Skript zur Vorlesung Bioinformatik I
gelesen Prof. Stadler

Sven Findeiß

11. Januar 2005

Inhaltsverzeichnis

1 Was ist Bioinformatik?

1.1 Informationsquellen

- **Thema der Bioinformatik I**
  Genbank (NCBI) → fr Sequenzinformation
- **Thema der Bioinformatik II**
  PDB (Brookhaven) → 3D – Strukturinformation über Proteine (u. wenige RNA – Moleküle)
  - aus Röntgenstrahlungsanalysen
  - aus Kernspinresonanz
- Heranziehen von bekannten Daten aus bekannten Sequenzen
- Konzept der Homologie

1.2 Problemstellung

String -to -String EDIT Problem
Nachweis bzw. Wissen über Homologie (Lösung über Algorithmen)

2 Alignments

2.1 DNA – Sequenzen

Wörter ber das Alphabet A,C,G,T
Achtung! da zweisträngig, müssen beide Stränge kontrolliert werden
N = A/G/C/T; R = G/A; Y = T/C

2.2 RNA – Sequenzen

Wörter über das Alphabet A,U,G, C
Y = U/C

2.3 Proteine

20 + 1 (Cystein; Selenocystein) + 1 Aminosäuren (Alphabet)
Definition Alphabet – A:

- Wort endlicher Länge
- ohne ϵ

2.4 Begriff der Homologie

Man spricht von `homolog`, wenn sich die 2 Sequenzen von einem gemeinsamen Verfahren ableiten lassen.

2.4.1 Alignment

```
A U G C G
| | | | |
A U G C G
```

2.4.2 Editieroperationen

Ein Alignment kann durch drei Editieroperationen verändert werden.

1. Deletion (Löschen einer Base aus der Sequenz)
2. Insertion (Einfügen einer Base in die Sequenz)
3. Substitution (A $\leftrightarrow$ U)

2.4.3 Forderung an Abbildungen

```
k < l
|   |
u < v
```

Wenn (k,u), (l,v) matches im Alignment und $k < l \rightarrow u < v$.

2.4.4 Drei Editiermöglichkeiten

1. Substitution (A $\leftrightarrow$ U)
   Kosten:
   $\sigma(i,j); i,j \in A$
   $\sigma(i,i) = \oslash$
   $\sigma(i,j) > 0; i \neq j$
   $\sigma(i,j) = \sigma(j,i)$

2. Insertion
   $\delta(i); i \in A$

3. Deletion
   $\delta(i); i \in A$

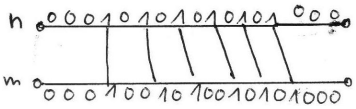
Folgende zwei Regeln müssen gelten:

1. $\sigma(i,j) \leq \delta(i) + \delta(j)$
2. $\sigma(i,j) + \sigma(j,k) \geq \sigma(i,k)$

2.5 Kostengünstiges Alignment

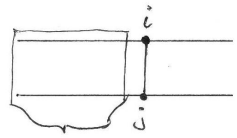
2.5.1 schlechte Möglichkeit

- 2 Sequenzen mit Länge n und m
  - Ausprobieren aller Alignments
  - günstigstes wird ausgewählt

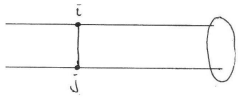


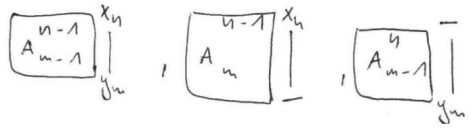
→ wieviele Alignments gibt es mit k Substitutionen
→ $\binom{n}{k} \cdot \binom{m}{k}$ → Produkt ist #Alignment mit genau k Substitutionen
#Alignments = $\sum_k^{min(n,m)} \binom{n}{k} \cdot \binom{m}{k} = \binom{n+m}{n}$

2.5.2 Bessere Möglichkeit



$cost\left(A_{j-1}^{a-1}\right) + cost\left(A_{j-1}^{i+1}\right) + \sigma(i,j)$ (so dass (i,j) Substitution





$cost\left(A_{m-1}^{n-1}\right) + \sigma(x_n, y_m), cost\left(A_m^{n-1}\right) + \delta(x_n), cost\left(A_{m-1}^{n}\right) + \delta(y_m)$
$min\left(mincost\left(A_{m-1}^{n-1}\right) + \sigma(x_n, y_m), mincost\left(A_m^{n-1}\right) + \delta(x_n), mincost\left(A_{m-1}^{n}\right) + \delta(y_m)\right)$

$$D_{ij} := mincost\left(A_j^i\right)$$
$$\rightarrow mincost\left(A_m^n\right) = D_{nm}$$
$$\rightarrow D_{nm} = min\left(D_{n-1,m-1} + \sigma(x_n, y_m); D_{n-1,m} + \delta(x_n); D_{n,m-1} + \delta(y_m)\right)$$
$$D_{0m} = \sum_{j-1}^{m} \delta(y_j)$$
$$D_{n0} = \sum_{i-1}^{n} \delta(x_i)$$

1. $D_{00} = 0$
2. for i = 1...n $D_{i0} = D_{i-1,0} + \delta(x_i)$
3. for j = 1...m $D_{0j} = D_{0,j-1} + \delta(y_i)$
4. `Hamming Distanz` for i = 1...n;
   for j = 1...m;

$$D_{ij} = min\left(D_{i-1,j-1} + \sigma(x_i, y_j); D_{i-1,j} + \delta(x_i); D_{i,j-1} + \delta(y_j)\right)$$

Aufwand:
(1-3)
O(n), O(m) worst = best Case
(4)
O(n · m) → quatratisch ≈ O(n^2)

Vorteil:
linearer Speicheraufwand, da nur vorhergehende Zeile q als Arbeitszeile gemerkt werden muss

Verbesserung:

- unelegant
  - merken, welche der 3 Alternativen im Schritt vorher die geringsten Kosten produziert hat
  - Pfad von (0,0) zu (n,m) beschreibt Alignment → Knoten ist match
  - ABER: Pfadspeicherung nötig → unelegant
- eleganter
  - Start bei (n,m)

– Anschauen der 3 Möglichkeiten, ob

$$D_{i-1,j-1} + \sigma(x_i, y_j) = D_{i,j} \quad oder D_{i-1,j} + \delta(x_i) = D_{i,j} oder D_{i,j-1} + \delta(y_j) = D_{i,j}$$

– Besser, da: i und j werden in jedem Schritt kleiner

Backtracking:
Unter Backtracking versteht man, dass wieder Herausfinden, welchen Weg man gegangen ist um zur richtigen Lösung gekommen zu sein.

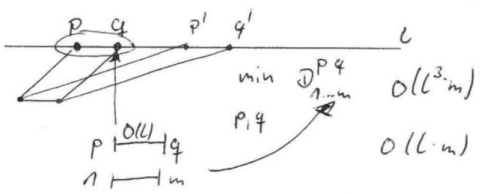
2.5.3 Needleman - Wunsch - Algorithmus

`Reflexion`, was bis hier gemacht wurde:

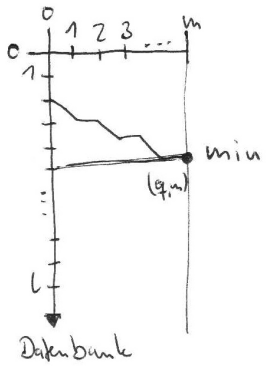
- ̈homologeÄlignments wurden verglichen

`Jetzt`

1. sehr langes Stück vs. sehr kuzes Stück

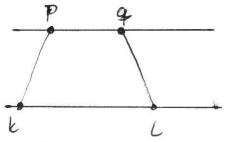


$D_{p-1,0} = \oslash \rightarrow$ hat keine Kosten



$D_{q,m} = minD_{q',m}$
q liefert letzte Substitution, die ungleich ist
in letzter Spalte steht optimales Alignment, d.h. min. Eintrag wird gesucht
Backtracking beginnt bei min.
→ Matrix ermöglicht auch Suche nach Kosten unterhalb eines Schwellenwertes

2. Suche des besten Ausschnitt – matches



Scoring – Funktion hier besonders wichtig
rightarrow Änderung der Kosten: $\sigma(x,x) < 0$
perfect match bekommt also negative Kosten, da sonst l̈eeres Alignmentïmmer beste Lösung

$$D_{p,k} =$$
normaler Alignment
$$D_{p-1,k-1} + \sigma(x_p, y_k); D_{p-1,k} + \delta(x_p); D_{p,k-1} + \delta(y_k)$$
neues Alignment
$$\sigma(x_p, y_k)$$

	A	G	C	C	A
C	1	1	-1	-1	1
C	1	1	-1	-2	0
C	1	1	-1	-2	-1

```
A G C C A
    | |
    C C C
```

- bestes Alignment durch Backtracking
  (a) start bei negativster Zahl
  (b) Rücklauf
  (c) bis zur ersten pos. Zahl
  (d) fertig

2.6 Wiederholung

2.6.1 globale Alignments und Needleman – Wunsch – Algorithmus

Sequenz $x = x_1, \ldots, x_m$ $y = y_1, \ldots, y_n \in \Sigma*$

$$F(i,j) := max(F(i-1,j-1) + s(x_i, y_j)); F(i-1,j) - d; F(i,j-1) - d)$$

F(i,j) = Summe für Alignments zwischen $x_1, \ldots, x_i$ und $y_1 \ldots y_i$
Ausgang: (0,0)
`max, da Interpretation von s und d unterschiedlich zur letzten Vorlesung`
→ F(m,n) groß (max)→ x und sehr ähnlich (Score)
→ letztes Mal: F(m,n) klein(min) → kurze Editierdistanz

2.6.2 lokale Alignments und Smith – Waterman – Algorithmus

$$F(i,j) := max(0; F(i-1,j-1) + s(x_i, y_j); F(i-1,j) - d; F(i,j-1) - d)$$
$$s := max_{i,j} F(i,j)$$

2.7 Wiederholungs – Treffen („Repeat – Matches„)

2.7.1 Ziel

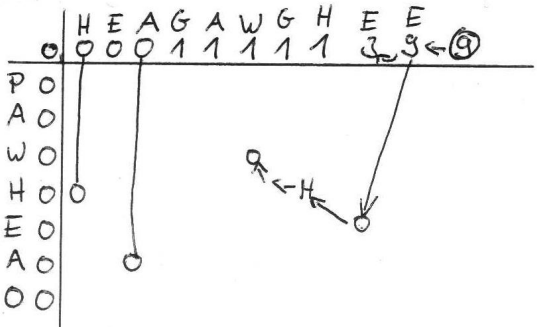
Partitioniere x in Regionen, die mit Teilsequenzen von y überdeckt werden.

2.7.2 Beispiel

x = HEAGAWGHEE
y = PAWHEAE
→
HEAGAWGHEE
HEA-AW-HE-
Rekurenzen (Funktionen, die von sich selbst abhängen = Rekursion) für dynamische Programmierung
Parameter T: Threshold, der mind – Score für matchende Blöcke

$$F(0,0) = 9$$
$$F(i,0) = max(F(i-1,0); F(i-1,j) - T) \rightarrow j \in [1;n]$$
$$F(i,j) = max(F(i,0); F(i-1,j-1) + s(x_i, y_j); F(i-1,j) - d; F(i,j-1) - d)$$

`NEU:` Starte Traceback $F(m+1,0)$

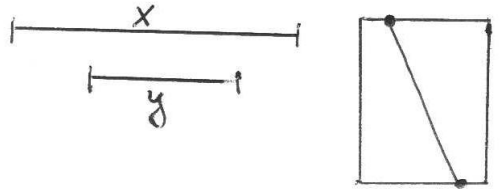


Bsp. in dem Buch besser erklärt

2.8 Overlap – Matches

2.8.1 Problem

Welcher Teilstring $\overline{x}$ von x hat das beste Alignment mit y.



$$
\begin{aligned}
&F(i,0) = 0\\
&F(i,j) = max(F(i-1,j-1) + s(x_i, y_i); F(i-1,j) - d; F(i,j-1) - d)\\
&F(0,j) = F(0,j-1) - d
\end{aligned}
$$

2.9 Gaps

$$
\begin{aligned}
F(i,j) = max(&F(i-1,j-1) + s(x_i, y_j);\\
&F(k,j) + \gamma(j-k)[k \in [0:i-1]];\\
&F(i,k) + \gamma(i-k)[k \in [0:j-1]])
\end{aligned}
$$

Gesamtkosten: $O(m \cdot n(n+m)) \rightarrow$ Zeit

2.9.1 Gotoh – Algorithmus (Gotoh 1988)

$O(m \cdot n)$ Zeit für affine Gap – Kosten

`Verbesserung:`

$$
\begin{aligned}
F(i,j) = max(&F(i-1,j-1) + s(x_i, y_j);\\
&F(k,j) + \gamma(k)[k \in [0:i-1]];\\
&F(i,k) + \gamma(k)[k \in [0:j-1]])
\end{aligned}
$$

offene Gap – Kosten: $\gamma(k) = -d - e \cdot k$
d = Gap – open cost
e = Gap – length cost

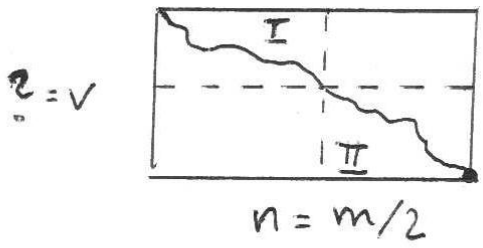
`Gotoh - Algorithmus`
3 Matrizen (m × n)

- M für Matches
- I_x für Gaps in x
- I_y für Gaps in y

$M(i,j) = max(M(i-1,j-1) + s(x_i, y_i); I_x(i-1,j) + s(x_i, y_i); I_y(i,j-1) + s(x_i, y_j))$
$I_x(i,j) = max(M(i-1,j) - d; I_x(i-1,j) - e)$
$I_y(i,j) = max(M(i,j-1) - d; I_y(i,j-1) - e)$
$(2^{18})^2 = 2^{36} = 2^{16}MB = 2^6 GB!$

2.9.2 Line – Space – Algorithmus (Hirschberg 1975, Myas Miller 1988)

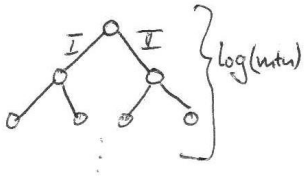
`Idee:`
Divide – and – Conquer



Hilfsgröße $c(i,j)$, fr $i \geq n$
Bei Ausrechnen von F(i,j) bezeichne (i‘,j‘) jene Indizes, von denen das Maximum gewählt wird, d.h. (i‘,j‘) ∈ {(i-1,j-1),(i-1,j),(i,j-1)}

```
c(i,j) = j‘, falls i = n\\
  c(i‘,j‘), sonst\\
```

c(m,n) liefert v. An Position (u,v) spalte Problem auf in Teilprobleme I und II und löse I und II rekursiv.
`maximale Rekursive Tiefe:`



log(m+n)

`Kosten auf jeder Ebene:`
$O(n \cdot m)$
`Gesamtaufwand:`
$O(m \cdot n \cdot log(m+n))$

2.10 Scoring Funktionen für paarweise Alignments

$$x = x_1, \ldots, x_n \text{ Zeitpunkt } 0$$
$$\downarrow t$$
$$y = y_1, \ldots, y_m \text{ Zwitpunkt t}$$

`Annahmen:`

1. Nur Substitution als Gegenstand der Veränderung
   $\rightarrow y_m = y_n$, da x und y gleich lang

$$x_1, x_2, x_3, \ldots, x_{n-1}, x_n$$
$$\downarrow, \downarrow, \ldots, \downarrow, \downarrow$$
$$y_1, y_2, y_3, \ldots, y_{m-1}, y_m$$

2. Sequenzpositionen sind unabhängig
3. Uniformität

Alle 3 Annahmen sind nicht in der Realität umsetzbar → max. Ignoranz der Biologie!

- es gibt Deletion und Insertion
- Mutationen treten wesentlich häufiger an gepaarten Positionen auf als an ungepaarten

`Mutationsrate:` Rate, mit der Mutationen versucht werden durchzusetzen.
`Substitutionsrate:` Rate, mit der Mutationen auch wirklich durchgesetzt werden → können tatsächlich beobachtet werden, da Selektion bereits erfolgt.
→ für uns erst einmal nur Substitutionsrate von Interesse
→ Prob steht fr Wahrscheinlichkeit
→ p(x) = Wahrscheinlichkeit, dass x auftritt
→ p(y—x) = Übergangswahrscheinlichkeit

$$Prob(y_1, y_2, \ldots, y_n; t | x_1, x_2, \ldots, x_n; 0) = \prod_{i=1}^{n} Prob(y_i; t | x_i; 0)$$

$$Prob(y_1, \ldots, y_n | t) = \sum_{x_1,\ldots,x_n} (y_1, \ldots, y_n; t | x_1, \ldots, x_n; 0) \cdot p(x_1, \ldots, x_n; 0)$$

$$= \sum_{x_1,\ldots,x_n} \prod_{i=1}^{n} Prob(y_i; t | x_i; 0) \cdot \prod_{i=1}^{n} p(x_i|0)$$

$$\sum_{x_1} \sum_{x_2} \sum_{x_3} \ldots \sum_{x_n} \cdot \prod_{i=1}^{n} Prob(y_i; t | x_i; 0) \cdot p(x_i|0)$$

$$= \prod_{i=1}^{n} \sum_{x_i} Prob(y_i; t | x_i; 0) \cdot p(x_i|0)$$

$$= \prod_{i=1}^{n} p(y_i|t)$$

$$p(y_i|t) = \sum_{x_i} Prob(y_i; t | x_i; 0) \cdot p(x_i; 0)$$

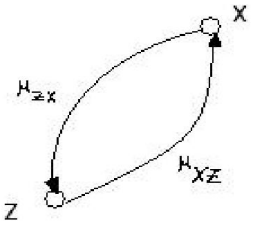
Änderungrate:

$$\frac{dp(x)}{dt}$$

$$= \sum_{Z \neq X} \mu_{XZ} \cdot p(Z) - \underbrace{\sum_{Z \neq X} \mu_{ZX}}_{\mu_{XX} := -\sum_{Z \neq X} \mu_{ZX}} \cdot p(x)$$

$$\underbrace{\sum_{Z \neq X} \mu_X Z \cdot p(Z)}_{\text{Wahrscheinlichkeit X zu erzeugen}}$$

$$\underbrace{- \sum_{Z \neq X} \mu_{ZX} \cdot p(X)}_{\text{Wahrscheinlichkeit Z zu erzeugen aus X}}$$



$$\sum_{Z \neq X} \mu_{ZX} \to \mu_{XX} := \sum_{Z \neq X} \mu_{ZX} = \sum_{Z} \mu_{XZ} \cdot p(Z)$$

$$\frac{d}{dt} \cdot p = \mu \cdot p \text{ mit } A \cdot B; p(0) = p_0$$

`Nebenrechnung zu Radioaktiven Zerfall:`

$$u = a \cdot u \qquad u(t) = K \cdot e^{at}$$
$$u`(t) = K \cdot ae^{at}$$
$$u(0) = K \cdot e^0 = K = u_0$$
$$u(t) = e^{at} \cdot u_0$$

`Nebenrechnung zu Taylor Polynom:`

$$e^{at} = 1 + \frac{at}{1!} + \frac{1}{2!}(at)^2 + \frac{1}{3!}(at)^3 + \dots$$

$$p(t) = e^{\mu \cdot t} \cdot p_0$$
$$e^{\mu \cdot t} = I + \frac{t}{1!}M + \frac{t^2}{2!}M^2 + \frac{t^3}{3!}M^3 + \dots$$
$$\underbrace{\left[e^{Mt}\right]_{KL}}_{\text{Eintrag na der KL-ten Stelle}} := \delta_{KL} + \frac{t}{1!}M_{KL} + \frac{t^2}{2!}(M^2)_{KL} + \dots$$

$\left[e^{Mt}\right]_{KL} \to$ mit $\delta = 1$, fr L = K; 0, sonst

$\to$ konvergiert für jedes beliebiges t $\to$ Nachweis, dass e^{Mt} funktioniert

$$p(t) = e^{Mt} \cdot p_0$$
$$P_X(t) = \sum_{Z} \left[e^{Mt}\right]_{XZ} \cdot p_0(Z)$$

$\to$ einfachstes Modell nach dem man Sequenzen evolvieren kann

2.10.1 Vergleich von zwei Modellen

`Modell I:`
"evolviert" $x \xrightarrow{t} y$
`Modell II:`
zufällig
`Frage:`
Was ist wahrscheinlicher?

$$p^{II}(x,y) = p(x) \cdot p(y)$$
Oder
$$p^{I}(x,y) = p(y|x) \cdot p(x)$$

$$\frac{p^I(x,y}{p^{II}(x,y)}$$
$$= \frac{p^I(x,y)}{p(x) \cdot p(y)}$$
$$= \prod_i \frac{p^I(x_i,y_i)}{p(x_i) \cdot p(y_i)} \rightarrow \texttt{odds ratio}$$
$$log \frac{p^I(x,y)}{p(x) \cdot p(y)}$$
$$= log \prod_i \frac{p^I(x_i,y_i)}{p(x_i) \cdot p(y_i)}$$
$$= \sum_i \underbrace{log \frac{p^I(x_i,y_i)}{p(x_i) \cdot p(y_i)}}_{\sigma(x_i,y_i)} \rightarrow \texttt{log odds ratio}$$

$\sigma(x_i, y_i) > 0$ Wenn x_i und y_i häufiger auftreten als sie im zufälligen Versuch erwartet werden.

- positiv bei Substitution
- negativ bei Deletion, Insertion

$$\frac{p^I(x,y)}{p(x) \cdot p(y)}$$
$$= \frac{\left[e^{tM}\right]_{XY} \cdot p(y)}{p(x) \cdot p(y)} \qquad \rightarrow \sigma(x,y) = log \frac{\left[e^{tM}\right]_{XY}}{p(x)}$$

`Wahrscheinlichkeit`, dass an einer bestimmten Position eine Substitution statt gefunden hat:

$$d = \sum_Y \sum_{XneqY} p(x|y)$$
$d \cdot N =$ Erwartungswert fr Substitutionen
$$d = \sum_Y \sum_{XneqY} \left[e^{tm}\right]_{XY}$$

2.11 Multiple Alignments (Bei Fragen Mail an dress@mis.mpg.de)

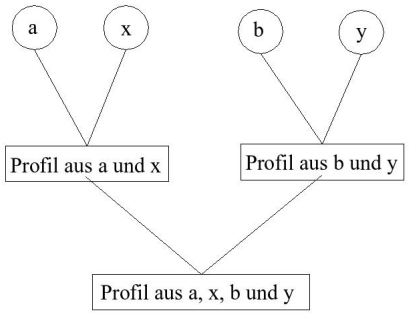
2.11.1 Clustal W

Clustal W ist das bekannteste Multiples Alignment Programm.

$$
\begin{array}{lll}
a_1 & \text{———} & a_n \\
b_1 & \text{———} & b_n \\
 & \text{———} & \\
 & \text{———} & \\
 & \cdots & \\
x_1 & \text{———} & x_n
\end{array}
$$

`Funktion:`

- alle Sequenzen werden erst einmal paarweise alignt → es entsteht der **guide tree**



Entstehung eines Profils

a_1	a_2	a_3	a_4
$-$	x_1	$-$	x_2
$\frac{a_1}{2}, \frac{blank}{2}$	$\frac{a_2}{2}, \frac{x_1}{2}$	$\frac{a_3}{2}$	$\frac{a_4}{2}, \frac{x_2}{2}$

`Beispiel:`
20 Aminosäuren

$R[01]$
$R[1,...,3] = R^n = \{x_1, \ldots, x_n\}$
$(\alpha_1, \ldots, \alpha_{20}, \alpha_{blank})$
$(x_1, \ldots, x_{20}, x_{blank})$
Bedingungen: $(x_i \geq 0; \sum_i x_i = 1)$ → Ähnlichkeit zu Wahrscheinlichkeiten!

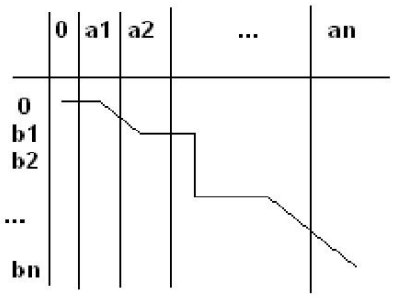
- danach werden Profile aligniert (dieses funktioniert wie bei normalen Sequenzen)

- das entstehende Profil, welches aus 2 Profilen gebildet wird, besteht aus einem Alignment aller 4 Ausgangssequenzen

`Problem:`
Man geht davon aus, dass alle Sequenzen einmal aus einer Ursequenz entstanden sind

2.11.2 DIALIGN (Burkhard Morgenstern, Thomas Werner)

Drei gegebene Sequenzen:
00001111
11110000
10101010

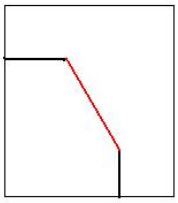


Möglichkeiten sie zu alignieren:

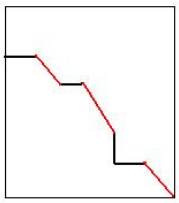
- die ersten beiden werden aligniert
  —-00001111
  11110000—-

- die letzten beiden werden aligniert
  11-11-0-0
  1-01-0-10
  1–1—-1

Die erste Möglichkeit ist wesentlich biologisch wahrscheinlich.



Möglichkeit 1

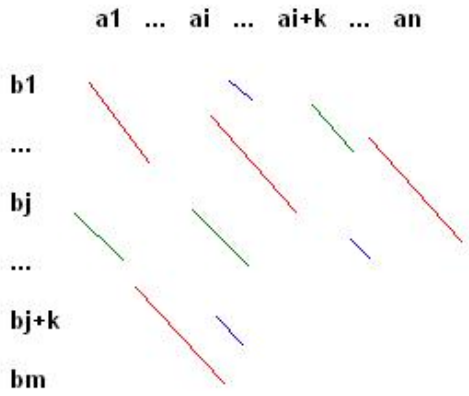


Möglichkeit 2

Es entstehen afine Cap – Kosten

$$\alpha + (k-1)\beta$$

gesucht sind nun möglichst optimale α und β
Die Länge des Matches ist auch entscheident für seinen Wert



`Gesucht:`

- System von Diagonalen, die aneinandergelegt eine lange durchgehende Gerade bilden
- der eine Teil der Diagonalen hört dort auf, wo der nächste beginnt
- die Summe ist am größten $-ln(p(----))$ (Wahrscheinlichkeit)

$$v = \begin{pmatrix} a_i, \ldots, a_{i+k} \\ b_j, \ldots, b_{j+k} \end{pmatrix} \in R^{\alpha}$$

Sortierung der Geraden nach der Größe.
und dann nimmt man die am besten passenden zusammen und wiederholt diesen Vorgang so oft bis alle Geraden verbunden sind und ein endgültiges Alignment entstanden ist.

2.12 Progressives Alignment

2.12.1 Dynamic Programming für N > 2 Sequenzen

`N = 3:`
Man nimmt die drei Sequenzen und aligniert sie genau so wie man 2 Sequenzen alignieren

würde.

$$x_1 \text{———} x_i$$
$$y_1 \text{———} y_j$$
$$z_1 \text{———} z_k$$

Es ergeben sich folgende drei Fälle.

`Fall 1:` $\begin{pmatrix} i-1 \\ \\ j-1 \\ \\ k-1 \end{pmatrix} \begin{matrix} x_i \\ | \\ y_j \\ | \\ z_k \end{matrix}$ `Fall 2:` $\begin{pmatrix} i-1 \\ \\ j-1 \\ \\ k-1 \end{pmatrix} \begin{matrix} x_i \\ | \\ y_j \\ | \\ z_k \end{matrix}$ `Fall 3:` $\begin{pmatrix} i-1 \\ \\ j-1 \\ \\ k-1 \end{pmatrix} \begin{matrix} x_i \\ | \\ y_j \\ | \\ z_k \end{matrix}$

$$D_{ijk} = min \begin{pmatrix} \text{Fall 1} \\ D_{i-1,j-1,k-1} + \sigma(x_i, y_j, z_k) \\ \text{Fall 2} \\ D_{i-1,j-1,k} + \sigma(x_i, y_i, --) \\ D_{i-1,j,k-1} + \sigma(x_i, --, z_k) \\ D_{i,j-1,k-1} + \sigma(--, y_j, z_k) \\ \text{Fall 3} \\ D_{i-1,j,k} + \sigma(x_i, --, --) \\ D_{i,j-1,k} + \sigma(--, y_j, --) \\ D_{i,j,k-1} + \sigma(--, --, z_k) \end{pmatrix}$$

`Allgemein:`

Es gibt$2^N - 1$ Fälle
MEM $O(n^N)$
CPU $O(n^N \cdot 2^N)$

→ für alle Fälle mit N > 3 nicht praktisch relevant, da der Aufwand unrealistisch hoch wird.

2.12.2 Alignieren von 2 Alignments

$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$	...	$\overline{x_n}$
x_{11}	x_{12}	x_{13}	...	x_{1n}
x_{21}	x_{22}	x_{23}	...	x_{2n}
...	...	...	...	...
x_{N1}	x_{N2}	x_{N3}	...	x_{Nn}

$Nxn; A \cup \{-\}$

$\overline{y_1}$	$\overline{y_2}$	$\overline{y_3}$	$\dots$	$\overline{y_n}$
y_{11}	y_{12}	y_{13}	$\dots$	y_{1m}
y_{21}	y_{22}	y_{23}	$\dots$	y_{2m}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
y_{M1}	y_{M2}	y_{M3}	$\dots$	y_{Mm}

$Mxm; A \cup \{-\}$

$\overline{x}$ = Vektor x
$-$ steht für Cap

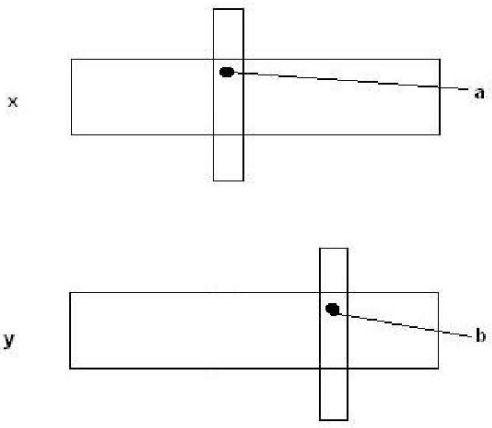
$$\overline{x_k} \in (A \cup \{-\})^N \backslash \{(-,-,\dots)\}$$
$$\overline{y_l} \in (A \cup \{-\})^M \backslash \{(-,\dots,-)\}$$

⇒ Spalten werden als Elemente über einem erweiterbaren Alphabet aufgefasst
⇒ man kann nun die einzelnen Vektoren matchen bzw. mismatchen
⇒ Problem ist die Kostenfunktion

2.12.3 "Sum of Peirs" Score

1. match

$$\sigma(\overline{x_k}, \overline{y_l}) = \sum_{a=1}^{N} \sum_{b=1}^{M} \sigma(x_{ak}, y_{bl})$$



`Mittelwert:`

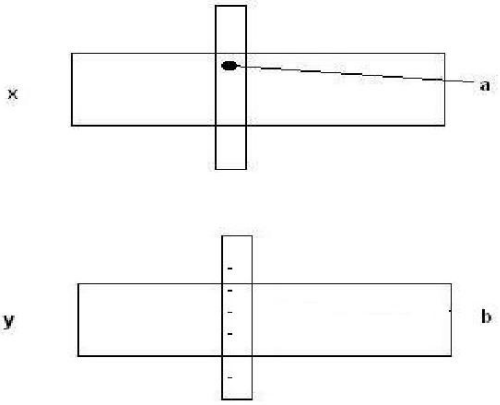
$$\sigma(x_k, y_l) = \frac{1}{N} \cdot \frac{1}{M} \cdot \sum \sigma(x_{ak}, y_{bl})$$

2. mismatch

$$\sigma(\overline{x_k}, -) = M \cdot \sum_{a=1}^{N} \underbrace{\sigma(x_{ak}, -)}_{\delta(x_{nk})}$$

$\rightarrow$ man fügt in das Alignment einen Vektor von Gaps ein

$$\sigma(-, \overline{y_l}) = M \cdot \sum_{b=1}^{M} \underbrace{\sigma(-, y_{bl})}_{\delta(y_{mk})}$$



$$\sigma(-, -) = \oslash(\text{keine Kosten})$$

Es gibt zwei Arten von Gaps bei affinen Gap – Kosten:
A U G $--$ $-$ $-$ $-$ $-$
$-$ extention gap
$--$ gap open
$\sigma(--, -) = konstant = 0$

```
A  U  G  - - - C
A  -- -  - - - C
```

$\Rightarrow$ Problem, wenn man sehr viele Alignments alignieren will

Bessere Möglichkeit?

$$\sum_{a=1}^{N}\sum_{b=1}^{M}\sigma(x_{ak}, y_{bk})$$

$- - - - - - - - - - - - - - - - - -$

NR: (wie oft kommt α oder β in der k-ten oder l-ten Spalte vor?)

$A' = A \cup \{--\}$

$n_k(\alpha) = |\{a | x_{ak} = \alpha \in A'\}|$

$m_l(\beta) = |\{b | y_{bl} = \beta \in A'\}|$

$- - - - - - - - - - - - - - - - - - -$

$$= \sum_{\alpha \in A'}\sum_{\beta \in A'} n_k(\alpha) \cdot n_l(\beta) \cdot \sigma(\alpha, \beta)$$

$\overline{x_k}$

$|$ bildet Spalte im fertigen Ergebnis $\overline{z_q}$ $\longrightarrow$

$\overline{y_l}$

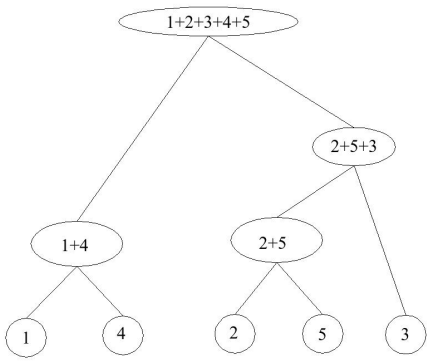
$\underbrace{n''_q(\alpha)}$ $= n_k(\alpha) + m_l(\alpha) \ldots$

wie oft kommt ein Buchstabe im Alignment vor $\Rightarrow$ Profile

$$O(n) \cdot O(|A|) \cdot N$$

Anmerkung: Ein Profil bekommt nie mehr Zeilen, es wird nur länger.
$\Rightarrow$ man fasst nun an immer paarweise Alignments (Sequenzen später Profile) zusammen.
$\Rightarrow$ Frage ist aber welche Alignments sollten zusammen gefasst werden.

Guide -- Tree:



möglichst gleiche/ ähnliche Alignments werden zusammengefasst

$$O(n) \cdot N \cdot O(|A|^2) \cdot O(N)$$

2.13 Probabilistische paarweise Alignments

$\mathring{A}$ =paarweises Alignment

$$
\begin{aligned}
Score(\mathring{A}) &= \sum_{i=1}^{n} Score\begin{pmatrix} x_{1i} \\ y_{2i} \end{pmatrix} \\
Sc\begin{pmatrix} x_{1i} \\ y_{2i} \end{pmatrix} &= K \cdot log \frac{p(x_{1i}, x_{2i})}{p(x_{1i}) \cdot p(x_{x_2 i})} \\
&= K \cdot log \prod_i \frac{p(x_{1i}, x_{2i})}{p(x_{1i}) \cdot p(x_{x_2 i})} \\
&= K \cdot log \frac{p(x_1, x_2)}{p(x_1) \cdot p(x_{x_2})} \\
p(x_1, x_2) &= Prob(\mathring{A}) \\
&= K \cdot logProb(\mathring{A}) - K \cdot log(p(x_1)) - K \cdot log(p(x_2)) \\
K \cdot log(p(x_1)) - K \cdot log(p(x_2)) &= C_0 \\
&= -C_0 + K \cdot logProb(\mathring{A})
\end{aligned}
$$

$$
\begin{aligned}
logProb(\mathring{A}) &= \frac{Score(\mathring{A}) + C_0}{K} \\
C_0' &= \frac{C_0}{K} \\
&\frac{1}{k} \cdot Score(\mathring{A}) + C_0' \\
Prob(\mathring{A}) &= e^{C_0'} \cdot e^{\frac{1}{K} \cdot Score(\mathring{A})}
\end{aligned}
$$

Wenn man alle Möglichkeiten zu alignieren aufsummiert erhält man den Wert 1 (Wahrscheinlichkeiten eines Ereignis).

$$\sum_{\mathring{A}} Prob(\mathring{A}) = 1$$

$$
\begin{aligned}
Prob(\mathring{A}) =& e^{C_0'} \sum_{\mathring{A}} e^{\frac{Score(\mathring{A})}{K}} \\
e^{C_0'} &= \frac{1}{Z} \\
\sum_{\mathring{A}} e^{\frac{Score(\mathring{A})}{K}} &= Z
\end{aligned}
$$

Z := Zustandsmenge des Alignment

$$\underline{Prob(\mathring{A}) = \frac{1}{Z} \cdot exp(\frac{Score(\mathring{A})}{K})}$$

$\Omega \subseteq \mathring{A}|$ Alignments von x_1 x_2
$\Omega^{(p,q)} \subseteq \mathring{A}|$ Alignments von x und y, so dass x_p und x_q gematcht ist

$$\begin{array}{ccc} x_1 \dots x_{p-1} & x_p & x_{p+1} \dots x_n \\ & | & \\ y_1 \dots y_{q-1} & y_q & y_{q+1} \dots y_m \end{array}$$

$$\begin{aligned} Prob(\mathring{A} \in \Omega) &= \sum_{\mathring{A} \in \Omega} Prob(\mathring{A}) \\ &= \frac{1}{Z} \sum_{\mathring{A} \in \Omega} exp(\frac{Score(\mathring{A})}{K}) \\ &\sum_{\mathring{A} \in \Omega} exp(\frac{Score(\mathring{A})}{K}) = Z(\Omega) \\ &= \frac{Z(\Omega)}{Z} \end{aligned}$$

`Anmerkung:`

$Prob(\mathring{A} \in \Omega^{(p,q)})$

- sehr nahe 1 $\Rightarrow$ p und q sind sehr wahrscheinlich gematcht wurden
- sehr nahe 0 $\Rightarrow$ p und q sind sehr wahrscheinlich nicht gematcht wurden

$$Sc \begin{pmatrix} x_1 \dots x_{p-1} & x_p & x_{p+1} \dots x_n \\ & | & \\ y_1 \dots y_{q-1} & y_q & y_{q+1} \dots y_m \end{pmatrix} = Sc \begin{pmatrix} x_1 \dots x_{p-1} \\ \\ y_1 \dots y_{p-1} \end{pmatrix} + \sigma(x_p, y_q) + Score \begin{pmatrix} x_{p+1} \dots x_n \\ \\ y_{q+1} \dots y_m \end{pmatrix}$$

$$\sum_{\mathring{A}\in\Omega^{p,q}} e^{Score(\mathring{A})} = \sum_{\mathring{A}'}\sum_{\mathring{A}''} e^{\frac{Score(\mathring{A}')+Score(\mathring{A}'')+\sigma(x_p,y_q)}{K}}$$

$$\sum_{\mathring{A}'} = \text{Alignment vom ersten Teil}$$

$$\sum_{\mathring{A}''} = \text{Alignment vom zweiten Teil}$$

$$= e^{\frac{\sigma(x_p,y_q)}{k}} \cdot \sum_{\mathring{A}'} e^{\frac{Score(\mathring{A}')}{K}} \cdot \sum_{\mathring{A}''} e^{\frac{Score(\mathring{A}'')}{K}}$$

$$\sum_{\mathring{A}'} e^{\frac{Score(\mathring{A}')}{K}} = Z_{p-1,q-1}$$

$$\sum_{\mathring{A}''} e^{\frac{Score(\mathring{A}'')}{K}} = \tilde{Z}_{p+1,q+1}$$

$$Z(\Omega^{(p,q)}) = Z_{p-1,q-1} \cdot \tilde{Z}_{p+1,q+1} \cdot e^{\frac{\sigma(x_p,y_q)}{K}}$$

$Z_n, m :=$ Zustandsmenge über die Summe aller Sequenzen
$\tilde{Z} :=$ Postfine
$Z_{n.m} = Z = \tilde{Z}_{1,1}$

$$Prob\begin{pmatrix} p \\ \vdots \\ q \end{pmatrix} = \frac{Z_{p-1,q-1} \cdot e^{\frac{\sigma(x_p,y_q)}{K}} \cdot \tilde{Z}_{p+1,q+1}}{Z}$$

$$
\begin{aligned}
Z_{kl} &= \begin{matrix} x_1 \text{———} x_k \\ y_1 \text{———} y_l \end{matrix} \\
&= \begin{pmatrix} \texttt{Fall 1:} & \\ x_1 \text{———} x_{k-1} & x_k \\ y_1 \text{———} y_{l-1} & y_l \\ \texttt{Fall 2:} & \\ x_1 \text{———} x_{k-1} & x_k \\ y_1 \text{———} y_l & - \\ \texttt{Fall 3:} & \\ x_1 \text{———} x_k & - \\ y_1 \text{———} y_{l-1} & y_l \end{pmatrix} = \begin{pmatrix} \texttt{Fall 1:} \\ Score(\mathring{A}_{k-1,l-1}) + \sigma(x_k, y_l) \\ \texttt{Fall 2:} \\ Score(\mathring{A}_{k-1,l}) + \sigma(x_k, -) \\ \texttt{Fall 3:} \\ Score(\mathring{A}_{k,l-1}) + \sigma(-, y_l) \end{pmatrix} \\
&= \sum_{\text{Alignments } \mathring{A}} e^{\frac{Score(\mathring{A})}{K}} \\
&= \sum_{\text{Alignments } \mathring{A} \text{ mit match}} e^{\frac{Score(A_{k-1,l-1})}{K}} \cdot e^{\frac{\sigma(x_k, y_l)}{K}} \\
&+ \sum_{\text{Alignments } \mathring{A} \text{ mit delet von } x_k} e^{\frac{Score(\mathring{A}_{k-1,l})}{K}} \cdot e^{\frac{\sigma(x_k, -)}{K}} \\
&+ \sum_{\text{Alignments } \mathring{A} \text{ mit insert von } y_l} e^{\frac{Score(\mathring{A}_{k,l-1})}{K}} \cdot e^{\frac{\sigma(-, y_l)}{K}} \\
&= Zk-1, l-1 \cdot e^{\frac{\sigma(x_k, y_l)}{K}} + Zk-1, l \cdot e^{\frac{\sigma(x_k, -)}{K}} + Zk, l-1 \cdot e^{\frac{\sigma(-, y_l)}{K}}
\end{aligned}
$$

`Needlman Wunsch Algorithmus`

$$
M_{kl} = max \begin{pmatrix} M_{k-1,l-1} + \sigma(x_k, y_l) \\ M_{k-1,l} + \sigma(x_k, -) \\ M_{k,l-1} + \sigma(-, y_l) \end{pmatrix}
$$

$$
\begin{array}{cc}
M_{00} = 0 & Z_{00} = 1 \\
M_{0l} = \sum_{j=1}^{l} \sigma(-, y_j) & e^{\frac{Score([\overline{y_1 y_2} \ldots \overline{y_l}])}{K}} \\
 & = e^{\sum_{j=1}^{l} \frac{\sigma(-, y_j)}{K}} \\
M_{k0} = \sum_{i=1}^{k} \sigma(x_i, -) & e^{\frac{Score([\underline{x_1 x_2} \ldots \underline{x_k}])}{K}} \\
 & = e^{\sum_{i=1}^{k} \frac{\sigma(x_i, -)}{K}}
\end{array}
$$

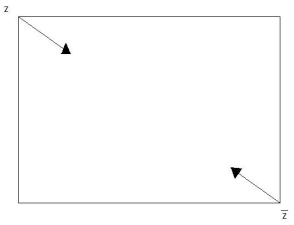
$$
f_{kl} := \frac{Z_{kl}}{exp(\frac{M_{kl}}{K})}
$$

$$\tilde{Z}_{i,j} = \begin{matrix} i & i+1 - - - - - - n; & i & i+1 - - - - - - n; & - & i - - - - - - n \\ | & & | & & | & \\ j & j+1 - - - - - - m; & - & j - - - - - - m; & j & j+1 - - - - - - m \end{matrix}$$

$$= \begin{pmatrix} \sigma(x_i, y_j) + Score(\mathring{A}^{i+1,j+1}) \\ \sigma(x_i, -) + Score(\mathring{A}^{i+1,j}) \\ \sigma(-, y_j) + Score(\mathring{A}^{i,j+1}) \end{pmatrix}$$

$$= \begin{matrix} \tilde{Z}_{i+1,j+1} \cdot e^{\frac{\sigma(x_i,y_j)}{K}} + \\ \tilde{Z}_{i+1,j} \cdot e^{\frac{\sigma(x_i,-)}{K}} + \\ \tilde{Z}_{i,j+1} \cdot e^{\frac{\sigma(-,y_j)}{K}} \end{matrix}$$

$$\tilde{Z}_{n+1,m+1} = 1$$



`gesucht:`
Wahrscheinlichkeiten die beagen wie wahrscheinlich man aus einer Richtung gekommen ist.

$$\begin{aligned} Z_{kl} = Z_{k-1,l-1} e^{\frac{\sigma(x_k,y_l)}{K}} \ldots \text{ match } \searrow \\ + Z_{k-1,l} e^{\frac{\sigma(x_k,-)}{K}} \ldots \text{ del } \downarrow \\ + Z_{k,l-1} e^{\frac{\sigma(-,y_l)}{K}} \ldots \text{ insert } \rightarrow \end{aligned}$$

2.13.1 Stochastisches Backtracking

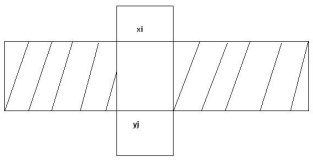
$$Prob(\searrow | k,l) = \frac{Z_{k-1,l-1} e^{\frac{\sigma(x_k,y_l)}{K}}}{Z_{kl}} \rightarrow k-1, l-1$$

$$Prob(\downarrow | k,l) = \frac{Z_{k-1,l} e^{\frac{\sigma(x_k,-)}{K}}}{Z_{kl}} \rightarrow k-1, l$$

$$Prob(\rightarrow | k,l) = \frac{Z_{k,l-1} e^{\frac{\sigma(-,y_l)}{K}}}{Z_{kl}} \rightarrow k, l-1$$

2.14 Suboptimale Alignments

U_{ij} = Score des optimalen Alignments in dem x_i und y_j matchen.

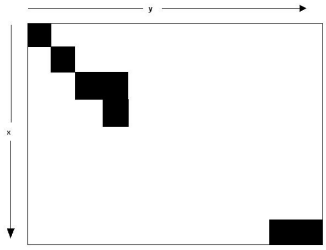


$$M_{i-1,j-1} + \sigma(x_i, y_j) + \tilde{M}_{i+1,j+1}$$

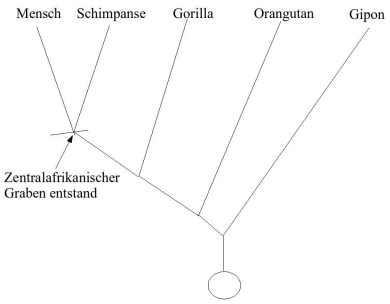
$$\tilde{M}_{i,j} = max \begin{pmatrix} \tilde{M}_{i+1,j+1} + \sigma(x_i, y_j), \\ \tilde{M}_{i+1,j} + \sigma(x_i, -), \\ \tilde{M}_{i,j+1} + \sigma(-, y_j), \end{pmatrix}$$

`Score des optimalen Alignments:`

$$M_{nm} = \tilde{M}_{11}$$



3 Phylogenetische (Stamm-) Bäume



D ist die Distanz zwischen Lebewesen.

3.1 Distanz Messung

- man greift sich ein Gen heraus, welches alle zu betrachtenden Lebewesen besitzen (Cytochrom C sehr kurzes Protein) und vergleicht dieses

A	B	C	D	E	F	G
—	—	—	—	—	—	
—	0	—	—	0	—	—

- man schreibt das Genom des Lebewesens auf und zählt statistisch wie oft welche Base vorkommt
- `allgemein:`
  - alignieren von Genen
  - Chromosomen oder Genome ungeeignet (Lage der Gene unterschiedlich, Chromosomensatz unterschiedlich)

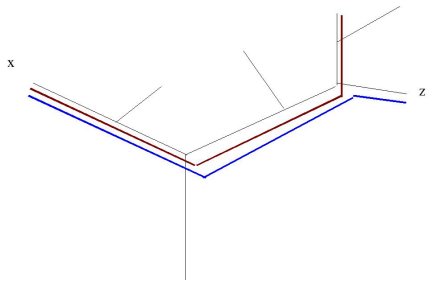
3.2 Genetische Distanz

3.2.1 Bedingungen

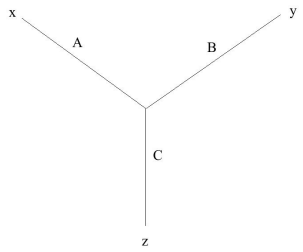
1. $D(x, x) = 0$
2. $D(x, y) > 0$
3. $D(x, y) = D(y, x)$ (Symmetrie)
4. $D(x, y) + D(y, z)$ `rot` $\geq D(x, z)$ `blau` (Dreiecksungleichung)

3.2.2 Ziel

Rekonstruktion eines Baumes, wobei die Kantenlänge die genetische Distanz symbolisiert. Wenn ein Baum existiert so muss die Dreiecksungleichung anwendbar sein.

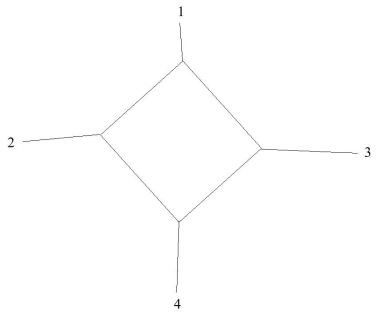


- bei 3 Punkten gibt es eine eindeutige Lösung (Dreiecksungleichung)

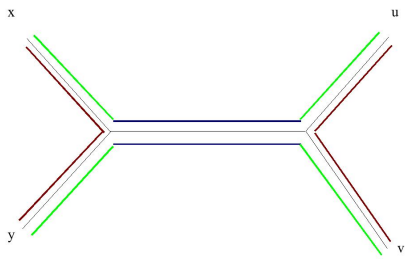


$$A = \frac{D(x,y) + D(x,z) - D(y,z)}{2}$$
$$B = \frac{D(y,x) + D(y,z) - D(x,z)}{2}$$
$$C = \frac{D(z,x) + D(z,y) - D(x,y)}{2}$$

- Problem ergibt sich bei 4 Punkten, da es hier keinen eindeutigen Baum gibt



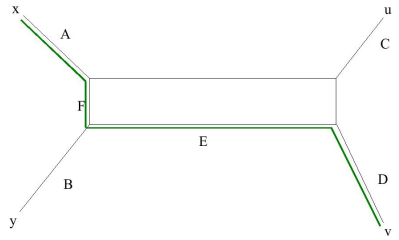
`Annahme:`
Es gäbe einen Baum x,y,u und v



So würden sich folgende Gleichungen für diesen ergeben:

$$D(x,y) + D(u,v) = 1 \cdot rot$$
$$D(x,u) + D(y,v) = 2 \cdot blau + 1 \cdot grün > 1 \cdot rot$$
$$D(v,x) + D(y,u) = 2 \cdot blau + 1 \cdot grün$$

Die größten zwei dieser Summen müssen gleich sein



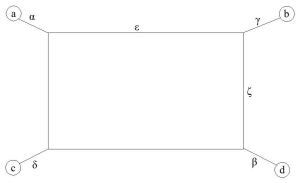
$$A = \frac{D(x,u) + D(x,y) - D(u,y)}{2}$$
$$B = \frac{D(y,v) + D(y,x) - D(x,v)}{2}$$
$$C = \frac{D(u,x) + D(u,v) - D(x,v)}{2}$$
$$D = \frac{D(v,y) + D(v,u) - D(y,u)}{2}$$
$$E = \frac{D(x,v) + D(y,u) - D(x,u) - D(y,v)}{2}$$
$$E = \frac{D(x,v) + D(y,u) - D(x,y) - D(u,v)}{2}$$

$$A + D + E + F = 2xv$$

Bei Boxstruktur wäre es:

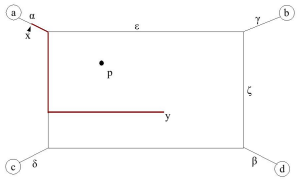
$$(D(x,y) + D(u,v)) \leq (D(x,u) + D(y,v)) < (D(x,v) + D(y,u))$$

$X = \{a,b,c,d\} \rightarrow$ 4 elementige Metrik
$D : X \times X \rightarrow R$
$D(x,y) = xy$



$$
\begin{aligned}
&oBdA: \\
&a, b, c, d \text{ mit} \\
&ab + cd \geq \\
&ac + bd \geq \\
&ad + bc \\
&\exists! \alpha, \beta, \gamma, \delta, \epsilon\zeta \in R \\
&\rightarrow \text{so dass } ac = \alpha + \epsilon + \gamma \\
&ab = \alpha + \epsilon + \zeta + \beta \\
&\alpha = ac + ad - cd \\
&\epsilon = ab + cd - (ad + bc) \\
&\zeta = ab + cd + (ac + bd)
\end{aligned}
$$

`Für mehr als 4 Elemente`
Manhatten – Matrix/ City – Block – Matrix



Beispiel, dass es für mehr als 4 Punkte immer min. 2 minimale Wege gibt → darum T(X,D)

$$
\begin{aligned}
&xy = D_{cb}((x_1, x_2), (y_1, y_2)) = |y_1 - x_1| + |y_2 - x_2| \\
&f_p : X \rightarrow R \\
&x \rightarrow D(p, x) \\
&p = p' \leftrightarrow f_p = f_{p'}, f_p(x) + f_p(y) \geq xy
\end{aligned}
$$

$f : X \rightarrow R$ ist von der Form $f = f_p$, für die obige Figur

1. $f(x) + f(y) \geq D(x, y)$

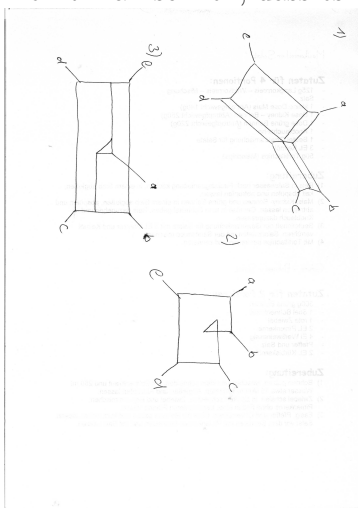
2. Für alle $x \in X$
   existiert $y \in X$ mit $f(x) + f(y) = D(x, y)$

Sei T(X,D) die Menge aller dieser f
`Annahme:`
D genügt der 4 – Punkte Bedingung (eine der Strecken ist 0, d.h. so groß wie die größte → kann weggelassen werden → läßt sich als Baum beschreiben) → wenn D ein Baum sein soll, dann muss diese Bedingung für 4 Punkte gelten ⇒ $T(X, D)$ ist „Baum„ für D.

3.3 Ziel aller Konstruktionen

$$D \approx D‘$$
$$textnormalwobeiD‘der4Punkte - -Bedinunggenügt$$

Es gibt aber keine schnelle und gut berechenbare Lösung. Wenn man nun eine Lösung sucht merkt man schnell, dass es mehrere Möglichkeiten gibt.



zu 1)

$$\zeta = \frac{1}{2} min \left(max \begin{pmatrix} xu + yv \\ xv + yu \end{pmatrix} - (xy + ux) \right)$$
$x, y \in \{a, e\}$
$u, v \in \{b, c, d\}$

X, D
$A \subseteq X \rightarrow B := X - A \Rightarrow$ B ist Komplement von A $\rightarrow$ Betrachtung zweier disjunkter Teilmengen
$A \cap B = \oslash$

$$\alpha^D_{A,B} := \frac{1}{2} min_{x,y \in A; u,v \in B} \left(max \begin{pmatrix} xu + yv \\ xv + yu \\ xy + uv \end{pmatrix} - xy - uy \right) \geq 0$$

$\alpha^D_{A,B} =$ Isolationsindex für den Split A, B

`Problem:`
Es gibt keinen genauen Splitpunkt. Man erhält aus einer Menge von N Spezies so ergeben sich $\frac{2^N - 2}{2}$ Splitmöglichkeiten. Überwindung nur möglich durch Wahl einer sehr kleinen (polinomialen) Teilmenge
`Satz:`
Gegeben X (mit #X = N) und $D : X \times X \rightarrow ReineMatritz.$ (#X bedeutet card(X))

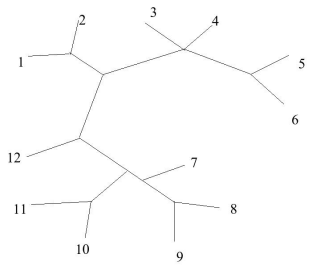
$$\#\{A, B\} \in \varphi(X) : EineKan\alpha^D_{A,B} \geq 0 \leq \binom{N}{2}$$
$$\zeta(X) := \{\{A, B\} \begin{pmatrix} \emptyset \neq A, B \leq X \\ A \cap B = \emptyset \\ A \cup B = X \end{pmatrix}\}$$

wenn ein weiteres Element in einem gegebenen Split einfügen will:

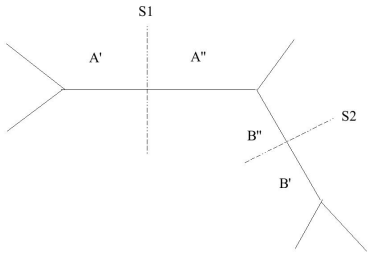
gegeben: $\{A', B'\}; X' \subseteq X; X' \cup \{X\}$
folgt: 2 Möglichkeiten$\{A' + x, B'\}$ oder $\{A', B' + x\}$ Problem wenn immer etwas dazu komm
Lösung:
$N = 4 : 6$
$N = 5 \leq 12$ aber auch $\leq \binom{5}{2} = 10 \rightarrow$ d.h. 2 mal geht etwas schief
nähert sich an $O(n^5)$ im Endeffekt

3.4 Splits

Baum durch Splits beschrieben:

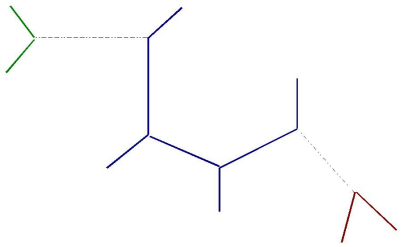


- eine Kante definiert immer einen Split
- 2N - 3 Splits → mehr als phylogenetisch relevant
- die Zerlegung ist eindeutig
- ein Split ist Teilung einer Menge in zwei Teilmengen
- Splitsystem ist eine Menge von Splits auf einer Menge

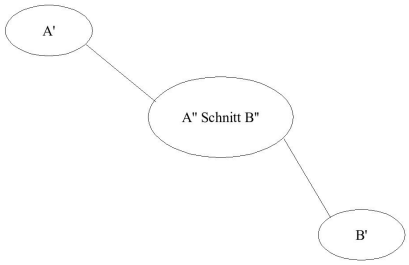


$$A' \cap B'$$
$$A' \cap B''$$
$$A'' \cap B'$$
$$A'' \cap B''$$

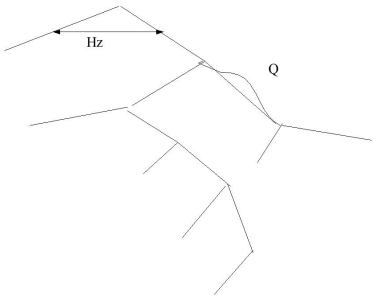
Genau einer dieser 4 Durchschnitte muss leer sein.
Wenn ein Baum idurch zwei Splits (gestrichelte Linien) geteilt wird entstehen immer genau 3 Teilbäume. Im Extremfall sind es Punkte. System von Splits δ auf X (Menge aller Knoten) ist kompatibel, wenn genau einer der Durchschnitte leer ist für je zwei Splits (A', A'') und $(B', B'') \in \delta$.



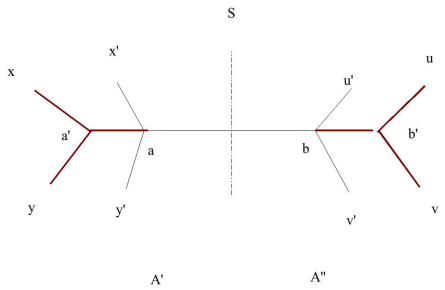
Baum $\Leftrightarrow$ kompatible Splitsystem



Das einfache Modell der Biologen für die Sequenzevolution:



HZ : Horizontaler Gentransfer
Q : Q - Variationen
$\Rightarrow$ es gibt viele „Störungen„, die einen solchen Baum nicht aufbauen lassen
$\Rightarrow$ der Großteil aller Effekte kann aber zu einem solchen Baum approximiert werden



$$D(a',b') = (D(x,u) + D(y,v) - D(x,y) - D(u,b)) \cdot \frac{1}{2}$$

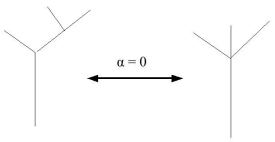
$$min_{x,y \in A';u,v \in A''} D(a',b') = D(a,b)$$

$$\alpha_S = \frac{1}{2} min_{x,y \in A';u,v \in A''}(xu + yv - xy - uv)$$

$\alpha_S =$ starker Isolationsindex

$D(x,v) = \sum_S \alpha_S = \sum_S \delta_S(x,y) \cdot \alpha_S \leq D(x,y)$ Gleicheit gilt wenn es ein Baum ist

`Problem:`
Wenn die Daten nicht exakt Baumartig sind, dann sind die konstruierten Distanzen immer kleiner als die eigentlichen realen Distanzen.



$\Rightarrow$ Splitverfahren sind sehr konservativ

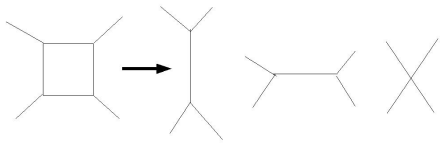
- Kanten die gefunden werden sind richtig
- der Baum ist aber ungenau

$$D(x,y) := f(T) = min \sum_{x,y} \left(\sum_{S \in T} \delta_S(x,y) \cdot c_S - D(x,y) \right)$$

c_S bestimmt die Länge

Wie gut passt mein konstruierter Baum auf meine vorhandenen Daten???
Dieses Verfahren gibt immer einen Baum aus egal was die Daten beschreiben.

Bsp:



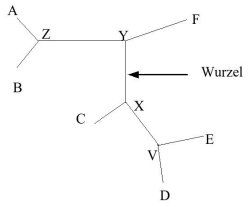
Daten mögliche Ergebnisse

B(n) = Anzahl der binären Bäume mit n nummerierten Blättern
B(n + 1) = B(n) Anzahl der Kanten von binären Bämen mit n Blättern
n Blätter, n - 2 innere Knoten
→ 2n - 2 Knoten
→ 2n - 3 Kanten
B(n + 1) = (2n - 3) B(n)
⇒ zu viele Bäume um sie alle durchzuprobieren

n	2n-3	B(n)
3	3	1
4	5	3
5	7	15
6	9	105
7	11	945

3.4.1 Maximum Porsimoney Problem

Maximum Porsimoney Problem bedeutet mit so wenig Aufwand wie möglich.



Wenn man diesen Baum so genau aufbauen kann → S: #Mutationen entlang $S \cdot \mu(S)$.
$\mu(T) = \sum_{S \in T} \mu(S)$ Gesammtzahl der Mutationen in T (Tree, Baum) mit S = Anzahl der Kanten.