen. Bei einer schlechten Komponente können nun zwei Fälle auftreten: Entweder es entsteht durch einen reversal eine gute Komponente (also divergente Kanten), oder es entsteht eine neue schlechte Komponente. Diese zwei Fälle haben H&P in „Simple“-Hürden und „Super“-Hürden unterschieden. Die exakten Definitionen von H&P lautet wie folgt:

„Simple“-Hürde: Wenn durch diese Komponente nicht zwei schlechte Komponenten getrennt werden.

„Super“-Hürde: Wenn bei einem reversal aus einer guten Komponente eine Hürde wird.

Nicht-Hürde: Wenn bei einem reversal nur gute Kreise entstehen.

Festung: Ein Graph aus einer ungeraden Anzahl von „Super“-Hürden.

In der folgenden Abbildung werden die Definitionen veranschaulicht:

Komponenten A, C, D und F sind Hürden. A, C und D sind „Simple“-Hürden, F ist eine „Super“-Hürde, B und E sind Nicht-Hürden. Man kann deutlich erkennen, dass bei jeder der Hürden mindestens zwei reversals nötig sind, um sie aufzulösen (eine um divergente Kanten zu erzeugen, eine um diese aufzulösen).

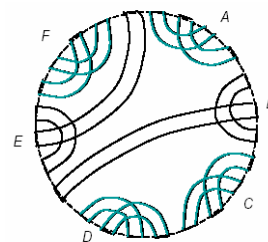


Abb 12

Um sich diesen Sachverhalt zu verdeutlichen, kann man formal eine der Hürden herausnehmen, und sich die Veränderung in dem verbleibenden Graphen anschauen. Nimmt man A heraus, entsteht keine neue Hürde. Nimmt man F heraus, so hat E die gleiche Form wie die übrigen Hürden; es wird also aus einer Nicht-Hürde eine Hürde (= „Super“-Hürde). Nimmt man C oder D heraus, so wird aus der entsprechend verbliebenen Hürde zwar eine „Super“-Hürde, es entsteht aber keine neue.

Entsprechend dieser Definitionen kann man nun drei reversal-Typen unterscheiden, die alle möglichen Fälle aufgreifen:

- Typ I: „safe“-reversal: Reversal über divergenten Kanten.
- Typ II: „hurdle-merging“: Ein reversal über zwei gegenüberliegenden Hürden, falls die Anzahl der Hürden gerade.
- Typ III: „hurdle-cutting“: Ein reversal über zwei gegenüberliegenden Hürden, falls die Anzahl der Hürden ungerade.

Der Unterschied von Typ II zu Typ III liegt in den gewählten Kanten für die reversals. Bei Typ II werden die inneren (von der jeweiligen Hürde eingeschlossenen) Kanten ausgetauscht, bei Kind III die jeweils äußeren.

e. Algorithmus I mit Laufzeitverhalten

Die nun festgelegten reversal-Typen werden von dem nun vorzustellenden Algorithmus genutzt:

repeat

if es gibt eine gute Komponente in $RDB(a)$

 führe Kind I reversal durch;

else if gerade Anzahl Hürden

 führe Kind II reversal durch;

else if ungerade Anzahl Hürden und min 1 Simple H.

 führe Kind III reversal durch;

else [fortress]

 Kind II reversal mit beliebigen Hürden;

until $a = \beta$.

Die Analyse des Algorithmus überlasse ich dem geneigten Leser, ich möchte nur noch auf einen Sachverhalt hinweisen: Der letzte Schritt (Fortress = Festung) kann nur ein einziges Mal erreicht werden, da ansonsten die Anzahl der Hürden gerade ist, oder immer mindestens eine „Simple“-Hürde entsteht.

Die Laufzeit wird von den Programmierern wie folgend angegeben:

Erstellen von RD	$O(n)$
Interleaving graph	$O(n^2)$
Komponenten	$O(n)$

$RD_{\beta}(a)$ (dynamisch)	$O(n^2)$
-----------------------------	----------

Aufgrund der dynamischen Programmierung benötigt dieser Teilschritt eine Gesamt-Laufzeit von $O(n^2)$.

Für einen reversal (2 edges)	$O(n^2)$
Prüfen ob schlechte Komp.(RD)	$O(n^2)$
Maximal n-mal	$O(n)$

Gesamtlaufzeit **$O(n^5)$**

Da bei jedem reversal jede Kante mit jeder überprüft wird (im schlimmsten Fall) ergibt sich $O(n^2)$; die Überprüfung auf schlechte Komponenten besteht aus den oberen drei Schritten: $O(n^2)$. Da im worst-case (wenn noch kein guter Kreis vorlag) das ganze n-mal durchgeführt werden muss, ergibt sich eine Gesamtlaufzeit von $O(n^5)$.

3. MCMC genome rearrangement

Wie in der Einführung erwähnt, kann der in dem Paper von Miklós vorgestellte Algorithmus zwei verschiedene Mutationen unterscheiden (Inversionen = reversals und Transpositionen). In Abbildung 13 werden diese näher erläutert.

Inversionen sind eine Umorientierung einer Sequenz, wobei es dem Algorithmus egal ist, ob hiermit einzelne Nukleotide oder durch Zahlen ersetzte „Cluster“ sind.

Transpositionen lokalisieren ganze Sequenzen an einer anderen Stelle und invertierte Transpositionen lokalisieren eine umorientierte Sequenz an einer anderen Stelle im Graphen.

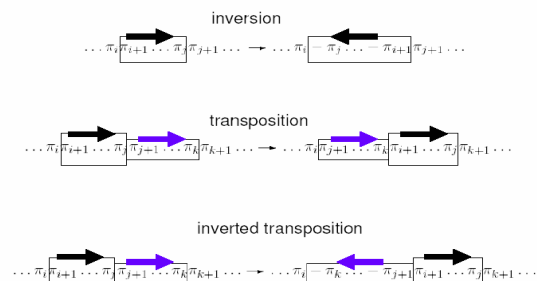


Abb 13

a. Methoden

Miklós nimmt eine stochastische Verteilung der Mutationen über die Zeit an. Aus diesem Grund weist er jedem Mutationstypus eine Wahrscheinlichkeit zu, die er mittels der Poisson-Verteilung auf einen Zeitraum t verteilt. Alle Mutationen werden voneinander als unabhängig betrachtet.

Mit den von Miklós festgelegten Wahrscheinlichkeiten α und β ergeben sich folgende Poisson-Verteilungen für Inversionen und Transpositionen:

$$(1) e^{-\alpha t} \frac{(\alpha t)^k}{k!} \text{ und}$$

$$(2) e^{-bt} \frac{(bt)^k}{k!}$$

l und m definieren die Gesamtzahlen der Inversionen und Transpositionen in einer Sequenz.

$$(3) l = \sum_{i=0}^{\infty} il_i$$

$$(4) m = \sum_{j=0}^{\infty} jm_j$$

Die Indizes l_i und m_j bezeichnen alle möglichen Permutationen von Teilsequenzen (als Index-Tupel) einer Sequenz durch, i und j zählen die Anzahlen für jede Mutation.

Beispiel:

Sequenz

0	1	2	3	...
1 Inversion + 0 T	2 Inversionen+0 T	2 Inversionen+1T	0 Inversionen+2T	
2 Transpositionen+ 0 Inversionen		0 Inversionen + 1 Transposition		
1 Inversion + 2 Transpositionen			1 Inversion+1T	

Die Summe addiert nun die Anzahlen der verschiedenen Mutationen auf:

$$l = 3 * 0 + 3 * 1 + 2 * 2 \dots \quad \{ 3 * 0 \sim \text{„3 mal kam keine Inversion vor“ etc} \}$$

$$m = 2 * 0 + 3 * 1 + 3 * 2 \dots$$

l_i sind die Anzahlen der Positionen, an denen i Inversionen stattgefunden haben. Die Summe über l_i entspricht also für i gegen unendlich der maximalen Anzahl der Inversionen. Wie in den Grundlagen angesprochen, sind für eine Inversion immer genau zwei Partner maßgebend, deswegen kann die maximal-Zahl für l_i berechnet werden als (analog m_j):

$$(5) \sum_{i=0}^{\infty} l_i = \binom{n+1}{2}$$

$$(6) \sum_{j=0}^{\infty} m_j = 3 \binom{n+1}{3}$$

Bei diesem Ansatz werden Transpositionen und invertierte Transpositionen zusammengefasst.

Da die Einzelereignisse als unabhängig betrachtet werden, kann die Gesamtwahrscheinlichkeit als Produkt der Einzelwahrscheinlichkeiten geschrieben werden:

$$(7) \prod_{i=0}^{\infty} \left(\frac{e^{-at} (at)^i}{i!} \right)^{l_i} \prod_{j=0}^{\infty} \left(\frac{e^{-bt} (bt)^j}{j!} \right)^{m_j}$$

Mit den Formeln (1), (2), (3) und (4) kann dies umgeformt werden zu

$$(8) \frac{e^{-\binom{n+1}{2} at} (at)^l e^{-3\binom{n+1}{3} bt} (bt)^m}{\prod_{i=0}^{\infty} (i!)^{l_i} \prod_{j=0}^{\infty} (j!)^{m_j}}$$

Formel (8) beschreibt die Gesamtwahrscheinlichkeit des Vorkommens der entsprechenden Mutationen ohne Reihenfolge. Um nur die Wahrscheinlichkeit eines einzelnen Datensatzes betrachten zu können, werden alle Permutationen (bzw. die Information über die Blöcke) wieder aus dem Produkt herausgezogen, mittels teilen durch Formel (9), womit Formel (10) entsteht.

$$(9) \frac{(l+m)!}{\prod_{i=0}^{\infty} (i!)^{l_i} \prod_{j=0}^{\infty} (j!)^{m_j}} \quad (\text{~ auf wie viele Arten kann man die Ereignisse ordnen})$$

$$(10) \frac{e^{-\binom{n+1}{2} at} (at)^l e^{-3\binom{n+1}{3} bt} (bt)^m}{(l+m)!} \quad (\text{~ mit Reihenfolge})$$

Formel (10) beschreibt die Wahrscheinlichkeit für den geordneten Datensatz, bzw. einen Pfad. Da in evolutionären Abstandbestimmungen Zeit und Rate nicht getrennt werden können, wird im folgenden übersichtlichshalber (at) zu a verkürzt.

b. MCMC Strategie

Als erstes muss errechnet werden, mit welcher Wahrscheinlichkeit ein bestimmter Datensatz G_1 der Länge n vorliegt. Da prinzipiell jede Permutation der Länge n vorliegen kann ergibt sich $n!$. In diesem speziellen Fall spielt allerdings die Orientierung der einzelnen Sequenzabschnitte eine wichtige Rolle. Bei zwei Richtungen kann man n Nukleotide 2^n mal verschieden orientieren. Zusammengefasst ergibt sich also

$$P_8(G_1) = 1 / (2^n n!).$$

Der likelihood einer MCMC kann mit dem Satz von Bayes bestimmt werden:

$$\begin{aligned}
& P(G_1, G_2 | \mathbf{a}, \mathbf{b}) \\
(11) \quad & = P_\infty(G_1) P(G_2 | G_1, \mathbf{a}, \mathbf{b}) \\
& = \frac{P(G_2 | G_1, \mathbf{a}, \mathbf{b})}{2^n n!}
\end{aligned}$$

$P(G_2 | G_1, \mathbf{a}, \mathbf{b})$: Die Wahrscheinlichkeit, dass wir G_2 bekommen, wenn wir G_1 , \mathbf{a} und \mathbf{b} bereits vorliegen haben.

Dies kann auch als die Summe aller Wahrscheinlichkeit der Pfade t von G_1 zu G_2 unter der Bedingung \mathbf{a} und \mathbf{b} interpretiert werden (Formel 12).

$$(12) \quad P(G_2 | G_1, \mathbf{a}, \mathbf{b}) = \sum_{t \in \text{Traj}(G_1, G_2)} P(t | \mathbf{a}, \mathbf{b})$$

Da Einzelwahrscheinlichkeiten nicht einfach aufaddiert werden können, ist diese Gleichung nicht zu berechnen. Deswegen nutzt Miklós die Fähigkeit einer MCMC, um \mathbf{a} und \mathbf{b} zu sampeln.

Die „posterior“- oder „auxiliary“-distribution eines MCMC Netzwerkes ergibt sich wieder aus dem Satz von Bayes:

$$\begin{aligned}
& P(\mathbf{a}, \mathbf{b} | G_1, G_2) \\
& = \frac{P(\mathbf{a}, \mathbf{b}, G_1, G_2)}{P(G_1, G_2)} \\
(13) \quad & = \frac{P_\infty(G_1)}{P(G_1, G_2)} P(G_2 | G_1, \mathbf{a}, \mathbf{b}) P(\mathbf{a}) P(\mathbf{b}) \\
& = \frac{\sum_{t \in \text{Traj}(G_1, G_2)} P(t | \mathbf{a}, \mathbf{b}) P(\mathbf{a}) P(\mathbf{b})}{2^n n! P(G_1, G_2)}
\end{aligned}$$

$P(G_1, G_2)$ wird als Dichte-Funktion der Parameter \mathbf{a} und \mathbf{b} gedeutet und kann wie folgend dargestellt werden:

$$(14) \quad P(G_1, G_2) = \int_{\mathbf{a}, \mathbf{b}} P(G_1, G_2 | \mathbf{a}, \mathbf{b}) P(\mathbf{a}) P(\mathbf{b}) d\mathbf{a} d\mathbf{b}$$

Um nun eine Aussage über \mathbf{a} und \mathbf{b} machen zu können wird über alle Pfade t und Parameter \mathbf{a} und \mathbf{b} gesampelt („joint-sampling“, Gibbs Sample, Geman & Geman 1984).

Beim MCMC ist es üblich konstante Parameter zu vernachlässigen. Mittels sehr niedrigen Eingangswahrscheinlichkeiten für \mathbf{a} und \mathbf{b} und Formel 10 ergibt sich für einen vorgegebenen Pfad t :

$$(15) \quad P(\mathbf{a}, \mathbf{b} | t) \propto e^{-\binom{n+1}{2} \mathbf{a}} e^{-3 \binom{n+1}{3} \mathbf{b}} \mathbf{b}^m$$

$$= f(\mathbf{a})g(\mathbf{b})$$

Nimmt man nun feste Werte für die Parameter \mathbf{a} und \mathbf{b} an (t ist gegeben), kann man einen Metropolis Independent Sampling – Schritt (MIS) durchführen. Es wird ein neuer Pfad t^* erstellt, mittels der auxiliary-distribution g , und mit der Wahrscheinlichkeit aus Formel 16 (Metropolis-Hastings-ratio) akzeptiert oder verworfen. Wird sie akzeptiert, dann wird t^* zu t .

$$(16) \quad \min \left\{ 1, \frac{P(t^* | \mathbf{a}, \mathbf{b})g(t)}{P(t | \mathbf{a}, \mathbf{b})g(t^*)} \right\}$$

Die Schnelligkeit der Berechnung und Qualität der Ergebnisse hängt sehr stark von der Qualität der auxiliary-distribution ab. Sie muss nahe der Ziel-Verteilung liegen, sonst werden richtige Pfad verworfen, und die Ergebnisse schlechter. Ab t würde der Metropolis-Hastings-ratio so klein, dass die meisten Pfade verworfen würden, und die Berechnung um ein vielfaches langsamer.

Der Algorithmus von Miklós hält sich an die in den Grundlagen dargestellten Konventionen, die vorzeichen-behaftete Permutation als Sequenz der Länge $2n$ darzustellen (vergleiche 2d). In circulären Genomen (Prokaryoten, einige Organellen,...) wird nur ein Anker eingeführt, der die linke und rechte Seite verknüpft. Es wird allen Sequenzen, die nicht den Anker beinhalten, erlaubt zu mutieren, ansonsten ist die Sequenz ungültig. Die Erstellung des RDs folgt analog zu Grundlagen 2d. Ebenso werden die in den Grundlagen vorgestellten reversal-Typen aufgegriffen und mit den Transpositionen erweitert, wobei hier zwischen benign (gutartig) und non-benign (bösaartig) unterschieden wird.

Miklós stellt drei Lemmata auf, die alle Fälle betrachten:

LEMMA 1:

Für jede Permutation länger als 1 gibt es entweder eine non-benign Transposition / invertierte Transposition oder die Länge ist ungerade und es liegt die Permutation $n/2, 1, n/2 + 1, 2, \dots, n/2 - 1$ vor.

BEWEIS:

Wenn eine Permutation einen Kreis enthält, und darauf eine Transposition oder invertierte Transposition stattfindet, wird eine schlechte Komponente erzeugt, oder es liegt genau der betrachtete Fall $n/2, 1, n/2 + 1, 2, \dots, n/2 - 1$ vor. Liegt beides nicht vor, so muss es mindestens eine non-benign

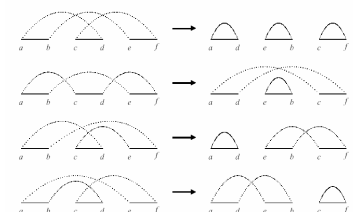


Abb 14

Transposition / invertierte Transposition geben. (zur Erläuterung Abb. 14)

LEMMA 2:

Für jede Permutation länger als 1 gibt es mindestens eine 0-Inversion (die keine Veränderung in der Anzahl der Kreise hervorruft) oder eine -1-Inversion (die die Anzahl der Kreise um 1 erniedrigt)

BEWEIS:

Wenn es keine 0-Inversion gibt, muss jeder Kreis eine Länge kürzer als drei haben und jede Inversion muss eine -1 – Inversion sein. Gibt es keine -1 – Inversion muss es mindestens einen Kreis mit der Länge drei geben, so dass es eine 0-Inversion geben kann. In den Grundlagen in 2c (breakpoint-graph) habe ich bereits dargestellt, dass eine Inversion entweder keine oder mindestens eine Änderung in der Anzahl der Kreise hervorruft.

LEMMA 3:

Für jede Permutation gibt es entweder eine +1 oder eine -1 – Inversion.

BEWEIS:

Auf jeder Permutation, die mindestens einen Kreis enthält, kann eine -1 – Inversion stattfinden, wenn die Länge aller Kreise kürzer als zwei ist (Lemma 2). Gibt es keinen Kreis, so gibt es mindestens eine +1-Inversion oder mindestens ein Kreis hat eine Länge größer als zwei, wodurch ein Kreis erzeugt werden kann (Abb.14).

Der Algorithmus von Miklós geht nun wie folgend vor:

- Um einen neuen Pfad vorzuschlagen, werden die Bögen aller Komponenten durchlaufen
- Bei dem Durchlauf werden die reality-edges mit +1 oder -1 markiert, je nach Orientierung (in welcher Richtung sie durchlaufen werden)
- Für jeden Bogen, der mindestens 3 reality-edges überspannt, werden zusätzlich alle möglichen Transpositionen und invertierte Transpositionen berechnet.
- Daraus wird die Gesamtzahl der möglichen Inversionen und Transpositionen bestimmt.
- Insgesamt können nun sechs Fälle auftreten, die in der Abb. 15 aufgeführt werden.
- Mit der in der Tafel verzeichneten Wahrscheinlichkeit wird nun eine dieser Mutationen für den neuen Pfad angenommen.
- Dies wird solange fortgeführt, bis aus G_1 G_2 geworden ist.

Benign mutation	We have		Benign mutations	Probability of choosing the subset of		Non-benign transpositions and inverted transpositions
	0-inversion	-1-inversion		0-inversions	-1-inversions	
Yes	Yes	Yes	0.99	0.005	0.004	0.001
Yes	Yes	No	0.99	0.009	-	0.001
Yes	No	Yes	0.99	-	0.009	0.001
No	Yes	Yes	-	0.95	0.04	0.01
No	No	Yes	-	-	0.99	0.01
Permutation $n/2, 1, n/2 + 1, 2 \dots n, n/2 - 1$			0.99	0.01	-	-

Abb 15

Die in Tafel 15 vermerkten Wahrscheinlichkeitswerte sind empirisch bestimmt worden. Lt. Miklós kann nicht gesagt werden, ob sie optimal oder auch nur suboptimal sind.

c. Ergebnisse / DERANGE II

Miklós hat seinen Algorithmus mit verschiedenen simulierten Daten und mit realen Daten der Fruchtfliege (*Drosophila melanogaster*) / humane mitochondriale DNA gegen DERANGE II (Blanchette) getestet.

In dem simulierten Datensätzen der Länge 50 Nukleotide (Sequenzen) waren in allen betrachteten Fällen die Anzahlen der Mutationen höher als simuliert.

An der Ergebnisse fallen mehrere Eigenheiten auf:

- Es werden sowohl Inversionen als auch Transpositionen vorhergesagt, auch wenn keine in den simulierten Daten vorhanden waren

Number of inversions		Number of transpositions	
Generated	Estimated	Generated	Estimated
5	4.96 ± 2.59	5	5.17 ± 0.73
0	2.64 ± 2.32	10	8.85 ± 1.10
10	6.46 ± 0.83	0	1.80 ± 0.21
10	8.58 ± 2.51	10	10.99 ± 1.43

Abb 16

- Waren entweder keine Inversionen oder Transpositionen vorhanden, so wurden die jeweils anderen Mutationen under-predicted.
- Inklusive der Fehlertoleranz erzeugt der Algorithmus gute Werte.

In den Vergleich mit DERANGE II von Blanchette et al. schneidet der Algorithmus von Miklós entsprechend gut ab.

DERANGE II unterscheidet entgegen dem stochastischem Ansatz nur zwischen guten und schlechten Mutationen, weswegen manche Pfade nicht begangen werden können, auch wenn sie vielleicht kürzer oder „natürlicher“ sind. Im biologischem Sinne kann auch eine schlecht erscheinende Mutation ein Selektionsvorteil entsprechend Darwins Evolutionstheorie sein. Dies wird durch die Wahrscheinlichkeiten in Miklós's Ansatz berücksichtigt.

Miklós	DERANGE II
6.76 ± 3.32 Inversionen 6.80 ± 1.12 Transpositionen	9 Inversionen 3 Transpositionen = 12 Mutationen
Mit aktuellen P-Werten: 2 Inversionen 9 Transpositionen = 11 Mutationen	Mit leicht veränderten Gewichten (nach oben oder unten) 13 Inversionen 3 Transpositionen

Der Algorithmus von Miklós findet also einen Pfad mit weniger Mutationen, allerdings mit deutlich mehr Transpositionen als der DERANGE II – Algorithmus, der immer eklatant mehr Inversionen findet. Auch scheint der Algorithmus von Miklós resistenter gegen Veränderung der Gewichte (Wahrscheinlichkeiten) zu sein.

4. Ausblick

Durch sorgfältige Erweiterung und/oder Verbesserung der experimentell bestimmten Wahrscheinlichkeitstafel von Miklós und der damit einhergehenden Erstellung der auxiliary-distribution, könnte der Algorithmus von Miklós ein probates Mittel sein, evolutionäre Abstände zu schätzen.

Allerdings gehen die Meinungen der verschiedenen Wissenschaftler über den Nutzen dieser Daten weit auseinander. Miklós selbst behauptet, dass Genom-überspannende Mutationen nicht soviel zum evolutionären Abstand beitragen, wie z.B. Gen-Mutationen. Andere Wissenschaftler hingegen sehen den Einfluss der hier behandelten Inversionen und Transpositionen als maßgeblich an (Schätzungen bis zu 97%).

„...a better implementation of the approach might improve both the theoretical time complexity and the running time in practice.“

“The proposed auxiliary distribution in the MIS step is far from the optimal.“

beides István Miklós

A Quellenverzeichnis

- [1] Miklós, “MCMC genome rearrangement”, Bioinformatics: Vol.19/2003, p. 130-137.
- [2] ramet.elte.hu/~miklosi/MathGenJan20.ppt, 28.07.04 .
- [3] gepard.bioinformatik.uni-saarland.de/html/BioinformatikIIIWS0304-Dateien/V5-Genome-Rearr.PDF, 28.07.04 .
- [4] 163.22.21.49/course/biology/Chapter_7_Genome_Rearrangement.doc, 28.07.04 .
- [5] www.inf.uos.de/theo/veranst/5_GenomeRearrangement.pdf, 28.07.04 .
- [6] engr.smu.edu/~saad/courses/cse8393/lectures/lecture17.pdf, 28.07.04 .
- [7] people.roqe.org/kr/docs/genomere-skript.pdf, 28.07.04 .
- [8] www.cs.wisc.edu/~tsa/cs638/lec13.pdf, 28.07.04 .