Alignment und Phylogenieschätzung

Dirk Metzler

http://www.cs.uni-frankfurt.de/~metzler/

Gastauftritt am 23. Januar 2003 in

Gisbert Schneider: *Grundlagen der Bioinformatik*

http://gecco.org.chemie.uni-frankfurt.de/bioinf.html

Alignment und Phylogenieschätzung

- paarweises Alignment
  - dynamische Programmierung: global, lokal
  - BLAST: Algorithmus, e-Werte
- Phylogenieschätzung
  - Distanz-basiert
  - Parsimonie vs. Maximum Likelihood

globales Alignment

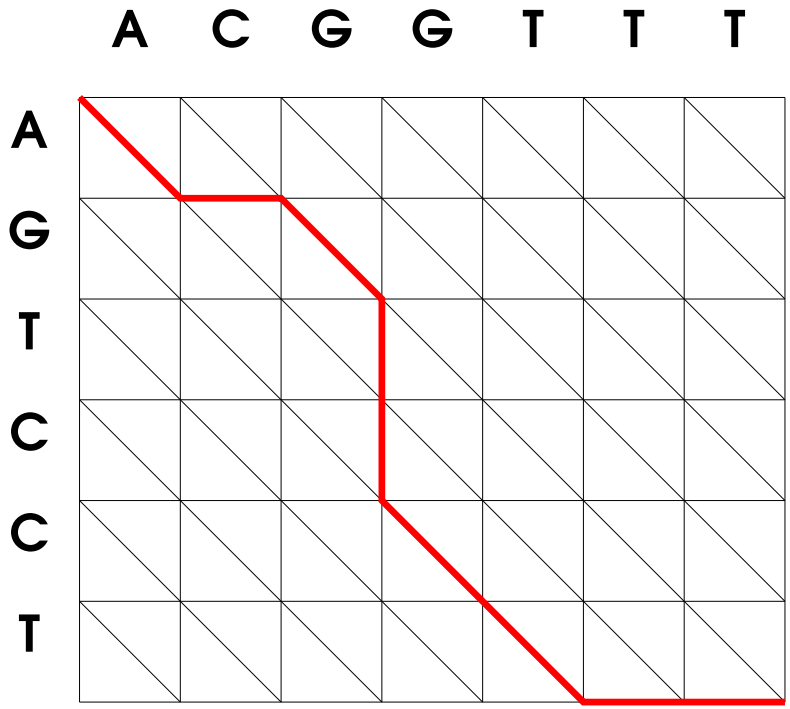
gegeben: zwei Sequenzen,

z.B. ACGGTTT, AGTCCT

gesucht:
das wahre Alignment,

– oder zumindest eins mit
optimalem Score, z.B.

```
ACG__GTTT
A_GTCCT__
```



Score-Funktion: einfaches Beispiel

match reward	$+1$
mismatch penalty	$-\mu$
gap open penalty	$-G$
gap extension penalty	$-g$

z.B.:

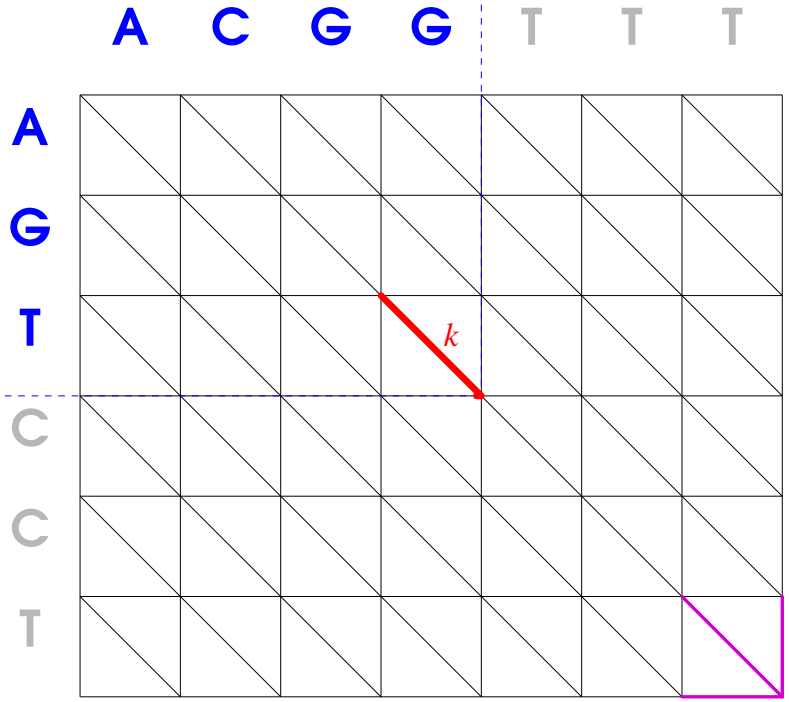
```
ACG__GTTT
A_GTCCT__
```

Score$= 3 - 1 \cdot \mu - 3 \cdot G - 2 \cdot g$

dynamische Programmierung nach Needleman & Wunsch (1970)

Gegeben: 2 Sequenzen, Score-Parameter (μ, G, g)

Gesucht: globales Alignment mit maximalem Score.

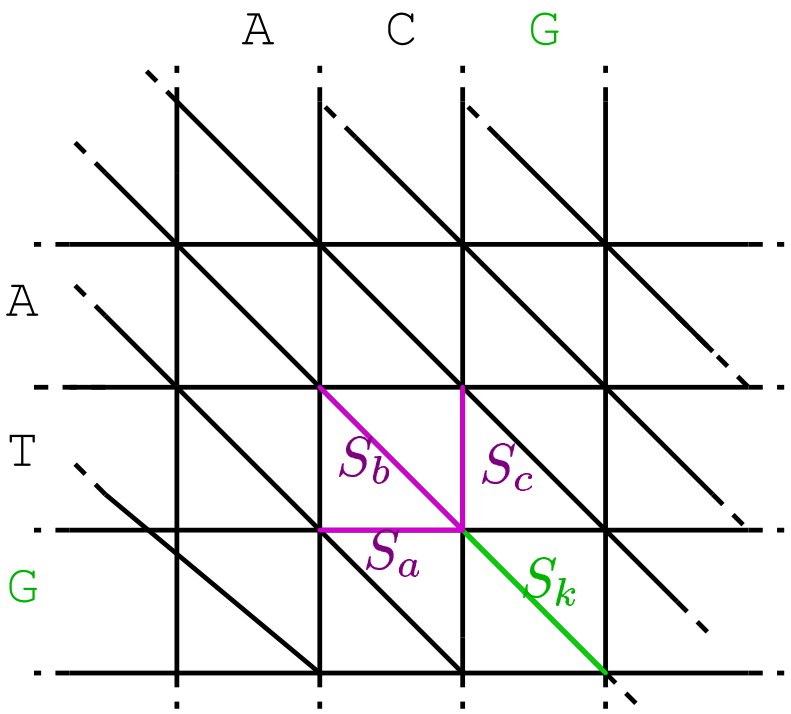


Teilschritt:
beschrifte jede Kante k mit

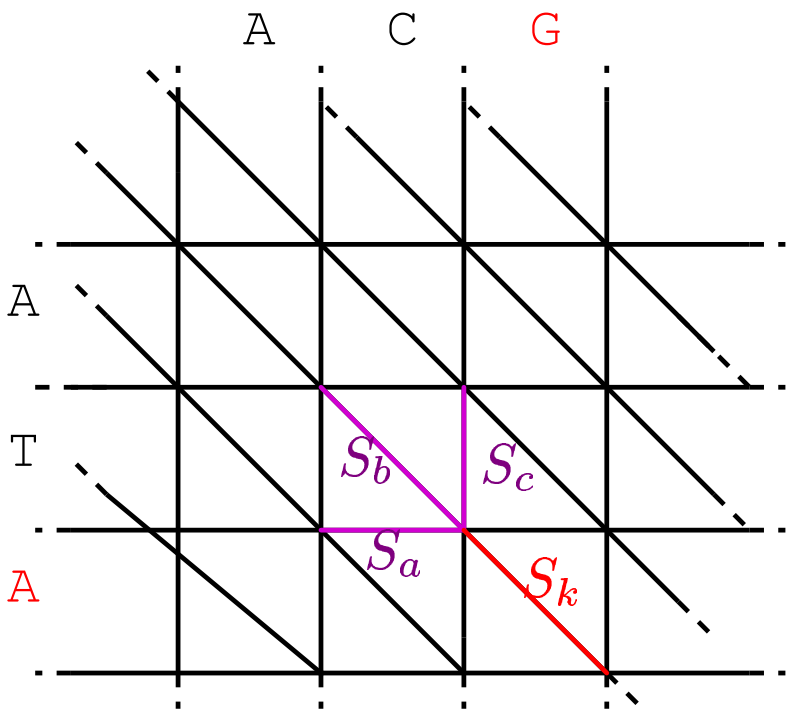
$$S_k$$

dem Score des besten aller in k enden-
den Alignments der Sequenzanfänge

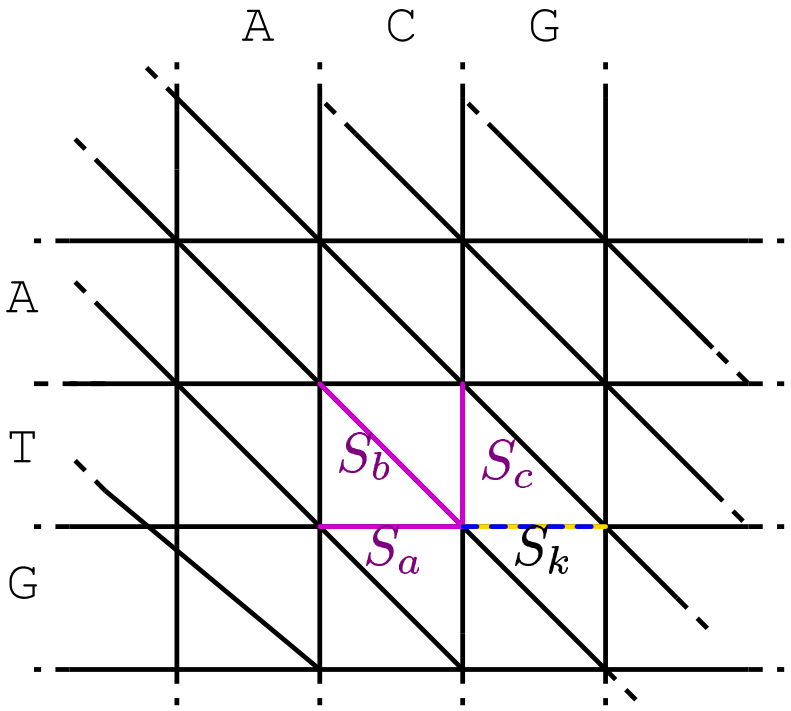
Der maximale Score eines globalen Alignments ist dann das Maximum der Labels der Kanten rechts unten.



$S_k = \max\{S_a, S_b, S_c\}+1$



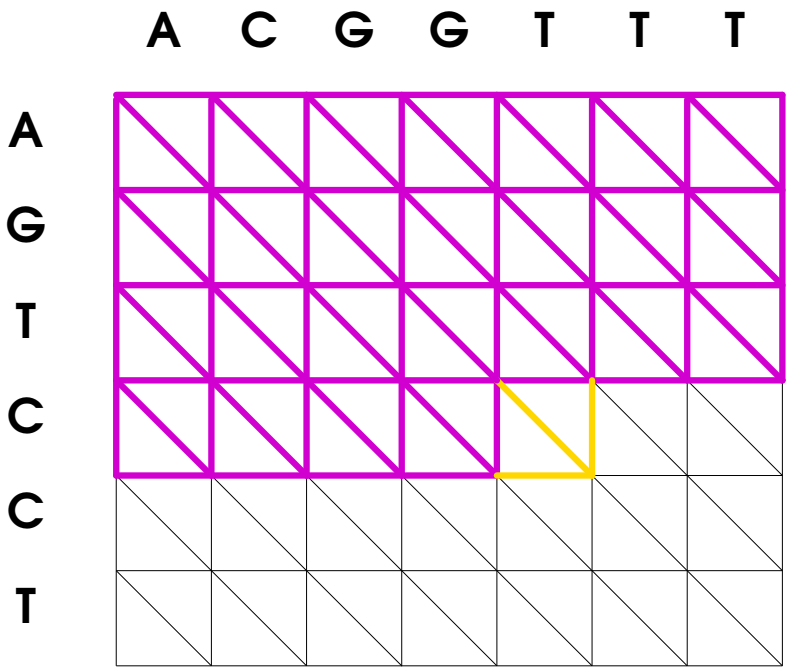
$S_k = \max\{S_a, S_b, S_c\}-\mu$



$$S_k = \max\{S_a - g, S_b - G, S_c - G\}$$

Beobachtung: S_k kann man flott berechnen wenn man die zu den links oben eingehenden Kanten gehörenden Werte S_a, S_b und S_c bereits kennt.

Idee der dynamischen Programmierung:

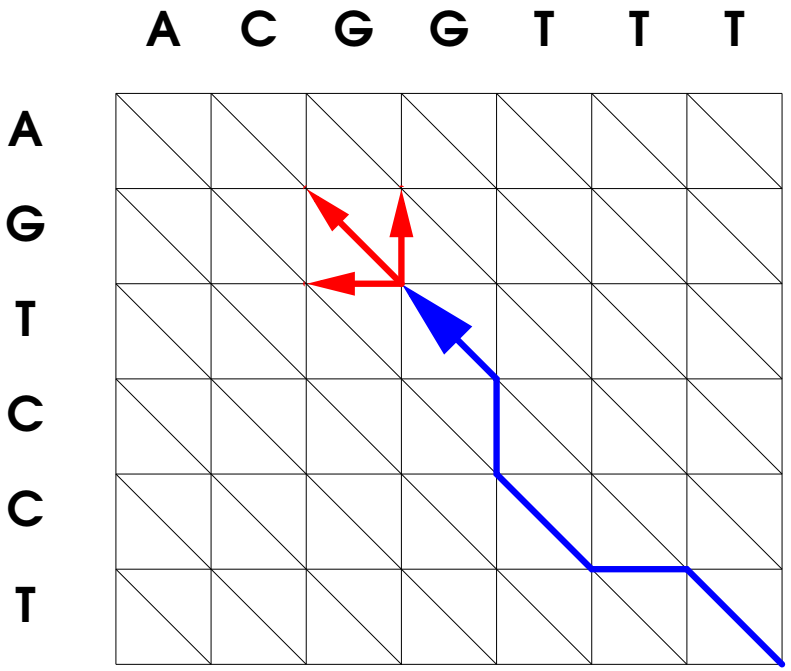


Berechne S_k für alle Kanten k Zeile für Zeile von links nach rechts.

Dann stehen die jeweils erforderlichen Werte S_a, S_b, S_c rechtzeitig zur Verfügung.

Letzter Schritt des Needleman-Wunsch-Algorithmus: Verfolge das Alignment zurück

Laufe von rechts unten nach links oben durch den Grafen. Entscheide Dich jeweils für die Kante, die das “Maximum”, also den Beitrag für die zuletzt gelaufene Kante geliefert hat.



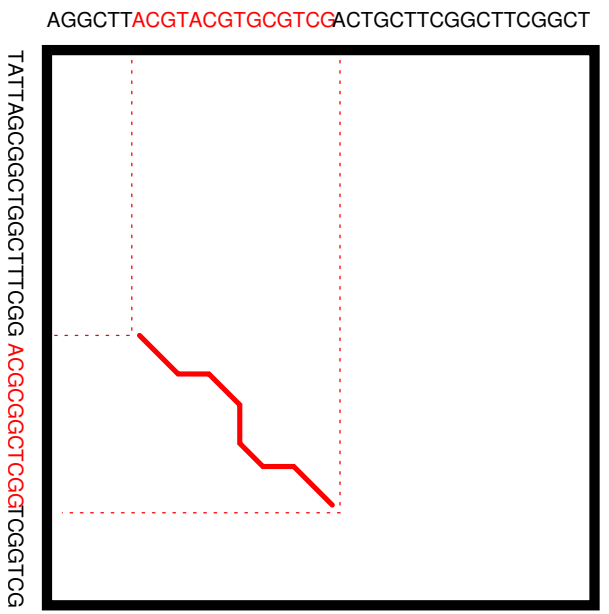
(Man mache sich klar wieso man im allgemeinen nicht das beste Alignment findet wenn man die Maxima von links oben nach rechts unten verfolgt!)

Der Smith-Waterman-Algorithmus (1981)

ist die Variante des Needleman-Wunsch-Algorithmus für *lokales Alignment*.

“Lokal” bedeutet, dass nur Teile der Sequenzen aligniert werden. M.a.W.: beim lokalen Alignment werden Gaps am Anfang und Ende nicht bestraft.

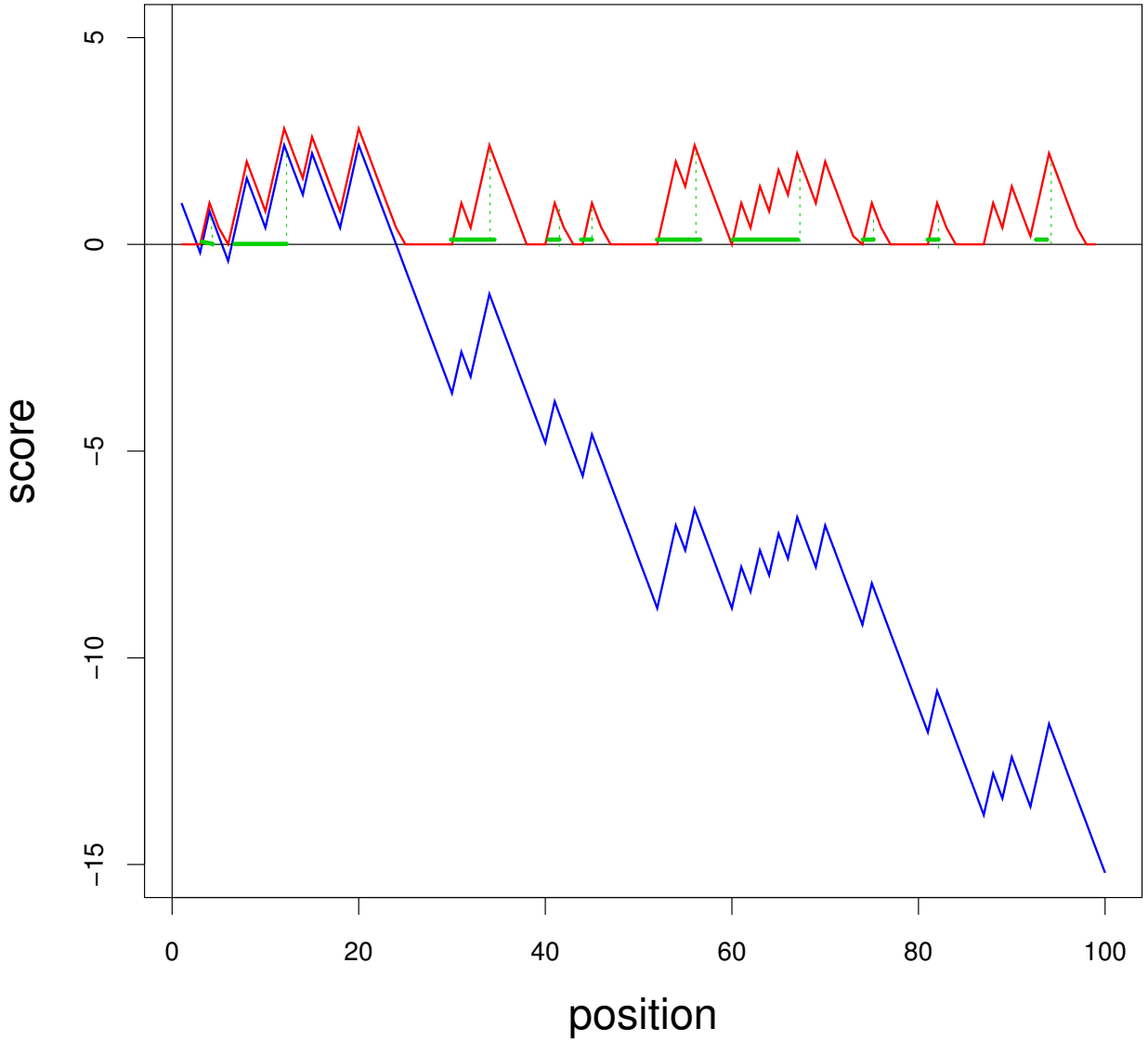
Wenn beim Alignieren längs eines Pfades negative Werte auftreten, lohnt es sich also nicht, weiter zu laufen. Stattdessen merkt man sich das bisherige Optimum und fängt wieder bei 0 an.



Die Rekursion zur Berechnung der Kanten-Labels S_k ist also nun z.B. falls k eine Mismatch-Kante ist:

$$S_k = \max\{0, S_a - \mu, S_b - \mu, S_c - \mu\}$$

Die Entwicklung des Scores längs einer Diagonale im Alignment-Graf:



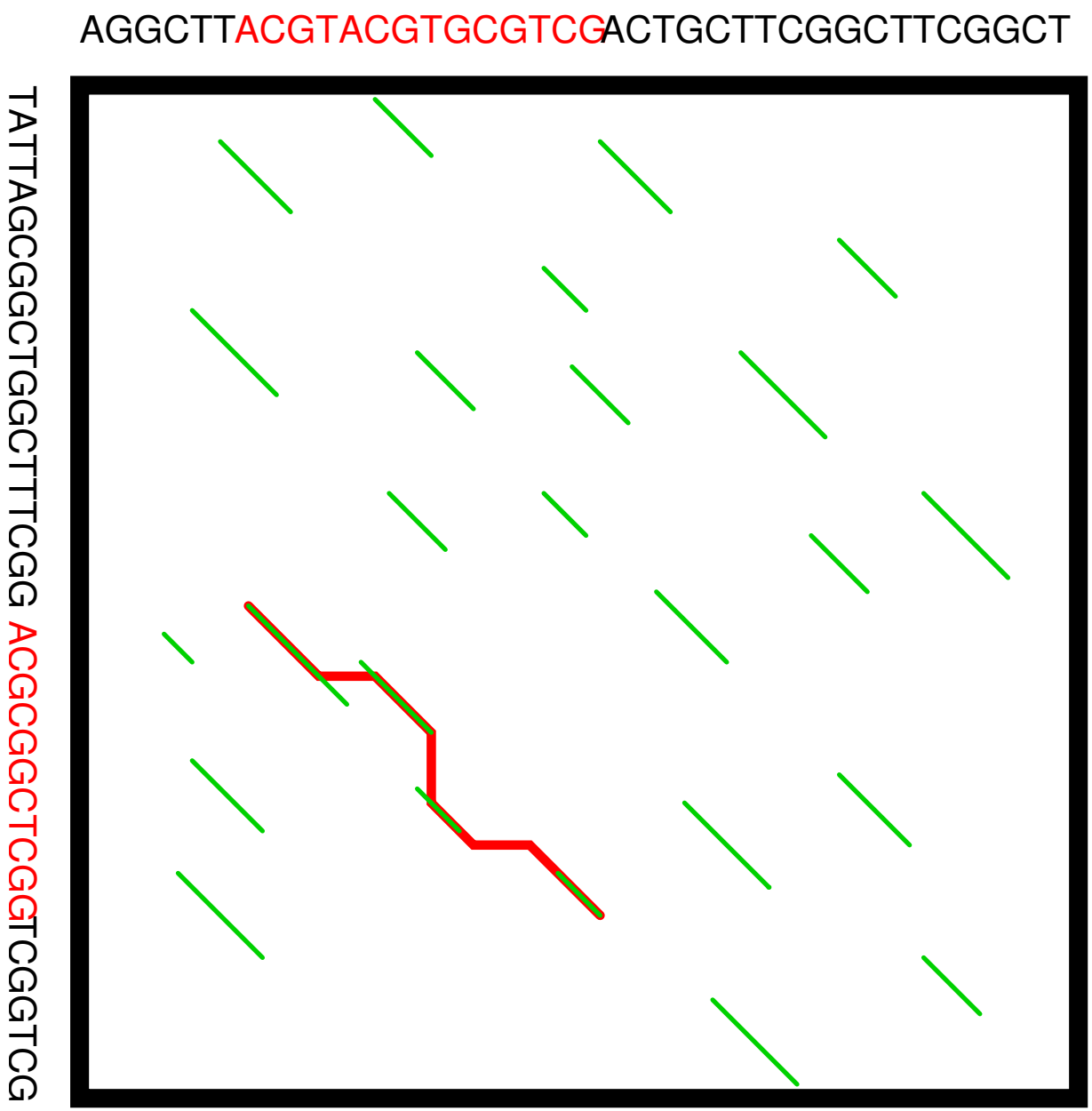
Also ohne Gaps!

$\mu = 0.6$

globaler score

localer score

MSP = maximal segment pair = lokal optimales gaploses Alignment



Oft ist das score-optimale lokale Alignment eine Verkettung von MSP's.

Darauf basiert unter anderem BLAST.

BLAST

Um zwei Sequenzen der Längen n und m mit dynamischer Programmierung zu alignieren (global oder lokal) muss man ca. $3 \cdot n \cdot m$ Kanten beschriften. Der Aufwand $O(nm)$ ist bei Datenbanksuche zu hoch.

BLAST ("Basic Local Alignment Search Tool") findet das score-optimale Alignment in "fast linearem" Aufwand "$O(n + m + \varepsilon)$", zumindest wenn es ein gutes MSP enthält.

BLAST gibt es in vielen Varianten, z.B. speziell für Proteine, für Nukleotide, für Datenbanksuchen, für den Vergleich zweier Sequenzen,...

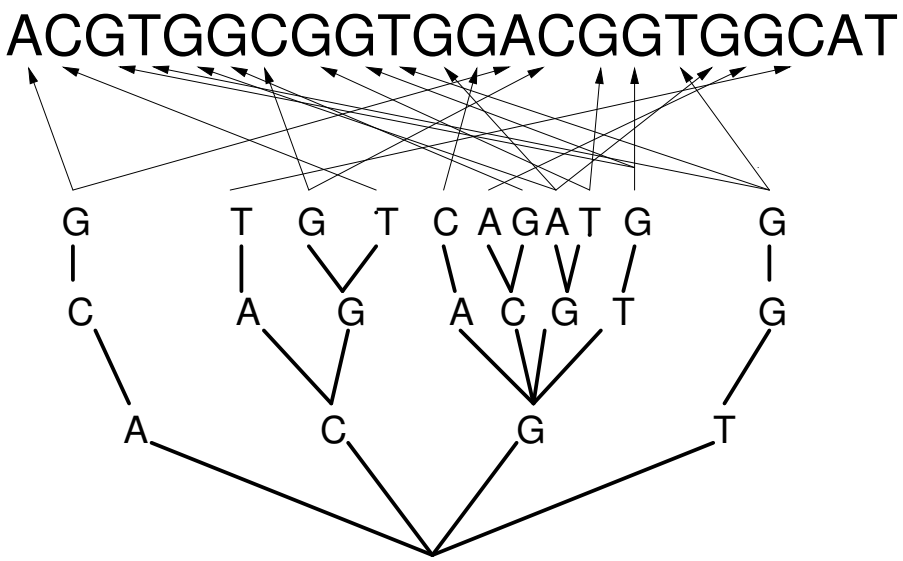
siehe

`http://www.ncbi.nlm.nih.gov/BLAST/`

Grundidee des BLAST-Algorithmus

(mal abgesehen von ∞ vielen Tricks)

Suche zunächst alle MSP's.



Speichere dazu alle in der ersten Sequenz vorkommenden Wörter der Länge der Länge w (z.B. $w = 3$ für Proteine, $w = 12$ für DNA) in einem Suchbaum (bzw. einen Automaten).
Suche die Wörter der zweiten Sequenz dann im Baum.

Versuche dann, die besten MSP's zu lokalen Alignments zu erweitern.

Signifikanz lokaler Alignments

Für zwei Sequenzen der Längen n und m gibt es ein lokales Alignment vom Score S.

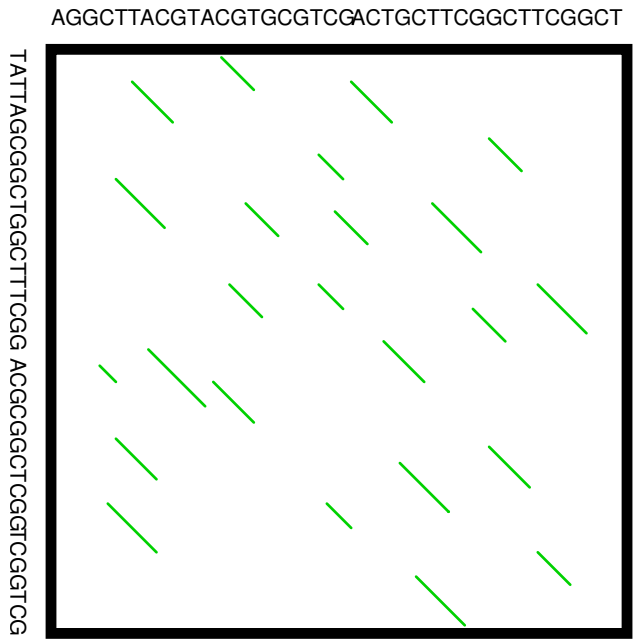
Ja kann denn das Zufall sein?

Modell: Die einzelnen Positionen innerhalb einer Sequenz sind voneinander unabhängig gemäß der Basen- bzw. Aminosäurehäufigkeiten besetzt.

Null-Hypothese: (wollen wir gerne verwerfen!) Die Sequenzen sind voneinander unabhängig.

e-Wert: Erwarte Anzahl lokaler Alignments mit Score $\geq S$ bei zwei Sequenzen der Längen n und m unter Annahme der Null-Hypothese.

Signifikanz lokaler *gaploser* Alignments bzw. MSP's

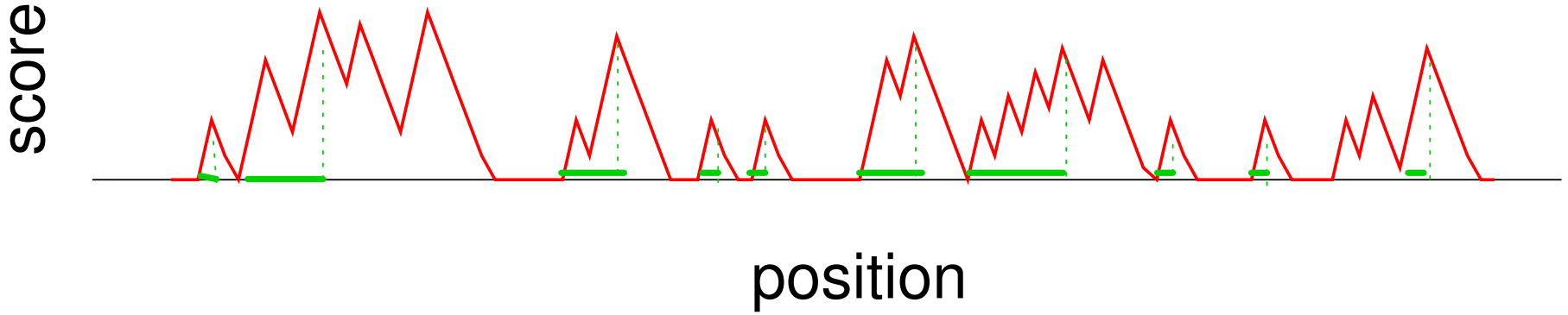


Die Wahrscheinlichkeit, dass an einer fest gewählten Stelle ein MSP mit Score $\geq S$ beginnt, ist

$$\sim k \cdot e^{-\lambda S}$$

(k und λ hängen von den Score-Parametern und den Basen- bzw. Aminosäurehäufigkeiten ab.)
Da es $n \cdot m$ mögliche Startpunkte gibt, folgt

$$e\text{-Wert} \sim nm \cdot k \cdot e^{-\lambda S}$$



Signifikanz lokaler Alignments mit Gaps

Während die Asymptotik für e-Werte im gaplosen Fall seit Anfang der ’90er bewiesen ist, macht der Fall mit Gaps nach wie vor Schwierigkeiten.

Man vermutet aber (und konnte auch für gewisse Fälle beweisen), dass ebenfalls eine Asymptotik der Form

$$e\text{-Wert} \sim nm \cdot \widetilde{k} \cdot e^{-\widetilde{\lambda} S}$$

gilt. Die Werte $\widetilde{k}$ und $\widetilde{\lambda}$ sind aber andere als im gaplosen Fall. Sie werden durch Simulationsstudien und Datenbankvergleiche bestimmt.

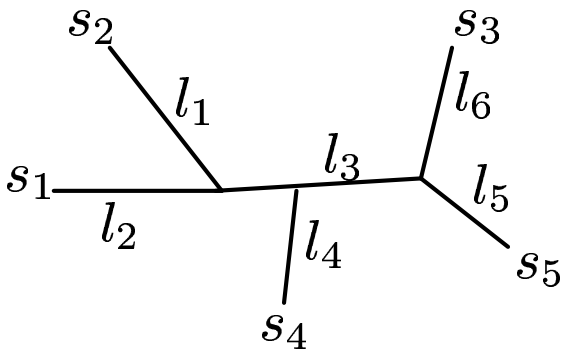
Distanzbasierte Phylogenieschätzung

Gegeben ist eine Menge $\{s_1, s_2, \ldots, s_n\}$ (z.B. von Sequenzen, Arten, Individuen, ...) und eine Distanzmatrix $(d_{ij})_{ij \leq n}$, d.h. es gilt für alle $i, j, k \leq n$:

$$d_{ij} = d_{ji}, \qquad d_{ij} + d_{jk} \geq d_{ik} \quad \text{(Dreieckungleichung)}, \qquad i = j \Leftrightarrow d_{ij} = 0$$

d_{ij} ist der (evtl. geschätzte) Abstand zwischen s_i und s_j.

Gesucht: ein binärer Baum, dessen Blätter mit $s_1, s_2, \ldots, s_n$ und dessen Kanten mit Längen $l_1, l_2, \ldots, l_k \in \mathbb{R}_{>0}$ beschriftet sind, so dass die Abstände der Knoten im Baum *möglichst gut* mit den d_{ij} übereinstimmen.



Distanzbasierte Phylogenieschätzung mit UPGMA

Unweighted **P**airwise **G**rouping **M**ethod using **A**rithmetic means

Clusterverfahren nach Sokal & Michener (1985)

für $i \leq n$ sei $C_i := \{s_i\}$
$\mathcal{C} := \{C_1, \ldots, C_n\}$
$m := n$ wiederhole

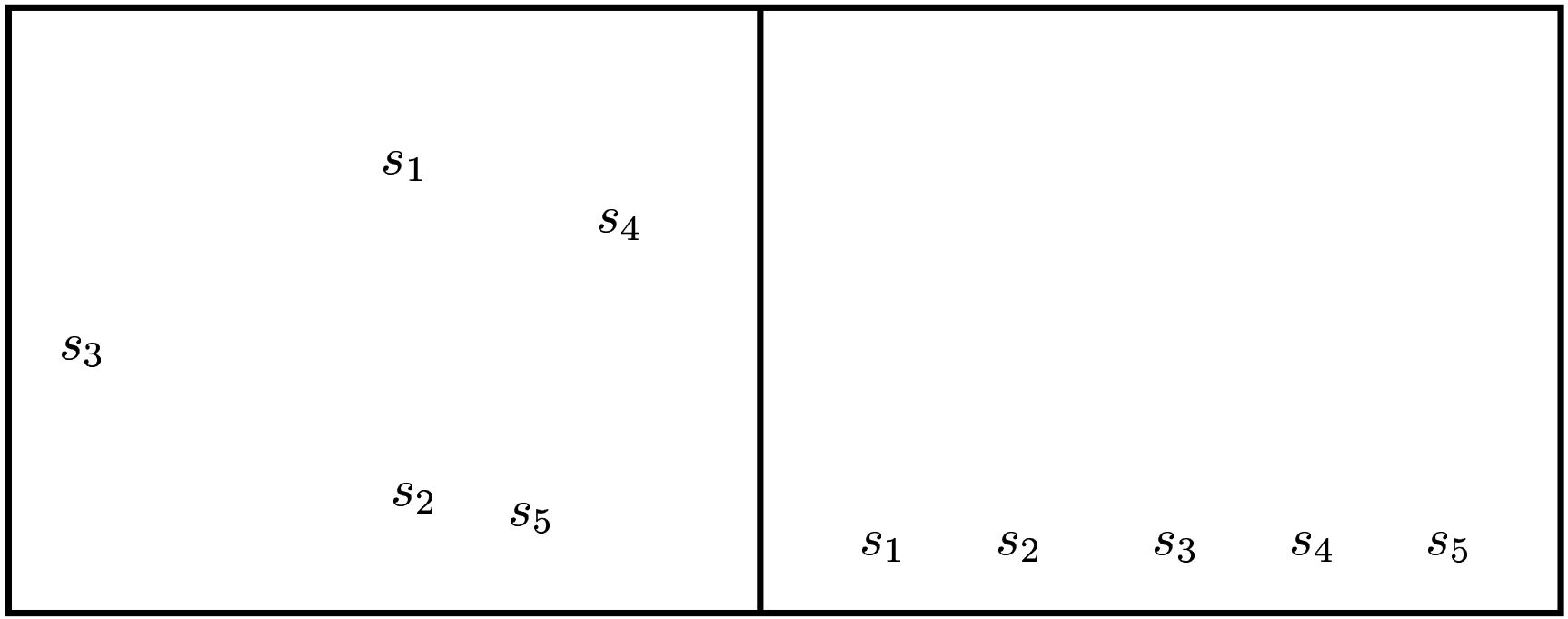
- $m := m + 1$
- Suche $C_i, C_j \in \mathcal{C}$ mit minimalem $d_{ij} > 0$
- $C_m := C_i \cup C_j$
- $\mathcal{C} := \mathcal{C} \cup \{C_m\} \setminus \{C_i, C_j\}$
- Für alle $C_k \in \mathcal{C}$ setze

$$d_{km} := d_{mk} := \frac{1}{|C_k| \cdot |C_m|} \sum_{s_x \in C_k, s_y \in C_m} d_{km}$$

bis $C_m = \{s_1, \ldots, s_n\}$.

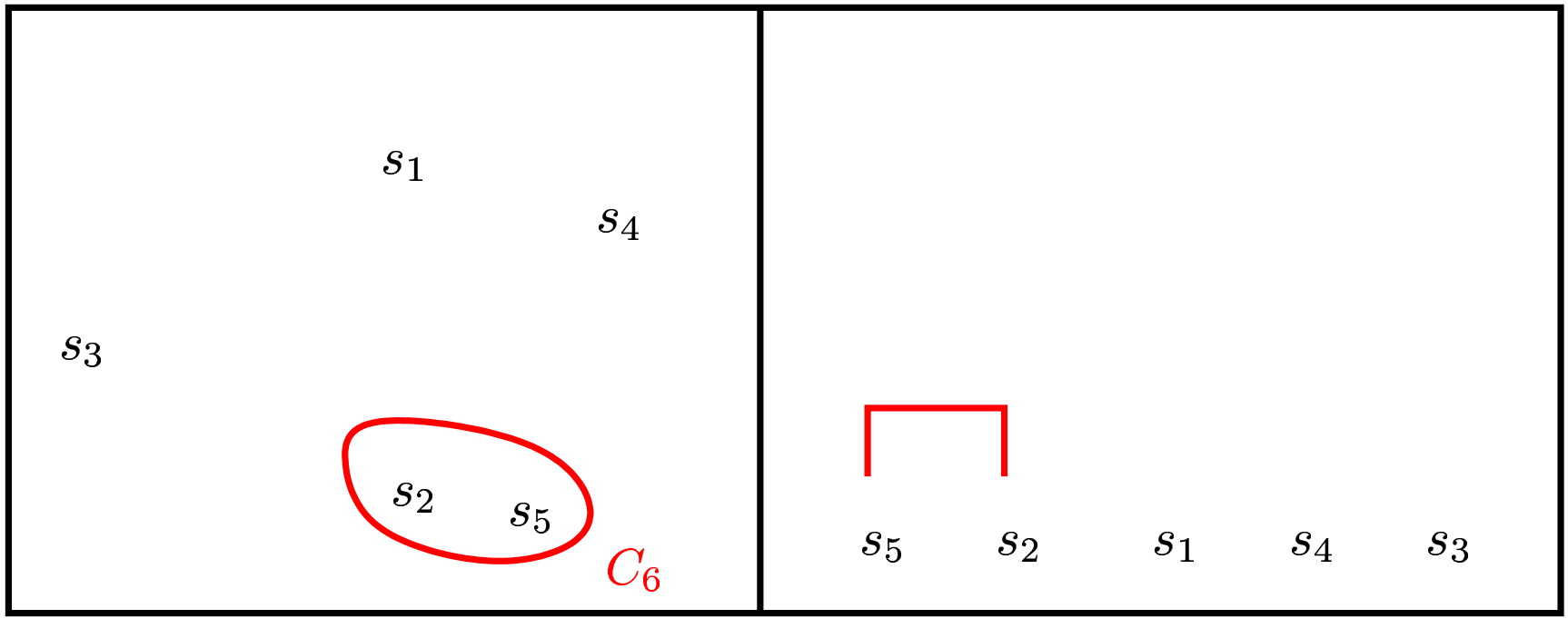
Beispiel für UPGMA

Distanzen entsprechen euklidischen Abständen auf linker Seite



Beispiel für UPGMA, Schritt 1

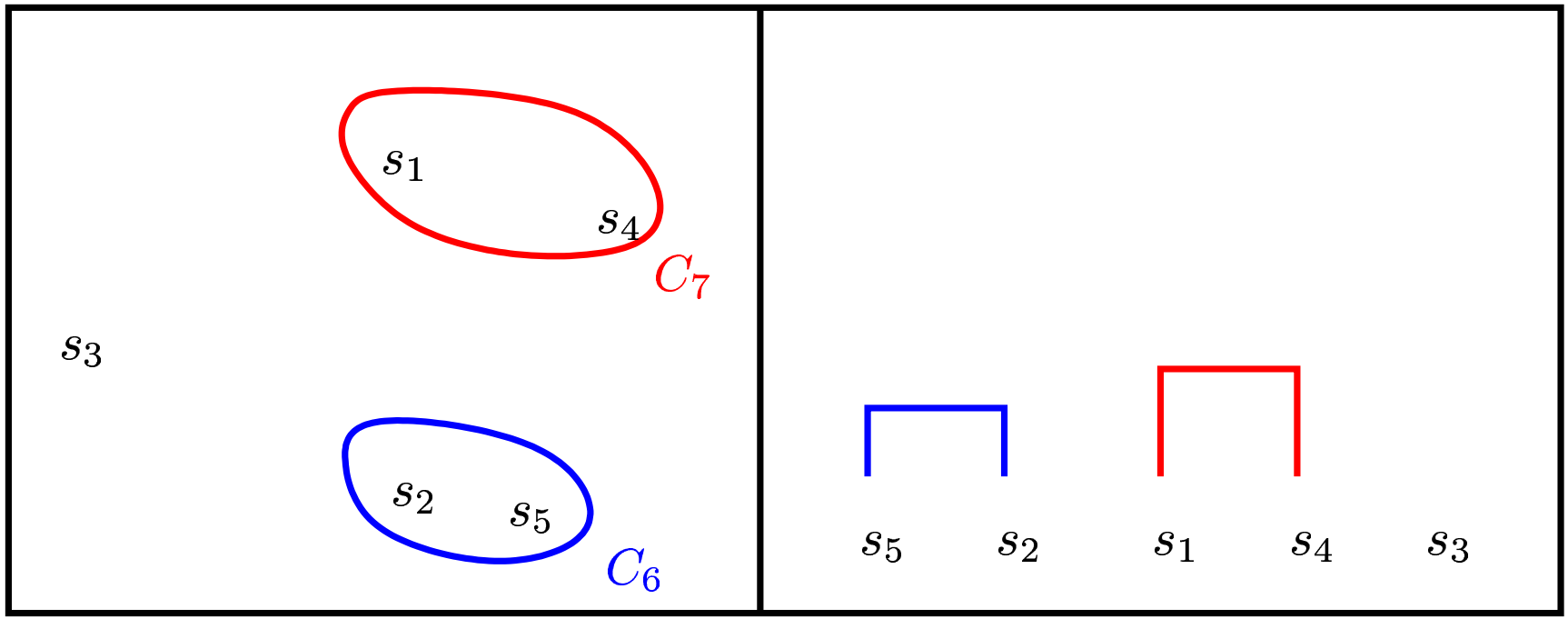
Distanzen entsprechen euklidischen Abständen auf linker Seite



Wähle Astlängen entsprechend der Abstände der s_i.

Beispiel für UPGMA, Schritt 2

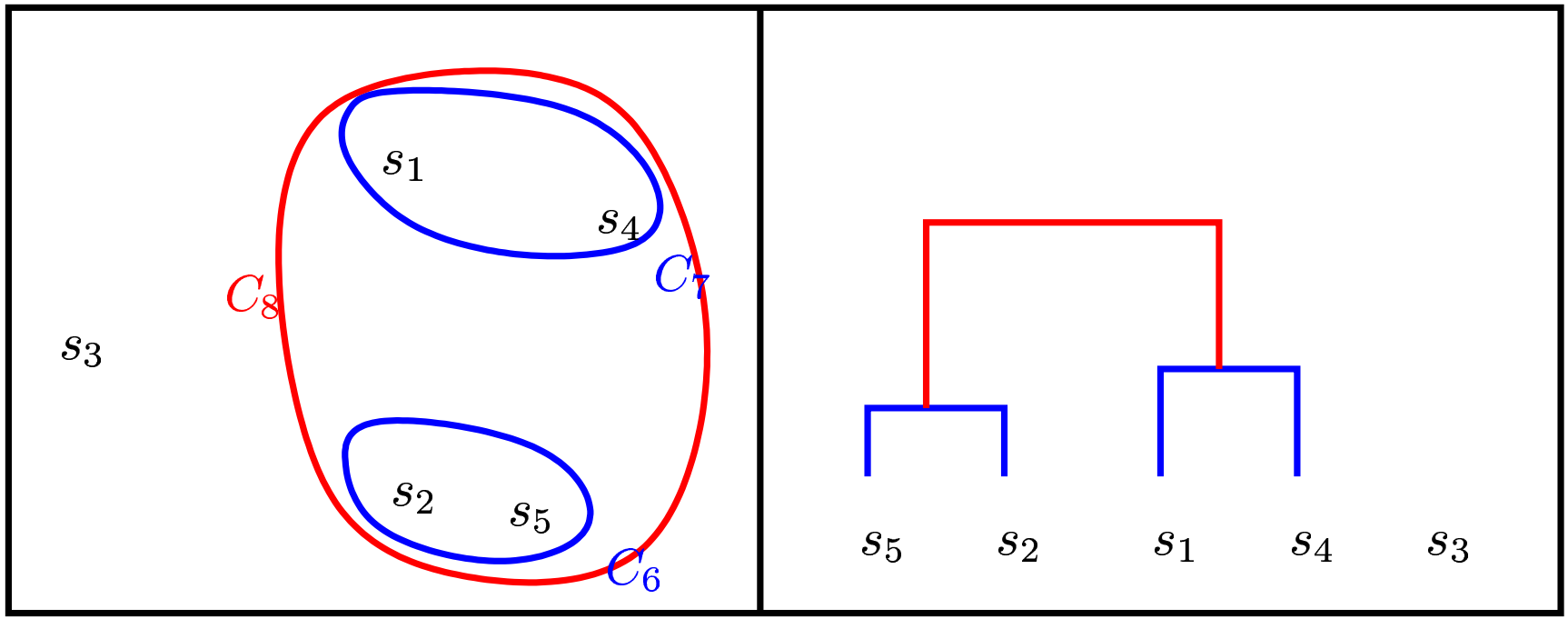
Distanzen entsprechen euklidischen Abständen auf linker Seite



Wähle Astlängen entsprechend der Abstände der s_i.

Beispiel für UPGMA, Schritt 3

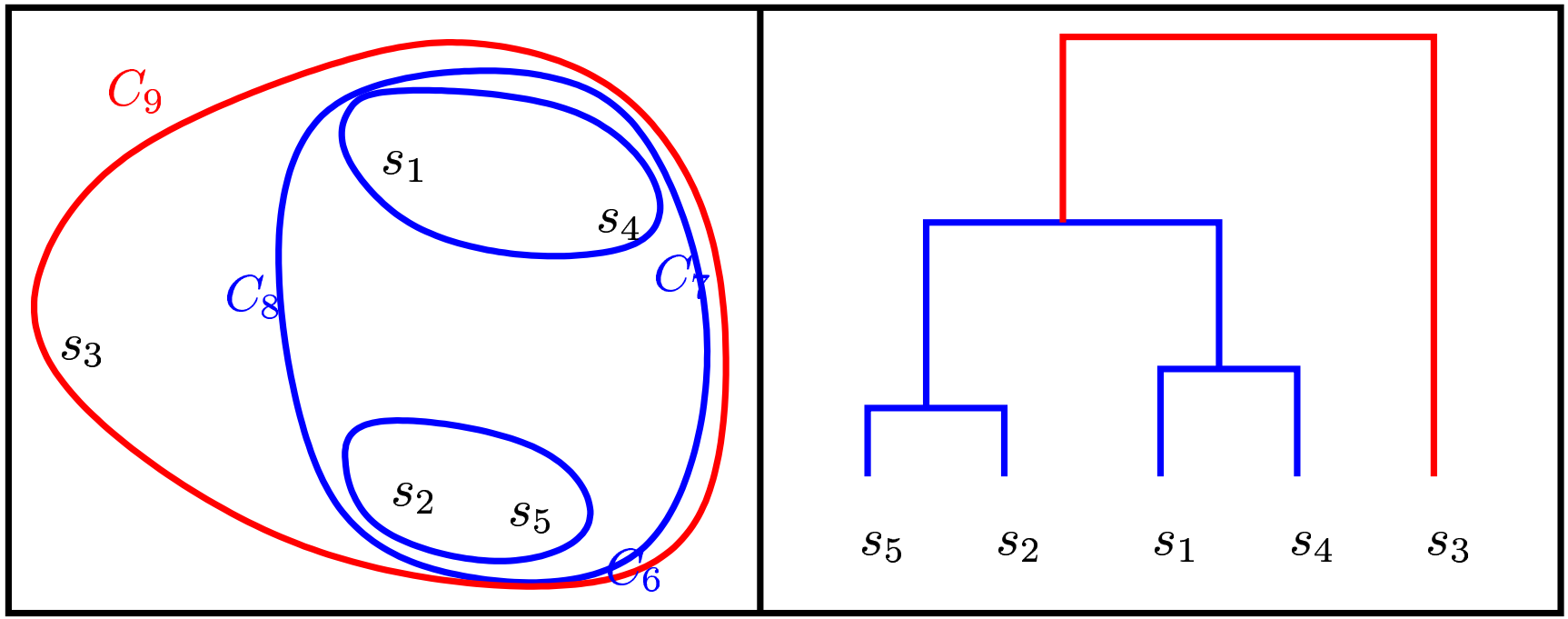
Distanzen entsprechen euklidischen Abständen auf linker Seite



Wähle Astlängen entsprechend der Abstände der Cluster.

Beispiel für UPGMA, Ergebnis

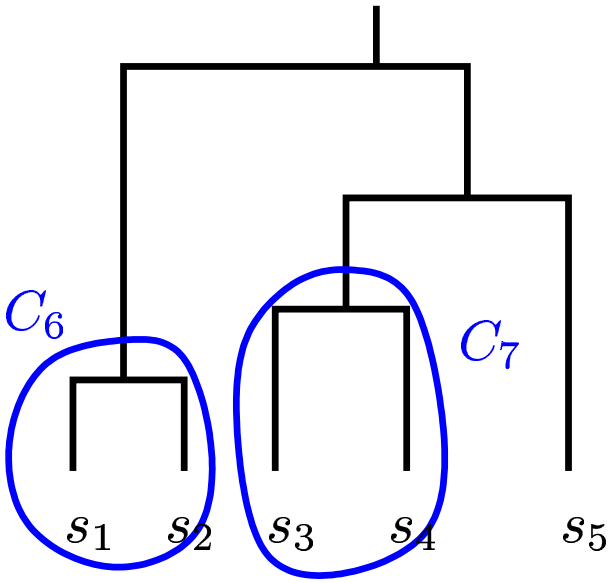
Distanzen entsprechen euklidischen Abständen auf linker Seite



Wähle Astlängen entsprechend der Abstände der Cluster.

Argument für UPGMA

Ideale Situation: Angenommen die Distanzen $\{d_{ij}\}$ sind exakt zu einem Baum kompatibel, der die Eigenschaft der molekularen Uhr erfüllt, d.h. alle Blätter sind gleich weit von einer Wurzel entfernt.



Dann gilt

$$d_{13} = d_{14} = d_{23} = d_{24}$$

Daraus folgt z.B.:

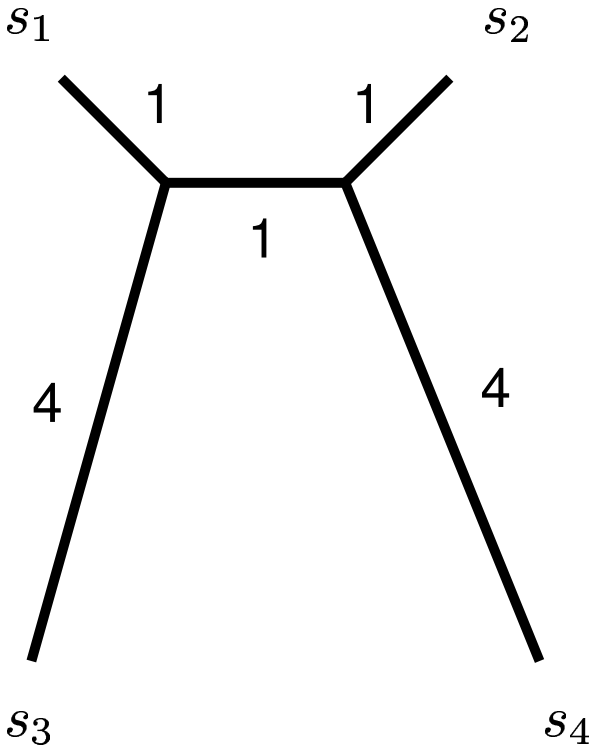
$$d_{67} = \frac{1}{2 \cdot 2} \cdot (d_{13} + d_{14} + d_{23} + d_{24}) = d_{13}$$

UPGMA behandelt also jedes Cluster wie ein Blatt und weist sinnvolle Distanzen zu.

Unter der Annahme der molekularen Uhr wird UPGMA den exakt passenden Baum finden, falls er existiert.

Argument gegen UPGMA

Betrachte als Beispiel folgenden Baum:

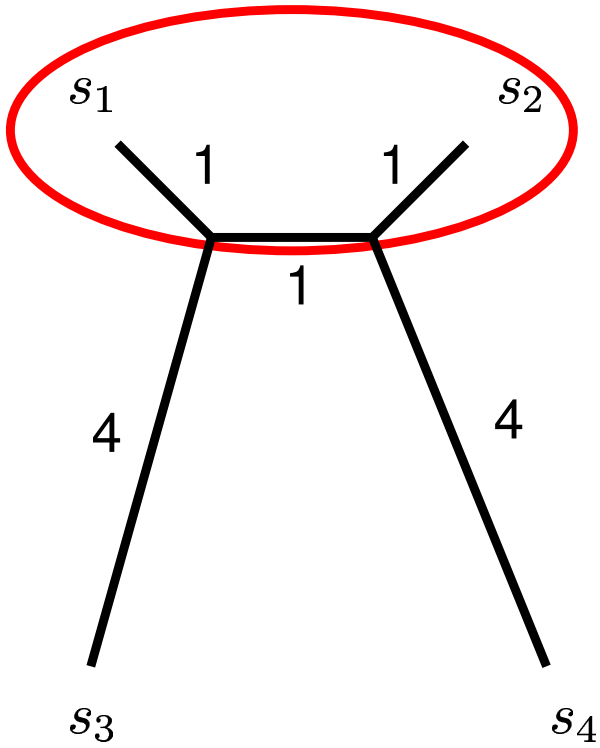


$$d_{12} = 3 \qquad d_{23} = 6$$
$$d_{13} = 5 \qquad d_{24} = 5$$
$$d_{14} = 6 \qquad d_{34} = 9$$

Was wuerde UPGMA tun?

Argument gegen UPGMA

Betrachte als Beispiel folgenden Baum:

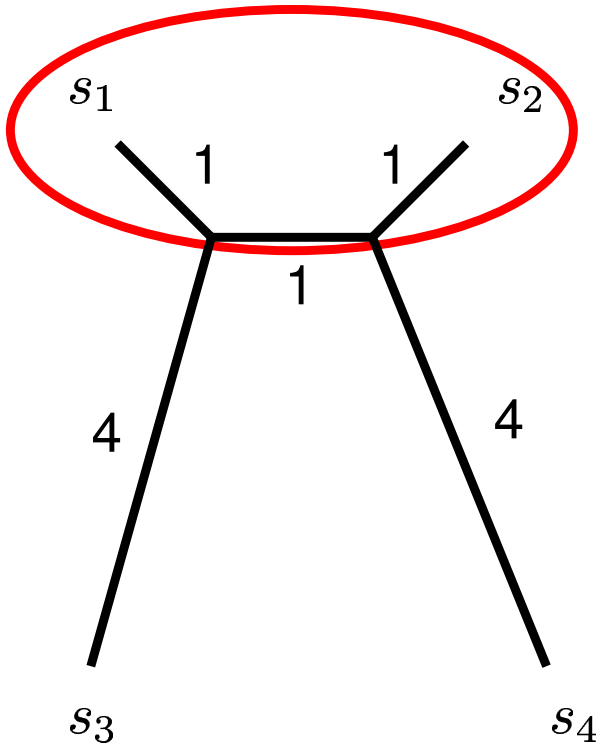


$$d_{12} = 3 \qquad d_{23} = 6$$
$$d_{13} = 5 \qquad d_{24} = 5$$
$$d_{14} = 6 \qquad d_{34} = 9$$

Was wuerde UPGMA tun? Es würde zuerst s_1 und s_2 vereinigen!

Argument gegen UPGMA

Betrachte als Beispiel folgenden Baum:



$$d_{12} = 3 \qquad d_{23} = 6$$
$$d_{13} = 5 \qquad d_{24} = 5$$
$$d_{14} = 6 \qquad d_{34} = 9$$

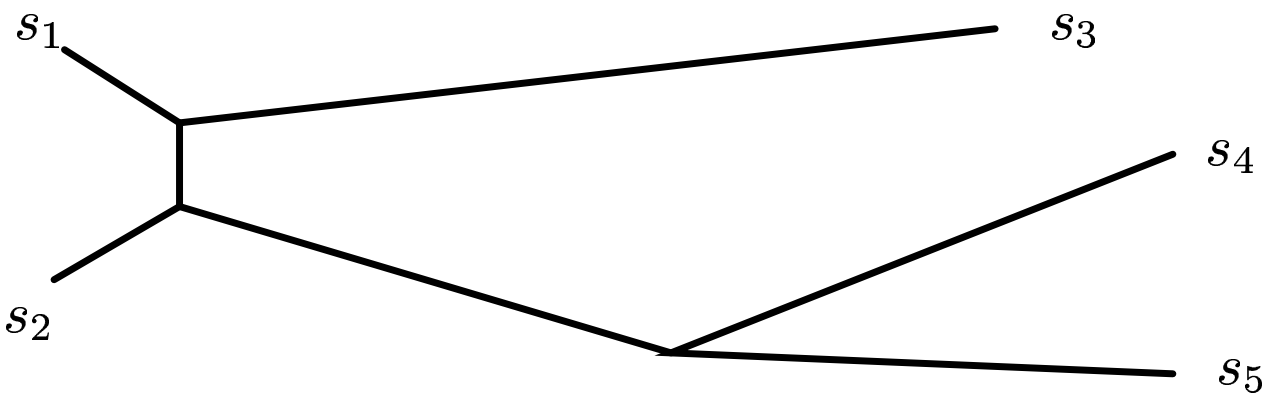
Was wuerde UPGMA tun? Es würde zuerst s_1 und s_2 vereinigen!

Gibt es einen Algorithmus, der einen exakt passenden Baum sicher findet? (sofern er existiert)

Neighbour Joining (Saitou & Nei, 1987)

Idee: Verwende gewichtete Distanzen

$$D_{ij} := d_{ij} - (r_i + r_j), \qquad \text{mit} \quad r_i = \frac{1}{n-2}\sum_k d_{ik} = \frac{n-1}{n-2} \cdot \overline{d_{i.}}$$

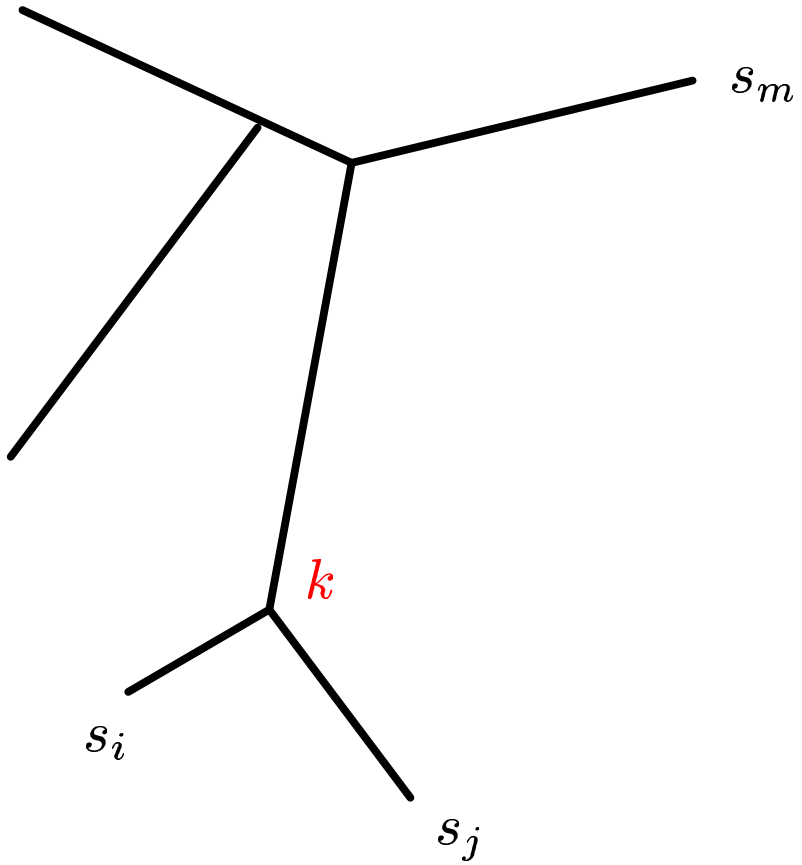


Neighbour Joining Algorithmus

Eingabe $T = \{s_1, \ldots, s_n\}$ mit Distanzen $(d_{ij})_{i,j \leq n}$

NeighbourJoining(T):

- fertig falls $n = 1$
- berechne alle D_{ij}
- finde in T die Knoten s_i und s_j mit minimalem D_{ij}
- definiere inneren Knoten k mit $\forall_m$: $d_{km} := \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$
- NeighbourJoining($\{k\} \cup T \setminus \{s_i, s_j\}$)



Satz: Falls ein Baum existiert, dessen Blätter $\{s_i\}_{i \leq n}$ exakt die Distanzen $(d_{ij})_{i,j \leq n}$ haben, wird *Neighbour Joining* diesen Baum finden.

Beweis: Nächstes Semester □

Satz: Falls ein Baum existiert, dessen Blätter $\{s_i\}_{i \leq n}$ exakt die Distanzen $(d_{ij})_{i,j \leq n}$ haben, wird *Neighbour Joining* diesen Baum finden.

Beweis: Nächstes Semester □

Problem: Meistens sind die Distanzen mit keinem Baum verträglich (z.B. weil es nur grobe Schätzungen sind).

Dann sind Verfahren erfolgreicher, die mehr Informationen als nur die ungefähren Distanzen ausnutzen!

Parsimonische Baumrekonstruktion

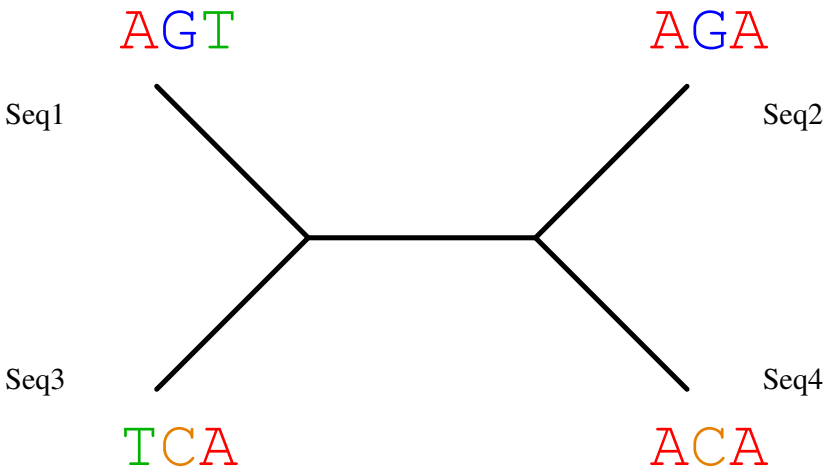
Gegeben: n homologe Sequenzen, z.B. DNA oder Proteine

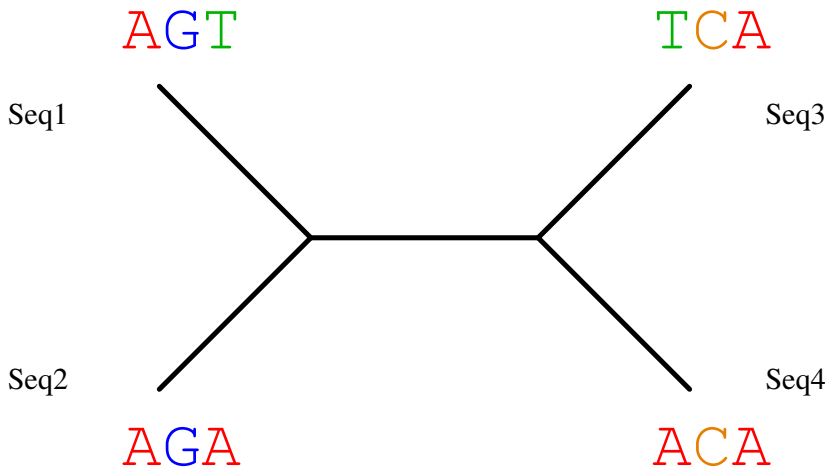
z.B. $n = 4$:

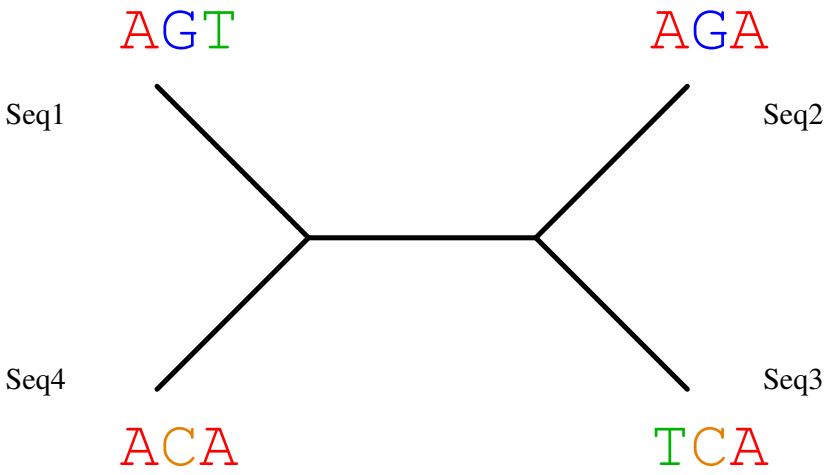
```
Seq1    GCAGGGTAC
Seq2    GCAGGGAAC
Seq3    GCTGGCAAC
Seq4    GCAGGCAAC
```

Gesucht: Stammbaum, der mit möglichst wenigen Substitutionen auskommt.

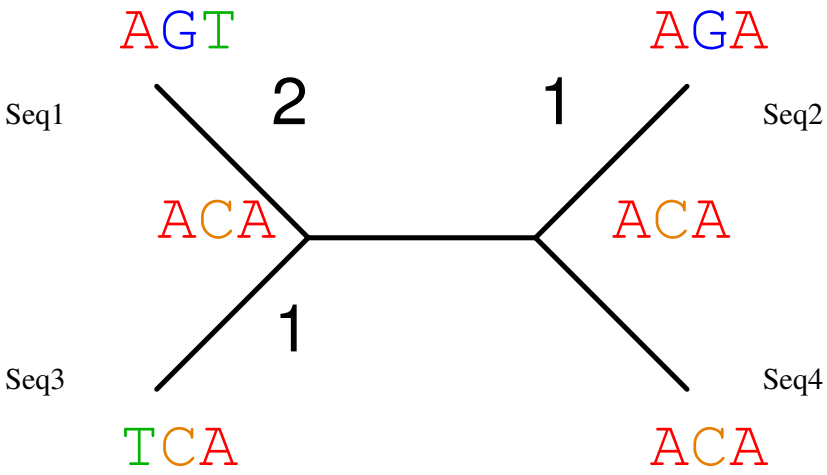
Welcher Baum ist der parsimonischte?

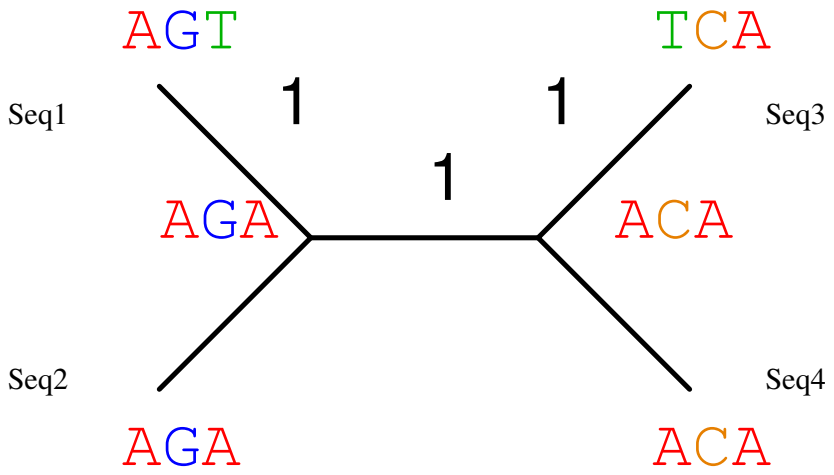


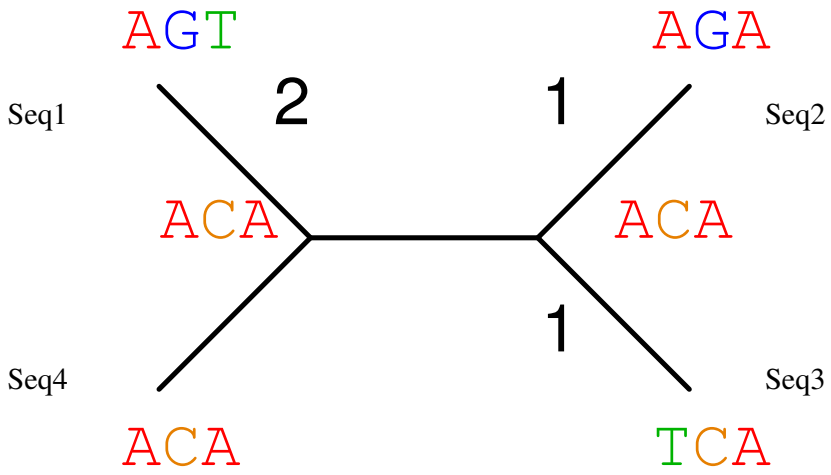




Welcher Baum ist der parsimonischte? Der da ↓





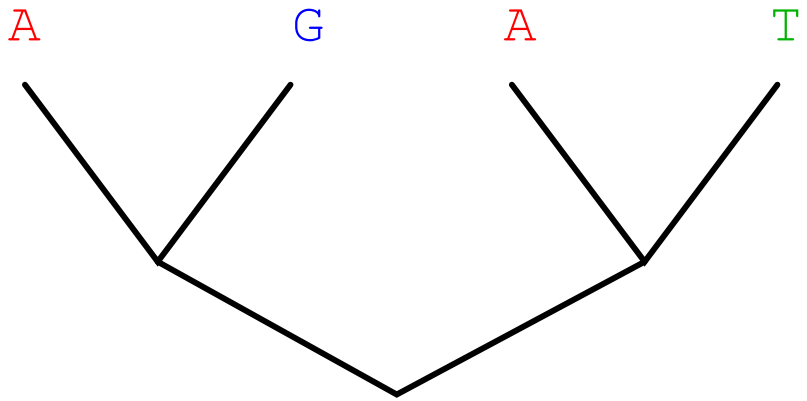


Gegeben: Baum (oBdA mit Wurzel), dessen Blätter mit Sequenzen beschriftet sind.

Gesucht: Mindestanzahl an nötigen Substitutionen

Algorithmus von Fitch (1971)

Idee: Beschrifte jeden Knoten k mit Menge M_k aller Beschriftungen, die optimal parsimonisch für darüberliegenden Teilbaum wären. (dynamische Programmierung!)

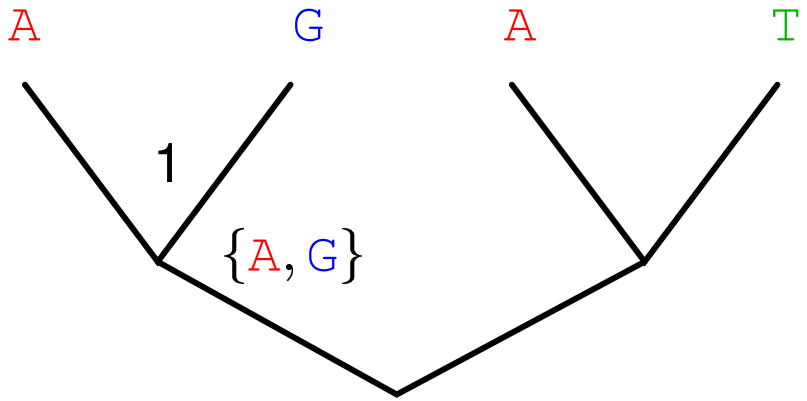


Gegeben: Baum (oBdA mit Wurzel), dessen Blätter mit Sequenzen beschriftet sind.

Gesucht: Mindestanzahl an nötigen Substitutionen

Algorithmus von Fitch (1971)

Idee: Beschrifte jeden Knoten k mit Menge M_k aller Beschriftungen, die optimal parsimonisch für darüberliegenden Teilbaum wären. (dynamische Programmierung!)

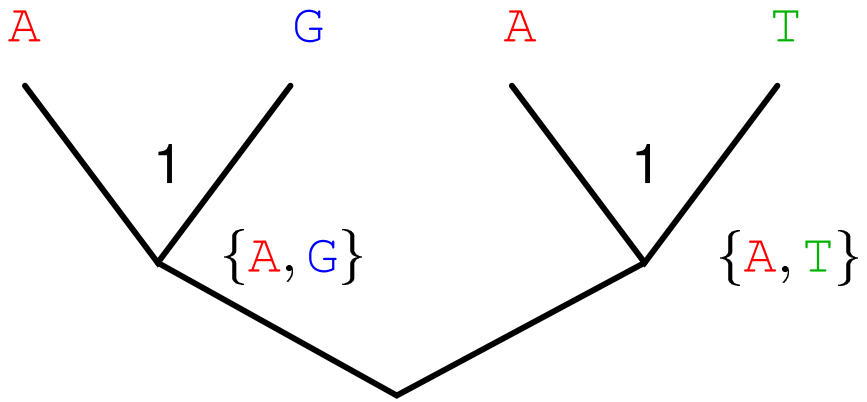


Gegeben: Baum (oBdA mit Wurzel), dessen Blätter mit Sequenzen beschriftet sind.

Gesucht: Mindestanzahl an nötigen Substitutionen

Algorithmus von Fitch (1971)

Idee: Beschrifte jeden Knoten k mit Menge M_k aller Beschriftungen, die optimal parsimonisch für darüberliegenden Teilbaum wären. (dynamische Programmierung!)

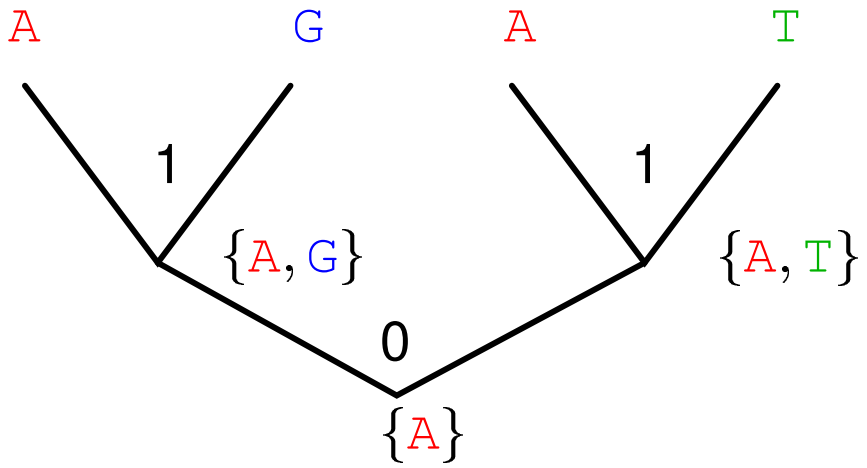


Gegeben: Baum (oBdA mit Wurzel), dessen Blätter mit Sequenzen beschriftet sind.

Gesucht: Mindestanzahl an nötigen Substitutionen

Algorithmus von Fitch (1971)

Idee: Beschrifte jeden Knoten k mit Menge M_k aller Beschriftungen, die optimal parsimonisch für darüberliegenden Teilbaum wären. (dynamische Programmierung!)



Algorithmus von Fitch, formal:

Führe folgende Schritte für jede Position aus:

$C := 0$

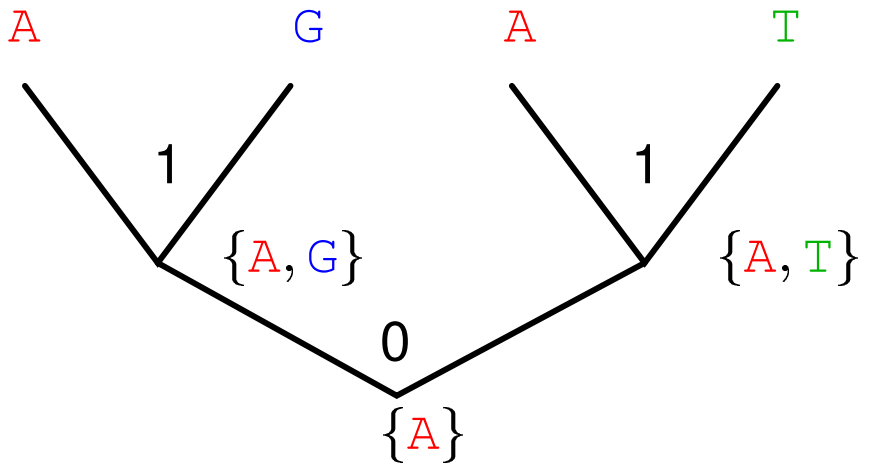
Für jedes Blatt b mit Beschriftung x setze $M_b := \{x\}$

Gehe “von oben nach unten” alle Knoten k mit Kindern i, j durch und setze:

Falls $M_i \cap M_j = \emptyset$:

- $M_k := M_i \cup M_j$
- $C := C + 1$

Sonst: $M_k := M_i \cap M_j$

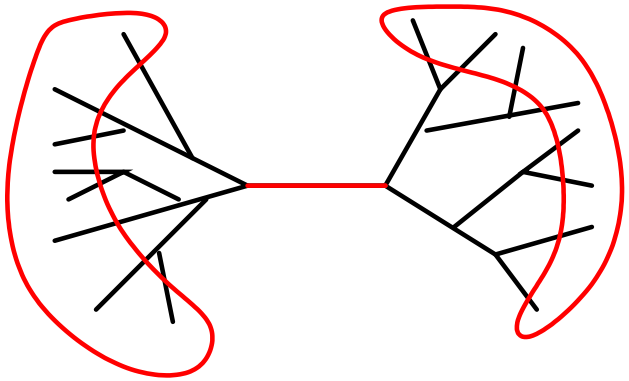


Problem der perfekten Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen 1 oder 2 verschiedene Zustände vor

gesucht: Gibt es einen perfekt parsimonischen Baum, d.h. einen in dem für jede Position höchstens eine Substitution vorkommt?

Lösung: Eine Kante des Baumes ist dasselb wie “Split” der Menge der Blätter in zwei Teilemengen.



Also: Unterteile die Menge der Blätter nach und nach für jede segregierende Position weiter auf bis ein Widerspruch auftritt – oder eben nicht.

verallgemeintertes Problem der perfekten Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen bis zu r verschiedene Zustände vor.

gesucht: Gibt es einen perfekt parsimonischen Baum, d.h. einen in dem an jeder Position weder Rückmutationen noch zwei Substitutionen zum selben Symbol vorkommen?

Komplexität: Falls r groß werden darf, ist dieses Problem NP-vollständig.

Aber: Für jedes feste $r \in \mathbb{N}$ existieren Polynomialzeitalgorithmen.

Problem der maximalen Parsimonie

gegeben: n homologe Sequenzen der Länge m. An jeder Position kommen 1 oder 2 verschiedene Zustände vor.

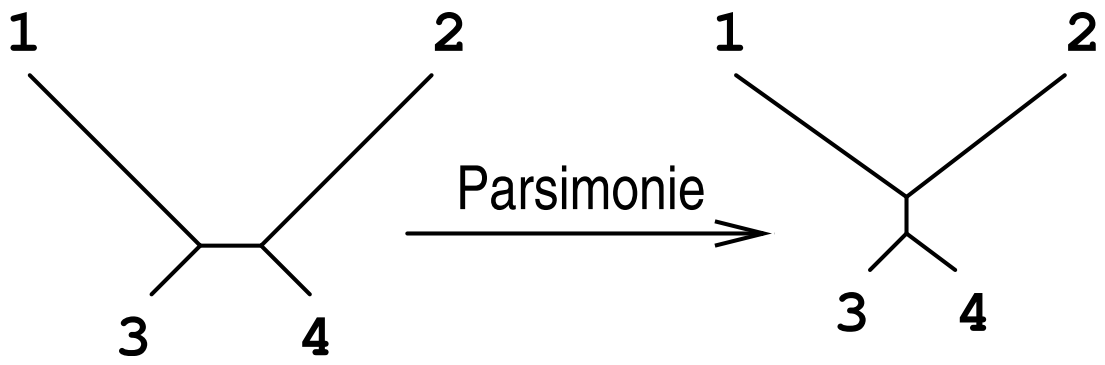
gesucht: ein Baum, der die minimale Anzahl an Substitutionen braucht

Komplexität: NP-vollständig

Aber: immerhin gibt es einen Algorithmus, der einen Baum liefert, der mit weniger als dem Doppelten der Mindestanzahl auskommt.

Grenzen des Parsimonie-Prinzips

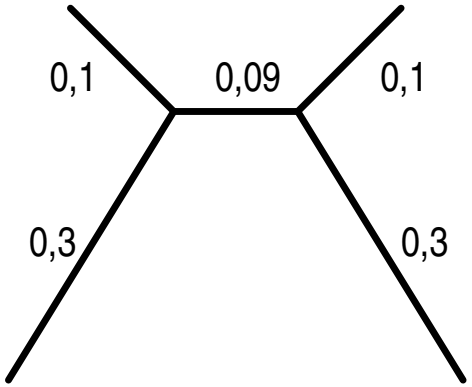
Problematisch wird's wenn Äste so lang sind, dass Rück- und Doppelmutationen auftreten. Die Länge socher Äste wird systematisch unterschätzt.



Besser wäre also ein Verfahren, das die Möglichkeit von Rück- und Doppelmutationen berücksichtigt.

Vergleich von Verfahren zur Phylogenieschätzung

Durbin et al. (1998) haben für mehrere Sequenzlängen je 1000 Quartette von $\{A, B\}$-Sequenzen längs dieses Baumes simuliert und damit die Verfahren verglichen. Die Astlängen sind mittlere Mutationshäufigkeiten pro Position.



Anteil richtig geschätzter Bäume:

Seq.läng.	Max.Pars.	Neigh.Join.	ML
20	39.6%	47.7%	41.9%
100	40.5%	63.5%	63.8%
500	40.4%	89.6%	90.4%
2000	35.3%	99.5%	99.7%

Das Maximum-Likelihood-Prinzip (ML)

Modell: Die Daten D sind Ergebnis eines zufälligen Prozesses. Ihre Wahrscheinlichkeit $\Pr_\theta(D)$ hängt vom unbekannten Parameter θ ab.

Gesucht: θ soll geschätzt werden!

ML-Prinzip: Der Schätzer $\widehat{\theta}$ ist der Wert, für den die Likelihood-Funktion

$$L_D(\theta) := \Pr_\theta(D)$$

maximal wird.

Das ML-Prinzip bei der Phylogenieschätzung

$D =$ Sequenzdaten (n homologe Sequenzen der Länge m)

$\theta =$ ein Baum

ML-Prinzip: $\widehat{\theta}$ ist der Baum, der $L_D(.)$ maximiert (der also die Sequenzdaten am wahrscheinlichsten macht).

Damit $L_D(\theta) := \Pr_\theta(D)$ definiert ist, benötigt man ein stochastisches Modell für den Substitutionsprozess, der längs der Kanten eines Baumes θ abläuft.

Spezifiziere also die Wahrscheinlichkeit

$$P_{x \to y}(t),$$

dass an einer Position, an der ein x steht, nach Zeit t ein y steht.

z.B.: einfachstes Modell für DNA-Evolution: Jukes-Cantor

All models are wrong

— but some are useful.

G.E.P. Box

DNA-Substitutionsmodell von Jukes & Cantor (1969)

Basenhäufigkeiten in Ursequenz: $(\pi(\mathtt{A}), \pi(\mathtt{C}), \pi(\mathtt{G}), \pi(\mathtt{T})) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$

Alle Position mutieren unabhängig voneinander mit Rate 1, d.h. die Wartzeit T einer Position auf die nächste Mutation ist exponential-1-verteilt, also:

$$\Pr(T > t) = e^{-t}$$

Wenn eine Mutation geschieht, wird die Base durch eine rein zufällige aus $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ ersetzt (bleibt also evtl. unverändert). Es folgt:

$$P_{x \to y}(t) = \begin{cases} (1 - e^{-t}) \cdot \frac{1}{4} & \text{falls } x \neq y \\ e^{-t} + (1 - e^{-t}) \cdot \frac{1}{4} & \text{falls } x = y \end{cases}$$

Jukes-Cantor ist sehr einfach!

Allgemeinere Modelle für DNA-Substitutionen (z.B. von Hasegawa, Kishino & Yano, 1985) berücksichtigen die ungleichen Basenhäufigkeiten und dass Transitionen eine höhere Rate haben als Transversionen.

Auch z.B. die Positionsabhängigkeit der Mutationsraten kann berücksichtigt werden (Tamura & Nei, 1993).

Substitutionsmodelle für Proteine sind implizit durch Score-Matrizen wie PAM und BLOSUM gegeben.

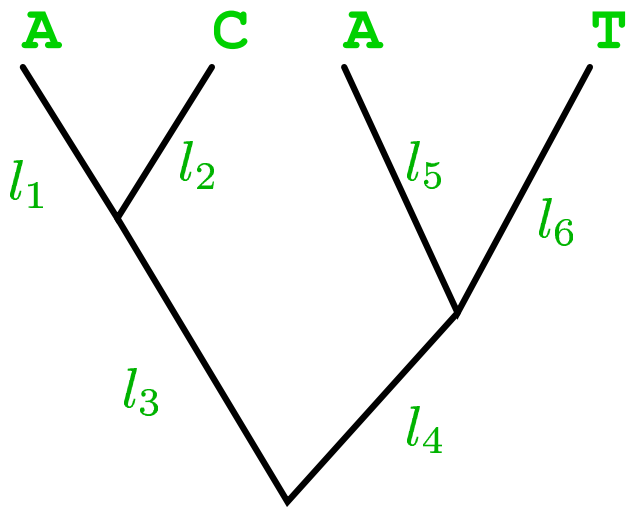
Berechnung der Likelihood eines Baums

Gegeben: (gewurzelter) Baum θ mit Astlängen $(l_i)_i$, Sequenzdaten D.

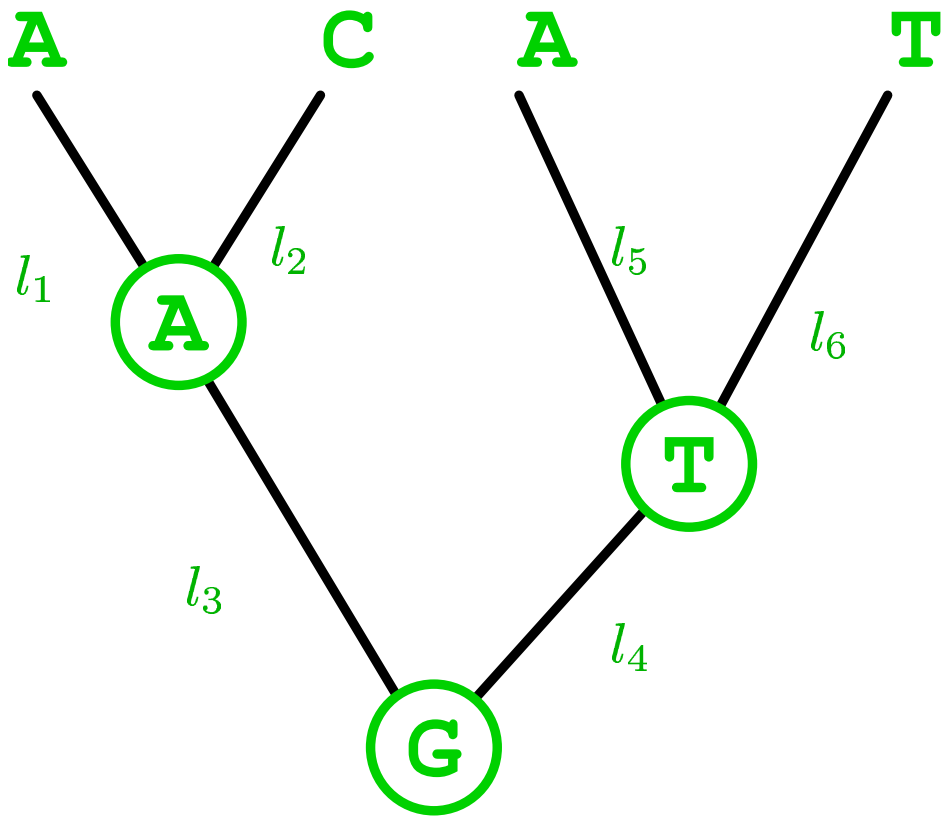
Gesucht: Die Likelihood $L_D(\theta) = \Pr_\theta(D)$ des Baumes für die gegeben Daten.

erste Idee: Berechne Likelihood positionsweise und multipliziere.

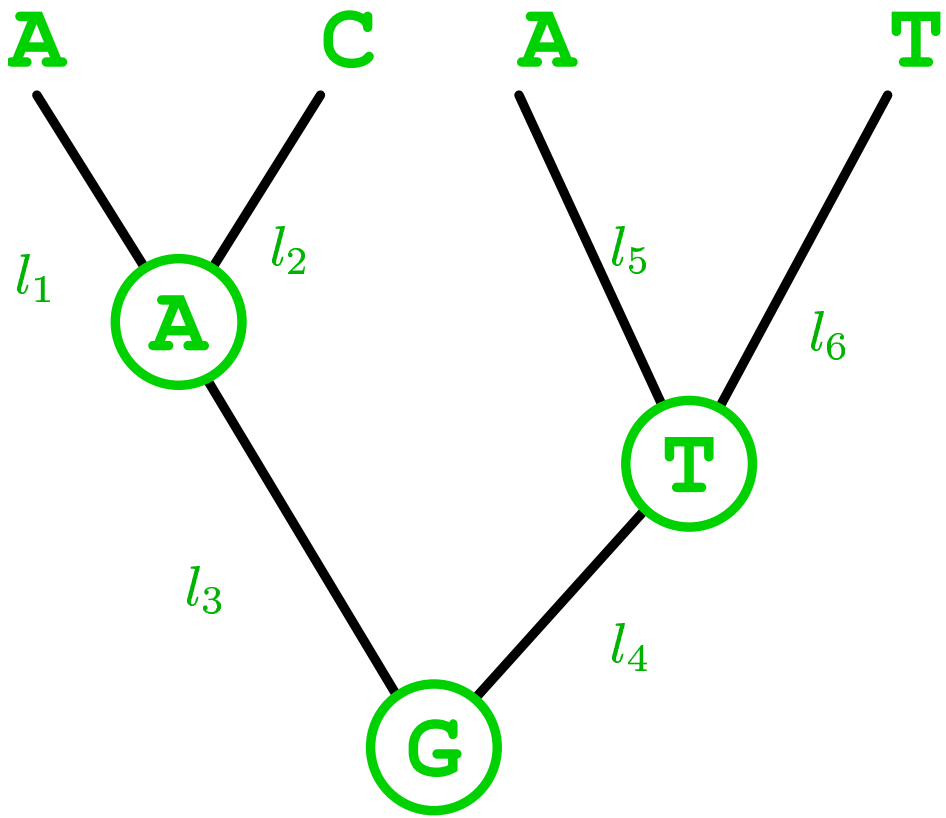
oBdA sei also die Sequenzlänge $m = 1$.



Angenommen D kennt auch die inneren Knoten:



Angenommen D kennt auch die inneren Knoten:



Dann ist es einfach:

$$L_D(\theta) = \pi(\mathtt{G}) \cdot P_{\mathtt{G}\to\mathtt{A}}(l_3) \cdot P_{\mathtt{G}\to\mathtt{T}}(l_4) \cdot P_{\mathtt{A}\to\mathtt{A}}(l_1) \cdot P_{\mathtt{A}\to\mathtt{C}}(l_2) \cdot P_{\mathtt{T}\to\mathtt{A}}(l_5) \cdot P_{\mathtt{T}\to\mathtt{T}}(l_6)$$

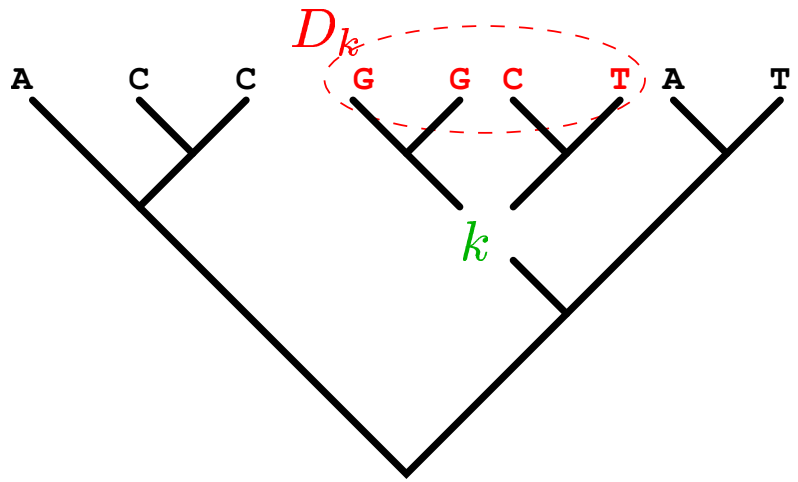
Was tun wenn die inneren Knoten nicht bekannt sind?

Felsensteins Pruning-Algorithmus (1981)

Wiedermal dynamische Programmierung!

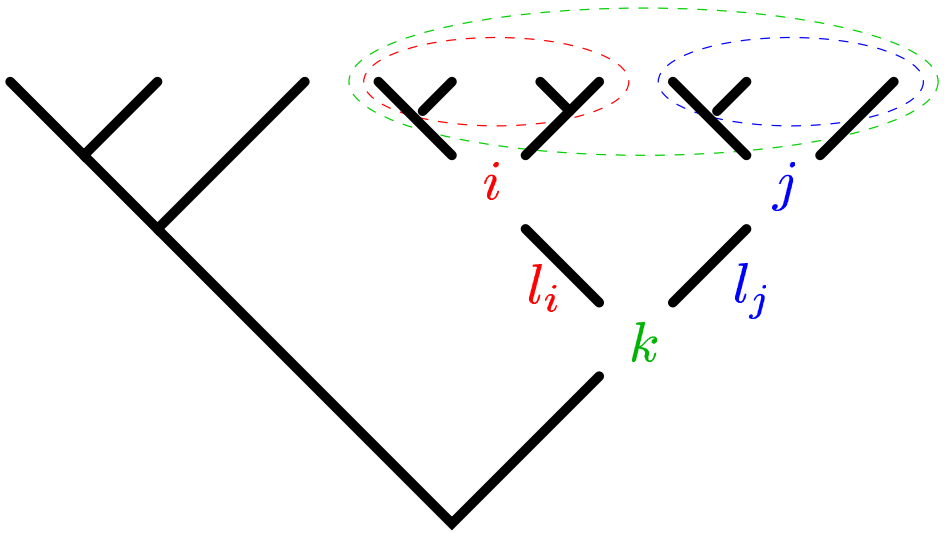
D_k der Datensatz an den Blättern über dem Knoten k.

Für Base x sei $W_k(x)$ die Wahrscheinlichkeit von D_k unter der Bedingung, dass bei k ein x steht.



Ist b ein Blatt mit Base y, so ist $W_b(y) = 1$ und $W_b(x) = 0$ für $x \neq y$.

Die Rekursion im Pruning-Algorithmus



Ist k ein Knoten mit Kindern i und j und Astlängen l_i und l_j, so gilt:

$$W_k(x) = \left(\sum_{y \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} P_{x \to y}(l_i) \cdot W_i(y) \right) \cdot \left(\sum_{z \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} P_{x \to z}(l_j) \cdot W_j(z) \right)$$

Felsensteins Pruning-Algorithmus: Letzter Schritt

Berechne mit obiger Rekursion von den Blättern bis zur Wurzel alle $W_k(x)$ (dynamische Programmierung!).

Ist r die Wurzel, so gilt $D_r = D$.

Damit erhalten wir das Ergebnis:

$$L_D(\theta) = \sum_{x \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \pi(x) \cdot W_r(x)$$

Schwieriges Problem

Gegeben n Sequenzen. Finde den Baum mit der höchsten Likelihood.

einige Lösungsansätze:

- Falls n klein ist: probiere alle Baumtopologien durch und variiere die Astlängen
- Gehe z.B. von NeighbourJoining-Baum aus und versuche durch leichtes Verändern des Baums nach und nach die Likelihood zu erhöhen.
- Nimm jeweils 4 Sequenzen und suche deren ML-Baum. Baue am Ende die Viererbäume zusammen. (“PUZZLE”; Strimmer, von Haeseler, 1996, `http://www.tree-puzzle.de/`)