[If you paid for this, you've been ripped off]

In eigener Sache:
Sollten Fehler egal welcher Art in der Ausarbeitung zu finden sein (davon geh ich mal aus, ich bin ein fauler Noob), bitte eine E-Mail an studentmule@gmx.at schicken, damit ich's ausbessern kann wenn Zeit ist und die Anfrage sinnvoll klingt. Ähm, in der E-Mail sollt natürlich drin stehen wo was falsch ist …

Die Mitschrift soll für alle interessierten Personen gratis zu haben sein – oder maximal eine gebrannte CD kosten.

So und jetzt genug mit dem unnötigen Gelaber.

Mitschrift zur Vorlesung "Grundlagen der Bioinformatik" von Ivo Hofacker (VO 300590)

Zur Vorlesung:
Vielen Dank an Chrissi, David und Flo für ihre Mitschriften.

Inhaltsverzeichnis:

Inhaltsverzeichnis:....1
Grundlagen der Bioinformatik....3
Einleitung....3
Einsatzgebiete der Bioinformatik:....3
Sequenzanalyse....4
Ausgehende Erwartungshaltungen der Sequenzanalyse:....4
Was ist eine Sequenz:....4
Grundlage der Sequenzanalyse....5
Sequenzvergleich mittels Hamming distance....5
Modifikation der Hemming distance für Proteinvergleich:....6
Bewertung mittels Edit distance....6
Anpassung der Edit distance an biologische Systeme....7
Alignments....8
Ein Sequenzvergleich liefert unterschiedliche Alignments....8
Ermittlung unterschiedlicher Alignments....9
(1) Kombinatorischer Ansatz:....9
(2) Rekursive Abzählung:....9
Fibonacci Numbers:....10
Ermittlung der Cost eines Alignments:....11
Needleman-Wunsch - Ermittlung eines Alignments mit minimalen Kosten....12
Beispiel:....12
Speicher- und Rechenaufwand nach Needleman-Wunsch....15
Hirschbergverfahren zur Speicherreduktion....16
Globale und lokale Alignments....17
Bewertung mittels Similarity....18
Lokale Alignments:....18
Smith-Waterman-Algorithmus zur Ermittlung lokaler Alignments....19
Deblocking....21
Bedingte Wahrscheinlichkeiten:....22
Wahrscheinlichkeiten von Gaps....23
PAM-Matrizen (Point accepted mutation)....23
BLOSUM-Matrizen (BLOcks of Amino Acid SUbstitution Matrix)....25
Gap-Kosten:....25
Affine Gap-Kosten:....26

Gotoh Algorithmus zur Berechnung affiner Gap-Kosten....27
Final state automaton:....29
Alignment nach Vingron und Argos....30
MEA (Maximum expected accuracy) Algorithmus....31
FASTA Algorithmus....32
BLAST (Basic Local Alignment Search Tool)....33
Multiple Alignments....35
Bewertung multipler Alignments durch Scores....36
Sum of pair score:....37
Weitere Möglichkeiten Sum of pair score:....37
Weighted sum of pair score....37
Berechnung multipler Alignments:....38
Carillo-Lipman-Begrenzung....39
Star Alignment....40
Progressive Alignments....40
Optimierung des Alignments durch Interative Methoden....42
T-Coffee (Tree-based Consistency Objective Function For alignment Evaluation)....42
ProbCons (Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences)....43
DIALIGN (Blockalignment)....43
DCA - Divide and Conquer Algorithmus....44
Phylogenetische Rekonstruktion - phylogenetische Bäume....45
Rooted Tree....45
Klammerdarstellung....45
Polytomie - Nicht vollständig aufgelöste Bäume:....46
Kladogramm....46
Additive Bäume....46
Ultrametrischer Baum....46
Splits....46
Distanzbasierte Methoden....47
Bünemann Theorem:....48
Agglomerative clustering....48
WPGMA (Weighted Pair Group Method with Arithmetic mean)....49
UPGMA (Unweighted Pair Group Method with Arithmetic mean)....49
Warts clustering:....49
Neighbour joining (Saitou und Nei)....50
Compatible splits:....52
Splitmetrik....53
Splitdecomposition:....53
Effizientere Splitzerlegungen:....53
Maximum parsimony Verfahren....54
Der Fitch Algorithmus....54
NNI (Nearest neighbour interchange)....55
Problem der maximum likelihood....56
Modelle für Mutationsraten:....56
Branch and Bound....59
Bootstrapping....59
Bayessche Methoden....61
Markoff Chain....61
Treepuzzle....62
Likelihood mapping....63
State-Modelle....64
Strukturelle Bioinformatik....66
Sekundärstrukturvorhersage....68

Grundlagen der Bioinformatik

08.10.2008

Einleitung

Bioinformatik ist eine Datenorientierte Wissenschaft.
Sie dient der Organisation von molekularbiologischen Experimentaldaten:

- Analyse der Daten
- Modell- und Hypothesenbildung

Die Organisation der Daten ist Vorraussetzung für eine Analyse, diese eine Voraussetzung für Modell- und Hypothesenbildung.

(1) Bioinformatik ist nicht gleichzusetzen mit (2) Computational biology
ad (1) neu
Datenbezogen → die verwendeten Daten stammen aus tatsächlichen Experimenten.
ad (2) älter
Reine Aufstellung von Hypothesen mit anschließender Prüfung dieser Hypothesen durch Computersimulationen z.B. Evolutionsmodelle mit hypothetischen Parametern.

Einsatzgebiete der Bioinformatik:

Das zentrale Dogma der Biologie gibt nur eine Aussage über den Informationsfluss, aber keine Mechanismen von Protein zu DNA.

DNA (Genom) → RNA (Genexpression) → Protein → Funktion

(Erweitert: DNA → RNA → Protein → Funktion)
Viren ncRNA

Daraus ergeben sich unterschiedliche Datenquellen bzw. Arten an Daten:

- Genomische Daten (durch Sequenzierung).
  Die Datenmenge der sequenzierten Genome wächst exponentiell:
  - pro Tag kommen ein paar neue bakterielle Genome dazu.
  - Sequenzierung von individuellen humanen Genomen, um Feinheiten unterscheiden zu können und Unterschiede zu erfassen (Allele).
- Expressionsdaten (durch Sequenzierung von Transkriptom sowie Microchips). Steht erst in der Anfangsphase der Erfassung, es gibt bisher nur wenig Strukturinformation, da diese Daten vor allem aus der Medizin stammen. Diese Daten sind schwer in die Datenstrukturen der Bioinformatik einzugliedern.
- Proteine und RNA (Proteinstrukturen und RNA-Strukturen für Funktion essentiell). Proteindatenbanken wachsen im Vergleich zu genomischen Datenbanken zwar schnell, aber haben im Vergleich immer noch einen kleineren Umfang.

Daraus ergeben sich verschiedene Einsatzgebiete der Bioinformatik:

- Sequenzanalyse für grundlegende Informationen zum Genom.
- Organisation von Expressionsdaten um z.B. Microarrays auszuwerten (Korrelationsanalyse).
- Vergleiche von Strukturen zur Erstellung von Proteinhierarchien (Proteinklassifizierung), Funktionen um Strukturen von ihren Sequenzen her vorauszusagen.
- Medizinisch relevante Daten mit der Genexpression in Verbindung setzen.

Sequenzanalyse

Ausgehende Erwartungshaltungen der Sequenzanalyse:

- Eine Sequenz bestimmt die Struktur, diese wiederum bestimmt die Funktion.
- In den meisten Fällen ist nur eine Sequenz verfügbar, aber keine Information über deren Struktur oder Funktion.
- Das Homolog zu einer bereits bekannten Sequenz kann eine ähnliche Struktur bilden und eine ähnliche Funktion ausüben.
- Sequenzen ändern sich schneller bzw. leichter als ihre Struktur bzw. Funktion. Eine bestimmte Struktur sagt nichts über ihre Sequenz aus (konvergente Evolution). Es wird daher bedeutend mehr unterschiedliche Sequenzen als Strukturen und Funktionen geben (unterschiedliche Lösungen für dieselben Herausforderungen).

Bsp. Zwei Sequenzen mit 60% Übereinstimmung
Eine mögliche Erwartung wäre, sind sie homolog, sind sie verwandt. Da sich Sequenzen schneller ändern als ihre Struktur bzw. Funktion, ist zu erwarten, dass sie eine ähnliche Struktur bzw. Funktion aufweisen, wenn sie sequenziell ähnlich sind.

Was ist eine Sequenz:

$x = x_1 \ldots x_i$ $x \in A$ für DNA … A = {A,T,C,G}
RNA … A = {A,U,C,G}
Protein … A = {20 AS}

Wieso kann man folgern, dass ähnliche Sequenzen tatsächlich verwandt sind und nicht zufällig ähnlich sind.

Bsp: Sequenz aus 100 Einheiten
$n = 100 \ldots 4^{100} = 2^{200}$ ($2^{10} \sim 1000$) … $2^{200} \sim 1000^{20}$
Diese möglichen unterschiedlichen Sequenzen mit vier unterschiedlichen Buchstaben und 100 Einheiten ist bereits größer als die Anzahl der Atome im Universum.
Die Sequenzen in einer Zelle sind dementsprechend nur ein geringer Teil aller möglichen Sequenzen.
Es ist daher sehr unwahrscheinlich, dass ähnliche Sequenzen nicht miteinander verwandt sind und dementsprechend ähnliche Funktion haben müssten.

Reminder:

- Homolog:
  Zwei ähnliche Sequenzen sind verwandt, ähnliche Sequenzen müssen nicht verwandt sein.
- Ortholog
  Zwei ähnliche Sequenzen in verschiedenen Organismen, sie erfüllen tatsächlich gleiche Aufgaben.
- Paralog:
  Zwei ähnliche Sequenzen im gleichen Genom, die verwandt sind, aber nicht dieselbe Funktion übernehmen. Entstehen durch Genduplikation, die unterschiedlichen Funktionen ergeben sich daraus, dass das entstandene Paralog die Funktion ändern kann, ohne dass negative Auswirkungen für den Organismus auftreten.

Die Vorlesung wird sich vor allem mit Homologien befassen.

Grundlage der Sequenzanalyse

- Zunächst wird geprüft, ob zwei Sequenzen ähnlicher sind, als man erwarten würde.
- Sind sie ähnlich, können sie homolog sein.
- Sind sie homolog, haben sie wahrscheinlich eine ähnliche Funktion.

Homologe Sequenzen entstehen durch Mutationen:

- Mutationen die keine Auswirkung auf die Funktion haben (stille Mutationen), haben keine nachteiligen Auswirkungen (keine Selektion). Solche Mutationen werden oft auftreten und Homologien entstehen lassen.
- Mutationen die die Funktion in positiver Weise verändern, sind demgegenüber sehr selten.
- Mutationen die die Funktion nachteilig verändern, werden ausselektiert und so gut wie nie erhalten bleiben.

Sequenzvergleich mittels Hamming distance

Bsp. RNA Sequenzvergleich:

```
AGUUCGAUGG
AGUCCGGUCG
```

Die Sequenzen sehen ähnlich aus, es finden sich nur drei Unterschiede in zehn Einheiten.

- Bei nur drei Unterschieden ist es sehr unwahrscheinlich, dass die beiden Sequenzen zufällig ähnlich sind.
- Es ergibt sich also 75% Sequenzidentität und 25% Unterschied.
- Hamming distance:

$$d_H(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i) \qquad \delta(x_i, y_i) = \begin{cases} 1 \text{ falls } x_i \neq y_i \\ 0 \text{ sonst} \end{cases}$$

x ... Sequenz A
y ... Sequenz B

Die Hemming distance gehört eigentlich zur Signalübertragung und liefert eine Aussage über die Unterschiede nach einer Übertragung.

Bsp. Proteinsequenzvergleich

```
----P----L--G
----A----I--F
```

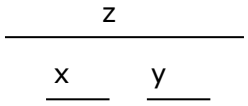
Bei Proteinen muss man die Auswirkung einer AS-Änderung mit einbeziehen. Es ist ein großer Unterschied, ob aus einer AS eine funktionell sehr ähnliche AS (z.B. Leu → Ile) oder eine funktionell verschiedene AS (z.B. Leu → Pro) wird.
Es ist auch von Bedeutung, an welcher Position im Protein (eher in der Mitte, eher am Ende, in einer funktionellen Sequenz, etc.) der Austausch stattfindet.
Bei Proteinvergleichen müssen Sequenzmutationen daher unterschiedlich bewertet werden.

Modifikation der Hemming distance für Proteinvergleich:

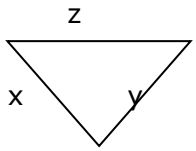
Bei $\delta(x_i,y_i)$ wird statt 0/1 bei $x_i \neq y_i$ eine Zahl eingeführt. Die Zahl wird über eine Distanzfunktion (Metrik) bestimmt.

Metrik:

- Distanz sollte symmetrisch sein … $d(x,y) = d(y,x)$
- Distanz mit sich selbst = 0 … $d(x,y) \geq 0$ und $d(x,y) = 0$ für $x = y$
- Dreiecksungleichung muss erfüllt sein:



kein Dreieck möglich, x und y zu kurz.



möglich, Seiten lang genug

… $d(x,y) \leq d(x,z) + d(y,z)$

$$\delta(\alpha,\beta) = \begin{cases} 0 \text{ falls } \alpha = \beta \\ > 0 \text{ sonst} \end{cases}$$

Erweiterte Hemming distance: $d_H(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i)$

d_H ist vor allem bei Punktmutationen sinnvoll, bei anderen Mutationen sind mehr Werkzeuge nötig, um sinnvolle Vergleichen zu erstellen.

Bewertung mittels Edit distance

Unter Edit distance versteht man die Minimalanzahl an Operationen (z.B. einfache Vertauschung), um Sequenz A in Sequenz B zu überführen.

- Mit jeder Operation kann ein Teil von Sequenz A ausgetauscht werden, bis Sequenz B erhalten wird.
- Jeder Operation wird eine cost >0 zugeordnet.
- Dadurch wird die "cost" ermittelt, die für die Umwandlung in Sequenz B nötig ist.
  cost = 0 bei Übereinstimmung
  cost > 0 bei Änderung der Sequenz

Beispiel: Die Hamming Distanz ist eine Editdistanz mit Punktmutationen als einzige erlaubte Operationen und Kosten von 1 pro Mutation.

Bei längeren Sequenzen oder Sequenzen die nur in Teilen übereinstimmen die durch verschieden lange nicht übereinstimmende Sequenzteile getrennt sind, benötigt man andere Werkzeuge (z.B. bei Deletion oder Insertionen).

Um eine Methode verwenden zu können, müssen daher Vorbedingung erfüllt sein:

- Der Satz von Operationen muss erlauben, dass jede Sequenz A in jede andere Sequenz B überführbar ist. Im Beispiel der RNA-Sequenz wird das durch einfache Vertauschung allein nicht möglich sein.
- Die Methode muss biologisch sinnvoll sein d.h. die verwendeten Operationen müssen tatsächliche biologische Mechanismen(z.B. Deletion) abbilden.

Bsp:

```
AGUUUCGAUCG
AGUU-CGAUGG
```

Hamming distance ist nur möglich, wenn ein zusätzliches Zeichen (-) ins verwendete Alphabet eingefügt wird, um Deletionen / Insertionen zu kompensieren.
Man erhält keine Aussage darüber, welche Sequenz der Vorfahr ist d.h. ob ein U inseriert oder deletiert wurde.
Im aktuellen Beispiel würden die Insertion/Deletion und die Punktmutation eine cost der Edit distance von 2 ergeben. Die cost berechnet sich dabei folgendermaßen:

- 1 für die Punktmutation durch $d(x,y) = d(y,x)$ … Mutation von C → G muss gleich groß sein wie G → C.
- 1 für die Insertion / Deletion durch $d(x,y) \geq 0$ … die cost darf nur dann 0 sein, wenn die beiden Sequenzen übereinstimmen.

Anpassung der Edit distance an biologische Systeme

Eine Überlegung welche Operationen biologisch Sinn machen liefert folgendes Ergebnis:

- Punktmutationen, treten am häufigsten auf
- Insertionen / Deletionen, sind seltener als Punkmutationen, treten aber ebenfalls häufig auf.
- Duplikationen
- Inversionen
- Kombinationen davon

Da nur Punktmutationen und Insertionen / Deletionen häufig auftreten, werden nur sie in weiterer Folge betrachtet.
Um das seltenere Auftreten von Insertionen / Deletionen in Bezug auf Punktmutationen zu berücksichtigen, erhalten sie eine entsprechend höhere cost zugewiesen.
In Proteinvergleichen wird auch je nach Punktmutation eine unterschiedliche cost vergeben, da ein Austausch von AS unterschiedliche Auswirkungen haben kann.

Durch die Erweiterung der Methoden ergeben sich auch Probleme. Bei einem Vergleich mittels Hamming distance werden einfach die einzelnen Einheiten von zwei Sequenz miteinander verglichen … bei 1000nt sind das 1000 Vergleiche.
Ein Vergleich mittels Edit distance ist bereits viel komplexer, weil mehr Operationen zur Verfügung stehen.
Es ist daher ein Algorithmus nötig, der nicht unendlich lang dauert. Daraus ergibt sich, dass bereits vor dem Vergleich eine Methode passend zu den zu vergleichenden Sequenzen gewählt wird.
Als Aufwandsabschätzung werden Rechenzeit und Speicheraufwand in Bezug zur Sequenzlänge gesetzt. Quadratisch oder kubisch wachsender Rechenaufwand ist vernünftig, exponentieller abzulehnen.

Alignments

Alignments können aus verschiedenen Blickwinkeln betrachtet werden:

(A) Ein Vergleich von zwei Sequenzen bzw. die Überführung einer Sequenz in eine zweite Sequenz durch bestimmte Operationen.

```
AGUUUCGAUCG
AGUU-CGAUGG
```

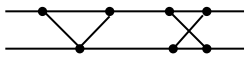
Statt der vergleichenden Ansicht würde es reichen, die erste Sequenz sowie "delete U_5, $C_{10} \rightarrow G_{10}$" anzuführen.

(B) Eine Kombination von jeweils zwei Buchstaben aus einem größeren Alphabet (z.B. {A,C,G,U,-}, hier mit der Ausnahme, dass zwei (-) nie in Kombination d.h. im Vergleich vorliegen dürfen).
Ein Alignment entspricht somit der Erweiterung einer Sequenz für den Vergleich. Die ursprüngliche Sequenz erhält man, indem man das erweiterte Alphabet (im Bsp. "-") aus der Sequenz entfernt.

(C) Ein Alignment kann auch als Reihe von Übereinstimmungen ("Match") betrachtet werden. Ein Match entspricht einer Übereinstimmung einer Einheit in Sequenz A mit einer Einheit in Sequenz B. Zwischen den Matches liegen Lücken ("Gaps"):

```
AGUUUC
||||  |
AGUU-C
```

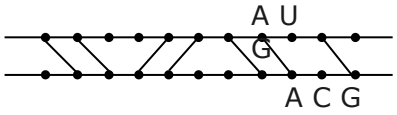
Bedingung für den Vergleich von Sequenzen:



Jede Einheit der Sequenz darf nur einmal für ein Match herangezogen werden, es darf keine Überkreuzungen geben!

Ein Sequenzvergleich liefert unterschiedliche Alignments

Bsp: Vergleich zweier Sequenzen



Daraus ergeben sich zwei unterschiedliche Alignments

```
AU-G          A-UG
A-CG   oder   AC-G
```

Beide Möglichkeiten sind vom Informationsgehalt gleichwertig. A und G liefern ein Match in beiden Sequenzen, wo aber Insertionen / Deletionen erfolgt sind, kann aus den Sequenzen nicht geschlossen werden.

Mit diesem Ansatz will man Gemeinsamkeiten und Unterschiede erhalten, um zu verstehen, wie z.B. ein Protein funktioniert und welche AS-Reste für die Funktion notwendig bzw. nicht notwendig sind.
Die Entwicklungsgeschichte wie die Sequenz entstanden ist, ist in diesem Fall nicht von Bedeutung.

Ermittlung unterschiedlicher Alignments

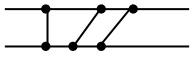
Es ergibt sich die Frage nach der möglichen Anzahl unterschiedlicher Alignments.

(1) Kombinatorischer Ansatz:

Es soll eine Sequenz A der Länge n mit Sequenz B der Länge m verglichen werden. Ermittelt werden soll die Anzahl der Alignments dieser Sequenzen mit genau k Matches. k wird in diesem Fall als Zahl vorher bestimmt und angenommen.

Bsp:
k = 3 … es werden drei zufällige Positionen in Sequenz A und drei in Sequenz B ausgesucht und geprüft, ob sich an diesen Positionen jeweils ein Match findet.

(A)



Diese Positionen werden der Reihe nach überprüft.

Berechnung von k Positionen aus einer Sequenz der Länge n

$$\binom{n}{k} = \binom{n}{n-k} = \frac{n!}{k!(n-k)!}$$

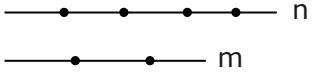
Die Anzahl der Alignments mit k Matches über zwei Sequenzen der Längen n und m:

$$\binom{n}{k}\binom{m}{k}$$

Alle Alignments:

$$\sum_k \binom{n}{k}\binom{m}{k} = \binom{n+m}{n} = \binom{n+m}{m}$$

(B)
Statt in beiden Sequenzen jeweils k Positionen auszuwählen die auf Match überprüft werden sollen, können auch in einer Sequenz Position auswählen die nicht auf Match geprüft werden.
Bsp:



Es werden i=6 Positionen ausgewählt.
In Sequenz A sollen k=4 Positionen ein Match ergeben, in Sequenz B sollen m-k=2 Positionen nicht auf Match geprüft werden.

Die Ansätze (A) und (B) liefern beide die Anzahl der Alignments von zwei Sequenzen mit Länge n und m:

$$\binom{n+m}{n} = \binom{n+m}{m}$$

(2) Rekursive Abzählung:

$S_{n,m}$ … Die Anzahl der möglichen Alignments der Sequenzen mit Länge n und m.

Bsp: Anzahl der möglichen Alignments mit Länge 1:

x_1 | x_1 - | - x_1	Es sind drei Alignments mit Sequenzlänge 1 möglich.
y_1 | - y_1 | y_1 -	$S_{1,1} = 3$

x_1 x_2 x_3 x_4 … x_k	$S_{k,0} = 1$
- - - - … -	

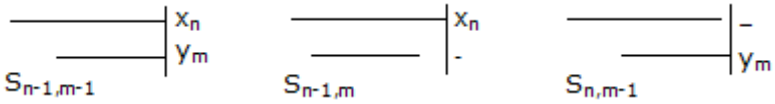
- - - - … -	$S_{0,k} = 1$
x_1 x_2 x_3 x_4 … x_k	

Es werden zwei Sequenzen der Längen n und m miteinander verglichen:

$x_1, x_2, \ldots, x_n$

$y_1, y_2, \ldots, y_m$

Dadurch ergeben sich am Ende des Alignments folgende Möglichkeiten:



Daraus ergibt sich die Anzahl der möglichen Alignments der beiden Sequenzen:

$S_{n,m} = S_{n-1,m-1} + S_{n-1,m} + S_{n,m-1}$

Jeder der erhaltenen Terme $S_{n-1,m-1}$, $S_{n-1,m}$ und $S_{n,m-1}$ kann genau wie $S_{n,m}$ nach dem selben Schema weiter in Einzelterme aufgespalten werden, bis die Berechnung der einzelnen Terme schnell und einfach abgewickelt werden kann, die einzelnen Terme werden anschließend einfach aufsummiert.
Es handelt sich dabei um eine Rekursion, die leicht programmierbar ist.

	1	2	3	4	…	n
1	1	1	1	1	…	1
2	1	3	5	7		
3	1	5	13			
4	1	7				
…	…					
m	1					

Fibonacci Numbers:

Die nächste FN ist die Summe der beiden Zuvorstehenden: 1,1,2,3,5,8,…
Die Berechnung der einzelnen Werte in der Matrix erfolgt über diese Art der Berechnung.

Ein Vergleich der beiden Methoden zur Ermittlung der Anzahl der Alignments zeigt einen Unterschied: $\binom{n+m}{n} \neq S_{n,m}$

Das ergibt sich daraus, dass bei der Rekursion folgende Sequenzalignments als unterschiedliche Alignments gewertet werden:

```
AU-C        A-UC
A-CG        AC-G
```

15.10.2008

Die Rekursion ausprogrammiert:

```
S(n,m)
{       if (n=0 || m=0)
                return 1;
        else
                return S(n-1,m) +
                       S(n,m-1) +
                       S(n-1,m-1);
}
        # hier wird innerhalb des ersten Aufrufs der Funktion S(n,m) die Funktion S(n,m)
        # dreimal mit anderen Werten – eben (n-1,m), (n,m-1) und (n-1,m-1) -
        # aufgerufen. Dieser Vorgang passiert so lange, bis entweder n oder m bei 0
        # angelangt sind.
```

Eine Berechnung dieser Art ist nur mit sehr kurzen Sequenzen zielführend. Weiters ist diese Art der Berechnung nicht sehr sinnvoll, da in der Rekursion mehrmals dieselben Werte berechnet werden.
Bsp: n=5, m=5

S(5,5) = S(4,5) + S(5,4) + S(4,4)
S(4,5) = S(4,4) + S(3,5) + S(3,4)

Sowohl in der Berechnung von S(5,5) als auch in der Berechnung von S(4,5) wird S(4,4) berechnet.

Aus diesem Grund werden Sequenzvergleich wie die oben genannten Fibonaccizahlen berechnet, um doppelte Berechnungen auszuschließen:

	1	2	3	4	…	n
1	1	1	1	1	…	1
2	1	3	5	7		
3	1	5	13			
4	1	7				
…	…					
m	1					

Man spricht im Zusammenhang (Verhinderung doppelter Berechnungen) von "dynamic programming".
z.B. Funktion fibo (n) {if n ≤ 2 return 1; else return fibo(n-1) + fibo(n-2);}

Ermittlung der Cost eines Alignments:

Um Ermitteln zu können, ob bei einem Alignment eine hohe Übereinstimmung zwischen den beiden Sequenzen gegeben ist oder nicht, wird die Cost berechnet.

ATAAG-C		$T_2 \rightarrow C_2$	$C_5 \rightarrow T_5$	In → C_6
ACAATCC	Cost:	1	1	3

Die gesamte Cost wäre in diesem Beispiel 5.

Zur Berechnung der Cost wird eine Matrix mit dem verwendeten Alphabet gebildet und die Cost der einzelnen Umwandlungen eingetragen. Es sind unterschiedliche Werte möglich, die Matrix muss aber symmetrisch befüllt werden, damit die damit ermittelten Distanzen stimmen.

δ	A	C	G	T
A	0	2	1	2
C	2	0	2	1
G	1	2	0	2
T	2	1	2	0

δ … Replacementcost

Die Umwandlung einer Base in dieselbe Base entspricht keiner Umwandlung und erhalt daher Cost 0. Die Umwandlung eines Purins in ein Purin bzw. eines Pyrimidins in ein Pyrimidin entspricht Cost 1, die Umwandlung eines Purins in ein Pyrimidin und umgekehrt einer Cost von 2.
Als Kosten für ein Gap g wird 3 angenommen, da Insertionen bzw. Deletionen seltener vorkommen als Punktmutationen.

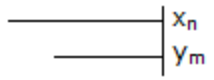
Bsp.:

ATAAG-C … hätte eine Cost von T→C = 1 + G→T = 2 + g = 3 = 6
ACAATCC
0100130

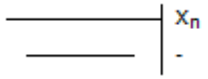
Needleman-Wunsch - Ermittlung eines Alignments mit minimalen Kosten

Hurra, jetzt geht's richtig los!

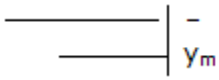
$D_{n,m}$ … ermittelt die bestmögliche Cost für ein Alignment der Sequenz $x_1 \dots x_n$ mit Sequenz $y_1 \dots y_m$.
Wie oben besprochen werden hier die beiden Sequenzen über ihre letzte Stelle verglichen:



… Als Formel gilt $D_{n,m} = D_{n-1,m-1} + \delta(x_n, y_m)$

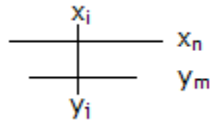


… Als Formel gilt $D_{n,m} = D_{n-1,m} + g$



… Als Formel gilt $D_{n,m} = D_{n,m-1} + g$

Um die geringste Distanz zu ermitteln, wird immer nur der niedrigste Wert für $D_{n,m}$ der drei genannten Formeln verwendet.



Es muss nicht die letzte Stelle verglichen werden, es können auch jeweils eine beliebige Position in jeder der beiden Sequenzen miteinander verglichen werden.

Es wird wieder der Minimalwert der drei möglichen Vergleiche zur Ermittlung der Cost verwendet um die geringste Distanz zu finden:

$$D_{i,j} = \min\begin{cases} D_{i-1,j-1} + \delta(x_i, y_j) \\ D_{i-1,j} + g \\ D_{i,j-1} + g \end{cases}$$

$\delta(x_i,y_j)$ … passende Replacementcost für die Elemente der Positionen x_i und y_j, aus Matrix ermitteln (s.o.)

Nicht vergessen:
$D_{0,j}$ … entspricht dem Vergleich der Position j mit einem Gap, die Cost wäre $j \cdot g$.
$D_{i,0}$ … $i \cdot g$

Beispiel:

Alignment zweier Sequenzen mit den oben angenommenen Kosten für Replacements.

Es werden die einzelnen Positionen der beiden Sequenzen herangezogen und mit den oben besprochenen Formeln die niedrigste Cost für diese Positionen ermittelt und notiert. Da bei diesem Alignment immer auf ein Match der letzten beiden Stellen geprüft wird, ist der allerletzte Wert rechts unten die Cost für das aktuelle Alignment. Diese Cost erhält man nur, wenn zuvor alle Positionen bestimmt wurden.

	D_j	A	T	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	2	3	6	9	12	15	
T	6	5	2	5	8	11	14	
C	9	8	5	4	7	10	13	
A	12	9	8	6	4	7	10	
A	15	12	11	9	6	5	**7**	
	y_j							

Die Cost für das Alignment wäre dementsprechend 7. Doch Stop, Halt, Timeout und so, wie genau kommt man zu den einzelnen Werten?

Die Sequenz wird Zeile für Zeile abgearbeitet und für jede Position die Cost ermittelt. Begonnen wird mit Position Zeile 1 Spalte 1:

	D_j	**A**	T	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3							
	y_j							

Wir verwenden folgende Formeln zur Berechnung der niedrigsten Distanz:

$$D_{i,j} = \min\begin{cases} D_{i-1,j-1} + \delta(x_i, y_j) \\ D_{i-1,j} + g \\ D_{i,j-1} + g \end{cases}$$

Wir setzen ein: i = 1, j = 1

$$D_{1,1} = \min\begin{cases} D_{0,0} + \delta(x_1, y_1) = 0 + 2 = 2 \\ D_{0,1} + g = 3 + 3 = 6 \\ D_{1,0} + g = 3 + 3 = 6 \end{cases}$$

$\delta(x_1,y_1)$ ist 2, da laut Tabelle eine Umwandlung von A nach T einer Cost von 2 entspricht. Die minimale Distanz für $D_{1,1}$ ist in diesem Beispiel demnach 2.

	D_j	**A**	T	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	**2**						

Wir gehen zur nächsten Position i = 2, j =1 über:

$$D_{2,1} = \min\begin{cases} D_{1,0} + \delta(x_2, y_1) = 3 + 0 = 3 \\ D_{1,1} + g = 2 + 3 = 5 \\ D_{2,0} + g = 6 + 3 = 9 \end{cases}$$

Die minimale Distanz für $D_{2,1}$ ist demnach 3.

	D_j	A	**T**	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	2	**3**					

Eins noch in Zeile 1: Position 1 = 3, j = 1:

$$D_{3,1} = \min\begin{cases} D_{2,0} + \delta(x_3, y_1) = 6 + 2 = 8 \\ D_{2,1} + g = 3 + 3 = 6 \\ D_{3,0} + g = 9 + 3 = 12 \end{cases}$$

Die minimale Distanz für $D_{3,1}$ ist demnach 6.

	D_j	A	T	**G**	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	2	3	**6**				

Wir nehmen jetzt an, wir hätten die erste paar Zeilen schon fertig berechnet und gehen an eine Position, um noch einen besonderen Wert anzuschauen:

	D_j	A	T	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	2	3	6	9	12	15	
T	6	5	2	5	8	11	14	
C	9	8	5	4	7	10	13	
A	12	**9**	**8**	6	4	7	10	
A	15	**12**						
	y_j							

Wir sind bei Position i = 2, j = 5.

$$D_{2,5} = \min\begin{cases} D_{1,4} + \delta(x_2, y_5) = 9 + 2 = 11 \\ D_{1,5} + g = 12 + 3 = 15 \\ D_{2,4} + g = 8 + 3 = 11 \end{cases}$$

Der Minimalwert für $D_{2,5}$ ist 11, es ist aber nicht eindeutig, welche Position für den erhaltenen Wert verantwortlich ist. Dieser Punkt ist wichtig, wenn wir uns gleich mit dem sog. Backtracing beschäftigen.

Durch die Berechnungen der minimalen Cost der einzelnen Positionen erhalten wir also die Cost des Alignments. Wenn wir zusätzlich noch einzeichnen, welche Werte zur Ermittlung der minimalen Cost verantwortlich waren, schaut das Ganze wie folgt aus:

	D_j	A	T	G	A	G	A	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	2	3	6	9	12	15	
T	6	5	2	5	8	11	14	
C	9	8	5	4	7	10	13	
A	12	9	8	6	4	7	10	
A	15	12	11	9	6	5	**7**	
	y_j							

Aus dieser Anordnung kann man jetzt Ableiten, wie man zur Cost 7 gekommen ist. Dieser Vorgang wird als "backtracing" bezeichnet und liefert im Beispiel folgenden Weg:

	D_j	A	T	G	A	G	**A**	
D_i	0	3	6	9	12	15	18	x_i – gap cost
T	3	**2**	3	6	9	12	15	
T	6	5	**2**	5	8	11	14	
C	9	8	5	**4**	7	10	13	
A	12	9	8	6	**4**	**7**	10	
A	15	12	11	9	6	5	**7**	
	y_j							

Daraus lässt sich folgendes ableiten:
Den Wert 7 erhält man nur, wenn über A und A an letzter Position ein Match stattfindet und man keine Gaps in die Sequenzen einführt.
Der verantwortliche Wert für den abschließenden Wert 7 eine Zeile höher war ebenfalls eine 7. Das Rückverfolgen der Zeilen wird als "Backtracing" bezeichnet.

Die Sequenz, die man durch Backtracing erhält ist demnach von hinten nach vorne:

```
     774422
x    AGAGTA
y    A-ACTT
```

Woher weis man, in welcher Sequenz der Gap liegt. Der Gap kommt von dort, wo die 3 von der Berechnung der Cost 7 hergenommen wurde (Sequenz x). Werden Gapkosten aus Sequenz y verwendet, um den Minimalwert zu erhalten, muss der Gap auch in Sequenz y liegen.

Betrachten wir noch einmal die weiter oben angesprochene Besonderheit, so kann man jetzt erkennen, dass die Möglichkeit besteht, dass man mittels Backtracing zwei verschiedene Sequenzen für ein Alignment erhält, wenn so eine zweideutige Position im Pfad des Backtracing liegt.
Nachdem beide Sequenzen gleichwertig sind, werden beide Sequenzen ausgegeben. Die Reihenfolge der Ausgabe hängt von der Programmierung ab und wird immer gleich ausfallen.

Das gesamte Verfahren ist die Methode nach Needleman-Wunsch.

Speicher- und Rechenaufwand nach Needleman-Wunsch

Es muss überlegt werden, was passiert, wenn die Sequenzen länger werden. Dabei müssen sowohl benötigter Speicher als auch Laufzeit bedingt durch den Rechenaufwand berücksichtigt werden.

$O(n \cdot m)$ … Die Rechenzeit wächst asymptotisch mit der Sequenzlänge. Auch der Speicherbedarf wächst quadratisch mit der Sequenzlänge d.h. bei Verdoppelung 4x so lang (diese Erkenntnis!).

Der Zeitaufwand für einen Eintrag in die Matrix bleibt immer gleich, daher wächst auch die Laufzeit quadratisch mit der Länge der Sequenzen.
Zusätzlich kommt noch das Backtracing dazu; Berechnungen die auf dem Pfad liegen, müssen noch einmal in der umgekehrten Richtung durchgeführt werden.

$O(n \cdot m)$ … Speicher
$O(n \cdot m)$ … Laufzeit
$O(n + m)$ … Backtracing

Der Speicher ist bei solchen Berechnungen der begrenzende Faktor. Bei Sequenzen von 30.000 x 30.000 dauert die Berechnung ~ 1 Sekunde, dafür sind 4 GB Speicher voll. Speicherorientierte Programmierung ist daher von essentieller Bedeutung.

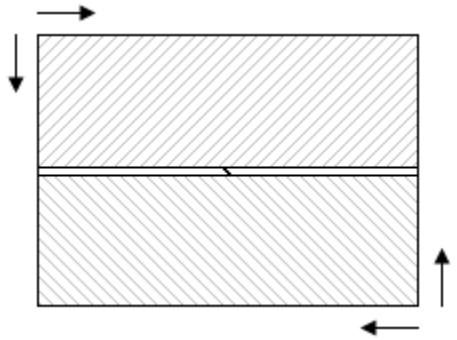
Wenn man z.B. ohne Backtracing arbeitet, wird immer nur die jeweils letzte Zeile der Matrix zur Berechnung der nächsten Zeile im Speicher benötigt.
Backtracing ist natürlich auch notwendig, da man ja auch die Sequenz zum besten Alignment wissen möchte.

Hirschbergverfahren zur Speicherreduktion

D_{ij} … bester Score für Alignment der Sequenz $x_1 \dots x_i$ mit $y_1 \dots y_j$.
D_{ij}^* … bester Score für Alignment der Sequenz $x_i \dots x_n$ mit $y_j \dots y_m$.
Auch bei diesem Ansatz wird wieder mit Rekursionen gearbeitet.

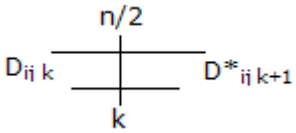
$$D_{i,j}^* = \min\begin{cases} D_{i,j}^*(x_{i+1}, y_{i+1}) + \delta(x_i, y_j) \\ D_{i,j}^*(x_{i+1}, y_i) + g \\ D_{i,j}^*(x_i, y_{i+1}) + g \end{cases}$$

$D_{1,1}^* = D_{n,m}$ ... Einfach ersten Teil der Sequenz berechnen.



Hälfte D_{ij}-Werte berechnen.

Hälfte D_{ij}^*-Werte berechnen.

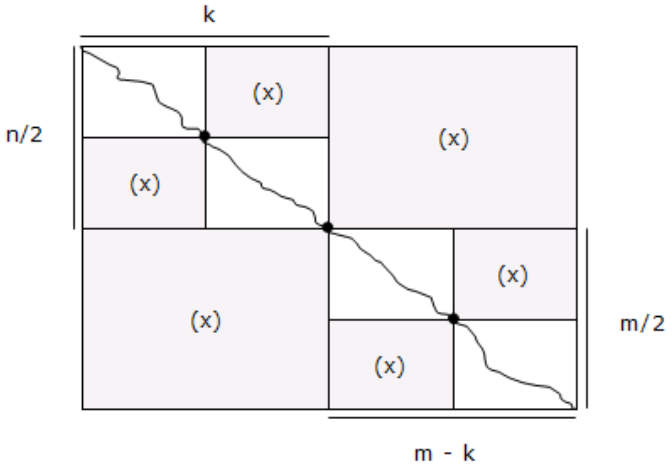


AUG | AGA
UUC | A–A

$D_{3,3} + D_{4,4}^* = 4 + 3 = 7$

$D_{opt} = \min(D_{n/2,k} + D_{n/2+1,k+1}^*)$ … ergibt ein Stück vom Backtrace.

Auf diese Art und Weise wird die Matrix immer weiter eingegrenzt und nach und nach die gesamte Sequenz ermittelt.



Es zunächst der Punkt in der Mitte berechnet. Durch die Bereiche (x) wird der Pfad dabei nie gehen, sie müssen daher nicht berechnet werden. Werden weitere Punkte berechnet, um die Sequenz zu erhalten, wird der Speicherbedarf mit jedem Schritt halbiert.

Der Aufwand ermittelt sich folgendermaßen:
Cost von D und D* = n m

$$\left(\frac{n}{2}k\right) + \left(\frac{n}{2}m - k\right) = \frac{n}{2}m$$

$$nm + \frac{1}{2}(nm) + \frac{1}{4}(nm) + \frac{1}{8}(nm) + \dots = nm\underbrace{\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right)}_{=2 \text{ geometrische Reihe}} = 2\,nm$$

Man erhält verdoppelten Rechenaufwand (2nm), aber nur noch linearen Speicherverbrauch statt quadratischem.

22.10.2008

Globale und lokale Alignments

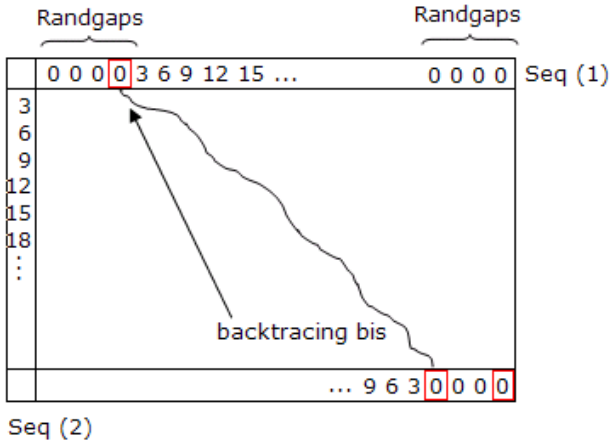
Man spricht im Zusammenhang mit den bisher bekannten Verfahren von globalen Alignments d.h. zwei Sequenzen werden vollständig verglichen.
In der Praxis z.B. bei Vergleich von RNA ergibt sich daraus folgendes Problem. Man weis nicht wirklich, wie lang das Transkript in der Realität ist – vielleicht fehlen am Beginn und/oder am Ende Teile der Sequenz.

Bsp.:

```
-----AUUCGAAU------- (1)
UACCGAUCC-AACUUCAGCU (2)
```

Bei (2) handelt es sich z.B. um eine vollständige Sequenz aus einer Datenbank, Sequenz (1) wurde unvollständige aus einem Organismus isoliert.
Das globale Alignment wird durch die teuren Gaps ein schlechtes Ergebnis liefern, obwohl die Sequenzen lokal ähnlich sind.
Die Gaps am Beginn und Ende dieser Sequenz sind eigentlich irrelevant. Ein für diesen Zweck notwendiger Algorithmus muss diesen Umstand berücksichtigen.

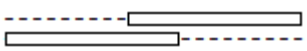
Es kann ein "halblokales" Alignment erzeugt werden, bei dem nur kurze Sequenzen mit einer langen Sequenz verglichen werden. Hier sind nur Gaps im verglichenen Bereich von Bedeutung, nicht aber Gaps am Rand der kurzen Sequenz.
Dabei ergibt sich ein weiteres Problem. Zu kurze Sequenzen können nicht verwendet werden, weil hier zu viele Alignments erhalten werden würden, die aber keine Relevanz im biologischen Sinn haben.



Die kurze Sequenz begrenzende Gaps erhalten daher als Cost 0.

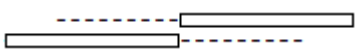
Der beste Wert des Alignments ist jetzt nicht der Wert rechts unten, sondern der erste vor den begrenzenden Gaps.

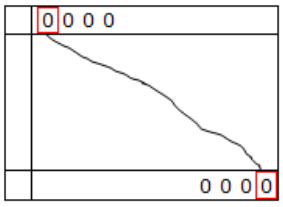
Alignment zwei unvollständiger Sequenzen:



In beiden Sequenzen werden die terminalen Gaps wieder als 0 bewertet (keine Insertionen / Deletionen sondern fehlende Informationen).

Doch Vorsicht: bisher wurde immer Edit distance berechnet. Das bestmögliche Alignment ist der niedrigste Wert d.h. man würde immer folgendes rauskriegen:





Durch Randgaps von 0 zu 0, das ist natürlich ein Unsinn, man muss die Randgaps hier anders lösen.

Bewertung mittels Similarity

Als Alternative werden die Ähnlichkeit (Similarity S_{ij}) der Sequenzen ermittelt:

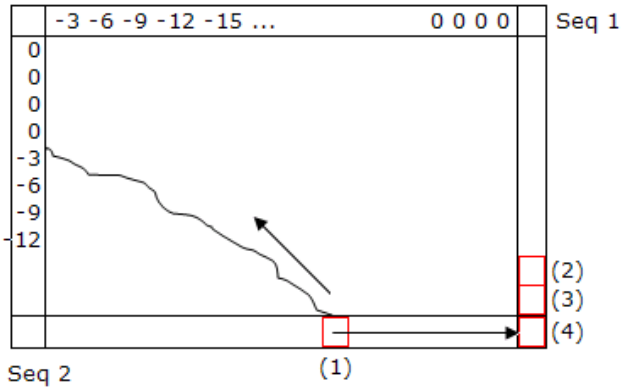
```
A  T  G  -  G  T
A  C  G  T  G  -
2 -1  2 -3  2 -3 = -1
```

Der Gesamtscore wäre -1 was einem schlechten Alignment entspricht. Je ähnlicher eine Sequenz, desto höher muss der Wert sein, den man erhält.

$$S_{i,j} = \max\begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$

Vorteil … Die Werte können sowohl positiv als auch negativ sein, dadurch erhält man im Vergleich mit eine der Edit distance eine flexiblere Bewertung, da das Ergebnis der Edit distance eher schwer zu beurteilen ist.

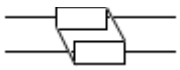
Bei globalen Alignments sind sowohl Edit distance als auch Similarity sinnvoll und Umrechnungen zwischen den Methoden möglich. Rechnet man das optimale Distance Alignment in Similarity um, erhält man auch tatsächlich den besten Wert Similarity.



Wieder zum Bsp: Gaps am Rand werden weiterhin mit 0, Gaps in der zu vergleichenden Sequenz negativ beurteilt.

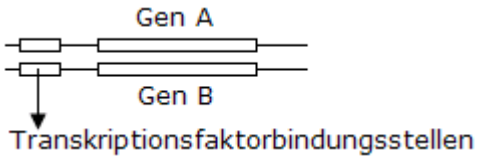
(2) … bestmöglicher Score mit zwei Endgaps in Sequenz (1)
(3) … bestmöglicher Score mit einem Endgap in Sequenz (1)
(4) … bestmöglicher Score ohne Endgap in Sequenz (1)
(1) … bester Score ohne Endgap in Sequenz (1) … von hier aus Backtracing, bis der Rand erreicht wird.

Lokale Alignments:



Bei lokalen Alignments ist nur interessant, ob Sequenz (1) ein Stück enthält, die zu einem Stück in Sequenz (2) sehr ähnlich ist.

Über dieses Verfahren können z.B. chromosomale Rearrangements oder Ähnlichkeiten zwischen Spezies (z.B. Mensch und Maus) ermittelt werden.
Die bisher betrachteten Methoden des Alignments waren nur für Insertionen bzw. Deletionen und Punktmutationen sinnvoll, andere Operationen waren zu rechenaufwändig.



Weiteres Beispiel: in den verglichenen Organismen finden sich unterschiedliche Gensequenzen aber sehr ähnliche TF Bindungsstellen.

Werden alle Subsequenzen von zwei Sequenzen der Länge n und m miteinander verglichen, so sind n^2 und m^2 Subsequenzen möglich.
Das ergibt einen Aufwand von $O(n\ m\ n^2\ m^2)$ und wäre damit extrem teuer.

Eine Anwendung der Edit distance ist in diesem Fall nicht sinnvoll, man arbeitet daher mit Similarity. Ein Alignment muss auf jeden Fall einen positiven Score liefern.

Smith-Waterman-Algorithmus zur Ermittlung lokaler Alignments

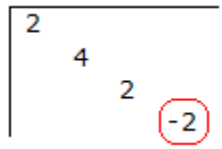
S_{ij} ist der best möglich Score für ein lokales Alignment das in x an Position i und in z an Position j endet. I.e. Ein Alignment der Teilsequenzen $x_k \dots x_i$
$y_l \dots y_j$

Bsp:

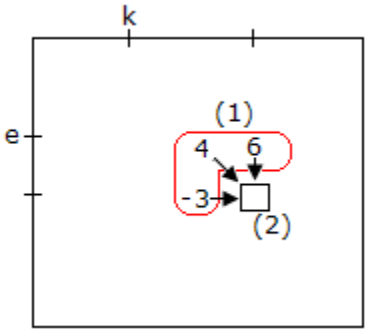
```
AT--GGC(A)
ATCCCGC(-)
```

Dieses Alignment kann nicht sinnvoll sein, weil ein Gap am Ende oder am Anfang einer Sequenz einen zusätzlichen negativen Wert in den Score des Alignments einbringt und den Wert des gesamten Alignments verschlechtert.
Lokale Alignments beginnen und enden daher sinnvollerweise nur mit einem Match.

```
AT--GGC
ATCCCGC
```



Zur Berechnung der Matrix:
ad -2) Negative Werte sind für lokale Alignments uninteressant, je positiver der Wert, desto besser das Alignment.



(1) z.B.
AT--GG|C_i
ATCCCG|C_j

(2) Hier findet sich ein Spezialfall:

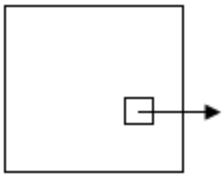
$$S_{i,j} = \max\begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ \mathbf{0 + \sigma(x_i, y_j)} \end{cases}$$

Da es sich um ein lokales Alignment handelt, kann es sein, dass der Wert (2) den Beginn des Alignments darstellt.

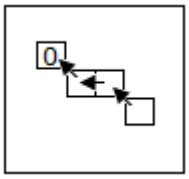
Liefert ein Schritt bei der Erstellung eines Alignments einen schlechten Wert wie bei (2), kann das vorangegangene Alignment abgeschlossen und ein neues Alignment an dieser Position begonnen werden. In so einem Fall würde $0+\sigma(x_i,y_j)$ einen höheren Score liefern als einer der anderen drei Möglichkeiten, es wird daher ein neues Alignment begonnen.

Nachdem nur positive Werte für ein Alignment von Interesse sind, kann man daher prinzipiell so vorgehen, dass man jede Position, die einen negativen Wert ergeben würde, statt dessen eine 0 einsetzt. Setzt man das als Regel zur Berechnung der Matrix ein, kann man weiterhin die bereits bekannte Formel zur Ermittlung der Similarity einsetzen.

$$S_{\mathrm{i,j}} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ 0 \end{cases}$$



Zur Ermittlung des besten Alignments wird der höchste Wert der Tabelle ermittelt. Von dieser Position ausgehend wird Backtracing durchgeführt, wobei immer der höchste vorangegangene Wert den Pfad beschreibt, da wird jetzt mit Similarity und nicht mit Edit distance arbeiten.

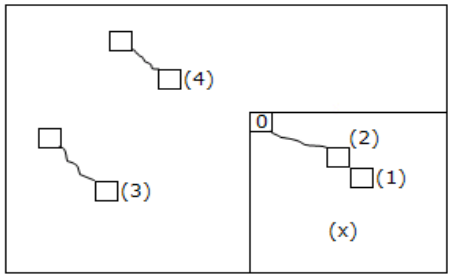


Erreicht man im Zuge des Backtracing eine 0, muss hier das lokale Alignment begonnen haben d.h. man ist entweder am Rand oder ein zuvor erhaltener negativer Wert wurde durch 0 ersetzt und dadurch ein neues Alignment gestartet.

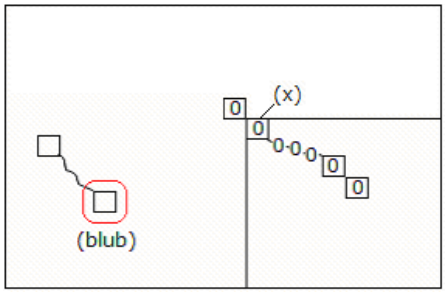
Durch diese Vorgangsweise bleibt man beim Smith-Waterman-Algorithmus mit einem Aufwand von O(n m).

Deblocking

Nachdem ich aber den Hals nicht vollkriegen kann und ALLES will und zwar am besten sofort, wie erhält man nicht nur das Beste, sondern möglichst viele gute Alignments nach ihrem Wert gereiht.



(1) ist der letzte Wert des besten Alignments, (2) wäre vom Wert her das Zweitbeste Alignment. Um die tatsächlich nächstbesten Alignments (3) & (4) zu finden, müssen alle Werte des besten Alignments (1) bis zu seinem Beginn für weiteres Backtracing ignoriert werden, um überhaupt andere, schlechtere Alignments finden zu können.
Dazu werden alle Werte, die Alignment (1) ausmachen, auf 0 gesetzt, damit sie nicht noch einmal gefunden werden. Daraus ergibt sich aber auch, dass dadurch weiteres Backtracing in diesem Zustand verfälscht werden würde, da die Werte links und unterhalb der eingesetzten 0 nicht mehr konsistent berechnet sind.
Setzt man ein Alignment gleich 0, müssen daher die Werte im Kästchen (x) neu berechnet werden.
Es sieht aus, als würde das enorme Berechnungskosten bringen, wenn man das für jedes Alignment neu machen müsste. But fear not, denn:



(x) Dieser Wert muss nicht von der vorangegangenen 0 stammen, er kann ja auch aus einem der zwei anderen vorangegangenen Werte gebildet worden sein. Es gibt daher eine 2/3 Chance, dass die Werte rechts vom Wert (x) gleich bleiben, ab hier muss die restliche Zeile nicht mehr neu berechnet werden.

Meistens sind daher nur einige Werte nach dem 0 Setzen eines Alignments neu zu berechnen, in dieser neuen, wieder konsistenten Matrix kann nach dem nächst besten Alignment (z.B. blub) gesucht werden.
Dieser Vorgang wird als Deblocking bezeichnet.

29.10.2008*

Bedingte Wahrscheinlichkeiten:

$P(A \cap B) = P(A) \cdot P(B)$ wenn A und B voneinander unabhängig sind.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B)}{P(B)} = P(A)$$ wenn A und B voneinander unabhängig sind.

$P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$ … Satz des Bayse, Reihenfolge ist vertauschbar.

R … zufällig ähnliche Sequenzen (R … random)
M … verwandte Sequenzen

$P(x, y \mid R) = P(x \mid R) \cdot P(y \mid R)$ … x,y … Sequenzen, die Wahrscheinlichkeiten sind zerlegbar.

$$P(x \mid R) = P(x_1 \mid R) \cdot P(x_2 \mid R) \ldots = \prod_{i=1}^{n} q_{x_i}$$

… Eine Sequenz wird in seine Einheiten / Buchstaben aufgespalten, es wird angenommen, dass die Wahrscheinlichkeiten für jeden Buchstaben gleich ist.

$$P(x, y \mid R) = P(x \mid R) \cdot P(y \mid R) = \prod_{i=1}^{n} q_{x_i} \cdot \prod_{i=1}^{m} q_{y_i}$$

$$P\left(\begin{matrix} x_1, x_2, \ldots, x_n \\ y_1, y_2, \ldots, y_m \end{matrix} \middle| M \right) = \prod_{i=1}^{n} P_{x_i y_i}$$

… beide Sequenzen sind gleich lang und enthalten keine Gaps.

q … Häufigkeit des entsprechenden Nukleotids / der AS.
$P_{a,b}$ … Häufigkeit eines Matchs in Form von Nukleotid-/AS-Paaren in Alignments oder einer Umwandlung von a zu b.

Odds ratio: $$\frac{P(x, y \mid M)}{P(x, y \mid R)} = \prod_{i=1}^{n} \frac{P_{x_i y_i}}{q_{x_i} q_{y_i}}$$

$$\log \frac{P(xy \mid M)}{P(xy \mid R)} = \sum_{i=1}^{n} \log \frac{P_{x_i y_i}}{q_{x_i} q_{y_i}} = \sum_{i=1}^{n} \sigma(x_i, y_i)$$

… Mittels des Logarithmus lassen sich die Scores des Alignments einfach aufaddieren, anstatt mit Wahrscheinlichkeiten rechnen zu müssen.

$\sigma(a, b) = \log \frac{P_{ab}}{q_a q_b}$ … "log-Odd"

Der Alignmentscore ist demnach durch $\sum_{i=1}^{n} \sigma(x_i, y_i)$ gegeben.

Der Score liefert ein negatives Ergebnis, wenn das Alignment seltener und ein positives Ergebnis, wenn das Alignment öfter als erwartet vorkommt.

Wahrscheinlichkeiten von Gaps

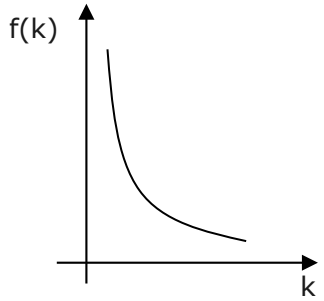
ATA

$$P(\text{Gap} \mid M) = \underset{\substack{\text{gap penalty}\\ \text{Gap der}\\ \text{Länge k}}}{f(k)} \cdot \prod_{i \in \text{Gap}} q_{x_i}$$

$$P(\text{Gap} \mid R) = \prod_{i \in \text{Gap}} q_{x_i}$$

$$\text{ratio} = \frac{P(\text{Gap} \mid M)}{P(\text{Gap} \mid R)} = \frac{f(k) \prod_{i \in \text{Gap}} q_{x_i}}{\prod_{i \in \text{Gap}} q_{x_i}} = f(k)$$

$$g(k) = k \cdot g_0 \Rightarrow g(R) = \log f(k) \Rightarrow f(k) = e^{kg_0}$$



g_0 muss für eine fallende Exponentialfunktion negativ sein.
Gapkosten werden empirisch geschätzt, dazu wird ein Satz guter, repräsentativer Alignments benötigt.

PAM-Matrizen (Point accepted mutation)

Bei PAM handelt es sich um Matrizen die der Scoreberechnung von Alignments dienen. Es gibt mehrere Matrizen, die auf tatsächlich beobachteten Mutationen in verwandten Proteinen beruhen; die entsprechenden Mutationen wurden einfach abgezählt und in Relation zueinander gesetzt. Die Matrizen sind 20x20 (für 20 AS), jede Zelle gibt eine Wahrscheinlichkeit für die Umwandlung einer AS in eine andere an.
Die einzelnen Matrizen unterscheiden sich darin, wie oft eine Mutation in einer Sequenz von 100 AS auftritt.

Wie kommt man zu solchen Matrizen:
$P_{a,b}$ ist eine Funktion der Zeit … $P_{a,b}(t)$. Je länger der gemeinsame Vorfahre zurückliegt, desto mehr Veränderungen konnten in den Sequenzen erfolgen.

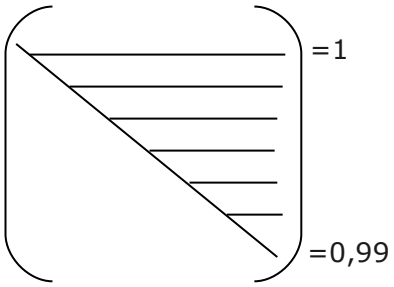
$\sigma(a,b) = \log \frac{P_{ab}(t)}{q_a q_b}$ … Zur Berechnung würden eigentlich mehrere Matrizen benötigt.

$P(b \mid a,t) = \frac{P_{ab}(t)}{q_a}$ … Wahrscheinlichkeit, dass sich Element b über die Zeit t zu Element a verändert hat.

$$B_{a,b} = A_{a,b} \Big/ \sum_c A_{a,c}$$

PAM1 entspricht einer Mutation auf 100AS, PAM250 entspricht 250 Mutationen auf 100AS. Die Angabe der Zahl entspricht der evolutionären Distanz (d.h. vergangene Zeit) zwischen den beiden Sequenzen; je größer die Zahl, desto größer die Distanz.
Die Umwandlungen der einzelnen AS in eine andere erhalten unterschiedliche Werte, da manche AS leichter mutiert werden können als andere.
Verwendet man eine Matrix die einer größeren evolutionären Distanz / Zeiteinheit entspricht als PAM1, werden einfach alle Werte in der Matrix mit einem bestimmten Faktor multipliziert.

Wie sieht so eine Matrix aus:



Matrix für die Zeit PAM1:
$\sum_{a \neq b} q_a \cdot q_c \cdot B_{a,b} = 0{,}01$

Die Zeilensumme muss im Mittel 1 sein, jede Zeile entspricht der Wahrscheinlichkeit der Umwandlung einer AS in eine andere.

Zur Ermittlung der Matrizen für größere Zeiteinheiten wird jede Zeile einfach mit einem Faktor bzw. Vektor multipliziert:
$S_{a,b}$ = P(b|a, t=1PAM)
S_n = P(b|a, t=nPAM) = S^n

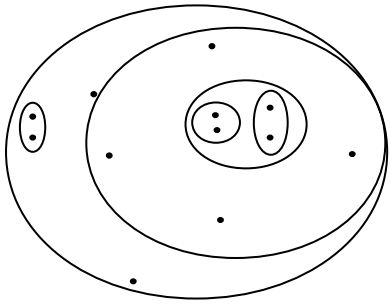
$$\underline{p}(t+1) = S \cdot \underline{p}$$
$$\underline{p}(t+2) = S \cdot \underline{p}(t+1) = S \cdot S \cdot \underline{p} = S^2 \cdot \underline{p}$$
$$P(n) = S^n \underline{p}(0)$$

(Bin mir nicht sicher, aber ich glaub hier wird einfach die Matrix vor der aktuellen Matrix mit Matrix PAM1 multipliziert d.h. braucht man PAM34, multipliziert man PAM33 mit PAM1).
Dabei ergibt sich das Problem, dass Wahrscheinlichkeiten aus anderen Sequenzvergleichen herangezogen und miteinander multipliziert werden. Treffen die Wahrscheinlichkeiten nicht auf die aktuell betrachteten Sequenzen zu, potenzieren sich die Fehler bei größeren Zeiteinheiten.

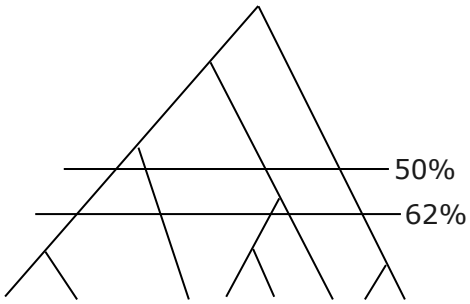
Diese Matrizen enthalten Umwandlungswahrscheinlichkeiten, noch keine Scores.
Scores mittels log-Odds: $\sigma(a,b) = \log \frac{P_{a,b}(t)}{q_a \cdot q_b} = \log \frac{P(b \mid a,t) \cdot q_a}{q_a \cdot q_b} = \log \frac{P(b \mid a,t)}{q_b}$

BLOSUM-Matrizen (BLOcks of Amino Acid SUbstitution Matrix)

Zur Erstellung der BLOSUM Matrizen wurden hochkonservierte Regionen von Proteinen herangezogen, die keine Gaps bei Alignments aufwiesen und erneut die vorkommenden AS abgezählt sowie deren Umwandlungswahrscheinlichkeit aufgenommen und die log-Odds berechnet.



Die Alignments werden mittels single linkage clustering erstellt. Dabei werden alle Alignments zusammengefasst, die bis zu einem bestimmten Prozentsatz Ähnlichkeit besitzen; man beginnt mit dabei mit der kleinsten Distanz zwischen zwei Punkten. D.h. alle die 90% Sequenzähnlichkeit besitzen, dann alle die 80% Ähnlichkeit besitzen etc.
Dadurch ergibt sich eine Hierarchie der Cluster die in eine Baumstruktur umgewandelt werden kann.



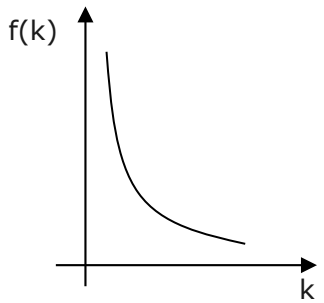
Wird der Baum an einer bestimmten Stelle geschnitten, ergeben sich eine bestimmte Anzahl an Clustern in dieser Schnittebene. Die Alignments in dieser Schnittebene sind zu einem bestimmten Prozentsatz identisch. Darüber ergeben sich die unterschiedlichen BLOSUM Matrizen. BLOSUM50 wäre eine Matrize, bei deren Erstellung die Alignments eine maximale Ähnlichkeit von 50% hatten, bei BLOSUM62 eine maximale Ähnlichkeit von 62% etc.

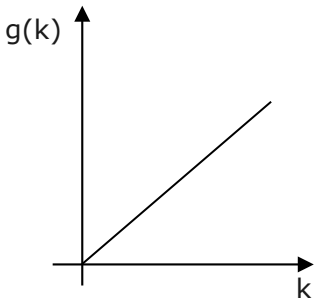
Bei PAM ergab eine hohe Zahl im Namen eine große evolutionäre Entfernung, bei BLOSUM ergibt eine hohe Zahl eine hohe Ähnlichkeit.

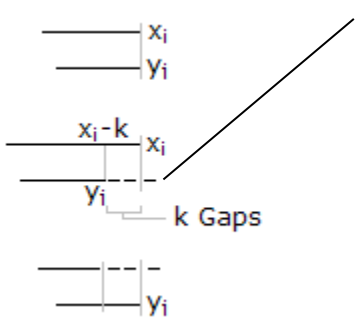
Werden BLOSUM Matrizen zur Bewertung eines Alignments herangezogen, so muss man entscheiden, wie nah die Sequenzen verwandt sind. Sind sie sehr nah verwandt, wird man eine BLOSUM Matrize mit hoher Zahl heranziehen, sonst eine mit niedriger Zahl.

Gap-Kosten:

$g(k) = \log f(k)$
$g(k) = k\, g_o$

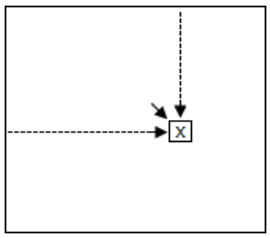






Länge k des Gaps entscheidend

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + \sigma(x_i, y_j) \\ \max\limits_{1 \le k \le i} S_{i-k,j} + g(k) \\ \max\limits_{1 \le k \le j} S_{i,j-k} + g(k) \end{cases}$$



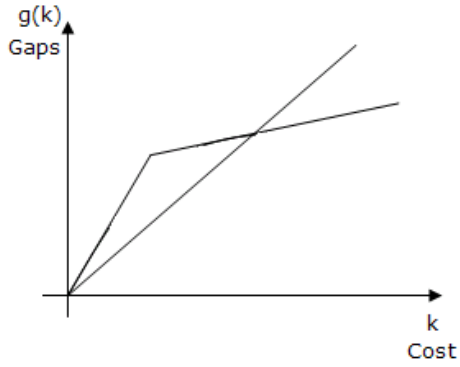
Das Problem bei genauer Berechnung der Gap-Kosten liegt im höheren Aufwand je länger die Sequenzen werden. Je länger ein Gap wird, desto weiter müsste bis zu seinem Anfang zurückgerechnet werden, um die Kosten für diesen immer länger werdenden Gap anzupassen.

Um zu Wert x zu erhalten, müssen alle vertikalen und horizontalen Werte berechnet werden (nicht mehr nur die drei umliegenden). Kosten: $O(n \cdot m \cdot (n+m))$. Die Kosten sind bei gleicher Sequenzlänge von $O(n^2)$ auf $O(n^3)$ gestiegen.

Je länger Gaps werden, desto seltener treten sie auf. Wenn Deletionen auftreten, dann betreffen sie allerdings keine kurzen Bereiche sondern über längere Strecken.
Ein lineares Gapmodell überschätzt lange und unterschätzt kurze Gaps. Eine Deletion der Länge 2 kann nicht dieselbe Wahrscheinlichkeit haben wie zwei Deletionen der Länge 1.

Affine Gap-Kosten:

Aus diesem Grund wird mit affinen Gap-Kosten gearbeitet.

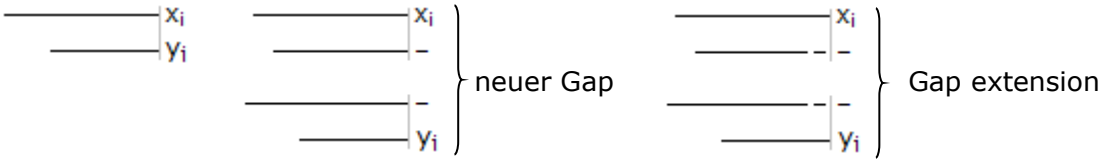


Man rechnet mit stückweis-linearer Funktion.

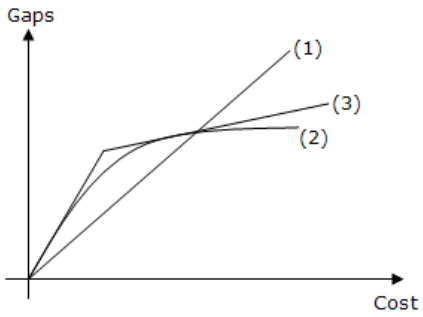
$g(k) = g_0 + g_{ext} < k-1$

g_0 … Gap open penalty "12"
g_{ext} … Gap extension "2"

Demnach ist die Öffnung eines Gaps teuer, eine Verlängerung ist aber günstiger als zwei kurze Gaps die eine ähnliche Länge ergeben.



05.11.2008

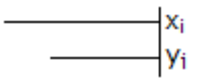


(1) … Needleman-Wunsch
Unser besserer Kostengap (2) … wird teuer (kubisch) in den Berechnungskosten
Kompromiss (3) ergibt sich durch lineare Schritte des Gap opening und Gap extension.
Dadurch muss nicht bis zu dem Punkt zurückgerechnet werden, an dem der Gap begann, es muss nur der Schritt zuvor darauf geprüft werde, ob es sich um ein Gap opening oder eine Gap extension handelt.

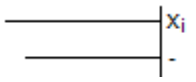
Warum werden überhaupt affine Gap-Kosten benötigt?
Lineare Gap-Kosten spiegeln biologische Gegebenheiten nicht wieder. Demgegenüber sind noch kompliziertere Gap-Kosten sind im Vergleich mit den verbundenen Ergebnissen dafür zu teuer.

Gotoh Algorithmus zur Berechnung affiner Gap-Kosten

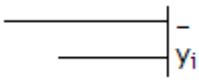
Bisher war S_{ij} der beste Score des Alignments der Sequenz $x_1 \dots x_i$ mit Sequenz $y_1 \dots y_j$.



M_{ij} … bester Score für Alignment $x_1 \dots x_i$ mit $y_1 \dots y_j$, das auf jeden Fall mit einem Match endet.

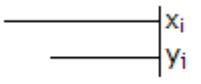


E_{ij} … bester Score für Alignment $x_1 \dots x_i$ mit $y_1 \dots y_j$, das mit einem Gap in y endet.

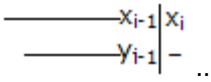


F_{ij} … bester Score für Alignment $x_1 \dots x_i$ mit $y_1 \dots y_j$, das mit einem Gap in x endet.

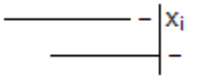
Daraus ergibt sich:
$S_{ij} = \max \{M_{ij}, E_{ij}, F_{ij}\}$ … benötigt Rekursion von allen drei Werten.



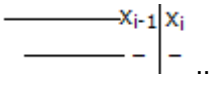
… $M_{ij} = S_{ij} + \sigma(x_i, y_j)$



… $E_{ij} = M_{i-1,j-1} + gap^{open}$



… $E_{ij} = F_{i-1,j} + gap^{open}$

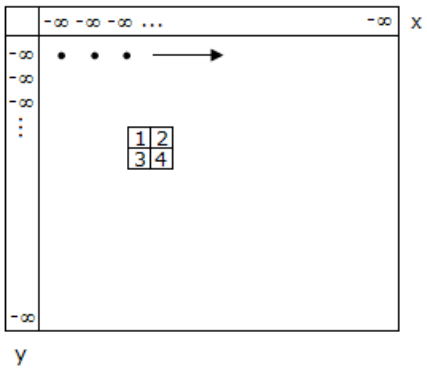


… $E_{ij} = E_{i-1,j} + gap^{ext}$

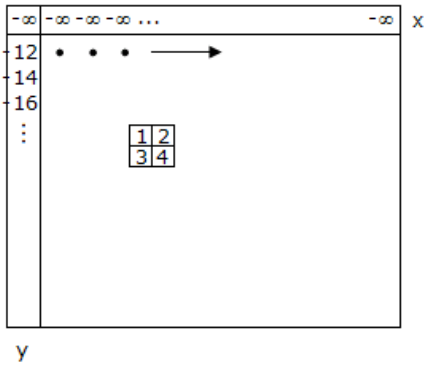
$$E_{i,j} = \max \begin{cases} M_{i-1,j} + gap^{open} \\ F_{i-1,j} + gap^{open} \\ E_{i-1,j} + gap^{ext} \end{cases}$$

$$F_{i,j} = \max \begin{cases} M_{i,j-1} + gap^{open} \\ F_{i,j-1} + gap^{ext} \\ E_{i,j-1} + gap^{open} \end{cases}$$

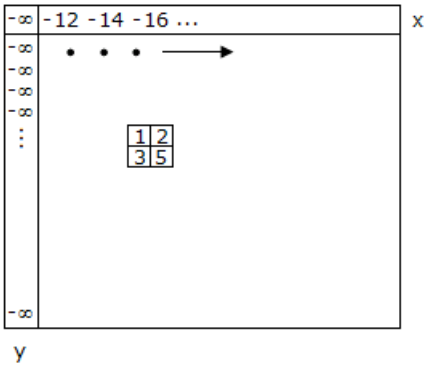
Wie teuer ist dieser Algorithmus
z.B. Gap open: -12
Gap extension: -2
Es müssen drei Matrizen initialisiert werden.



$M_{i,0} = M_{0,j}$ ist nicht erlaubt,
$M_{i,0} = M_{0,j} = -\infty$, i,j>0
$M_{0,0} = 0$



$E_{i,0} = gap^{open} + (i-1)\, gap^{ext}$, i>0
$E_{0,j} = -\infty$

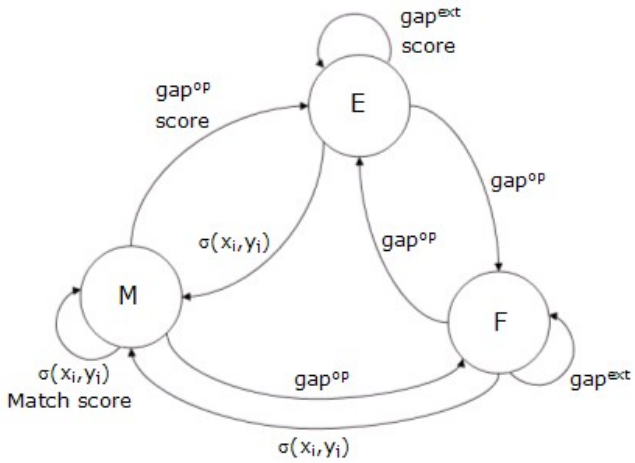


$F_{0,j} = gap^{open} + (j+1)\, gap^{ext}$, j>0
$F_{i,0} = -\infty$

Speicheraufwand … drei Matrizen mit O(n m)
Rechenaufwand … es müssen in allen drei Matrizen gleichzeitig die jeweiligen Bereiche berechnet werden, da die nächsten Werte in allen drei Matrizen voneinander abhängen. Um irgendwo (2) zu berechnen, wird überall (1) benötigt. Um (5) zu berechnen, wird überall (3) benötigt, um (4) zu berechnen, wird überall (1) benötigt usw.
Der Aufwand entspricht Needleman-Wunsch um den Faktor 3 erhöht.

Der Algorithmus ist auch auf lokale Alignments mittles Smith-Waterman anwendbar. Es wird der größte Wert in der Matrix gesucht und Backtracing durchgeführt, bis man auf 0 stößt. Das Backtracing ist wegen der drei Matrizen etwas komplizierter; der Pfad springt von einer Matrix zur nächsten.

Final state automaton:



Ein Final state automaton ist lediglich eine Art der Darstellung von Zuständen (states) und Regeln für Übergänge (transitions), die diese States ineinander überführen können.

Hier das final state automaton zum Gotoh Algorithmus.

Bsp:

```
A T T C G - T
A – - C – T T
M→E→E→M→etc.
1 2 3 4
```

(1)Gap open (2) Gap extension (3) Match (4) etc.

Der oben gekennzeichnete Teil der Sequenz kann auf zwei Arten betrachtet werden:

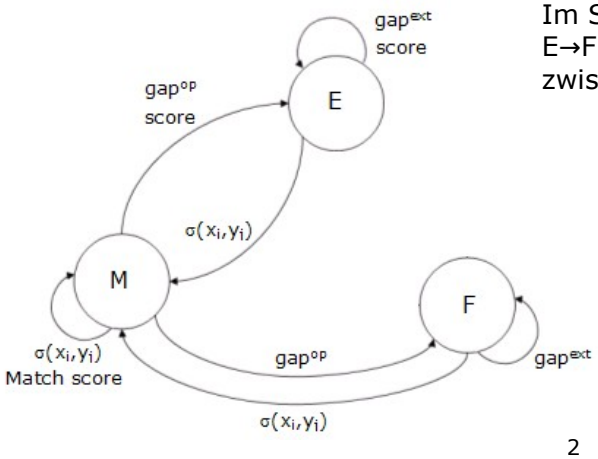
```
(1)G –  und  (2)– G
   – T          T –
```

Es gibt in diesem Teil der Sequenz eigentlich keinen Unterschied zwischen (1) und (2) in Aussage oder Bewertung des Alignments der Sequenz.
Aus diesem Grund wird z.B. der Übergang von F zu E im State Diagramm wird verboten. Dadurch gibt's nur die Version Gap oben gefolgt von Gap unten (2), aber nicht mehr die Möglichkeit Gap unten gefolgt von Gap oben (1).
Daraus folgt aus E_{ij} = max {etc. wird der F-Term, aus F_{ij} = max {etc. der E-Term entfernt. Dadurch wird jeweils ein Rechenschritt gespart. Ausgezeichnet. But wait, there's more!

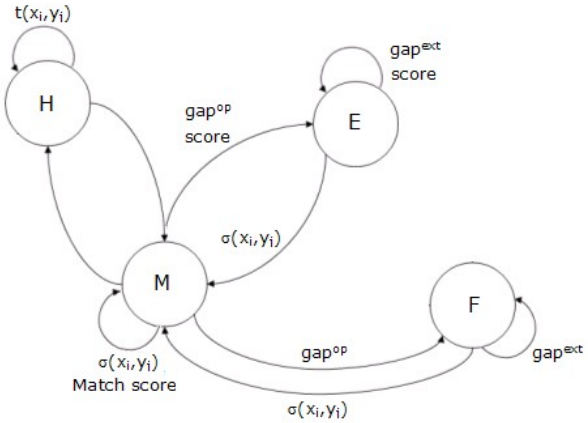
Um das Ganze noch besser zu machen wird folgende Regel eingeführt:
Score $2g^{ext} < \sigma(a,b)$

(1) `GG--` liefert einen schlechteren Score als
`--TT`
(2) `GG`
`TT`

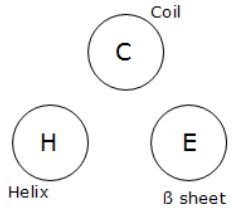
Dadurch kommt (1) im Vergleich mit (2) einfach nicht mehr vor.



Im State Diagramm fällt dann auch der Fall E→F weg, es gibt dann keine Verbindung mehr zwischen E und F.

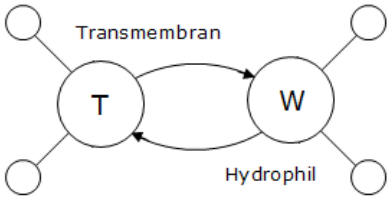


Es wird ein zusätzlicher State einführt: High Fidelity state. In diesem State können keine Gaps entstehen. In einem solchen State könnten z.B. andere Matchscores als in State M eingeführt werden.
Solch ein State könnte eine hoch konservierte Region abbilden, in der Punktmutationen seltener vorkommen und daher teurer bewertet werden.



z.B. eigene Matrizen für Helices, ß-Sheets, coiled coils statt normaler BLOSUM Matrix.

z.B. Membranprotein: wenn in einer α-Helix plötzlich huch, wer bist denn du ein Prolin sitzt, werden die Kosten für einen Mismatch viel größer sein, als bei einer anderen hydrophoben AS an dieser Stelle.



Unter Verwendung solcher Bewertungsmatrizen kann man weiters Membran- und extrazelluläre bzw. intrazelluläre Domänen vorhersagen.

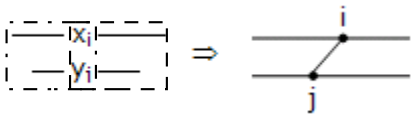
Alignment nach Vingron und Argos

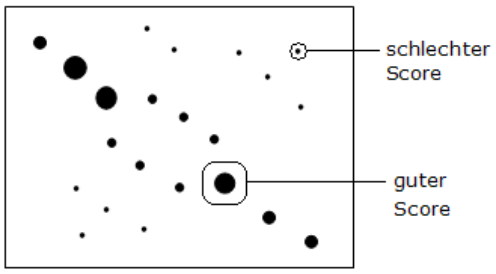
Üblicherweise erhält man ja nicht nur ein Alignment, ein Sequenzvergleich liefert meist vom Score her mehrere gute Alignments, die ebenfalls für Auswertungen interessant sind z.B. bei Homologievergleichen.

Bisher wurden mittels Gotoh berechnet:
M_{ij}, E_{ij}, F_{ij} der Teile $x_1 \dots x_i$ und $y_1 \dots y_j$ der Sequenzen x (Länge n) und y (Länge m).

Zusätzlich wird jetzt berechnet:
$\hat{M}_{ij}$, $\hat{E}_{ij}$, $\hat{F}_{ij}$ der Teile $x_i \dots x_n$ und $y_j \dots y_m$ der Sequenzen x und y, also das bestmögliche Alignment über die Mitte der Sequenz bis zu ihrem Ende.

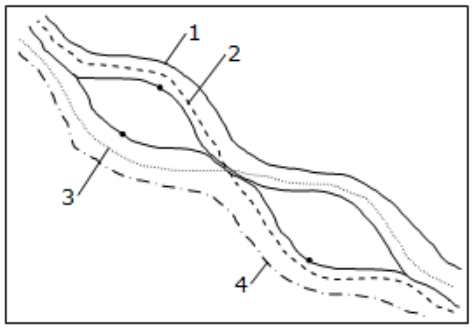
$M_{ij} + \hat{M}_{ij} - \sigma(x_i,y_j)$ ... der Match score für x_i und y_j wird doppelt gerechnet,
daher $(-\sigma(x_i,y_j))$





Es wird der beste Score unter der Voraussetzung berechnet, dass i und j ein Match bilden. Die Scores werden graphisch ausgegeben, wobei die Markierung desto ausgeprägter wird, desto besser der Score in der Matrix ist.

Dadurch wird bereits optisch sichtbar, wo gute und wo schlechte Alignments liegen. Im Beispiel finden sich zwei gleich gute Alignments, beide sind interessant …
Backtracing für linken Teil in M, E, F,
Backtracing für rechten Teil der Sequenz in $\hat{M}$, $\hat{E}$, $\hat{F}$.



Es wird immer nur ein Wert im Alignment festgelegt. Je nachdem in welchem Punkt man das Alignment festlegt, (1,2,3) erhält man drei unterschiedliche Alignments. Alignment 4 erhält man nie, da das obere Alignment immer besser sein wird. Dazu bräuchte man zwei Punkte in der unteren unsicheren Regionen.

*

Jedes Alignment liefert Scores, die Scores wurden von Wahrscheinlichkeiten abgeleitet.

… $p(A) \sim e^{\beta S(A)}$ p(A) … Wahrscheinlichkeit Alignment, S(A) … Score Alignment, ß … Skalierung der Scores

$$S(a,b) = c \cdot \log \frac{p_{a,b}}{q_a \cdot q_b} \Rightarrow ß = \frac{1}{c}$$

Dadurch kann jedem Alignment eine Wahrscheinlichkeit zugeordnet werden.

$z = \sum_{\text{Alignm.}} e^{\beta S(A)} \Rightarrow p(A) = \frac{1}{z} e^{\beta S(A)}$ … Zustandssumme aller Alignments

Das ist extrem unpraktikabel, weil man ja viele Alignments erhält.

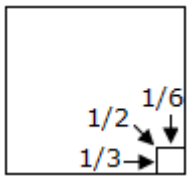
$z_{ij} = z_{i-1,j-1} \cdot e^{\beta\sigma(x_i, y_i)} + z_{i-1,j} \cdot e^{\beta_{gap}} + z_{i,j-1} \cdot e^{\beta_{gap}}$ … Zustandssummenvariante des Needleman-Wunsch Algorithmus für lineare Gap-Kosten.

Eine Zustandssumme mit obligatem Match

z^M äquivalent zu Matrix M

$\hat{z}^M$ äquivalent zu Matrix $\hat{M}$

$$P_{ij} = \frac{z_{ij}^{M} \cdot \hat{z}_{ij}^{M} / e^{\beta\sigma(x_i, y_i)}}{z}$$



Stochastisches Backtracing:

$p(A) = \frac{1}{z} e^{\beta S(A)}$

Man erhält Wahrscheinlichkeiten für Backtracing, gute Alignments sind viel Wahrscheinlicher als schlechte.

MEA (Maximum expected accuracy) Algorithmus

Dieser Algorithmus wird z.B. vom Programm probcons verwendet.
Es wird zunächst die Genauigkeit (wahrscheinliche Anzahl der Matches durch die Länge der kürzeren Sequenz) berechnet und anschließend der Pfad durch die Matrix mit der höchsten Summe ermittelt. Dieser Pfad entspricht dem besten Alignment, es werden keine Gap-Kosten benötigt (Gap mit Needleman-Wunsch = 0).

12.11.2008*

```
GCATCGGC          k=2
CGATCGCCATCG
```

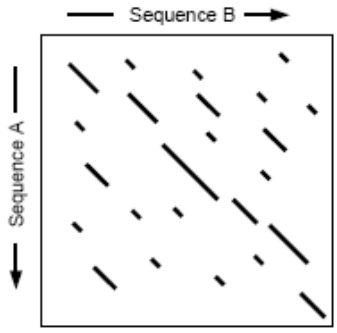
AT	3	3,9	0,6
CA	2	4,8	0,6
CC	–	1	0,6
CG	5	5,11	5,–1
GC	1,7	6	0,6
GG	6	–	
TC	4	4,10	

Auf Diagonale 6 … 4 2er Tuppel

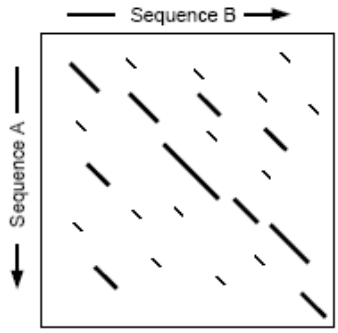
Liste aufstellen dauert so lange wie Länge der Sequenz

FASTA Algorithmus

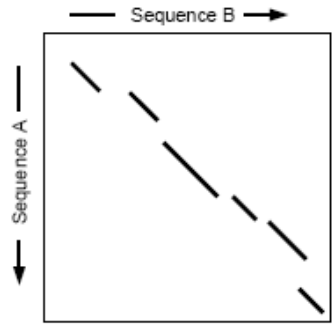
Beim FASTA Algorithmus handelt es sich um eine Heuristik ("mit begrenztem Wissen und wenig Zeit zu guten Lösungen zu kommen"), die eine Sequenz gegen Sequenzen in einer Datenbank in 4 Schritten mittels lokalen Alignments vergleicht.



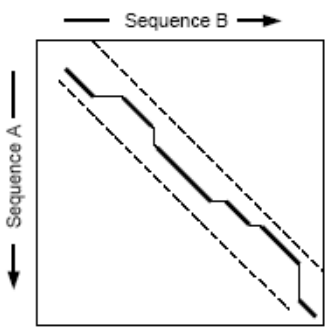
1) Zuerst werden bestimmte Sequenzen (common key words) aus einer Tabelle ausgelesen und geprüft ob diese exakt in der zu untersuchenden Sequenz enthalten sind. Die Länge dieser Sequenzen kann zu Beginn definiert werden.
Die erhaltenen Regionen werden über eine Scoring Matrix (z.B. BLOSUM50) bewertet und die besten 10 ermittelten Regionen mittels Dot Plot ausgegeben.



2) Die in Schritt 1) erhaltenen Sequenzen werden erneut mit kürzeren key words berechnet und mit einer Scoring Matrix (z.B. PAM) bewertet, um ihre Scores besser unterscheiden zu können.



3) Es werden alle gefundenen Regionen entfernt, deren Score einen bestimmten Wert nicht überschreitet. Anschließend wird geprüft, ob die verbleibenden Regionen zu einem Alignment zusammengefügt werden können. Falls nötigt werden zwischen Regionen Gaps eingeführt.



4) Der Score des erhaltenen Alignments wird mittels Smith-Waterman ermittelt.

Bilder von:
http://en.wikipedia.org/wiki/FASTA

Schlüssel des Algorithmus ist die anfängliche Suche nach Key Words (k). Die Suche dauert je länger, je mehr Key Words in der zur Verfügung stehenden Datenbank vorhanden sind.
Je länger die für die Suche definierten Key words sein sollen, desto unwahrscheinlicher wird ein Match auf die Sequenz, wenn keine sehr hoch konservierte Region enthalten ist, dafür wird die Suche umso schneller, je länger die Key words angenommen werden.
Für Proteine werden üblicherweise Werte von k=2, für Nukleotide k=4-6 verwendet.

FASTA ist die älteste Heuristik und durch das kleine Alphabet für Nukleotidalignments besser geeignet, da bei AS kompliziertere Scoring Matrizen verwendet werden müssen.

Weiters ist im Gegensatz zu Nukleotidalignments nicht nur die Ähnlichkeit der Sequenzen sondern auch der tatsächliche Austausch der AS von Interesse, da sie bei der Bildung eines sinnvollen Alignments eine Rolle spielen. Es wird daher bereits im Zuge des Vergleichs mit den Key words ein Score benötigt.
Es wird demnach für sinnvolle Proteinalignments ein anderer Algorithmus benötigt.

BLAST (Basic Local Alignment Search Tool)

Auch BLAST vergleicht eine Sequenz gegen Sequenzen einer vorhandenen Datenbank. BLAST baut darauf auf, dass so genannte high-scoring segment pairs (HSP) in jedem sinnvollen Alignment zweier Sequenzen zu finden sind. Um diese HSPs in den verglichenen Sequenzen zu ermitteln, wird eine Heuristik verwendet, die ähnlich dem Smith-Waterman Algorithmus funktioniert und zwar weniger genau aber dafür ein vielfaches schneller ist.

1) Zuerst werden über bestimmte Algorithmen bekannte nicht sinnvolle Regionen (z.B. Tandem repeats) maskiert.

2) Anschließend wird aus der Query-Sequenz eine Liste von k-words erstellt.
Bsp.: Query Sequenz MRDPYNKLIS mit k=3 ergibt folgende k-words:
MRD, RDP, DPY, PYN, NKL, KLI, LIS

3) Jedes Element der k-words wird über eine Scoring Matrix mit jedem möglichen 3-Buchstabenwort verglichen und deren Score ermittelt:

	PYN	PYN	PYN	…
	PYN	PFN	PFQ	…
Score:	20	16	10	…

Es werden nur die 3 Buchstabenworte behalten, deren Score nicht unter einen gewissen Wert fällt ("high-scoring words"), diese Worte werden in eine weitere Liste der eingetragen.

4) Die high-scoring words werden jetzt mit den Sequenzen in der Datenbank verglichen. Wird ein exakter Match gefunden, wird diese Sequenz markiert. Das gefundene high-scoring word in der Datenbanksequenz wird jetzt mit dem zugehörigen k-word in der Query Sequenz zu einem Match gebracht.

Query Sequence	MRDPYNKLIS
Database	SKAPFNPLAS

Es wird eine weitere Liste erstellt, die die Position jedes gefundenen high-scoring words (z.B. {PYN 3233, 33322323, 32323} {PFN 10123, 2567, 3358})in der Datenbank zu einem k-words (z.B. PYN) enthält.

5) Die gefundenen Paare werden jetzt hergenommen und ihre Umgebung (z.B. +/- 2 AS) mittels einer Scoring Matrix (z.B. BLOSUM) bewertet:

```
Query Sequence    M R D P Y N K L I S
Database 1        M H E P Y N D V P W
                  5 0 2   20 -1-1-3-3     … Score von 20

Query Sequence    M R D P Y N K L I S
Database 2        M H D P F N K V P W
                  5 0 7   16  7-1-3-3     … Score von 39
```

Die Datenbanksequenz 2 liefert demnach einen besseren Score als Datenbanksequenz 1, weil die Umgebung mehr oder bessere Match Scores liefert als die direkte Übereinstimmung des gefundenen Wortes.
Auf diese Art und Weise werden HSPs erstellt. Es werden nur Sequenzen behalten, deren HSPs einen gewissen Score überschreiten.

6) Finden sich zwei HSPs in unmittelbarer Nähe, können diese zusammengezogen werden.

7) Abschließend wird die statistische Signifikanz der gefundenen HSPs überprüft; d.h. eine Zuordnung der Alignmentwahrscheinlichkeit:

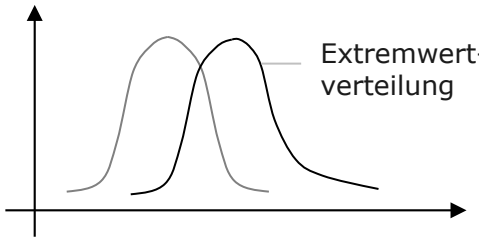
$$\sigma(a,b) = \log \frac{P_{a,b}}{q_a q_b}$$

Würde man eine Zufallssequenz gegen die Datenbank suchen, wäre der zuvor erhaltene Score in Bezug dazu besser oder schlechter?

```
Query Sequence    M R D P Y N K L I S
Database 2        M H D P F N K V P W
                  5 0 7   16  7-1-3-3     … Score von 39
```

$$\underset{\text{erwartete Score}}{\langle\sigma\rangle} = \sum_{a,b} q_a q_b \log \frac{P_{a,b}}{q_a q_b}$$

Was für einen Score würde man erwarten, wenn zwei zufällige Sequenzen verglichen werden würden?



Eine Summe an Zufallszahlen liefert eine Gaußverteilung
Sucht man den besten Score aus allen möglichen Alignments erhält man eine Extremwertverteilung.
Die Verteilung verschiebt sich zu höheren Werten und wird unsymmetrisch.

Die Wahrscheinlichkeit einen Score S zu erhalten der größer oder gleich einem Wert x ist ergibt sich durch folgende Formel:

$P(s \geq x) = 1 - \exp(-e^{-\lambda(x-\mu)}) = 1 - \exp(-K\, m\, n\, e^{-\lambda x})$ … wird als P-Value ausgegeben.

$$\mu = \frac{\log(Kmn)}{\lambda}$$

λ, K … abhängig von Score matrix, Gap-Kosten, Buchstabenhäufigkeit in der Sequenz
m, n … Länge von Query- und Datenbanksequenz

E-Value (expect value): Anzahl wie oft ein solcher P-Value bei einer zufälligen Suche auf die Datenbank erhalten werden würde. Dieser Wert ist dementsprechend proportional zur Größe der Querysequenz bzw. der Datenbanksequenz.

$P(s \geq x) = 1 - e^{-E(x)} \Rightarrow E(x) = K\ m\ n\ e^{-\lambda x}$ P- und E-Values können ineinander umgerechnet werden

Gehe von Ursprungsverteilung zu Extremwertverteilung.

$E << 1 \quad 1 - e^{-E} \approx 1 - (1 - E) = E$ … nach Taylorreihe

$s' = \frac{\lambda s - K}{\ln 2}$ s' … Bitscore, hängt von n und m ab, also von der Länge der Sequenzen … wird in Bit angegeben.

$E(s')=nm2^{-s}$ … E-Value hängt von Größe der Datenbank ab.

E-Values können nicht verglichen werden, da Datenbanken unterschiedlich groß sind und sich über die Jahre erweitern.
Scores können auch nicht direkt miteinander verglichen werden, da vielleicht unterschiedliche Score Matrizen verwendet wurden.
Der Bitscore ist nicht von Größe der Datenbank (n,m) abhängig, und normalisiert so die E-Values um Vergleiche möglich zu machen.

19.11.2008*

Multiple Alignments

$$A = \begin{cases} A_1 = \{a_{11}, a_{12}, \ldots, a_{1n}\} \\ \quad \ldots \\ A_R = \{a_{R1}, a_{R2}, \ldots, a_{Rn}\} \end{cases}$$

$a_{ij} \in A$
$A^* = A \cup \{-\}$ ... Vereinigung Alignment A mit Gaps

A* entspricht demnach einer Matrix A die alle Sequenzen enthält, wobei die Sequenzen so mit Gaps versetzt wurden, damit ein gesamtes Alignment über alle Sequenzen entstehen kann. Dabei darf keine Spalte des Alignments nur Gaps enthalten.

$$A^* = \begin{pmatrix} a_{11}\ * & & \\ & & \\ & & a_{Rl}\ * \end{pmatrix}$$ … A* Matrix R x l $\quad a_{ij}* \in A*$

$\max(n_1 \ldots n_R) \leq l \leq \sum n_i$ ... Summe über die Sequenzlänge

Werden die Gaps aus A_i* (einer Zeile der Matrix A*) wieder entfernt, erhält man die Ausgangssequenz A_i.

Bsp: Multiples Alignment der Worte "apple, pepper, paper"

$$\begin{pmatrix} - & a & p & p & l & e & - \\ p & a & p & - & - & e & r \\ p & e & p & p & - & e & r \end{pmatrix}$$ … Matrix A* mit Gaps, würden Gaps entfernt, würde man die Matrix A mit den ursprünglichen Worten apple, pepper, paper erhalten.

Ein paarweises Alignment folgt Regeln, wie eine Sequenz in die andere überführt werden kann.
Aus multiplen Alignments können Subalignments gelöst werden, die wie paarweise Alignments behandelt werden können.

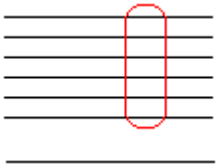
Die Subalignments können gleiche Wertigkeit haben:

```
pap-er          pa-per
pepper          pepper
```

Sie können untereinander auch nicht kompatibel sein:

```
-apple-     pa-per      -apple-
pap—er      pepper      pepp-er
```

Im Gegensatz dazu können aus multiplen Alignments mehr Informationen über Homologien ermittelt werden.
Dieser Umstand ergibt sich aus hoch konservierten Blöcken in den einzelnen Sequenzen, die nur wenig Gaps und eine hohe Übereinstimmung der Sequenzabfolge enthalten und daher immer untereinander stehen werden.



Vergleicht man zusätzlich eine Einzelsequenz mit dem zuvor erstellten multiplen Alignment, kann sehr schnell über die konservierte Region geprüft werden, ob die Sequenz mit den Sequenzen des multiplen Alignments verwandt ist oder nicht.

Multiple Alignments erlauben so phylogenetische Klassifizierung und Ordnung von verglichenen Sequenzen in Hierarchien z.B. Einteilung von Proteinsequenzen in Proteinfamilien.

Bewertung multipler Alignments durch Scores

$\sigma(a,b) = \log \frac{P_{a,b}}{q_a \cdot q_b}$ … herkömmlicher Substitution Score

$\sigma(a,b,c) = \log \frac{P_{a,b,c}}{q_a \cdot q_b \cdot q_c}$ … Substitution Score für den Vergleich von drei Sequenzen. Dabei ergibt sich aber das Problem, dass die Matrizen immer größer werden, je mehr Sequenzen verglichen werden.

Zur Bewertung multipler Alignments werden daher die einzelnen Spalten der verglichenen Sequenzen herangezogen, mit einem Score versehen und deren Summe gebildet.

$$S(A^*) = \sum_{i=1}^{l} s(a_{1i}^*, a_{2i}^*, \ldots, a_{Ri}^*)$$ … Score für Spalte i.

Um einer Spalte eines multiplen Alignments einen passenden Score zuzuordnen, wird ein Konsensusscore verwendet. Dabei wird der häufigste Buchstabe in einer Spalte als Referenz herangezogen.

$$c(A^*) = \sum_{j=1}^{l} \sum_{i=1}^{R} d(x_j, a_{ij}^*)$$ x_j … Consensussequenz

Dabei ergeben sich die Probleme, dass die Consensussequenz nicht genau definiert und aufwändig zu ermitteln ist.

Um diesen Problemen zu begegnen wird ein anderer Mechanismus verwendet.

Sum of pair score:

Das multiple Alignment wird in alle paarweisen Alignments zerlegt, diese werden gewertet und die erhaltenen Scores addiert.

$$S_{Sp}(A^*) = \sum_{1\le p<q\le r} S(A_p^*, A_q^*) = \sum_{j=1}^{l} \sum_{1\le p<q\le r} s(a_{pj}^*, a_{qj}^*)$$

l … Länge des Alignments
a_{pj}* … Position Zeile p Spalte j

$$s(a,b) = \begin{cases} \sigma(a,b) & \text{falls } a \neq gap \wedge b \neq gap \\ -g & \text{falls } a \vee b = gap \\ 0 & \text{falls } a = b = gap \end{cases}$$

… einzusetzende Scores

Weitere Möglichkeiten Sum of pair score:

1)
Spalte eines multiplen Alignments mittels BLOSUM62 bewerten:

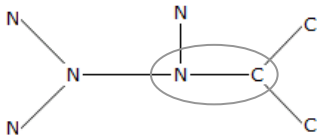
```
N
N
N
C
C
```

BLOSUM62 Score:

```
N – N =  6
N – C = -3
C – C =  9
```

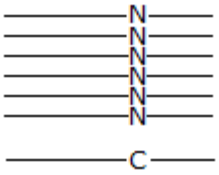
Score des Alignments:
1σ (C,C)+6σ(N,C) +3σ(N,N) = 9-18+18 = 9

2)
Eine weitere, sehr elegante Methode kann angewendet werden, wenn der phylogenetische Baum einer Sequenz bekannt ist – worin auch gleich der Nachteil liegt…



Nur an der markierten Position ist eine Mutation aufgetreten. Zusätzlich gibt es weiters 4 Zeitabschnitte, an denen N und 2 Zeitabschnitte, an denen C gleich geblieben sind.

Daraus kann wieder mittels einer passenden Matrize der Score ermittelt werden:
2σ(C,C) + 4σ(N,N)+1σ(N,C) = 18+24-3 = 39



Bei der Bewertung dieses Alignments ergibt sich das Problem, dass bei der bisherigen Bewertung die eine Sequenz mit C keinen großen Einfluss auf den Score haben wird:

$$\frac{n(n-1)}{2}\sigma(N,N) + n\sigma(N-C)$$

Dieser Fall kann auftreten, wenn z.B. Sequenzen zwischen Organismen verglichen werden, von Organismus A aber um ein vielfaches mehr Sequenzen vorhanden sind als von Organismus B.

Weighted sum of pair score

Aus diesem Grund wird jeder Sequenz des Alignments zusätzlich entsprechend ihres Vorkommens ein Gewicht zugeordnet. Dadurch können zwei Sequenzen auch zwei mal im selben multiplen Alignment vorkommen ohne die Bewertung überproportional zu verfälschen.

$$S_{SP}(A^*) = \sum_{1\le p<q\le r} w_p w_q s(A_p^*, A_q^*)$$

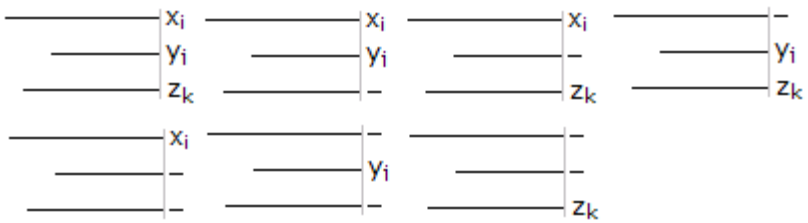
$w_p w_q$ … Faktoren für Gewichtung

Berechnung multipler Alignments:

Es sollen drei Sequenzen miteinander verglichen werden.

$x_1 \dots x_n$ x
$y_1 \dots y_m$ y
$z_1 \dots z_p$ z

Klassischer Needleman-Wunsch Algorithmus auf drei Sequenzen erweitert:



7 Möglichkeiten

$$S_{i,j,k} = \begin{cases} x_1 \dots x_i \\ y_1 \dots y_j \\ z_1 \dots z_k \end{cases}$$

Im Fall von r Sequenzen erhält man 2^r-1 Möglichkeiten, da der Fall, dass nur Gaps vorhanden sind ausgeschlossen wird.

$$S_{i,j,k} = \begin{cases} S_{i-1,j-1,k-1} + S(x_i, y_i, z_i) \\ S_{i-1,j-1,k} + S(x_i, y_i, -) \\ S_{i-1,j,k-1} + S(x_i, -, z_i) \\ S_{i,j-1,k-1} + S(-, y_i, z_i) \\ S_{i-1,j,k} + S(x_i, -, -) \\ S_{i,j-1,k} + S(-, y_i, -) \\ S_{i,j,k-1} + S(-, -, z_i) \end{cases}$$

Man erhält eine dreidimensionale Matrix. Dadurch ergeben sich kubische Kosten für Speicher und Rechenzeit.

Rechenhzeit … $O(N^3)$
Speicher … $O(N^3)$

Für r Sequenzen ergeben sich folgende Kosten:

Rechenzeit … $O(n^r 2^r)$
Speicher … $O(n^r)$

Werden die Sequenzen verlängert, hat das nur wenig Auswirkung auf die Kosten, wird die Anzahl der Sequenzen erhöht, steigen die Kosten exponentiell. Es gibt daher keinen sauberen Algorithmus, es muss auf Heuristiken zurückgegriffen werden.

26.11.2008*

$S^*(A_p,A_q)$ … best möglicher Score für A*
A* … best möglicher Score für A

$$S(A^*) \le \sum_{p,q} S^*(A_p, A_q)$$

$$S(A^*) = \sum_{p,q} S(A_{p,q}{}^*) = S(A_{u,v}{}^*) + \sum_{p<q} S(A_{p,q}{}^*) \le S^*(A_r, A_u) + \sum_{\substack{p<q \\ (p,q)\neq(u,v)}} S(A_{p,q}{}^*)$$

… multiples Alignment

$\sum_{\substack{p<q \\ (p,q)\neq(u,v)}} S(A_{p,q}{}^*)$ … Summe der paarweise Alignments

Carillo-Lipman-Begrenzung

Die Berechnung nach Carillo-Lipman beruht darauf, dass über die Berechnung des Scores des multiplen Alignments sowie der Berechnung der Summe der Scores jeweils des besten Pfades aller paarweisen Alignments der Sequenzen ermittelt werden.
Der Score des multiplen Alignments ergibt eine obere Schranke, der Gesamtscore der paarweisen Alignments eine untere Schranke. Wird der Score der unteren Schranke vom Score der oberen Schranke abgezogen, erhält man einen Wert.
Es werden jetzt wieder die paarweisen Alignments hergenommen und der jeweilige Pfad des besten Alignments betrachtet. Es werden rund um den Pfad nur jede Positionen weiter betrachtet, die maximal um den oben betrachteten Wert verschoben sind, alle anderen Positionen werden bei der weiteren Berechnung des multiplen Alignments ignoriert.

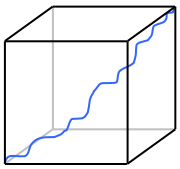
Bsp: Es werden drei Sequenzen miteinander verglichen.
Berechnung Score multiples Alignment liefert U = 18
Die Berechnung des Scores der besten paarweisen Alignments der drei Sequenzen liefert die Werte 5,6,5, was einen Gesamtscore von L=16 ergibt.
Die ermittelte Schranke ist demnach U-L=2.

Matrix Vergleich Sequenz 1 mit Sequenz 2:

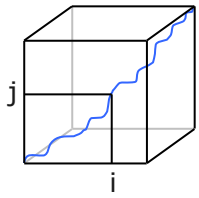
```
0 1 2 3 4 5
1 1 2 3 4 5
2 2 2 3 4 5
3 3 2 3 4 5
4 4 3 3 3 4
5 5 4 3 4 4
6 6 5 4 4 5
```

Matrix mit bestem Pfad +jeweils 2 Positionen, alle anderen Positionen werden nicht mehr beachtet:

```
  1 2 * * *
1 1 2 3 * *
2 2 2 3 * *
3 3 2 3 4 *
* 4 3 3 3 4
* * 4 3 4 4
* * * 4 4 5
```



Bei einem Alignment von drei Sequenzen entspricht der Pfad durch die Matrizen einer Wanderung durch einen Würfel von einer Ecke zur anderen. Dabei kommen keine Gaps vor.
Der Pfad durch den Würfel kann auf die Wände projiziert werden.



$S_{ij}^*(A_u,A_v)$ … Best möglicher Score für das Alignment der Sequenzen u,v die durch Punkt (i,j) geht (Punkt i,j=Projektion des Pfades auf der Wand).

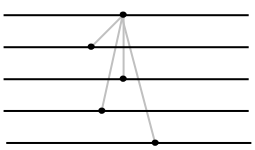
$$S(A^{*ij}) \le S_{ij}^*(A_u, A_v) + \sum_{\substack{p<q \\ (p,q)\neq(u,v)}} S(A_{p,q}^{*ij})$$

$$B \le S_{ij}^*(A_u, A_v) + \sum_{\substack{p<q \\ (p,q)\neq(u,v)}} S^*(A_p, A_q)$$

B … ist eine untere Schranke für zugelassene Scores.

Durch gegebene Schranken ergeben sich eben Flächen, die nicht berechnet werden müssen.
Der Aufwand verringert sich aber nur, wenn es sich um ein gutes Alignment handelt und große Teile nicht beachtet werden. Bei schlechten Alignments liefert die Schranke einen zu großen Wert und es können nur kleine Teile weggelassen werden.

Star Alignment



Bei Star alignments wird über paarweise Alignments zunächst die Sequenz aus dem multiplen Alignment ermittelt, das die größte Ähnlichkeit mit allen anderen Sequenzen besitzt. Diese Sequenz wird als "center of star" verwendet d.h. alle Sequenzen werden bei der Anordnung im multiplen Alignment an diesem "center of star" durch Einführung von Gaps ausgerichtet.

(1) Suche die zentrale Sequenz C, sodass $\sum_j D(A_i, A_j)$ minimal wird. D … Abstand zu allen anderen Sequenzen.

Bsp: A = (CGCTTTA,ACGTT,GCTAG)

1) paarweise Alignments, erste Sequenz wird als zentrale Sequenz verwendet

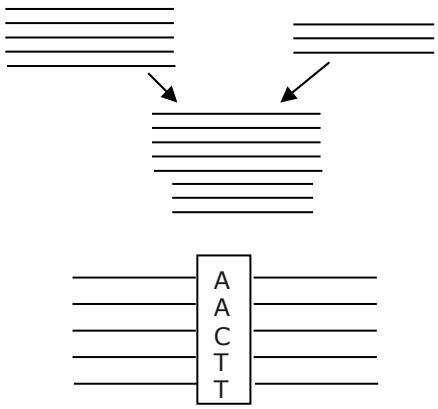
```
-CGCTTTA        CGCTTTA-
ACG--TT-        -GC--TAG
```

2) Multiples Alignment an zentraler Sequenz ausgerichtet erstellen

```
-CGCTTTA-
ACG--TT--
--GC--TAG
```

$D(A_C^*) \leq 2D(A)$ $D(A_C^*)$ … Distanz des Star Alignments
D(A) … bestmögliches Alignment

Progressive Alignments



Bei progressive Alignments handelt es sich um eine Heuristik, die mehrere multiple Alignments erzeugt und diese Schritt für Schritt vereint. Dabei werden zu Beginn paarweise Alignments erstellt und diese Alignments Schrittweise ihrem Verwandtschaftsverhältnis nach (von nah verwandt zu fern verwandt) zu größeren Alignments zusammengefasst.
Dazu muss im ersten Schritt eine Baumstruktur ("guide tree") erstellt werden, anhand dessen im zweiten Schritt das multiple Alignment zusammengestellt wird.

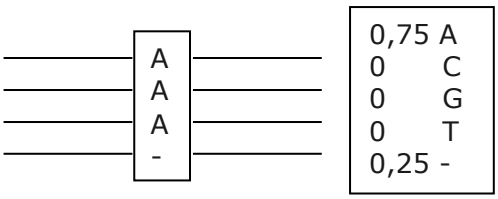
Die Berechnung des Sum of pair Scores ist bei hoher Sequenzanzahl sehr aufwändig. Ist die Häufigkeit von des verwendeten Alphabets der Sequenzen (z.B. A,C,T,G) bekannt, kann der Sum of pair Score einfacher errechnet werden.
Zu diesem Zweck werden Profile erstellt, die die Häufigkeit der Buchstaben enthalten.

0,4	A
0,2	C
0	G
0,4	T
0	-

Anteilhäufigkeit (Summe = 1).

Das Alignment des Alignments wird mit Profile ausgerechnet.

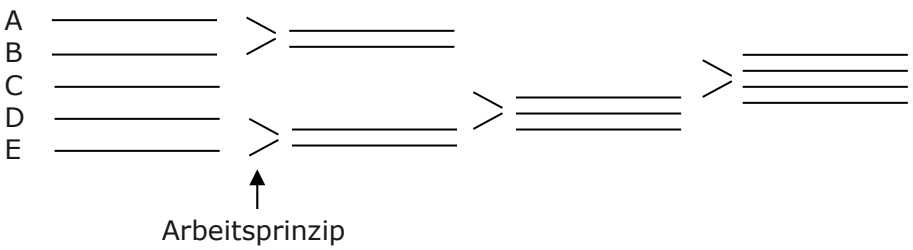
```
-CGCTTTA-
ACG-TT---
--GC--TAG
```



$A = \{A_i, \ldots, A_r\}$
$C = \{\{A_1\},\{A_2\}, \ldots\}$ … Menge an Alignments

while $|c| \geq 2$
$c = c - \{A_p^*, A_q^*\}$

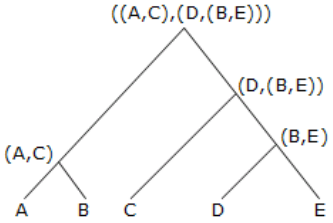
$A_S^* = \text{align}(A_p^*, A_q^*)$
$c = c \cup \{A_S^*\}$



Die Zahl der Schritte die für die Erstellung des Alignments nötig sind entspricht der Anzahl der Sequenzen -1.

Durch diese Vorgangsweise ergibt sich je nach Reihenfolge der Alignments ein unterschiedliches Endergebnis. Durch Neuerstellung des Alignments könnte so der Wert des Alignments verbessert oder verschlechtert werden.

Prinzip des progressive pairwise Alignments:



(1) Berechne alle paarweisen Alignments, daraus Distanzen zwischen den Sequenzen (D_{pq}).
(2) Berechne den Guide tree aus den ermittelten Distanzen.
(3) Ermittle aus dem Baum das progressive Alignment $O(r^2N^2+rN^2)$

Im Detail:
(1) Berechen alle paarweisen Alignments und die Distanzen zwischen ihnen (D_{pq}).
(2) Ermittle u,v für die minimale Distanz D_{uv}.
(3) Lösche u,v aus D.
(4) Berechne alle profile Alignments von S
(5) Distanzen zu S in Matrix D ein (→2)

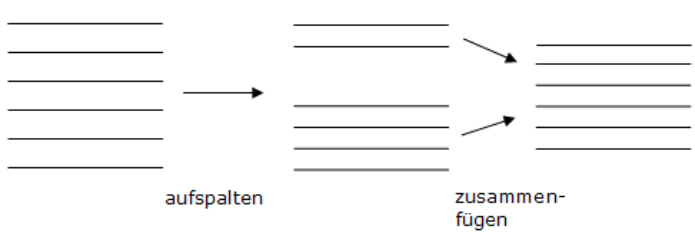
Iterative Methoden zu progressive pairwise aligments

(1) Guide tree Optimerung
- 1) Berechne alle pairwise alignments … D_{pq}
- 2) Berechne guide tree
- 3) Progressives Alignment
- 4) Berechne neuen guide tree aus multiplem Alignment
  - Falls der guide tree unterschiedlich ist, wird mit Schritt 3 fortgefahren.
  - Das neue Alignment wird acceptiert, wenn es besseren Sum-of-pair score hat als das vorhergehende

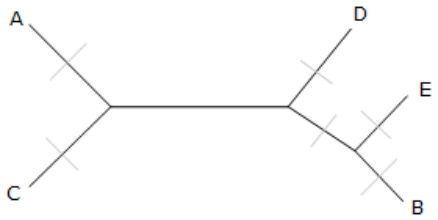
- Stop bei unverändertem quide tree, oder wenn das Alignment nicht besser wird

Optimierung des Alignments durch Interative Methoden

(2) Optimierung durch aufspalten und re-alinieren



Das Endalignment ist eventuell besser als das Ausgangsalignment. Dieser Vorgang wird so lange wiederholt, bis kein besseres Ergebnis mehr erzielt wird. $O(rn^2)$

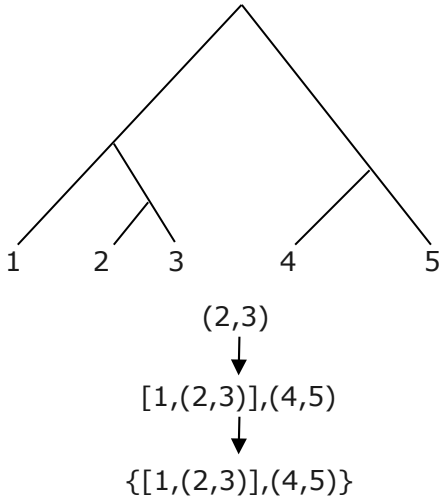


Gute Schnittstellen sind die Äste des Baumes.

- Teile einen Ast um zwei Subalignments zu erhalten.
- Die beiden Subalignments werden zu einem neuen Alignment zusammengefasst. Das neue Alignment wird akzeptiert, wenn es einen besseren Wert liefert.

03.12.2008*

Der Nachteil progressiver Alignments liegt darin, dass im paarweisen Alignment gesetzte Gaps im multiplen Alignment erhalten bleiben.
Der Vorteil liegt in der Geschwindigkeit $O(rn^2)$.



1) Paarweise Alignments erstellen $O(r^2N^2)$
2) Guide tree erzeugen
3) Progressive alignment über den Guide tree
4) Score Alignment

5) Ist der Score besser oder schlechter?
   a) Ist er schlechter zurück zu 2)
   b) Ist er besser Alignment in zwei Subgroups spalten und bei 3) Vorgang wiederholen um Subalignments zu verbessern.

   Bei Aufspalten in Subgroups können schlecht gesetzte Gaps entfernt werden.

T-Coffee (Tree-based Consistency Objective Function For alignment Evaluation)

Ein Problem bei paarweise globalen Alignments besteht darin, dass auch zufällige und nicht nur konservierte Sequenzen ein Match erzeugen können. Dadurch steigt zwar der Score des Alignments, die Bedeutung der konservierten Sequenzen sinkt aber dadurch

Die progressive Alignment Methode T-Coffee erzeugt lokale paarweise Alignments aus den einzelnen Sequenzen des multiplen Alignments. Die so erhaltenen Alignments erhalten einen Score und werden in eine Library geschrieben.
Diese Scores werden bei der Berechnung des Scores des progressiven Alignments beibehalten.

Ein Match in einem paarweisen Alignment bedingt damit bereits einen Match im multiplen Alignment und bestimmt damit den Score des multiplen Alignments mit.

ProbCons (Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences)

http://probcons.stanford.edu/

1) Im ersten Schritt wird die Wahrscheinlichkeit eines Matchs zwischen jeder der Sequenzen des multiplen Alignments ermittelt.

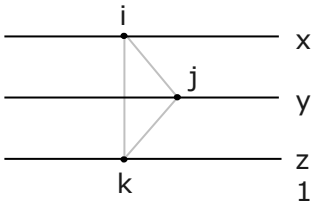
$p_{ij}^{xy} = P(x_i \sim y_i)$

2) Für jedes paarweise Alignment wird die Anzahl zu erwartender Matchs zwischen den Sequenzen E_a ermittelt.

3) Probabilistic consistency transformation:

$$P_{ij}^{xy} = \frac{1}{S} \sum_{z_S} \sum_{z_k} P_{ik}^{xz} P_{jk}^{yz}$$

Über diese Formel wird die Similarity der paarweisen Alignments in die Berechnung der Match-Scores mit einbezogen.

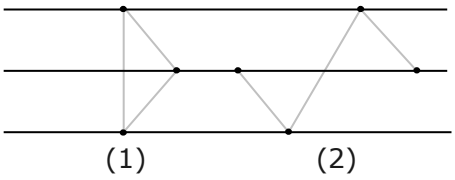


$P_{ik}^{xz} = \sum P_{ij}^{xy} P_{jk}^{yz}$ … P_{ij}^{xy} p_{jk}^{yz} … geben Schluss auf P_{ik}^{xz}

3) Erzeugung eines Guide Trees durch hierarchisches Clustering. Als Maß der Ähnlichkeit wird E_a aus Schritt 2) verwendet.

5) Progressives Alignment: Die Sequenzen werden nach dem Guide tree hierarchisch nach und nach aligned. Als Score wird Sum of pair score verwendet.

Aufwand:
1) $O(r^2N^2)$
2) $O(r^3n^3)$
3) $O(rN^2)$

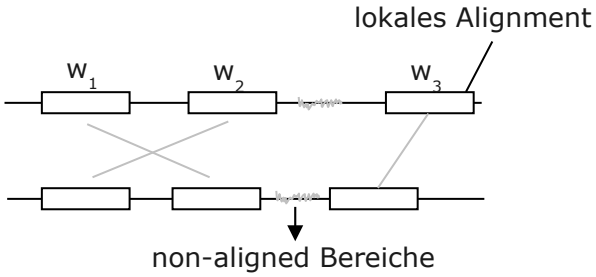


(1) besserer Score als (2)

DIALIGN (Blockalignment)

DIALIGN erzeugt Gapfreie Paare ähnlicher Segmente von verglichenen Sequenzen. Diese Segmente werden als Diagonale bezeichnet und erhalten einen durch die Ähnlichkeit der Segmente bestimmten Score.
Der Algorithmus versucht jetzt weiter ein Alignment zu erstellen, das einen möglichst hohen Score durch die einzelnen gefundenen Diagonalen liefert.

```
ACGacaAG
A-GcggCG
ACTt..C-
```

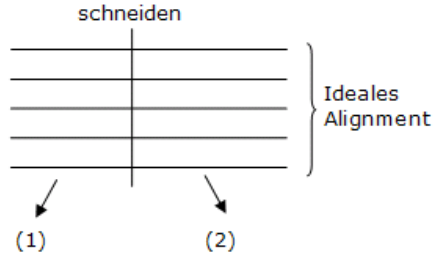


Es können nur konsistente Sequenzen miteinander verglichen werden. … w_1+w_3 oder w_2+w_3

1) "Fragments" … kurze gap-freie paarweise Alignments
2) Align weights w(f) (Scores)
   w(f)=-log p(f)
   Gesamtscore für multiple Aligments … $\sum_f w(f)$
3) Gready heuristic
   a) select best fragment m
      Gesamtscore für multiple Aligments … $\max \sum_f w(f)$
   b) delete all fragments incompatible with f_m
      Gesamtscore für multiple Aligments … $\min \prod p(f)$

DCA - Divide and Conquer Algorithmus

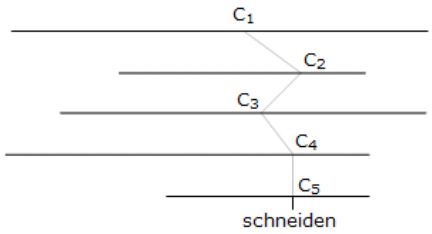
Das Prinzip von Divide an Conquer liegt darin, dass ein komplexes Problem in kleine Teile aufgeteilt wird, die einzeln gelöst und deren Lösungen zu einer Gesamtlösung zusammengesetzt wird.



(1) Bestes Alignment der vorderen Sequenz
(2) Bestes Aignment der hinteren Sequenz

Man erhält nur halb so lange Sequenzen, wodurch sich der Rechenaufwand verringert.

Ein Problem das sich ergibt liegt in der Frage nach dem besten Schnittpunkt. Denn liegt der Schnitt genau in einer konservierten Sequenz, werden auch die Subalignments keine guten Ergebnisse liefern.



DCA (A_1, A_2, …, A_r, L)

if min(A_1, A_2, …, Ar) ≤ L
   compute optimal alignment
else
   compute cut-point (C_1, …, C_r) # Heißt: suche c_1 … c_r, sodass C minimal
   compute DCA (A^1, … C_1, A^2,…C_2, … A^r…C_r) hinterer Teil
            DCA (A^1_{C+1} … n_1, A^r_{Cr+1}…n_r) vorderer Teil

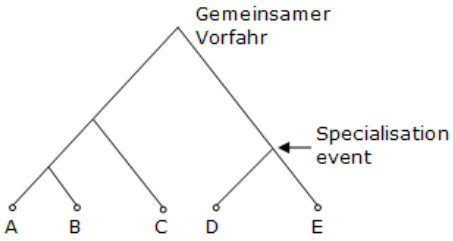
Gute Schnittpunkte werden folgendermaßen berechnet:
S_{ij}* … bestmöglicher Score des Alignments der Sequenz, das durch i,j verläuft.

[S(x,y) - S_{ij}xy] = w_{ij}^{xy} … Verschlechterung des Alignments wenn es durch i,j läuft.

$C(c_1,\ldots,c_r) = \sum_{x,y} w^{xy}{}_{c_x,c_y}$ … Summe über alle Paare von Sequenzen und durch Kosten für Cutpoints.

10.12.2008*

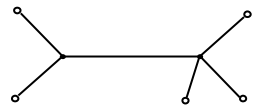
Phylogenetische Rekonstruktion - phylogenetische Bäume



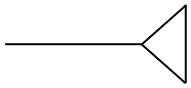
Phylogenetische Bäume geben die Entwicklung unterschiedlicher Spezies aus einem gemeinsamen Vorfahren wieder. Die Trennung in voneinander verschiedene Organismen erfolgt an bestimmten Punkten im Baum, den specialisation events. Dieses Modell schließt horizontalen Gentransfer wie z.B. bei Übertragung mittels F-Duktion nicht mit ein.

Der Definition nach ist so ein Baum ein zusammenhängender Graph ohne Kreisschlüsse.

Baum ist ein zusammenhängender Graph ohne Kreise (-schluss)

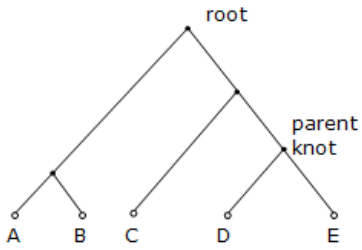


- Blätter (Leaves) … entsprechen einer exakten Spezies
- Interne Knoten … hypothetische Vorfahren

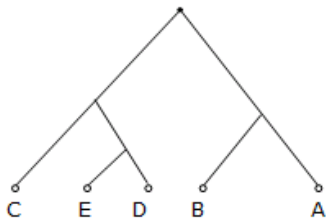


Durch den Kreisschluss kann es sich hier um keinen Baum handeln.

Rooted Tree



root entspricht dem gemeinsamen Vorfahren aller im Baum enthaltenen Organismen.
Phylogenetische Bäume sind ungeordnete Bäume; alle Subbäume sind gleichberechtigt und können rotiert werden.

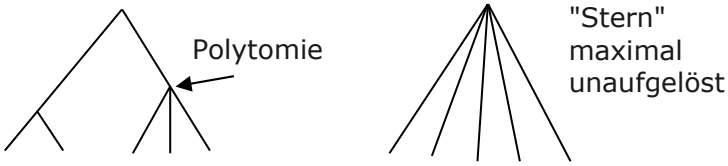


Der Baum könnte daher auch so aussehen und wäre dem oben angeführten gleichgestellt.

Klammerdarstellung

Um einen Baum eindeutig darzustellen, kann man sich der Klammerdarstellung bedienen.
Baum (1) ((A,B),(C,(D,E)))
Baum (2) ((C(E,D))(B,A))

Polytomie - Nicht vollständig aufgelöste Bäume:



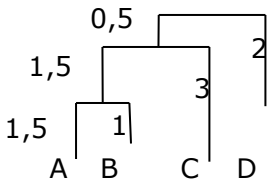
Für Polytomie gibt es zwei Gründe:

- Zu einem bestimmten Zeitpunkt sind z.B. aus einer Spezies mehrere zugleich hervorgegangen. In diesem Zusammenhang spricht man von hard polytomy.
- Soft polytomy entsteht dadurch, dass in sehr kurzem Zeitraum aufeinander folgend zwei oder mehr Spezies entstanden sind.

Kladogramm

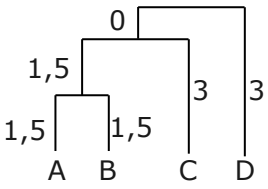
Die Länge der Äste ist nicht von Bedeutung, es gibt demnach weder Kantenlängen noch Distanzen. Ein Kladogramm zeigt lediglich die Verwandtschaftsverhältnisse aber nicht den genauern Grad der Verwandtschaft an.

Additive Bäume



Wird den Kanten zusätzlich eine Gewichtung zugeordnet, erhält man Distanzen zwischen den Spezies. Dadurch kann ein Grad der Verwandtschaft ermittelt werden.
D(AC) = 1,5+1,5+3 = 6

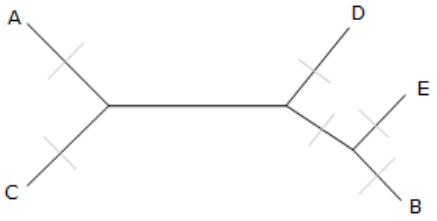
Ultrametrischer Baum



In einem ultrametrischen Baum liegen alle Children auf einer Ebene.

Ein ultrametrischer Baum erfüllt die Axiome der Ultrametrik. Alle Dreiecke sind gleichschenklig d.h. bei drei Distanzen sind zwei davon gleich lang, die dritte weist eine andere Länge auf.
$d(x, y) \leq \max(d(x, z), d(y, z))$

Splits



Eine andere Art Bäume zu beschreiben sind Splits.

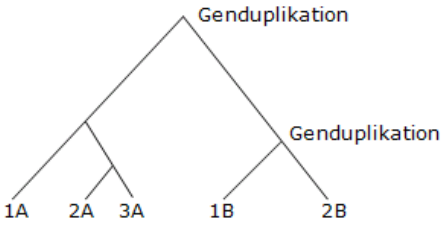
Jede Kante eines solchen Baumes kann ein Split sein. Wird eine Kante aus dem Baum entfernt, hängen die beiden erhaltenen Subbäume nicht mehr zusammen.

Mögliche Splits: AC | DEB, ACD | EB etc.
gegen

Ein Split entspricht damit einer Möglichkeit, eine Menge in zwei Submengen zu zerlegen, jeder Split entspricht einer Kante.

Den einzelnen Submengen der Splits kann man in der Phylogenie oft bestimmte Eigenschaften (character) zuweisen, die nur die jeweilige Submenge, nicht aber die andere Submenge besitzt z.B. können Chordata in Invertebrata und Vertebrata getrennt werden, die beide bestimmte Eigenschaften aufweisen.

Nicht jede Eigenschaft ist geeignet einen Baum zu generieren (Warmblüter, Kaltblüter) d.h. die einzelnen Splits sind untereinander nicht kompatibel.



Früher wurden hauptsächlich species trees erstellt, um die Verwandtschaftsverhältnisse zwischen den Spezies darzustellen, heute werden hauptsächlich gene trees erstellt.
Unterschiedliche Spezies entstehen durch Duplikation von Genen und anschließende unterschiedliche Veränderung der doppelten Gene.

Distanzbasierte Methoden

Es können beliebige Eigenschaften verwendet werden, um Distanzen zu definieren. Distanz aber nicht nur bei Phylogenie interessant, sondern immer wenn man viele Daten hat und in einem Baum darstellen möchte.

Ein additiver Baum d^T_{ij} enthält gemessene Distanzen d^M_{ij} von zwei Spezies die bekannt sind.
Gesucht ist der additive Baum T, bei dem $d^M_{ij} \sim d^T_{ij}$.

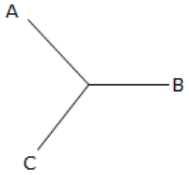
$$\sum_{ij} \left(d^M_{ij} - d^T_{ij}\right)^2 \quad \text{minimal}$$

Das Problem lässt sich so nicht einfach lösen. Man müsste alle möglichen Baumtopologien durchlaufen, in so einem Fall steigen die Kosten exponentiell.
Es muss daher mit einer Heuristik gearbeitet werden.

Zunächst wird die Frage gestellt, ob ein Baum existiert, dessen Distanzen gleich sind.

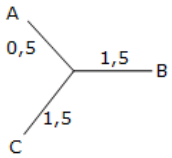
$$\sum_{ij} \left(d^M_{ij} - d^T_{ij}\right)^2 = 0$$

Bsp. Für drei Spezies:



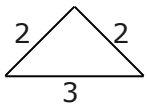
Können die Kanten im Baum so sein, dass die Distanz 0 entspricht.

	A	B	C
A	0	2	2
B		0	3
C			0



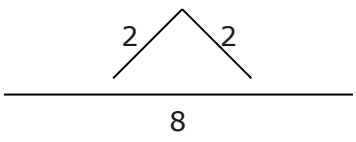
Es gibt drei Constraints (Kanten AB, AC, BC) und drei Variablen.

Mit dieser Matrix wäre das möglich, da die Dreiecksgleichung erfüllt ist.

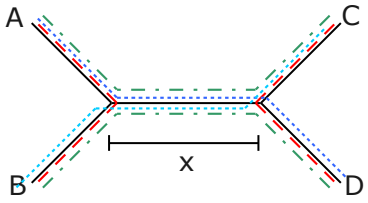


	A	B	C
A	0	2	2
B		0	8
C			0

Hier würde das nicht gehen, da hier kein Dreieck möglich ist.

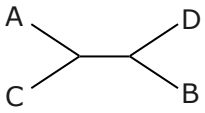


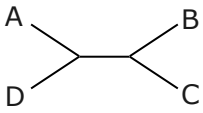
Für vier Spezies:



Es sind sechs Gleichungen mit fünf Unbekannten zu lösen.
Ermittlung von x:
$x = 0{,}5 \cdot (d(A,D) + d(B,C)) - d(AB) - d(CD)$
$x = (d(A,C) + d(B,D)) - (d(A,D) + d(B,C))$ muss 0 ergeben.
$x = 0{,}5 \cdot (d(AB) + d(CD)) - d(A,D) - d(B,C)$ ergibt -x

Es gibt aber noch zwei weitere Topologien, die berücksichtigt werden müssen:

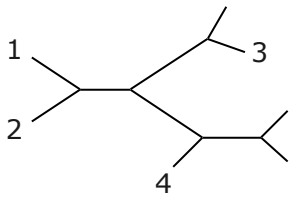




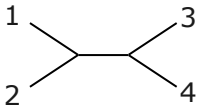
Hier wird x nach demselben Schema ermittelt.

Das Ergebnis wird verwendet, um zu entscheiden, um welchen der drei Bäume es sich handelt.

Großer metrischer Baum:



… entspricht



Auf diese Art und Weise können die Entfernungen in großen Bäumen ermittelt werden.

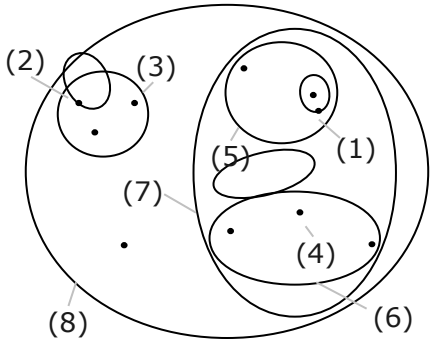
$d_{ij} + d_{kl} - d_{ik} - d_{jl}$

Bünemann Theorem:

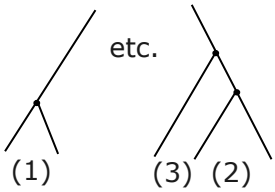
$$d_{ij} + d_{kl} \leq \max (d_{ik} + d_{jl}, d_{il} + d_{jk})$$

Werden mit dieser Formel vier Distanzen geprüft und die Formel erfüllt, kann mit den Distanzen ein Baum erstellt werden, der $d^M_{ij} = d^T_{ij}$ entspricht. Ist die Gleichung nicht erfüllt, kann nur ein Baum erstellt werden, der ungefähr passt ($d^M_{ij} \sim d^T_{ij}$).

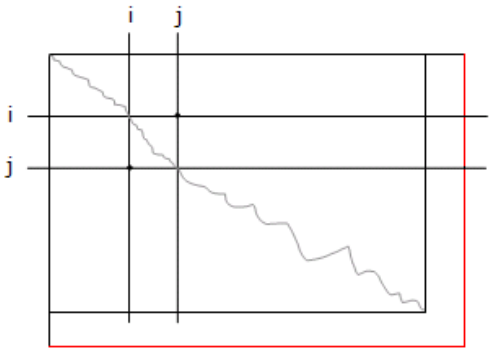
Agglomerative clustering



Die zwei am nächsten beieinander liegenden Daten werden zusammengefasst und am unteren Ende des Baumes angeschrieben. Diese Punkte werden ab jetzt nur noch als ein Punkt behandelt. Es werden dann die nächsten Punkte gesucht, die am nächsten beieinander liegen.



Diese Punkte werden nach dem Zusammenfassen als nur noch ein Punkt betrachtet … aus n Punkten n-1 Punkte und weitersuchen.

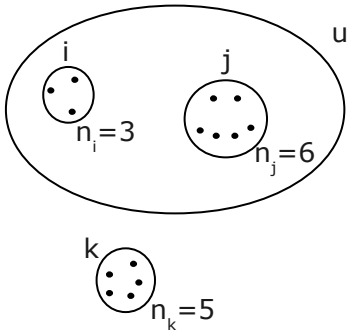


(1) Man erhält eine Distanzmatrix D_{ij}
(2) Suche Paar mit kleinster Distanz (ij), fasse (ij) zusammen.
(3) Lösche (ij) aus der Matrix, füge Cluster u (ij) als neue Zeile und Spalte ein.
Dieser Vorgang wird wiederholt, bis die Matrix zu einem Wert zusammengeschrumpft ist.

Die Cluster können auf unterschiedliche Art berechnet werden.

WPGMA (Weighted Pair Group Method with Arithmetic mean)

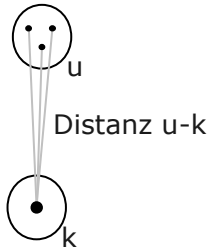
$d_{ku} = 0{,}5 \cdot (d_{ki} + d_{kj})$ … Distanz zu einem Cluster.



… u enthält somit 9 Punkte
… k enthält somit 5 Punkte

Mittelwert aus k und u bilden. ABER: in j sind mehr Punkte als in i enthalten. Daher wäre es sinnvoll j stärker zu gewichten und das in die Mittelwertberechnung einfließen zu lassen.

UPGMA (Unweighted Pair Group Method with Arithmetic mean)



Aus diesem Grund wird vor allem UPGMA verwendet, in dem diese Gewichtung entfernt wird und Distanzen gleich gewertet werden.

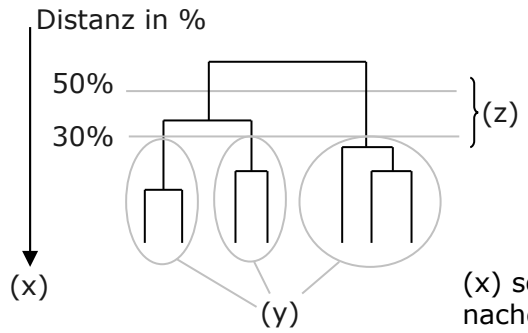
$$d_{ku} = \frac{n_i \cdot d_{ki} + n_j \cdot d_{kj}}{n_i + n_j}$$

$d_{ku} = \min d_{ki}, d_{kj})$ … single linkage
$d_{ku} = \max(d_{ki}, d_{kj})$ … complete clustering

Warts clustering:

Bei zwei zusammenfassten Punkten wird der Schwerpunkt weiter verwendet.

Agglomerative clustering führt immer zu ultrametrischen Bäumen.



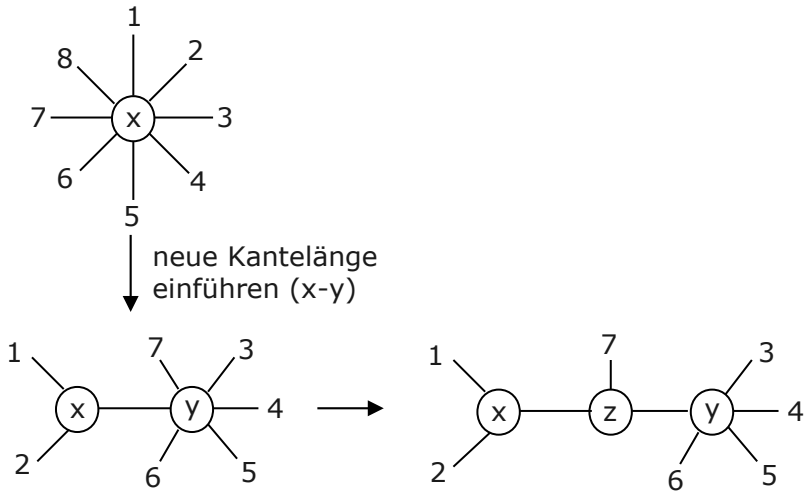
Wird bei (z) geschnitten, wird der Baum in Gruppen eingeteilt, je weiter unten der Schnitt, desto mehr Gruppen erhält man. Bei einer Million Datensätzen wird der Baum erstellt, geschnitten und z.B. nur 100 Gruppen ausgewertet. Die Ansicht der Daten ist so in verschiedener Auflösung möglich.

(x) so Distanz der Gruppen zueinander anschauen, je nachdem ob single linkage oder complete clustering eben mindestens 30% Distanz zu- oder maximal 30% Distanz voneinander.
Bei (y) weis man daher, dass man Gruppen mit Mindestabstand von 30% betrachtet.

17.12.2008*

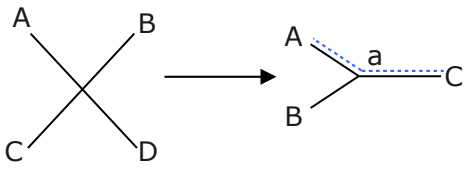
Neighbour joining (Saitou und Nei)

Durch die Einführung neuer Kanten kann aus paarweisen Distanzen die Kantenlänge des Baumes ermittelt werden.
Durch entfernen einzelner Kanten kann die Länge des Baumes wieder verkürzt werden. Dabei werden zwei Knoten zusammengefasst und Beobachtet, ob sich die Gesamtlänge des Baumes stark verkürzt. Ist das nicht der Fall, werden diese beiden Knoten nahe beieinander liegen.
Liegen sie nicht nah beieinander, wird sich die Gesamtlänge des Baumes stark verkürzen.



$S_0 = \frac{1}{N-1}\sum d_{ij}$ … Gesamtlänge des Baumes

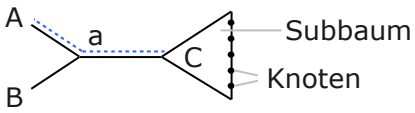
Bsp:



C und D wurden zusammengefügt

$$a = \frac{1}{2}(d_{AC} + d_{AB} - d_{BC})$$

In der Realität wird nicht mit bekannten Distanzen sondern mit mittleren Distanzen über alle Punkte gearbeitet.



$$a = \frac{1}{2}\left(d_{ij} + \frac{1}{N-2}\sum_{k\neq ij} d_{ik} - d_{jk}\right)$$

$$L_{xy} = \frac{1}{2(N-2)}\left[\sum_{k=3}^{N}(d_{ik} + d_{2k}) - (N-2)d_{12} - 2\sum_{i=3}^{N} L_{iy}\right]$$

L_{xy} … Distanz zwischen x und y … Distanz der neu eingeführten Kante.

$\frac{1}{2(N-2)}$ … weil die Länge xy 2(N-2)x durchlaufen wird

$-2\sum_{i=3}^{N} L_{iy}$ … weil jede Kante zweimal gezählt wird

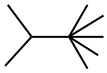
Gesamtlänge des neuen Baumes:

$$S_{12} = L_{xy} + (L_{1x} + L_{2x}) + \sum_{i=3}^{N} L_{iy} = \frac{1}{2(N-2)}\sum_{k=3}^{N}(d_{1k} + d_{2k}) + \frac{1}{2}d_{12} + \frac{1}{N-2}\sum_{3\leq i\leq j} d_{ij}$$

Der nächste Punkt nach Einführung von Kanten ist es, den Baum zu schrumpfen lassen. Suche das Paar ij, sodass S_{ij} minimal wird.

$$r_i = \sum_{j \neq i} d_{ij} \qquad Q_{ij} = \frac{d_{ij} - r_i + r_j}{N - 2}$$

Die Gesamtlänge schrumpft umso mehr, je mehr die zwei Knoten gemein haben.

Von  auf  schrumpft die Gesamtlänge des Baums, wenn die Daten einer Baumstruktur entsprechen.

Kosten von $O(n^3)$

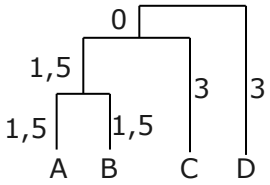
Bsp: Unterschied zwischen Clustering und Neigbour joining, ausgehend von einer Distanzmatrix:

	A	B	C	D
A	0	3	5	6
B		0	6	5
C			0	9
D				0

… Agglomaratives clustering durchführen: man sucht kleinsten Wert

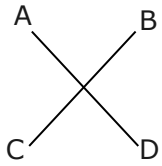
	(AB)	C	D
(AB)	0	6	6
C		0	9
D			0

Neue Matrix:
Für C und D jeweils wird das max AC/BC und AD/BD aus der vorherigen Matrix verwendet.



C und D sind jetzt sechs voneinander entfernt und nicht neun wie in der Matrix. Das ist das Problem der ultrametrischen Bäume. Das Ergebnis ist nicht korrekt, wenn Baum nicht genau ultrametrisch ist.

Selber Baum mit Neighbour joining:



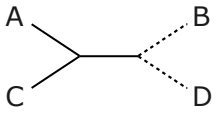
$$r_i = \sum_{j \neq i} d_{ij}$$

$r_A = 3 + 5 + 6 = 14$
$r_B = 3 + 6 + 5 = 14$
$r_C = 5 + 6 + 9 = 20$
$r_D = 6 + 5 + 9 = 20$

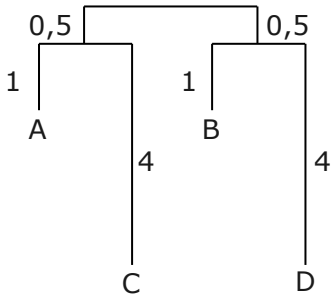
Q_{ij} für Clustering berechnen: $Q_{ij} = \frac{d_{ij} - r_i + r_j}{N - 2}$

$Q_{AB} = 3 - 28/2 = -11$
$Q_{AC} = 5 - 34/2 = -12$
$Q_{AD} = 6 - 34/2 = -11$
$Q_{BC} = 6 - 34/2 = -11$
$Q_{BD} = 5 - 34/2 = -12$
$Q_{CD} = 9 - 40/2 = -11$

Es wurden zwei Minima ermittelt (AC bzw. BD). Eines der Minima (AC) wird geclustert.



$a = 0{,}5 \cdot (d_{AB} + d_{AD} - d_{CB} - d_{CD}) = 3 + 6 - 6 - 9 = -6$
bzw.
$a = 1/4\,(2d_{AC} + d_{AB} + d_{AD} - d_{BC} - d_{CD})$
$= 1/4\,(10 + 3 + 6 - 6 - 9) = 1$



Jede Kante in einem Baum die gestrichen wird, ist äquivalent zu einem Split des Baums, da er dann in zwei Subbäume zerfällt.
Ein Baum kann daher auch als Menge an Splits angesehen werden … $S{:}A, \overline{A}$
$A + \overline{A}$ Gesamtmenge des Baums, weder A noch $\overline{A}$ dürfen leer sein.

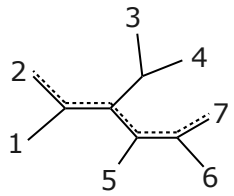
$$d(x,y) = \sum_S d_S(x,y)\alpha_S$$

Split S, Gewicht des Splits α_S

1 falls S separiert x,y
0 sonst

Bsp: 1234567 split … 123|4567 … $d_{13} = 0$ nicht separiert
$d_{15} = 1$ separiert

z.B.

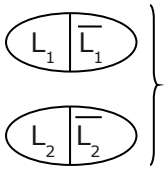


Bei N Elementen gibt's 2^{N-1}-1 mögliche Splits. 2^{N-1}-1 weil Mengen nicht leer sein dürfen (0|1234 darf nicht möglich sein).
Splits wachsen exponentiell, während die Zahl der Elemente im Baum linear wächst.

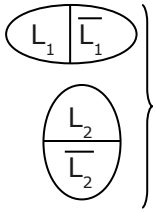
Compatible splits:
Vier mögliche Schnittmengen

L_1, $\overline{L_1}$	wird ein weiterer Split eingeführt	$L_1 \cap L_2$
L_2, $\overline{L_2}$	wird eine der Untermengen weiter zerteilt	$\overline{L_1} \cap L_2$
		$L_1 \cap \overline{L_2}$
		$\overline{L_1} \cap \overline{L_2}$

Mengen müssen für einen Schnitt kompatibel sein wenn sie bereits einmal geschnitten wurden. Es kann kein weiterer Schnitt zwischen einem Punkt in Menge eins und einem Punkt in Menge zwei mehr stattfinden, es dürfen nur noch Schnitte innerhalb der Submengen stattfinden.



für weitere Schnitte kompatibel

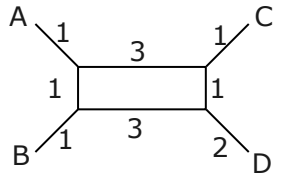


nicht kompatibel

Splitmetrik

Jede Distanz auf einem Baum ist eine Splitmetrik, aber nicht jede Splitmetrik muss in einer baumartigen Struktur liegen.

Eine Splitmetrik ist eine Distanz, die in der Form $d(x,y) = \sum_S d_S(x,y)\alpha_S$ darstellbar ist.



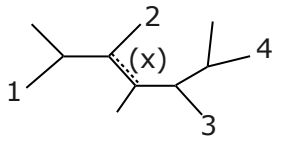
Splitgraph:

Split A gegen den Rest $\alpha_A = 1$
B| gegen Rest $\alpha_B = 1$
AB|CD $\alpha = 3$
AC|BD $\alpha = 1$

} Splits nicht kompatibel, sind unterschiedlich gewichtet, nur einer von beiden kann in einem Baum vorkommen. Der Split AB|CD hat mehr Gewicht, ein solcher Baum ist daher wahrscheinlicher.

Anhand des Splitgraphen kann man baumähnliche Strukturen ablesen oder entscheiden, dass es sich sicher um keinen Baum handelt.

Splitdecomposition:



(x) soll berechnet werden.
$x = 0{,}5\ (d_{13} + d_{24} - d_{12} - d_{34})$

$$\alpha_S = \frac{1}{2}\min_{ij,kl}(d_{ik} + d_{il} - d_{ij} - d_{kl})$$

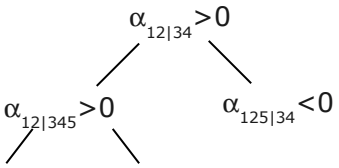
$\min_{ij,kl}$ … auf unterschiedlichen Seiten des Splits

Bsp. $\alpha_{13|24}$ … Dieser Split ist in diesem Baum nicht möglich. Man würde eine negative Zahl für das Gewicht des Splits erhalten. Demnach sind nur positive α von Interesse! $\alpha_S > 0$. Auf diese Art können O(n) bis $O(n^2)$ Splits entstehen, aber nicht mehr.

Effizientere Splitzerlegungen:

Es wird mit drei bis vier Spezies begonnen und Splits eingeführt. Erhält man ein negatives Gewicht, können hier keine weiteren Splits eingefügt werden.
Erhält man aber ein positives Gewicht, können weitere Spezies hinzugefügt und neue Splits eingefügt werden.

Bsp: For i=4…N



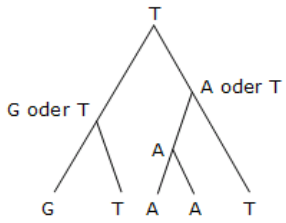
Für alle alten Splits versuche i links oder rechts hinzuzufügen. Dieser Schnitt für sich kostet $O(n^3)$.

Das liefert theoretische exponentielle Splits, tatsächlich aber fallen viele Splits wieder weg, man erhält O(n) bis $O(n^2)$ Splits.

07.01.2009*

Maximum parsimony Verfahren

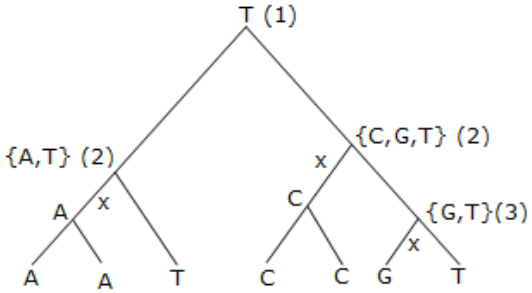
Über das maximum parsimony Verfahren wird der Baum gesucht, der mit den geringsten Änderungen einer bekannten Sequenz zugeordnet werden kann oder am besten einen Satz verwandter Spezies abbilden kann.



Es wird am unteren Ende des Baumes begonnen und nach oben gewandert. Bei jedem erreichten Knoten kann ein Set an Buchstaben stehen, je nachdem welche Children sich von diesem Knoten abgespalten haben.
Manchmal ist es nicht das sinnvollste möglichst wenige Mutationen auszurechnen z.B. bei sehr weit entfernten Organismen.
Es wird daher nach gemeinsamen Vorläufern gesucht und der entsprechend beste Baum gewählt.

Der Fitch Algorithmus

Wie wird jetzt der Baum ermittelt, der die wenigsten Mutationen aufweist:



1) Von den unteren Knoten her werden die möglichen Mengen der Buchstaben bis zur Wurzel ermittelt.
Dabei wird Hierarchisch verfahren. v sei ein interner Knoten z.B. Position (3). Dieser Knoten hat zwei Kinder w (S(w) hat den Wert G) und u (S(u) hat den Wert T). Dadurch muss der interne Knoten v die Werte der Kinder vereinigen $S(v) = S(w) \cap S(u) = \{G,T\}$.

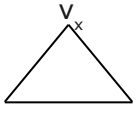
$$S(v) = \begin{cases} S(u) \cap S(w) & \text{falls } \neq 0 \\ S(u) \cup S(w) & \text{sonst} \end{cases}$$

2) Der Baum wird jetzt von oben nach unten durchlaufen und jedem Knoten genau ein möglicher Wert seiner Menge zugewiesen.

$S^{neu}(u) = S(u) \cap S(v)$ v father of u

Auf diese Art kann die Baumlänge effizient ermittelt werden, der beste Baum wird aber nicht ermittelt.

Will man Mutationen bewerten (z.B. ein AS-Austausch) werden Kosten vergeben:

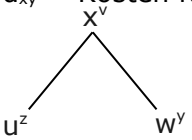


Cv(x) sind die Kosten für den Subbaum, der an Knoten v beginnt, wenn x der dort stehende Buchstabe ist.

Für die Leafs gilt dann:

Initialisierung: $Cv(x) = \begin{cases} 0 & \text{wenn } x = x_v \\ \infty & \text{sonst} \end{cases}$

d_{xy} = Kosten für Mutation x→y

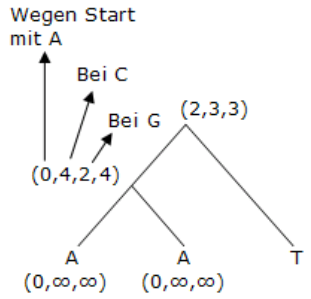


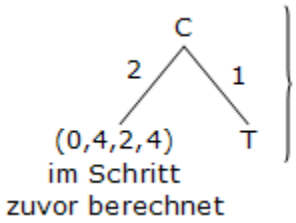
$$Cv(x) = \min_{Y \in A} Cw(y) + d_{xy} + \min_{z} Cu(z) + d_{xz}$$

Beispiel:

	A	C	G	T
A	0	2	1	2
C		0	2	1
G			0	2
T				0

min (d_{CA} +0, 0+4, d_{CG} + 2, d_{CT} + 4) = 2





Möglichkeiten:
min(d_{CA} + 0, 0+4,…)

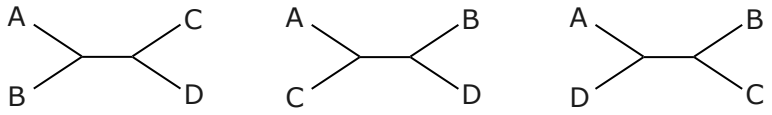
d_{CA} … Kosten für Mutation
+0 … Kosten wenn zuvor ein A steht
0 + 4 … Mutation gratis, aber: 4 weil zuvor ein C steht.

Lösung des kleinen parsimony Problems (gegebener Baum)
Kosten: O(N(A))
N … Zahl der Spezies
A … Größe des Alphabets

Das große Parsimony Problem ist schwer zu lösen, da viele Topologien durchsucht werden müssen.

NNI (Nearest neighbour interchange)

Es handelt sich um eine heuristische Methode um die Wahrscheinlichkeit (Likelihood) eines gegebenen unrooted Baumes zu prüfen, indem die Äste des Baumes vertauscht werden.

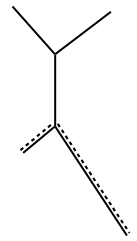


Durch die Vertauschung der Nachbarn erhält man neue Bäume, die bei erneuter Berechnung bessere Verwandtschaftsverhältnisse aufweisen können als der Originalbaum.

Problem der maximum likelihood

Unter maximum likelihood estimation versteht man das Verfahren, ein mathematisches Modell an reale Daten anzupassen, um das Modell zu verfeinern.

Annahme: folgender Baum:



Je länger eine Kante ist, desto wahrscheinlicher und damit billiger wird eine Mutation.

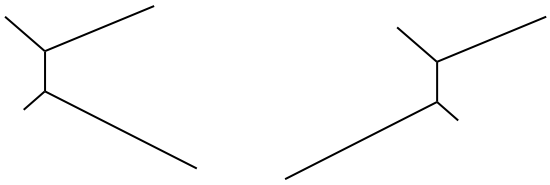
Für einen gegebenen Baum kann man sich Wahrscheinlichkeiten ausrechnen.

$P(Model/data) = P(A/B) \cdot P(B) = P(B/A) \cdot P(A)$

$$\frac{P(Data/Model) \cdot P(Model)}{P(Data)}$$

P(Data/Model) … likelihood
P(Model) … Annahme alle Modelle gleich
P(Data) … konstant

Baum und Kantenlängen sind bekannt, dadurch ist die Wahrscheinlichkeit für Daten bei dem gegebenem Modell berechenbar.



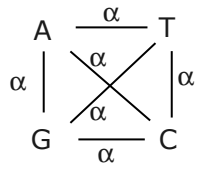
Zwei Bäume die möglich sind

$P(x|y \cdot t)$ ist die Wahrscheinlichkeit, dass irgendwo ein x steht, wenn früher ein y stand.

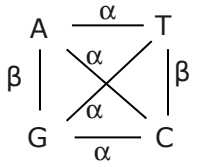
Für diese Berechnung ist ein Modell der Sequenzevolution nötig.

1. Annahme: jede Sequenz entwickelt sich unabhängig von der anderen.
2. Annahme: Mutationsraten sind zu jeder Zeit gleich.

Modelle für Mutationsraten:



Im einfachsten Modell (Jukes-Cantor) sind die Mutationsraten für alle Nukleotide gleich.



Das Modell nach Kimura unterscheidet zwischen Transition (Pur→Pur, Pyr→Pyr) und Transversion (Pur→Pyr, Pyr→Pur), da sie unterschiedlich oft stattfinden.

$$Q = \begin{pmatrix} A & C & G & T \\ -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}$$

Die Mutationsrate von A zu einer der anderen Basen für Jukes-Cantor kann in einer Ratenmatrix Q dargestellt werden. Jede Zeile addiert sich auf 0.

$$P = \begin{pmatrix} P_A \\ P_C \\ P_G \\ P_T \end{pmatrix}$$ … Wahrscheinlichkeit, dass nach gegebener Zeit τ dort ein A,C,… steht:

$$\frac{d}{dt} P_x = \sum_y P_y \cdot q_{yx} \qquad \frac{d}{dt} \vec{P} = p \cdot Q$$

$\vec{p}(t) = \vec{P}(o)\, e^{tQ}$ … e^{tQ} … der Exponent entspricht einer Matrix

Lösung: Reihenentwicklung der e-Funktion

$e^Q = 1 + Q + ½ Q^2 + …$

Allerdings wird die Matrix üblicherweise diagonalisiert:

$A = V^{-1}DV$
$A^2 = V^{-1}DVV^{-1}DV = V^{-1}D^2V$
$A^3 = V^{-1}D^3V$

$$e^{\text{Diagonalmatrix}} = \begin{pmatrix} e^{\lambda_1} & 0 & 0 \\ 0 & e^{\lambda_2} & 0 \\ 0 & 0 & e^{\lambda_3} \end{pmatrix}$$

$P(t)=P(o)e^{tQ}$

$$Px(t) = \sum_y P_y^{(0)} \cdot P(x \mid y, t)$$

$P(x|xt) = 1/4 + 3/4\ e^{-4\alpha t}$ … wenn $t \to \infty$ geht wird $e^t = 0$

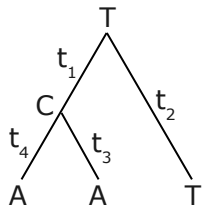
Nach sehr langer Zeit stellt sich also ein Gleichgewicht ein, wobei jedes Nukleotid eine Wahrscheinlichkeit von 1/4 hat.

$P(x|y,t) = 3/4\ (1-e^{4\alpha t})$
$\langle d(t) \rangle = n\ 3/4\ (1-e^{4\alpha t})$

$$\frac{4}{3}\frac{d}{n} = 1 - e^{-4\alpha t} \Rightarrow 1 - \frac{4}{3}\frac{d}{n} = e^{-4\alpha t} \Rightarrow t = \frac{1}{-4\alpha} \ln(1 - \frac{4}{3}\frac{d}{n})$$ … "Jukes-Cantor correction"

Gibt Zusammenhang zwischen Distanz und Zeit an.

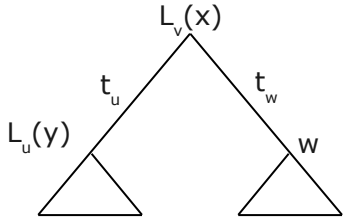
Zurück zur Maximum likelihood Methode



Likelihood α dieses Baumes:

$\alpha = P(T) \cdot P(C|T,t_1) \cdot P(T|T,t_2) \cdot P(A|C,t_3) \cdot P(A|C,t_4)$
Gesucht ist aber die beste Baumvariante.

$L_v(x)$ = Likelihood des Subbaums unter v gegebene Belegung bei v ist x.



$$v(x) = \left(\sum_{y \in A} P(y \mid x, t_u) \cdot L_u(y)\right) \cdot \left(\sum_{y \in A} P(y \mid x, t_w) \cdot L_w(y)\right)$$

Leafs: $L_v(x) = \begin{cases} 1 & \text{für } x = x_v \\ 0 & \text{sonst} \end{cases}$

Die Methode deswegen so schwierig, weil man für jede Baumtopologie auch noch die optimalen Kantenlängen berechnen muss.

Wie das geht?
$L(Data|T,\theta)$

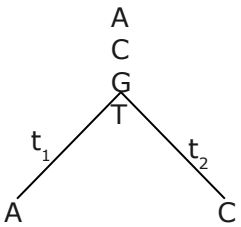
Optimum suchen: $\frac{\partial}{\partial\theta} L(Data \mid T, \theta) = 0$

$$L = \sum_{xy} \overset{l}{L}(x) \cdot \overset{r}{L}(x) \cdot P(x \mid y, t)$$

Die Methode ist recht teuer (für jede Baumtopologie optimale Kantenlänge ermitteln)
Vorteil: maximum parsimony berücksichtigt Kantenlängen nicht.

14.01.2009*

Beispiel für maximum Likelihood; die Wahrscheinlichkeit, dass die Sequenz die von Interesse ist, durch ein bestimmtes Szenario entstanden ist.



Likelihood für diesen Baum berechnen. Wieso ist die Berechnung der Likelihood teurer als parsimony?

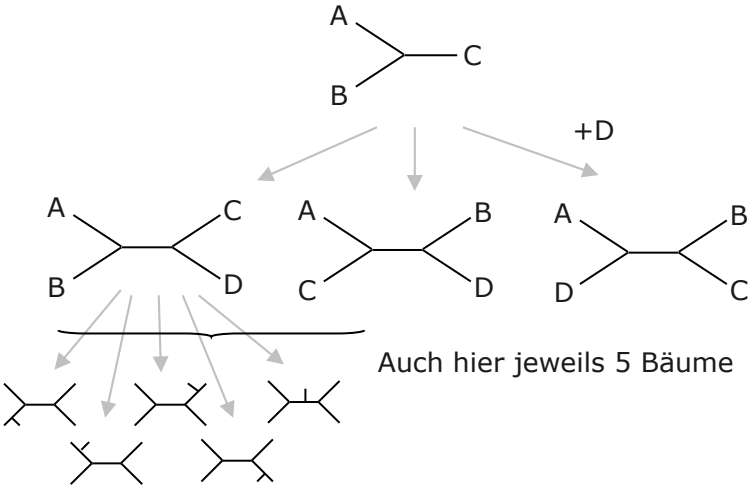
Weil man Likelihood für jede der möglichen Kantenlängen berechnen muss.
Vorteil: die Methode liefert die genauesten Ergebnisse.
Nachteil: Es muss ein passendes Modell der Sequenzevolution verwendet werden.

Algorithmus und Kosten:
Minimum parsimony: schnell (Topologie gegeben)
Maximum parsimony, Maximum likelihood: beide teuer

Bei 20 Spezies erhält man $2 \cdot 10^{20}$ mögliche Baumtopologien. Das ist selbst für max Parsimony teuer.

Branch and Bound

In der Praxis wird daher der nach Branch and Bound berechnet. Die 1020 Baumtopologien werden rekursiv generiert.



Das Verfahren wird mit drei Spezies begonnen und eine neue Spezies eingeführt (Branch Schritt). In diesem Schritt werden alle möglichen Topologien erstellt und die Baumlänge jeder Topologie berechnet.
Überschreitet eine Baumtopologie eine zuvor festgelegt Länge, wird ihre Verarbeitung abgebrochen, bei den anderen Topologien wird eine weitere Spezies eingebracht und alle möglichen Topologien ermittelt.

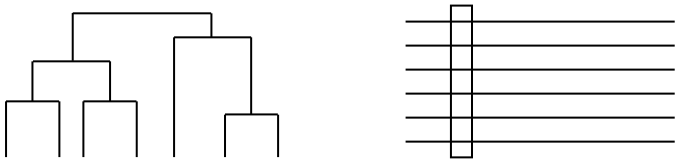
Die Berechnung der Topologien kann über Neigbour joining oder Lösung des großen Parsimony Problems durchgeführt werden.

Gesuchter optimaler Baum $S_{opt} \leq S_{NJ}$

Problem: Branch and Bound spart nicht immer essenziell was, da beispielsweise erst zu einem späten Zeitpunkt ein Zweig wegfällt.

Bootstrapping

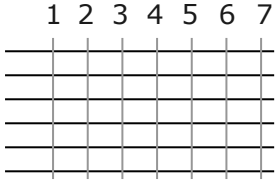
Erzeugt man einen Neighbour joining Baum, kann von Interesse sein, wie zuverlässig bestehende Kanten des Baumes sind:



Bis jetzt wurden bei Baumtopologien Spalten unabhängig voneinander betrachtet.

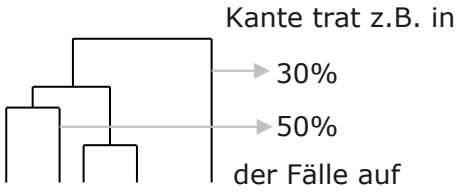
Zu diesem Zweck gibt es zwei statistische Methoden, Jackknifing und Bootstrapping.
1. Möglichkeit: Jackknifing: Es wird die Hälfte der Spalten entfernt und geprüft, ob man einen vergleichbaren Baum erhält?

2. Möglichkeit: Bootstrapping
Bsp: Alignment mit 7 Spalten



Daraus wird ein neues Alignment erzeugt, indem man Spalten zufällig auswählt (z.B. 322466) und den Phylogeniealgorithmus erneut laufen lassen.

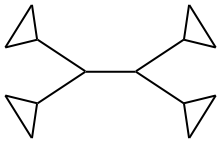
Das Ergebnis der statistischen Auswertung vergleicht man nun mit anfangs untersuchten Baum.



Je mehr Daten vorhanden sind, desto eher wird ein durch den Algorithmus ermittelter Baum ein "wahrer" Baum sein. Je kürzer ein Baum, desto größer wird die Wahrscheinlichkeit, dass man einen falschen Baum erhält.
Darüber gibt Bootstrapping Auskunft: Es wird ermittelt, ob genügend lange Sequenzen vorhanden sind, damit ein der Wahrscheinlichkeit nach richtiger Baum erhalten wird.

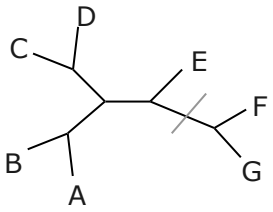
Bootstrapping sagt nur etwas über Fehler aufgrund der Datenmenge aus. Systematische Fehler im Evolutionsmodell werden dadurch nicht ermittelt.
z.B. horizontaler Gentransfer, Sequenzen, die sich durch genetische Selektion gemeinsam entwickelt haben.

Zurück zu Neighbour joining, maximaler Likelihood, parsimony



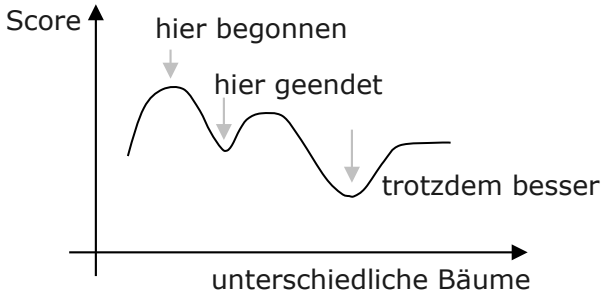
Es wurde mittels Neighbour joining ein Subbaum im Gesamtbaum ermittelt, der guten Bootstrapsupport hat.

Mittels NNI kann dieser Baum verbessert werden, indem versucht wird, Subbäume oder Äste zu vertauschen.



Eine Erweiterung von NNI ist Tree grafting

Es wird überall etwas abgeschnitten. Findet sich nach einer gewissen Zeit trotz aller Umstellungen kein besserer Baum, wurde ein lokales Optimum erreicht.



Bayessche Methoden

Es handelt sich um eine Heuristik von maximum Likelihood Methoden.

$$P(\text{Tree} \mid \text{Data}) = \frac{P(\text{Data} \mid \text{Tree}) \cdot P(\text{Tree})}{P(\text{Data})}$$

P(Data|Tree) … Wahrscheinlichkeit, dass ein Baum unseren Datensatz produziert.
P(Tree) … prior Wahrscheinlichkeit irgendeinen Baum zu erhalten, bleibt konstant.
P(Data) … durch vorgegebene Daten ist auch das konstant.

Da andere Faktoren konstant sind ist P(Tree|Data) $\propto$ P(Data|Tree)

Markoff Chain

A ——— B Wurden durch NNI verbunden.
P(A) P(B)

Methode, die von einer Möglichkeit zur anderen springt und die Anzahl an beobachteten A soll $\propto$ P(A) und bei B $\propto$ P(B) sein … Marcoff chain

$x_0 \cdot x_1 \cdot \ldots \cdot x_t$

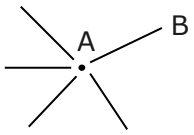
$P(x_{t+1}|x_t)$

Marcoff Modell 0.Ordnung: Liefert Nukleotidsequenzen anhand ihrer relativen Häufigkeit.

1. Ordnung: P(C|ACCT)

A	C	C	T						

A —NNI— B $P(A)\cdot P(A\rightarrow B)=P(B)\cdot P(B\rightarrow A)$
P(A) P(B)



Wie groß ist der Übergangsbereich zu B oder anderen?
$P(A)\cdot P(A\rightarrow B) = P(B)\cdot P(B\rightarrow A)$ … muss erfüllt sein.

Metropolisregel: $P(A\rightarrow B) = \begin{cases} 1 & \text{falls } P(B) > P(A) \\ \frac{P(B)}{P(A)} & \text{sonst} \end{cases}$

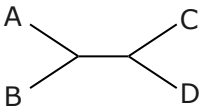
Es wird mit einer beliebigen Baumtopologie begonnen. Von den vielen NNIs wird eine ausgewählt und nach $P(A)\cdot P(A\rightarrow B) = P(B)\cdot P(B\rightarrow A)$ entschieden, ob ein Übergang stattfindet oder nicht.

Die Methode bleibt nicht in lokalen Minima gefangen.

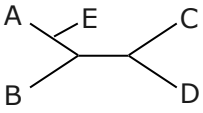
Treepuzzle

Es werden alle möglichen 4-Tupl aus einem Speziespools gewählt und dafür die Maximum likelihood berechnet.

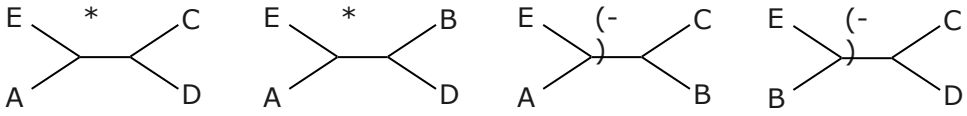
(1) Berechne für jedes Quartett den ML Baum.
Spezies alle in einer Reihe aufschreiben.
Arten 1-4: Baum:



Spezies anhängen (schon alle Quartette berechnet):



Wie gut ist das? Es resultiert folgendes Quartett:



Wird eine Spezies an anderer Stelle angehängt, wird erneut eine Zahl an Quartetten erhalten, die gute und schlechte Ergebnisse liefert.

(2a) Bringe Spezies in zufällige Reihenfolge
(2b) Wähle besten Baum für die ersten vier.
(2c) füge die weitere Spezies nacheinander hinzu, wähle jeweils den Baum mit der kleinsten Zahl von Widersprüchen (zu den in (1) berechneten Quartetten).
Der Baum, den man erhält, hängt von der Reihenfolge der Spezies ab!
(3) Wiederhole (2) mit unterschiedlichen Reihenfolgen. Man erhält einen Satz möglicher Bäume.
(4) Bilde den strikten Konsensusbaum:
Wähle alle Splits (müssen kompatibel sein), die in > 50% der Bäume vorkommen.

100 Bäume: Split 1 in 51 Bäumen mindestens ein Baum mit Überschneidung
Split 2 in 51 Bäumen

Ist das nicht möglich, handelt es sich um einen unaufgelösten Baum.

Aufwand der Methode:
(1) Die Berechnung der Quartette ist der aufwändigste Schritt der Methode.
Anzahl der Quartette bei n Spezies:

$$\binom{N}{4} = \frac{N!}{4!(N-4)!} = \frac{N\cdot(N-1)\cdot(N-2)\cdot(N-4)}{1\cdot 2\cdot 3\cdot 4} \qquad O(N^4)$$

Bsp: 20 Spezies $3\cdot\binom{20}{4} = 14535$ statt 10^{20} Baumtopologien

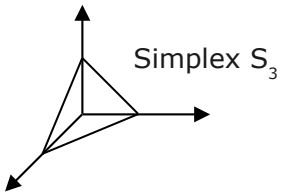
Die Methode ist eine Heuristik (es wird kein ML Baum erhalten), aber es findet eine Annäherung an einen ML Baum statt.

Likelihood mapping

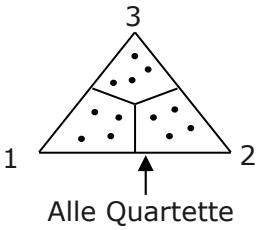
Es handelt sich dabei nicht unbedingt um eine Baumrekonstruktionsmethode.

(1) Berechne für jedes Quartett die likelihood der drei Möglichen Topologien L_1, L_2, L_3. Es handelt sich dabei nicht um die Wahrscheinlichkeiten der Bäume, sondern um die der Sequenz gegenüber dem Baum.

Jeder Topologie eine Wahrscheinlichkeit zuordnen: $P_i = \frac{L_i}{L_1 + L_2 + L_3}$



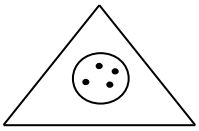
Die Wahrscheinlichkeiten müssen sich auf 1 addieren.



Jeder der drei Bäume wird als Punkt in einem Dreieck dargestellt.

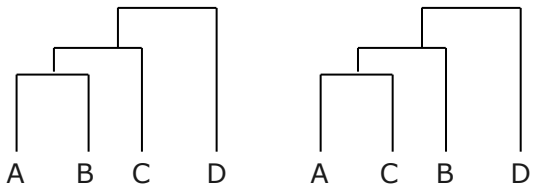
Punkt wo alle Topologien eines Quartetts gleich wahrscheinlich sind.
Simplex $S_n = \{x_1,x_2,\ldots x_n \in \mathbb{R}+|x_1+x_2+\ldots+x_n=1\}$

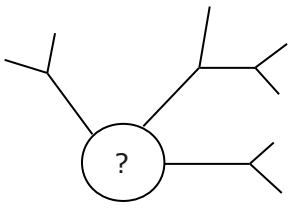
Eine andere Möglichkeit:



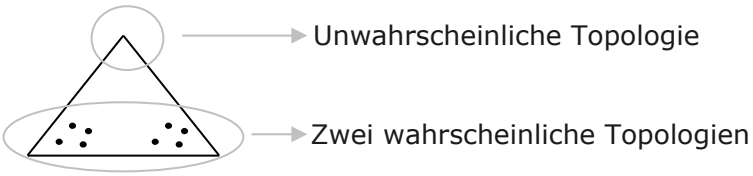
Nicht so gut wie oben, da die Wahrscheinlichkeiten der Topologien nah beieinander liegen.

Beispiele, die gut für likelihood mapping geeignet sind: z.B. Streitfragen:





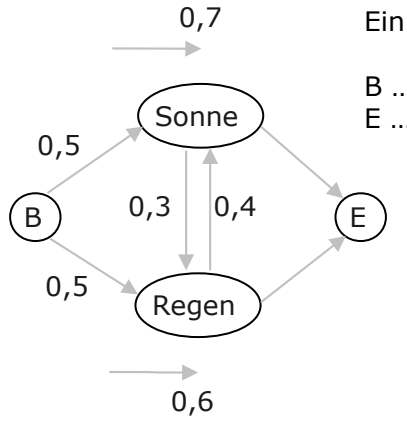
Ein anderes Beispiel: typischerweise sind alte Zweige am schwersten rekonstruierbar:



Ist ein Verfahren, dass nicht unbedingt eine Antwort liefert.

28.01.2009

State-Modelle



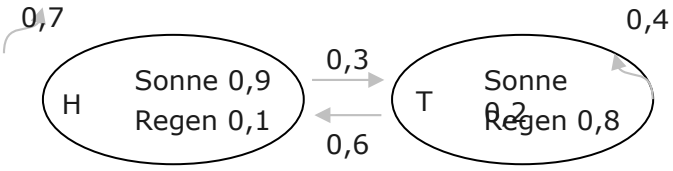
Ein kleines Wahrscheinlichkeitsmodell der Wettervorhersage.

B … Begin state
E … End state

Die Übergänge von State zu State sind durch Wahrscheinlichkeiten bestimmt.

Die Gesamtwahrscheinlichkeit einer State-Änderung muss = 1 (100%) sein.

Ein etwas komplizierteres Modell zur Wettervorhersage, das bereits mehr Punkte berücksichtigt.

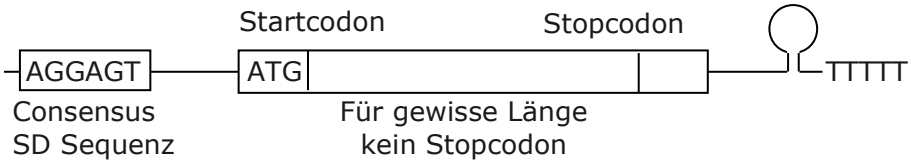


H … Hochdruck
T … Tiefdruck

Transition probabilities
Emission probabilities

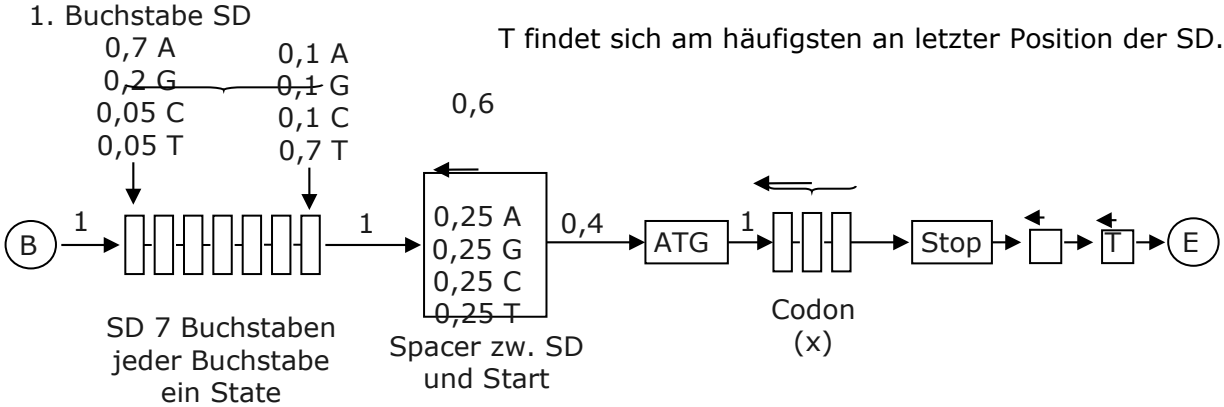
Ein solches Modell wird als Hidden Markov model bezeichnet. Die State Information geht verloren, wenn man nur Regen/Sonne nicht aber das Barometer betrachtet.

Bsp: Vorhersage Proteinkodierende Gene in einem Genom:



(1) Modell, das bei Gensuche diese einzelnen Teile abbildet … jedes Teil (SD; Start, Stop, etc.) entspricht einem eigenen State.

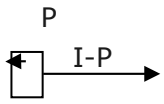
Hidden Markov model für die Erzeugung einer Gensequenz.

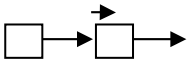


T findet sich am häufigsten an letzter Position der SD.

(x) Codon … hier können weitere Vorgaben eingebracht werden z.B. Abhängigkeit des letzten nt von den nts zuvor, um der Realität besser zu entsprechen.

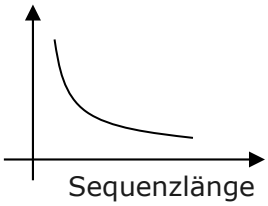
Allgemeine Berechnung einer wahrscheinlichen Längenverteilung:



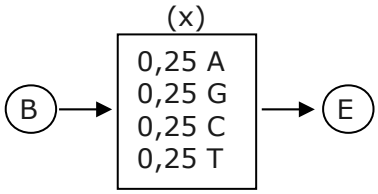


Beispiel um eine Mindestlänge zu erhalten.

Sequenzlänge	
1	I-P
2	$P \cdot I\text{-}P$
3	$P^2 \cdot I\text{-}P$
k	$P^{k-1} \cdot I\text{-}P$



Das Modell liefert eine Wahrscheinlichkeit zur erzeugten Sequenz d.h. wie wahrscheinlich es ist, dass diese Sequenz auftritt.



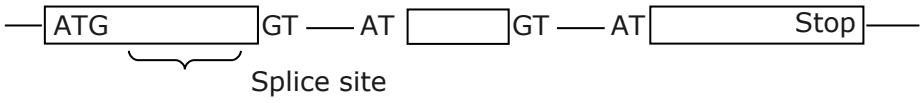
Hidden Markov model für den intergenischen Raum

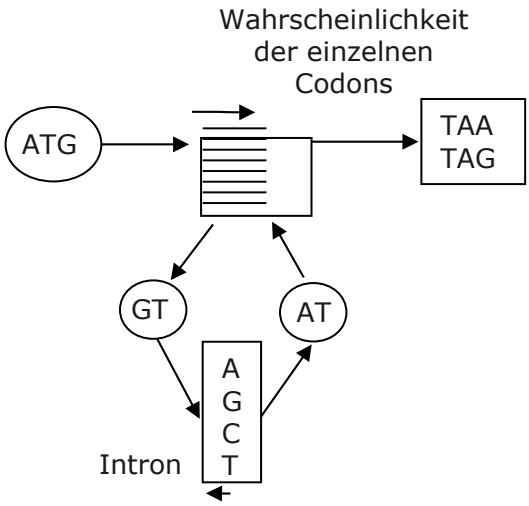
(x) je nach GC/AT Verteilung können die Werte abweichen.

Wie wird fine tuning eines solchen Modells durchgeführt, damit es besser der Realität entspricht?

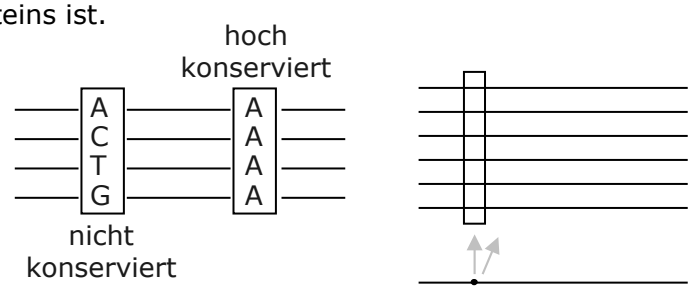
- Bereits sequenzierte Gene hernehmen.
- Die Wahrscheinlichkeit im Modell so ändern, dass die bereits bekannten Sequenzen mit hoher Wahrscheinlichkeit produziert werden.

Bsp. Modell für eukaryotische Gene:



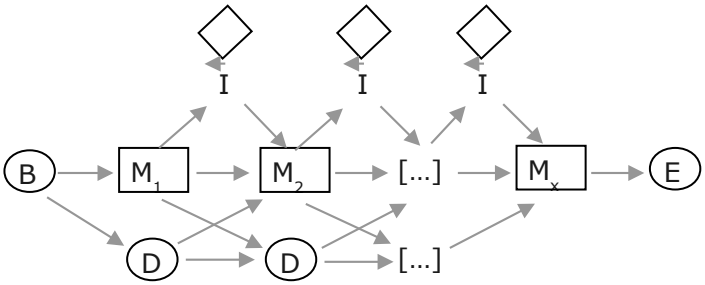


Bezieht man bei einem Alignment der Sequenz auf Profile die konservierte Sequenzen in die Scoreberechnung mit ein, erhält man positionsspezifische Scores.
z.B. auch wenn in einem Teil der Sequenz nie ein Gap weil's z.B. die Active Site eines Proteins ist.



Drei Möglichkeiten
Match
Deletion
Insertion

Ein Hidden Markov model aus einem erhaltenen Alignment erstellen.



M … Match
I … Insertion
D … Deletion

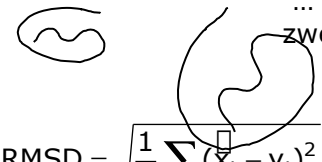
Die Wahrscheinlichkeiten für das Modell müssen mit konservierten Regionen abgeglichen werden d.h. wie schlecht es wäre, aus einer konservierten Sequenz etwas zu entfernen.

Strukturelle Bioinformatik

Es gibt ja die erstaunliche Idee, dass die Sequenz eines Proteins die Struktur bestimmt. Weiters gibt es die Erwartung, dass die Struktur stärker konserviert ist als die Sequenz. Strukturhomologie ist viel unwahrscheinlicher zufällig als Sequenzhomologie.

Strukturvergleiche sind viel schwieriger als Sequenzvergleiche. Probleme ergeben sich bei Röntgenkristallanalyse, wenn die Versuchsbedingungen nicht exakt identisch sind, dann sind die Ergebnisse nicht vergleichbar.

RMSD: $x_i = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ … räumliche Position des x_i Atoms im Protein



… Moleküle sollen so überlagert werden, dass die Distanz zwischen zwei Atomen in den zwei Molekülen 0 wird.

$$RMSD = \sqrt{\frac{1}{N}\sum_i (x_i - y_i)^2}$$

N … # Atome
x_i … Atom i in Molekül X
y_i … Atom i in Molekül Y

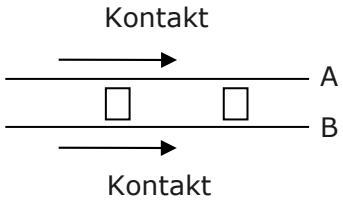
Möglichkeit … Distanzmatrizen für die zwei Moleküle

$$D_x = \begin{matrix} 0 & d_{12} & d_{13} & \dots \\ \dots & & & \\ & \dots & & \\ & & \dots & \end{matrix}$$

D_y = …

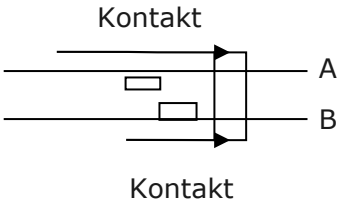
Abstände der Atome in Molekül X Abstände der Moleküle in Atom Y

Meist werden hier nur der Backbone oder überhaupt nur C_α Atome mit den Residues verwendet, um zu ermitteln, ob die Residues in Kontakt kommen können. Zwei Residues wären z.B. in Kontakt, wenn sie weniger als 6 Å voneinander entfernt sind.



guter Score

Es wird zunächst ein Sequenzalignment durchgeführt und anschließend geprüft, wo sich strukturell in Sequenz A ein Kontakt innerhalb der Sequenz findet. Weiters wird geprüft, ob in Sequenz B strukturell an dieser Position ebenfalls ein Kontakt ist. Das liefert einen guten Score, wenn sich dieser Kontakt in beiden Sequenzen findet, andernfalls wird ein schlechter Score ausgegeben … Structural alignment.

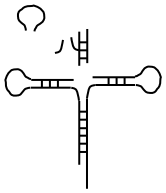


Es werden immer Paare betrachten. Mit welchem Alignment kann das ermittelt werden? Man sieht sich zwei Spalten an, über den Score bestimmt wird. Needleman-Wunsch schaut z.B. nur eine (die letzte) Spalte an und vergibt danach den Score, das wäre nicht sinnvoll für unser Problem.

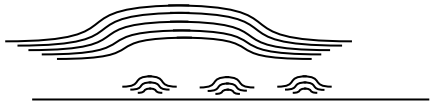
schlechter Score

Für dieses Problem ist vermutlich keine effiziente Lösung möglich. Um es wenigstens zu vereinfachen werden Proteine in Stücke zerschnitten (z.B. in Sekundärstrukturen wie α-Helix oder ß-Faltblatt) und diese Stücke verglichen.

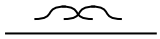
Bsp. Sekundärstrukturabbildung in tRNA:



… Aufschlüsselung der Paarung

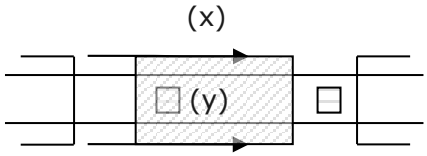


Jede Base kann nur eine Paarung eingehen (nested) oder benachbarte Paarungen entstehen.

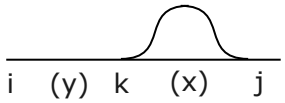


Überschneidende Paarungen werden als Pseudoknoten bezeichnet.

Problem bei Alignments:



Überschneidende Paarungen (Pseudoknoten) dürfen nicht stattfinden, d.h. weitere Paarungen dürfen nur innerhalb von (y) oder außerhalb der beiden in Kontakt stehenden AS (x) erfolgen.



Eingrenzung auf zwei weitere unabhängige Sequenzen (y) und (x), wenn keine Pseudoknoten erlaubt sind.

$E_{ij} = \max E_{ij}, \max_k I + E_{kH,j-1} + E_{i,k-1}$ … $O(N^3)$

Es handelt sich um eine kubische Laufzeit und ist damit teuerer als Needleman-Wunsch, weil die Aufspaltung komplizierter ist.

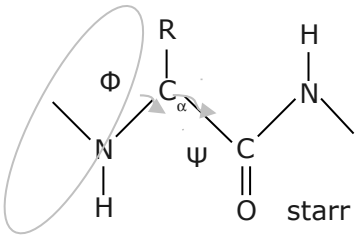
Sekundärstrukturvorhersage

z.B.

CCHHHCCCEEECCCCEEEEECC

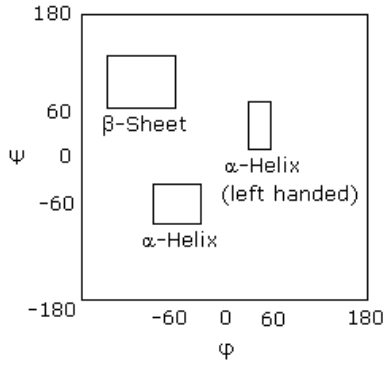
H … Helix
E … Betastrand
C … Random Coil

Peptidbindung:



Φ,Ψ … Torsionswinkel im Backbone, nur hier ist das Protein wirklich flexibel.

Ramachandran plot für Winkel bei Sekundärstrukturen.



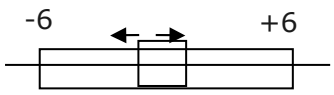
Für Glycin und Prolin würde der Ramachandran plot ganz anders aussehen.

Über einen solchen Plot kann ermittelt werden, wie oft eine AS in einer Sekundärstruktur (Helix, ß-Strand, etc.) vorkommt.
z.B. wie oft kommt Alanin in einer α-Helix in Bezug zum Gesamtvorkommen des Alanins vor:

$$\frac{P(Ala)\alpha}{P(Ala)}$$

Annahme: wenn mehrere AS in Folge auftreten, die oft in einer α-Helix liegen, dann findet sich hier eine α-Helix.
Aber: Vorhersagen dieser Art sind sehr ungenau – Fehlerquote 50%.
Warum? Weil nur eine AS ohne Kontext betrachtet wird. … **True Fassman Algorithmus**

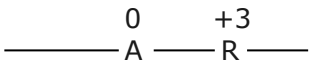
Verbesserter Algorithmus:



Die AS links und rechts der betrachteten AS werden berücksichtigt. Je mehr AS betrachtet werden, desto komplexer wird auch die Wahrscheinlichkeitsberechnung über die Statistik:

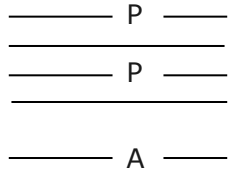
$$\frac{P(-6...AS...+6)\alpha}{P(-6...AS...+6)}$$

Es können auch Kombinationen betrachten werden:



wie wahrscheinlich ist eine Helix bei A, wenn sich in +3 ein R findet.
… solche Algorithmen kommen auf 60% korrekte Vorhersage.
… nimmt man noch Homologievergleiche dazu, kommt man auf 70%.

z.B.



Meine Sequenz hat ein Alanin, das könnte in einer Helix sitzen, da an derselben Position in homologen Sequenzen ein P steht, ist es schon unwahrscheinlicher, dass sich hier wirklich eine Helix findet.