UA RUHR

Algorithmen auf Sequenzen

Paarweiser Sequenzvergleich: Erweiterungen und Verbesserungen

Sven Rahmann

Genominformatik
Universitätsklinikum Essen
Universität Duisburg-Essen
Universitätsallianz Ruhr

Erweiterungen und Verbesserungen

UA RUHR

Zu den Grundtypen des paarweisen Sequenzalignments lassen sich verschiedene Verallgemeinerungen und Verbesserungen betrachten, wie

- allgemeinere Gapkosten (insbes. affine Gapkosten)
- Alignments unter Nebenbedingungen
- Alignment-Berechnung mit linearem Platzbedarf
- Längennormalisiertes Alignment

Affine Gapkosten

- Gaps sind Insertionen oder Deletionen.
- Bisher hat ein Gap der Länge ℓ einen Score von $g(\ell) = -\gamma \cdot \ell$, wobei $\gamma \geq 0$ die Gapkosten sind (negativer Score eines Gaps).
- Diese Annahme ist z.B. bei biologischen Sequenzen nicht realistisch.

```
attccgacagaaagatac        attccgacagaaagatac
att-c--c-g----atac   vs.  attccg--------atac
```

Affine Gapkosten

- Gaps sind Insertionen oder Deletionen.
- Bisher hat ein Gap der Länge ℓ einen Score von $g(\ell) = -\gamma \cdot \ell$, wobei $\gamma \geq 0$ die Gapkosten sind (negativer Score eines Gaps).
- Diese Annahme ist z.B. bei biologischen Sequenzen nicht realistisch.

```
attccgacagaaagatac        attccgacagaaagatac
att-c--c-g----atac   vs.  attccg--------atac
```

- Sinnvoller ist, die Kosten für das Eröffnen eines Gaps höher anzusetzen als die für das Verlängern eines Gaps.

Affine Gapkosten

- Eine relativ einfache, aber schon relativ realistische Modellierung erreichen wir durch **affine Gapkosten**

$$g(\ell) := -c - \gamma(\ell - 1) \text{ für } \ell \geq 1.$$

- Dabei gibt die Konstante $c \geq 0$ die Gapöffnungskosten (**gap open penalty**) und $\gamma \geq 0$ die Gaperweiterungskosten (**gap extension penalty**) an. Sinnvoll ist $c \geq \gamma \geq 0$.

Affine Gapkosten

- Eine relativ einfache, aber schon relativ realistische Modellierung erreichen wir durch **affine Gapkosten**

$$g(\ell) := -c - \gamma(\ell - 1) \text{ für } \ell \geq 1.$$

- Dabei gibt die Konstante $c \geq 0$ die Gapöffnungskosten (**gap open penalty**) und $\gamma \geq 0$ die Gaperweiterungskosten (**gap extension penalty**) an.
  Sinnvoll ist $c \geq \gamma \geq 0$.
- Direkte Erweiterung des Alignment-Graphen:
  Füge **alle** horizontalen und vertikalen Kanten (über alle Gaplängen) ein.
  Damit **beliebige** Gapkosten möglich, aber Laufzeit $\mathcal{O}(mn(m+n))$ bzw. $\mathcal{O}(n^3)$.

Globales Alignment mit affinen Gapkosten

- Ziel: Algorithmus mit Laufzeit $\mathcal{O}(mn)$
- Dazu: Nachhalten, ob Gap bereits geöffnet

Globales Alignment mit affinen Gapkosten

- Ziel: Algorithmus mit Laufzeit $\mathcal{O}(mn)$
- Dazu: Nachhalten, ob Gap bereits geöffnet
- $S[i][j] := \max\left\{ score(A) \mid \begin{array}{l} A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j] \end{array} \right\}$.
- $V[i][j] := \max\left\{ score(A) \mid \begin{array}{l} A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j] \\ \text{das mit einem Gap } (-) \text{ in } t \text{ endet} \end{array} \right\}$,
- $H[i][j] := \max\left\{ score(A) \mid \begin{array}{l} A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j] \\ \text{das mit einem Gap } (-) \text{ in } s \text{ endet} \end{array} \right\}$.

Globales Alignment mit affinen Gapkosten

- Ziel: Algorithmus mit Laufzeit $\mathcal{O}(mn)$
- Dazu: Nachhalten, ob Gap bereits geöffnet
- $S[i][j] := \max\{score(A) \mid A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j]\}$.
- $V[i][j] := \max\left\{score(A) \mid \begin{array}{l} A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j] \\ \text{das mit einem Gap } (-) \text{ in } t \text{ endet} \end{array}\right\}$,
- $H[i][j] := \max\left\{score(A) \mid \begin{array}{l} A \text{ ist ein Alignment von } s[:i] \text{ und } t[:j] \\ \text{das mit einem Gap } (-) \text{ in } s \text{ endet} \end{array}\right\}$.

$$
\begin{aligned}
V[i][j] &= \max\{S[i-1][j] - c,\ V[i-1][j] - \gamma\},\\
H[i][j] &= \max\{S[i][j-1] - c,\ H[i][j-1] - \gamma\},\\
S[i][j] &= \max\{S[i-1][j-1] + \text{score}(s[i-1], t[j-1]),\ V[i][j],\ H[i][j]\}.
\end{aligned}
$$

Globales Alignment mit affinen Gapkosten

Initialisierung:

$$
\begin{aligned}
S[0][0] &= 0, \\
V[0][j] &= -\infty \text{ für alle } 0 \leq j \leq n, \\
S[i][0] &= V[i][0] = g(i) \text{ für alle } 1 \leq i \leq m, \\
H[i][0] &= -\infty \text{ für alle } 0 \leq i \leq m, \\
S[0][j] &= H[0][j] = g(j) \text{ für alle } 1 \leq j \leq m.
\end{aligned}
$$

Globales Alignment mit affinen Gapkosten

Initialisierung:

$$
\begin{aligned}
S[0][0] &= 0, \\
V[0][j] &= -\infty \text{ für alle } 0 \leq j \leq n, \\
S[i][0] &= V[i][0] = g(i) \text{ für alle } 1 \leq i \leq m, \\
H[i][0] &= -\infty \text{ für alle } 0 \leq i \leq m, \\
S[0][j] &= H[0][j] = g(j) \text{ für alle } 1 \leq j \leq m.
\end{aligned}
$$

Traceback:
Anpassung notwendig; nachverfolgen, aus welcher Tabelle (S, V, H) Optimum stammt

Globale Alignments mit Nebenbedingungen

- Gesucht ist ein optimales Alignment, das durch gegebenen Knoten (i, j) führt.
- Wenn (i, j) nicht auf einem insgesamt optimalen Pfad liegt, ist so ein Alignment suboptimal.

Globale Alignments mit Nebenbedingungen

- Gesucht ist ein optimales Alignment, das durch gegebenen Knoten (i, j) führt.
- Wenn (i, j) nicht auf einem insgesamt optimalen Pfad liegt, ist so ein Alignment suboptimal.
- Lösung: zwei Alignments berechnen, Kosten addieren:

$$(0, 0) \to (i, j) \text{ und } (i, j) \to (m, n).$$

Globale Alignments mit Nebenbedingungen

- Gesucht ist ein optimales Alignment, das durch gegebenen Knoten (i, j) führt.
- Wenn (i, j) nicht auf einem insgesamt optimalen Pfad liegt, ist so ein Alignment suboptimal.
- Lösung: zwei Alignments berechnen, Kosten addieren:

$$(0, 0) \to (i, j) \text{ und } (i, j) \to (m, n).$$

- Optimale Scores durch (i, j) **für alle** (i, j)?

Globale Alignments mit Nebenbedingungen

- Gesucht ist ein optimales Alignment, das durch gegebenen Knoten (i, j) führt.
- Wenn (i, j) nicht auf einem insgesamt optimalen Pfad liegt, ist so ein Alignment suboptimal.
- Lösung: zwei Alignments berechnen, Kosten addieren:

$$(0, 0) \to (i, j) \text{ und } (i, j) \to (m, n).$$

- Optimale Scores durch (i, j) **für alle** (i, j)?
- Iteriere über alle (i, j), Laufzeit insgesamt $\mathcal{O}(m^2 n^2)$.
- Geht das schneller?

Globale Alignments mit Nebenbedingungen

Verbesserung:

- Optimale Scores $(0,0) \to (i,j)$ für alle (i,j) liegen vor: $S[i][j]$
- Scores $(i,j) \to (m,n)$ sind Scores $(0,0) \to (m-i, n-j)$ der reversen Strings.

Globale Alignments mit Nebenbedingungen

Verbesserung:

- Optimale Scores $(0,0) \to (i,j)$ für alle (i,j) liegen vor: $S[i][j]$
- Scores $(i,j) \to (m,n)$ sind Scores $(0,0) \to (m-i, n-j)$ der reversen Strings.
- Matrix $R[i][j]$ mit optimalen Scores von $(m,n) \to (i,j)$ (rückwärts berchnet)
- Aufpassen bei der Indizierung!
- Summiere $S + R$: optimaler Score aller Pfade durch (i,j)

Beispiel

$s =$ andi und $t =$ handy.

	ϵ	h	a	n	d	y
ϵ	0	-1	-2	-3	-4	-5
a	-1	-1	0	-1	-2	-3
n	-2	-2	-1	1	0	-1
d	-3	-3	-2	0	2	1
i	-4	-4	-3	-1	1	1

1	2	0	-2	-4	-4	a
-1	0	1	-1	-3	-3	n
-3	-2	-1	0	-2	-2	d
-5	-4	-3	-2	-1	-1	i
-5	-4	-3	-2	-1	0	ϵ
h	a	n	d	y	ϵ	

Beispiel

$s =$ andi und $t =$ handy.

	ϵ	h	a	n	d	y
ϵ	0	-1	-2	-3	-4	-5
a	-1	-1	0	-1	-2	-3
n	-2	-2	-1	1	0	-1
d	-3	-3	-2	0	2	1
i	-4	-4	-3	-1	1	1

1	2	0	-2	-4	-4	a
-1	0	1	-1	-3	-3	n
-3	-2	-1	0	-2	-2	d
-5	-4	-3	-2	-1	-1	i
-5	-4	-3	-2	-1	0	ϵ
h	a	n	d	y	ϵ	

$\searrow$ $+$ $\swarrow$

1	1	-2	-5	-8	-9
-2	-1	1	-2	-5	-6
-5	-4	-2	1	-2	-3
-8	-7	-5	-2	1	0
-9	-8	-6	-3	0	1

Globale Alignments mit linearem Platzbedarf

- Wir können den optimalen Alignmentscore mit $\mathcal{O}(\min(m, n))$ Platz berechnen.
- Für das optimale Alignment benötigen wir (bisher) jedoch die vollständige Traceback-Matrix T, die $\mathcal{O}(mn)$ Platz benötigt.

Globale Alignments mit linearem Platzbedarf

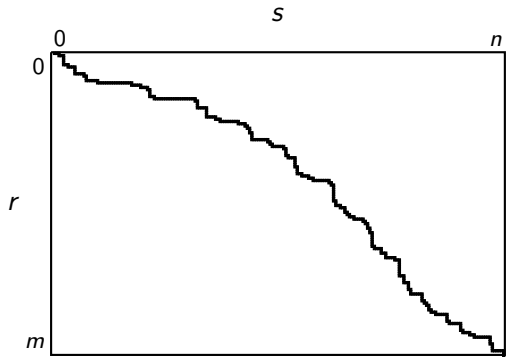
- Wir können den optimalen Alignmentscore mit $\mathcal{O}(\min(m, n))$ Platz berechnen.
- Für das optimale Alignment benötigen wir (bisher) jedoch die vollständige Traceback-Matrix T, die $\mathcal{O}(mn)$ Platz benötigt.
- Nach einer Idee von Daniel S. Hirschberg aus dem Jahr 1975 lässt sich das Speichern der vollständigen Traceback-Matrix vermeiden und der Speicherbedarf auf $\mathcal{O}(m + n)$ reduzieren.
- Dabei verdoppelt sich (etwa) die Laufzeit; sie bleibt also $\mathcal{O}(mn)$.

Globale Alignments mit linearem Platzbedarf

- Wir können den optimalen Alignmentscore mit $\mathcal{O}(\min(m, n))$ Platz berechnen.
- Für das optimale Alignment benötigen wir (bisher) jedoch die vollständige Traceback-Matrix T, die $\mathcal{O}(mn)$ Platz benötigt.
- Nach einer Idee von Daniel S. Hirschberg aus dem Jahr 1975 lässt sich das Speichern der vollständigen Traceback-Matrix vermeiden und der Speicherbedarf auf $\mathcal{O}(m + n)$ reduzieren.
- Dabei verdoppelt sich (etwa) die Laufzeit; sie bleibt also $\mathcal{O}(mn)$.
- Hauptidee: Durch welche(n) Knoten $(i^*, n/2)$ verläuft der optimale Pfad?
  Ist i^* bekannt, zerfällt das Problem in zwei Teilprobleme (links oben, rechts unten)
  ("Divide-and-Conquer-Strategie")

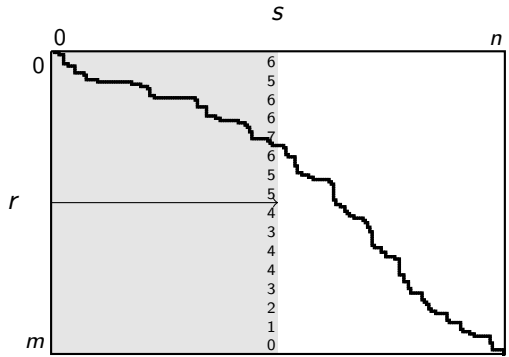
Hirschberg-Algorithmus

Zerlegung des Problems in zwei Teilprobleme:



Hirschberg-Algorithmus

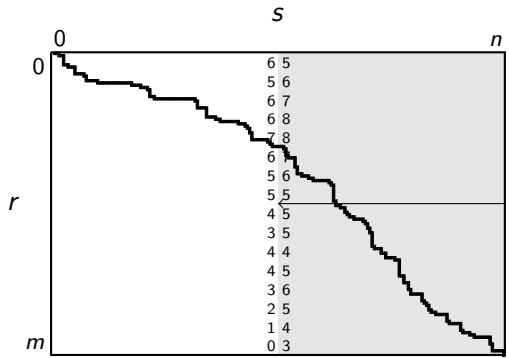
Zerlegung des Problems in zwei Teilprobleme:



Mit dem Vorwärtsalignment von 0 bis $n/2$ alignieren

Hirschberg-Algorithmus

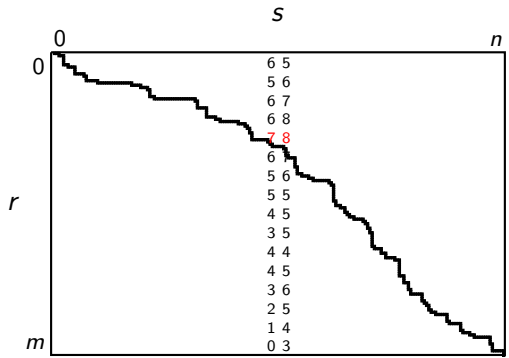
Zerlegung des Problems in zwei Teilprobleme:



Mit dem Rückwärtsalignment von n bis $n/2$ alignieren

Hirschberg-Algorithmus

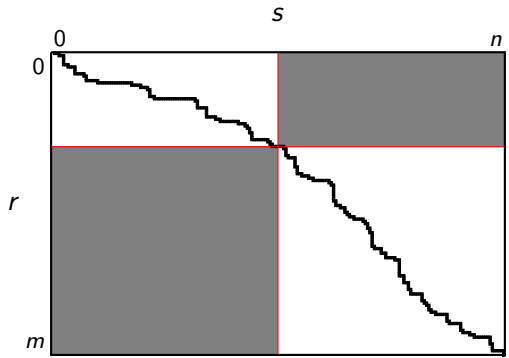
Zerlegung des Problems in zwei Teilprobleme:



Knoten $(i^*, n/2)$ mit dem optimalen Score bestimmen

Hirschberg-Algorithmus

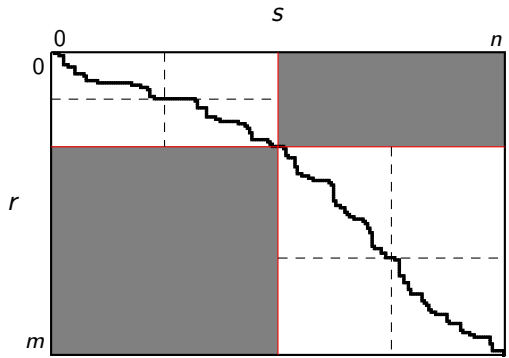
Zerlegung des Problems in zwei Teilprobleme:



Alignments $(0, 0) \rightarrow (i^*, n/2)$ und $(i^*, n/2) \rightarrow (m, n)$ durchführen

Hirschberg-Algorithmus

Zerlegung des Problems in zwei Teilprobleme:



Alignments so lange teilen, bis $r[i : i']$ oder $s[j : j']$ Größe 1 hat

Analyse des Hirschberg-Algorithmus

- Optimaler Pfad wird “von der Mitte aus” bestimmt. Platz: $\mathcal{O}(m + n)$
- Platz für Scorewerte: $\mathcal{O}(\min(m, n))$
- Insgesamt: $\mathcal{O}(m + n)$

Analyse des Hirschberg-Algorithmus

- Optimaler Pfad wird “von der Mitte aus” bestimmt. Platz: $\mathcal{O}(m+n)$
- Platz für Scorewerte: $\mathcal{O}(\min(m,n))$
- Insgesamt: $\mathcal{O}(m+n)$

- In der ersten Iteration Iteration werden die Alignments $(0,0) \rightarrow (m, n/2)$ und $(0, n/2) \leftarrow (m,n)$ berechnet: $\mathcal{O}(mn)$ Zeit

Analyse des Hirschberg-Algorithmus

- Optimaler Pfad wird "von der Mitte aus" bestimmt. Platz: $\mathcal{O}(m+n)$
- Platz für Scorewerte: $\mathcal{O}(\min(m,n))$
- Insgesamt: $\mathcal{O}(m+n)$

- In der ersten Iteration Iteration werden die Alignments $(0,0) \rightarrow (m, n/2)$ und $(0, n/2) \leftarrow (m,n)$ berechnet: $\mathcal{O}(mn)$ Zeit
- Nach jedem Rekursionsschritt fällt die Hälfte der verbleibenden Matrix weg.

Analyse des Hirschberg-Algorithmus

- Optimaler Pfad wird "von der Mitte aus" bestimmt. Platz: $\mathcal{O}(m+n)$
- Platz für Scorewerte: $\mathcal{O}(\min(m,n))$
- Insgesamt: $\mathcal{O}(m+n)$

- In der ersten Iteration Iteration werden die Alignments $(0,0) \to (m, n/2)$ und $(0, n/2) \leftarrow (m,n)$ berechnet: $\mathcal{O}(mn)$ Zeit
- Nach jedem Rekursionsschritt fällt die Hälfte der verbleibenden Matrix weg.
- Gesamtlaufzeit $\mathcal{O}(mn) \cdot (1 + 1/2 + 1/4 + \dots) = 2 \cdot \mathcal{O}(mn) = \mathcal{O}(mn)$

Zusammenfassung: Hirschberg-Algorithmus

- Mit dem Hirschberg-Algorithmus kann man globales Alignment mit linearem Platzbedarf $\mathcal{O}(m+n)$ berechnen.
- Die asymptotische Laufzeit bleibt bei $\mathcal{O}(mn)$ (Verdopplung)
- Alignment-Varianten wie semiglobales, "Free End Gaps" oder lokales Alignment können diese Methode nutzen, indem bei den Methoden die optimalen Start- und Endpunkte vorab bestimmt werden ($\mathcal{O}(m+n)$ Platz) und anschießend ein globales Alignment vom Start bis zum Ende durchgeführt wird.

Konzeptionelle Probleme des lokalen Alignments

- Obwohl sich die Nutzung des lokalen Alignments universell durchgesetzt hat, leidet dieses Verfahren an konzeptionellen Problemen.
- Der besteht in der Additivität der Zielfunktion

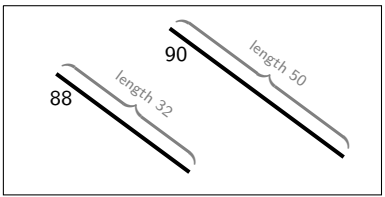
$$score(A) := \sum_{0 \leq i < |A|} score(A_i)$$

Konzeptionelle Probleme des lokalen Alignments

Definition (Schatten-Effekt)

Bei längeren Alignments mit vielen Edit-Operationen können bessere Scores als bei kürzeren exakteren Alignments erzielt werden.

Möglicherweise ist das kürzere Alignment biologisch interessanter.



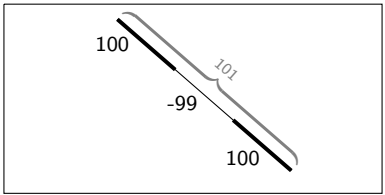
Konzeptionelle Probleme des lokalen Alignments

UA RUHR

Definition (Mosaik-Effekt)

Bei abwechselnd gut/schlecht/gut alignierenden Bereichen kann es dazu kommen, dass der komplette Bereich einen größeren Score hat als die kürzeren guten Bereiche individuell.

Interessant wären aber eigentlich die guten kurzen Alignments separat.



Längennormalisiertes Alignment

- Aufgrund des additiven Verfahrens können lange Alignments mit weniger guten Regionen insgesamt einen besseren Score erhalten als kurze gute Alignments.
- Es liegt nahe, einen Score zu definieren, der **längennormalisiert** ist.

Längennormalisiertes Alignment

- Aufgrund des additiven Verfahrens können lange Alignments mit weniger guten Regionen insgesamt einen besseren Score erhalten als kurze gute Alignments.
- Es liegt nahe, einen Score zu definieren, der **längennormalisiert** ist.
- Achtung: $score(A)/|A|$ wäre für sehr kurze Alignments (etwa eine Spalte / Zeile) kaum zu überbieten.

Längennormalisiertes Alignment

Eine pragmatische Lösung ist das Einführen eines Parameters $L > 0$, der eine "Mindestlänge" vorgaukelt:

$$NormScore_L(A) := \frac{1}{|A| + L} \cdot \sum_{0 \leq i < |A|} Score(A_i).$$

Längennormalisiertes Alignment

Eine pragmatische Lösung ist das Einführen eines Parameters $L > 0$, der eine "Mindestlänge" vorgaukelt:

$$NormScore_L(A) := \frac{1}{|A| + L} \cdot \sum_{0 \leq i < |A|} Score(A_i).$$

Probleme:

- Um L zu bestimmen, müsste schon vor dem Alignieren die Länge der möglichen Alignments bekannt sein.
- Die bekannten DP-Algorithmen können nicht mehr direkt angewendet werden, da der Score nicht mehr additiv ist.

Längennormalisiertes Alignment

Arslan und andere (2001) definieren folgenden Hilfs-Score:

$$\begin{aligned} DScore_{\lambda,L}(A) &:= Score(A) - \lambda(|A| + L) \\ &= -\lambda L + \sum_{i=1}^{|A|} (Score(A_i) - \lambda) \end{aligned}$$

Längennormalisiertes Alignment

Arslan und andere (2001) definieren folgenden Hilfs-Score:

$$\begin{aligned} DScore_{\lambda,L}(A) &:= Score(A) - \lambda(|A| + L) \\ &= -\lambda L + \sum_{i=1}^{|A|} (Score(A_i) - \lambda) \end{aligned}$$

(Alle Score-Werte werden effektiv um λ verringert.)
Der Offset $-\lambda L$ wirkt sich nicht auf das Optimierungsproblem aus.

Somit kann der bekannte DP-Algorithmus für lokales Alignment (für festes λ) angewendet werden.

Längennormalisiertes Alignment

- Man kann zeigen, dass immer ein $\lambda^* \geq 0$ existiert, so dass die Lösungen von $\max_A DScore_{\lambda^*,L}(A)$ und $\max_A NormScore_L(A)$ übereinstimmen.
- Durch binäre Suche und wiederholtes Alignieren kann λ^* gefunden werden, in der Praxis mit 3–5 Iterationen.

Längennormalisiertes Alignment

- Man kann zeigen, dass immer ein $\lambda^* \geq 0$ existiert, so dass die Lösungen von $\max_A DScore_{\lambda^*,L}(A)$ und $\max_A NormScore_L(A)$ übereinstimmen.
- Durch binäre Suche und wiederholtes Alignieren kann λ^* gefunden werden, in der Praxis mit 3–5 Iterationen.
- Trotzdem hat sich längennormalisiertes Alignment bisher nicht durchgesetzt, vielleicht wegen der Beliebigkeit des Parameters L.