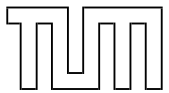
Algorithmische Bioinformatik 1

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

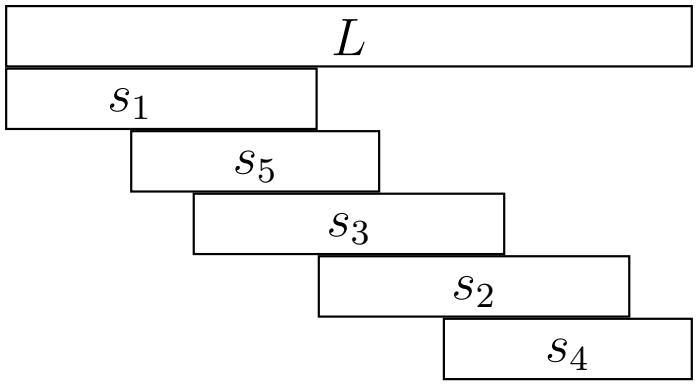
Sommersemester 2009

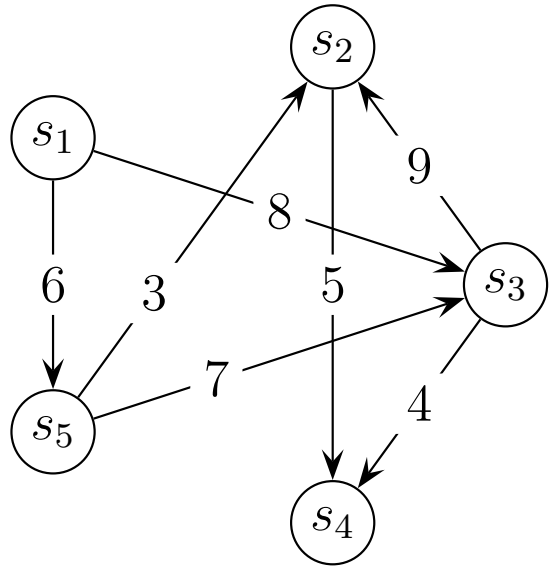


Übersicht

1. Fragment Assembly
   - Fragment Layout
2. Mehrfaches Sequenzen-Alignment

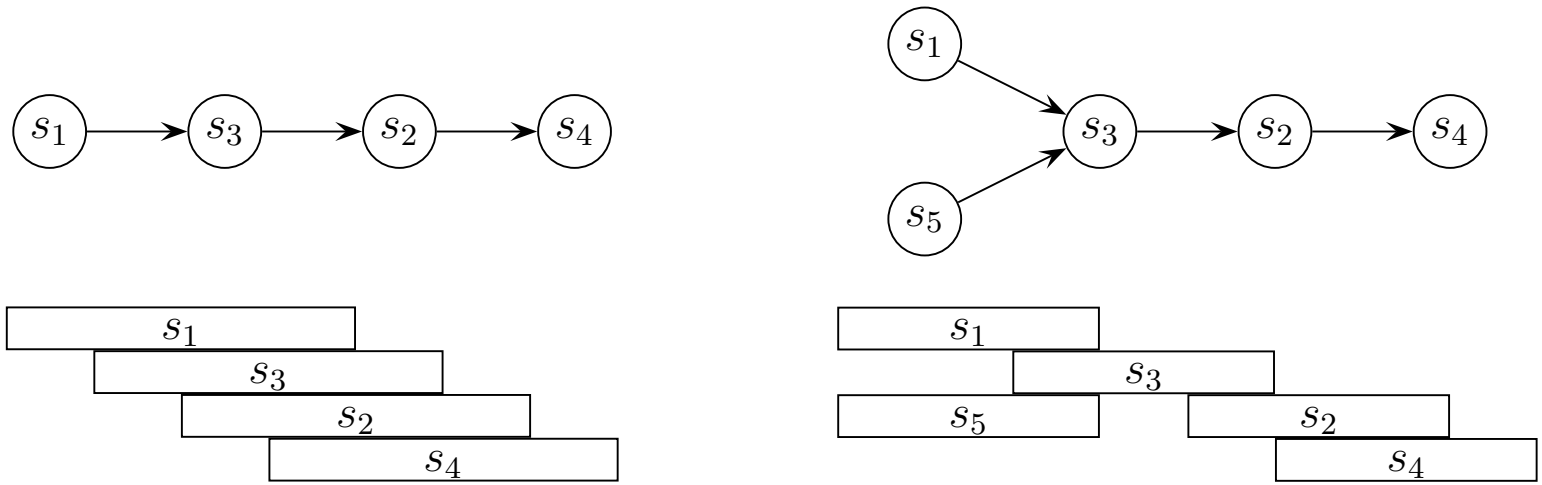
Layout und schwere Kanten im Overlap-Graph





Skizze: Overlap-Graph G mit den schwersten Kanten

Maximale Pfade/Spannbäume und Layouts



Skizze: Greedy-Path und maximaler Spannbaum für G

Maximale Pfade/Spannbäume und Layouts

- Konstruiert man aus dem Overlap-Graphen mit Hilfe der Greedy-Methode einen Pfad, so werden zuerst die Kanten (s_3, s_2) und (s_1, s_3) in den Pfad aufgenommen.
- Kante (s_5, s_3) kann dann nicht mehr aufgenommen werden, da sonst Knoten s_3 einen Eingangsgrad größer als 1 hätte.
- Der resultierende Pfad ist im linken Teil der Abbildung dargestellt.
- Knoten s_5 kann nicht in den Pfad integriert werden.
- Schwäche des Greedy-Ansatzes: kann den Pfad nicht vollständig rekonstruieren bzw. sogar ganz falsche Layouts konstruieren

Maximale Pfade/Spannbäume und Layouts

- Konstruieren wir nun ebenfalls mit der Greedy-Methode einen Spannbaum des Overlap-Graphen, so erhalten wir den im rechten Teil der Abbildung angegebenen Baum.
- Daraus lässt sich ein Layout rekonstruieren, bei dem nur die relative Lage von s_1 und s_5 ungeklärt bleibt.

- Die Greedy-Methode zur Konstruktion von minimalen Spannbäumen ist auch als Algorithmus von Kruskal bekannt.
- In diesem Fall muss nicht unbedingt ein Spannbaum entstehen (sondern nur ein Spannwald), da nur Kanten mit einem gewissen Mindestgewicht berücksichtigt werden.

Übersicht

1 Fragment Assembly

2 Mehrfaches Sequenzen-Alignment
- Maße für mehrfache Sequenzen-Alignments
- Dynamische Programmierung

Mehrfache Sequenzen-Alignments

- Definition analog zum Fall paarweiser Sequenzen-Alignments
- $\overline{\Sigma} = \Sigma \cup \{-\}$, wobei $- \notin \Sigma$

Definition

Seien $s_1, \ldots, s_k \in \Sigma^*$.

Eine Folge $(\overline{s}_1, \ldots, \overline{s}_k) \in \left(\overline{\Sigma}^n\right)^k$ heißt mehrfaches Alignment für die Sequenzen $s_1, \ldots, s_k$, wenn gilt:

- Für alle $j \in [1 : k]$ gilt $|\overline{s}_j| = n$.
- Für alle $j \in [1 : k]$ gilt $\overline{s}_j|_\Sigma = s_j$.
- Für alle $i \in [1 : n]$ gilt: aus $\overline{s}_{1,i} = \overline{s}_{2,i} = \cdots = \overline{s}_{k,i}$ folgt $\overline{s}_{1,i} \neq -$.

Mit $\mathcal{A}(s_1, \ldots, s_k)$ bezeichnen wir die Menge aller mehrfachen Sequenzen-Alignments von $s_1, \ldots, s_k \in \Sigma^*$.

Induzierte paarweise Alignments

Definition

Seien $s_1, \ldots, s_k \in \Sigma^*$ und $(\overline{s}_1, \ldots, \overline{s}_k)$ ein mehrfaches Alignment hierfür.

Dann ist $(\overline{s}_i, \overline{s}_j)$ mit $i \neq j \in [1 : k]$ ein projiziertes paarweises Alignment.

Das paarweise Alignment $(\overline{t}_i, \overline{t}_j)$ heißt induziertes paarweises Alignment, wenn es aus dem projizierten paarweisen Alignment $(\overline{s}_i, \overline{s}_j)$ entsteht, in dem alle Paare $(-, -)$ eliminiert werden.

Alignment-Distanz

- Im Folgenden bezeichnen wir mit $\overline{\Sigma}_0^k$ alle k-Tupel aus $\overline{\Sigma}$, die nicht nur aus Leerzeichen bestehen, d.h. $\overline{\Sigma}_0^k := \overline{\Sigma}^k \setminus \{(-, \cdots, -)\}$.

Definition

Sei $w : \overline{\Sigma}_0^k \to \mathbb{R}_+$ eine Kostenfunktion für ein Distanzmaß eines k-fachen Alignments $(\overline{s}_1, \ldots, \overline{s}_k)$ für $s_1, \ldots, s_k$, dann ist

$$w(\overline{s}_1, \ldots, \overline{s}_k) := \sum_{i=1}^{|\overline{s}_1|} w(\overline{s}_{1,i}, \ldots, \overline{s}_{k,i})$$

die Distanz des Alignments $(\overline{s}_1, \ldots, \overline{s}_k)$. Die so genannte Alignment-Distanz der Sequenzen $s_1, \ldots, s_k$ ist definiert als

$$\overline{d}_w(s_1, \ldots, s_k) = \min \{w((\overline{s}_1, \ldots, \overline{s}_k)) \ : \ (\overline{s}_1, \ldots, \overline{s}_k) \in \mathcal{A}(s_1, \ldots, s_k)\}$$

Bedingungen an das Distanzmaß

Wie bei paarweisen Alignments sollte die dem Distanzmaß zugrunde liegende Kostenfunktion w wieder den Bedingungen einer Metrik entsprechen:

- Symmetrie:

$$\forall (a_1, \ldots, a_k) \in \overline{\Sigma}_0^k : \forall \pi \in S(k) : w(a_1, \ldots, a_k) = w(a_{\pi(1)}, \ldots, a_{\pi(k)})$$

- Dreiecks-Ungleichung:

$$\begin{aligned} \forall (a_1, \ldots, a_k) \in \overline{\Sigma}_0^k : \forall x \in \overline{\Sigma} : \quad & w(a_1, \ldots, a_i, \ldots, a_j, \ldots, a_k) \\ \leq \quad & w(a_1, \ldots, a_i, \ldots, x, \ldots, a_k) \\ & + w(a_1, \ldots, x, \ldots, a_j, \ldots a_k). \end{aligned}$$

- Definitheit:

$$\forall (a_1, \ldots, a_k) \in \overline{\Sigma}_0^k : w(a_1, \ldots, a_k) = 0 \quad \Leftrightarrow \quad a_1 = \cdots = a_k.$$

Alignment-Ähnlichkeit

Definition

Sei $w : \overline{\Sigma}_0^k \to \mathbb{R}$ eine Kostenfunktion für ein Ähnlichkeitsmaß eines k-fachen Alignments $(\overline{s}_1, \ldots, \overline{s}_k)$ für $s_1, \ldots, s_k$, dann ist

$$w(\overline{s}_1, \ldots, \overline{s}_k) := \sum_{i=1}^{|\overline{s}_1|} w(\overline{s}_{1,i}, \ldots, \overline{s}_{k,i})$$

die Ähnlichkeit des Alignments $(\overline{s}_1, \ldots, \overline{s}_k)$.

Die Alignment-Ähnlichkeit der Sequenzen $s_1, \ldots, s_k$ ist definiert als

$$\overline{s}_w(s_1, \ldots, s_k) = \max\{w((\overline{s}_1, \ldots, \overline{s}_k)) \ : \ (\overline{s}_1, \ldots, \overline{s}_k) \in \mathcal{A}(s_1, \ldots, s_k)\}$$

Bedingungen an das Ähnlichkeitsmaß

- Auch diese sollte wieder symmetrisch sein.
- Weiter sollen positive Werte Ähnlichkeit und negative Werte Unähnlichkeit von Spalten repräsentieren.

- Wenn die zugrunde liegenden Kostenfunktionen und die Verwendung von Alignments klar ist, schreibt man auch $d(s_1, \ldots, s_k)$ statt $\overline{d}_w(s_1, \ldots, s_k)$ und $s(s_1, \ldots, s_k)$ statt $\overline{s}_w(s_1, \ldots, s_k)$.

Optimale mehrfache Sequenzen-Alignments

Definition

Sei $w : \overline{\Sigma}_0^k \to \mathbb{R}$ eine Kostenfunktion für ein Distanzmaß d bzw. Ähnlichkeitsmaß s eines k-fachen Alignments.

Ein k-faches Alignment $(\overline{s}_1, \ldots, \overline{s}_k)$ für $s_1, \ldots, s_k \in \Sigma^*$ heißt optimal, wenn

$$w(\overline{s}_1, \ldots, \overline{s}_k) = d(s_1, \ldots, s_k)$$

bzw.

$$w(\overline{s}_1, \ldots, \overline{s}_k) = s(s_1, \ldots, s_k).$$

Anmerkung

- Bei den verwendeten konkreten Kostenfunktionen von mehrfachen Alignments wird manchmal auch auf die Symmetrie verzichtet.

- Wir werden dabei im Folgenden die Kostenfunktion auf Zeichentupel $w : \overline{\Sigma}^k \to \mathbb{R}$ auf Sequenzen von Zeichentupeln (d.h. auf mehrere Sequenzen gleicher Länge) in kanonischer Weise fortsetzen:
$w(\overline{s}_1, \ldots, \overline{s}_k) := \sum_{r=1}^{n} w(\overline{s}_{1,r}, \ldots, \overline{s}_{k,r})$ für $\overline{s}_1, \ldots, \overline{s}_k \in \overline{\Sigma}^n$.

Spezielle Kostenfunktionen

- Es ist aufwendig, Kostenfunktionen auf $\overline{\Sigma}_0^k$ zu definieren.

$\Rightarrow$ Man betrachtet spezielle Kostenfunktionen, die aus Kostenfunktionen auf $\overline{\Sigma}^2$ aufgebaut sind.

- Dabei gilt hier allerdings $w(-,-) = 0$, da dies in einer Projektion eines mehrfachen Sequenzen Alignment zum einen nicht wirklich die Distanz zwischen den beiden betrachteten Sequenzen erhöht und zum anderen die Ähnlichkeit dadurch weder erhöht noch erniedrigt wird.

Spezielle Kostenfunktionen

Definition

Sei $G = ([1:k], E)$ ein (zusammenhängender) Graph auf $[1:k]$.

Sei weiter $w' : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine Kostenfunktion für ein Distanz- bzw. ein Ähnlichkeitsmaß.

Dann heißt $w : \overline{\Sigma}_0^k \to \mathbb{R}$ eine von G induzierte Kostenfunktion für ein Distanz- bzw. Ähnlichkeitsmaß, wenn für alle $(a_1, \ldots, a_k) \in \overline{\Sigma}_0^k$ gilt:

$$w(a_1, \ldots, a_k) = \sum_{\{i,j\} \in E} w'(a_i, a_j),$$

wobei hier $w'(-,-) := 0$ definiert ist.

Die im Folgenden definierten einfachen Kostenfunktionen für mehrfache Sequenzen-Alignments sind im Prinzip alles induzierte Kostenfunktionen.

Sum-of-Pairs Cost (SP)

Definition

Ist G ein vollständiger Graph, so heißt die durch G induzierte Kostenfunktion Sum-of-Pairs-Kostenfunktion (kurz SP-Kostenfunktion):

$$w(a_1, \ldots, a_k) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} \tilde{w}(a_i, a_j),$$

wobei $\tilde{w} : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine gewöhnliche Kostenfunktion für ein Distanz- oder Ähnlichkeitsmaß eines paarweisen Alignments ist.

Sum-of-Pairs Cost (SP)

Entsprechender Abstand oder Ähnlichkeit für mehrfache Alignments:

$$w(\overline{s}_1, \ldots, \overline{s}_k) := \sum_{i=1}^{k} \sum_{j=i+1}^{k} \tilde{w}(\overline{s}_i, \overline{s}_j).$$

Die Berechnungszeit für ein $(a_1, \ldots, a_k) \in \overline{\Sigma}_0^k$ ist $\mathcal{O}(k^2)$.

Anmerkung: im Folgenden wird meist als Kostenfunktion die oben erwähnte Sum-of-Pairs-Kostenfunktion verwendet.

Das zugehörige Distanz bzw. Ähnlichkeitsmaß wird dann oft auch als Sum-of-Pairs-Distanz bzw. Sum-of-Pairs-Ähnlichkeit oder kurz als SP-Distanz bzw. SP-Ähnlichkeit bezeichnet.

Sum-of-Pairs Cost (SP)

Theorem

Sei $w : \overline{\Sigma}_0^2 \to \mathbb{R}_+$ *eine Kostenfunktion für ein SP-Distanzmaß* d *und sei* $C \in \mathbb{R}_+$ *so gewählt, dass* $w' : \overline{\Sigma}_0^2 \to \mathbb{R}$ *mit*

$$\begin{aligned} w'(a,b) &= C - w(a,b) \text{ und} \\ w'(a,-) &= \frac{C}{2} - w(a,-) \end{aligned}$$

für alle $a, b \in \Sigma$ *eine Kostenfunktion für ein SP-Ähnlichkeitsmaß* s *ist.*

Dann gilt für alle $(s_1, \ldots, s_k) \in (\Sigma^*)^k$*:*

$$d(s_1, \ldots, s_k) + s(s_1, \ldots, s_k) = \frac{C(k-1)}{2} \sum_{i=1}^{k} |s_i|$$

Sum-of-Pairs Cost (SP)

Allgemeiner gilt sogar:

Theorem

Sei G ein r-regulärer Graph auf $[1:k]$.

Sei $w : \overline{\Sigma}_0^k \to \mathbb{R}_+$ eine Kostenfunktion für ein durch G induziertes Distanzmaß d und sei $C \in \mathbb{R}_+$ so gewählt, dass $w' : \overline{\Sigma}_0^2 \to \mathbb{R}$ mit $w'(a,b) = C - w(a,b)$ und $w'(a,-) = \frac{C}{2} - w(a,-)$ für alle $a, b \in \Sigma$ eine Kostenfunktion für ein durch G induziertes Ähnlichkeitsmaß s ist.

Dann gilt für alle $(s_1, \ldots, s_k) \in (\Sigma^*)^k$:

$$d(s_1, \ldots, s_k) + s(s_1, \ldots, s_k) = \frac{Cr}{2} \sum_{i=1}^{k} |s_i|$$

Fixed-Tree Cost

Die feste Baum-Kostenfunktion wird durch einen ungewurzelten Baum auf allen Sequenzen induziert.

Definition

Ist G ein Baum mit $V(G) = [1 : k]$, so heißt die durch G induzierte Kostenfunktion feste Baum-Kostenfunktion:

$$w(a_1, \ldots, a_k) = \sum_{\{i,j\} \in E} \tilde{w}(a_i, a_j),$$

wobei $\tilde{w} : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine gewöhnliche Kostenfunktion für ein Distanz- oder Ähnlichkeitsmaß eines paarweisen Alignments ist.

Die Berechnungszeit für ein $(a_1, \ldots, a_k) \in \overline{\Sigma}_0^k$ ist $\mathcal{O}(k)$.

Center-Star Cost

Die feste Stern-Kostenfunktion wird durch einen speziellen Baum induziert, wobei alle $k-1$ Blätter an einem inneren Knoten hängen

Definition

Ist $G = ([1:k], \{\{1,j\} : j \in [2:k]\})$ ein Stern, so heißt die durch G induzierte Kostenfunktion Stern-Kostenfunktion

$$w(a_1, \ldots, a_k) = \sum_{\{i,j\} \in E} \tilde{w}(a_i, a_j),$$

wobei $\tilde{w} : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine gewöhnliche Kostenfunktion für ein Distanz- oder Ähnlichkeitsmaß eines paarweisen Alignments ist.

Die Berechnungszeit für ein $(a_1, \ldots, a_k) \in \overline{\Sigma}_0^k$ ist $\mathcal{O}(k)$.

Consensus Cost

Hierbei handelt es sich im Wesentlichen auch um eine Stern-Kostenfunktion, wobei die Sequenzen jedoch nur den Blättern zugeordnet sind, und für die Wurzel das Zeichen ausgesucht wird, das die Stern-Kostenfunktion minimiert.

Definition

Die Konsensus-Kostenfunktion ist definiert durch

$$w(a_1, \ldots, a_k) = opt \left\{ \sum_{i=1}^{k} \tilde{w}(a_i, c) \ : \ c \in \overline{\Sigma} \right\},$$

wobei $\tilde{w} : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine gewöhnliche Kostenfunktion für ein Distanz- bzw. Ähnlichkeitsmaß eines paarweisen Alignments ist und $opt \in \{\min \max\}$, je nach Maß.

Die Berechnungszeit für ein $a \in \overline{\Sigma}_0^k$ ist $\mathcal{O}(|\Sigma| \cdot k)$.

Tree Cost

Verallgemeinerung der Konsensus-Kostenfunktion
Jeder innere Knoten hat Grad genau drei und man sucht die beste Belegung der inneren Knoten mit Zeichen aus $\overline{\Sigma}$.

Definition

Sei $([1 : 2k-2], E)$ ein freier Baum, wobei alle inneren Knoten den Grad 3 besitzen und die inneren Knoten durch $([k+1 : 2k-2])$ bezeichnet sind. Die allgemeine Baum-Kostenfunktion ist definiert durch

$$w(a_1, \ldots, a_k) = opt \left\{ \sum_{\{i,j\} \in E} \tilde{w}(a_i, a_j) \ : \ (a_{k+1}, \ldots, a_{2k-2}) \in \overline{\Sigma}^{k-2} \right\},$$

wobei $\tilde{w} : \overline{\Sigma}_0^2 \to \mathbb{R}$ eine gewöhnliche Kostenfunktion für Distanz- oder Ähnlichkeitsmaß eines paarweisen Alignments ist und $opt \in \{\min \max\}$, je nach Maß.

Tree Cost

Die Berechnungszeit für ein $(a_1, \ldots, a_k) \in \overline{\Sigma}_0^k$ ist $\mathcal{O}(|\Sigma| \cdot k^2)$.

Der Beweis der Berechnungszeit ist etwas aufwendiger und bedarf einer geschickten dynamischen Programmierung.

In Varianten dieser Kostenfunktion werden auch andere Bäume mit größerem Grad zugelassen, oder es wird auch über alle möglichen Topologien optimiert (oder einer eingeschränkten Teilmenge davon).

Dynamische Programmierung

- Verallgemeinerung der Methode der Dynamischen Programmierung von paarweisen auf mehrfache Sequenzen-Alignments

- Aufgrund der großen Laufzeit ist dieses Verfahren aber eher von theoretischem Interesse bzw. nur für wenige oder kurze Sequenzen sinnvoll einsetzbar.

Rekursionsgleichungen

Im Folgenden sei $D[\vec{x}]$ für $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{N}_0^k$ der Wert eines optimalen mehrfachen Sequenzen-Alignments für

$s_{1,1} \cdots s_{1,x_1}$, $s_{2,1} \cdots s_{2,x_2}$, $\ldots$ $s_{k-1,1} \cdots s_{k-1,x_{k-1}}$ und $s_{k,1} \cdots s_{k,x_k}$.

Theorem

Seien $s_1, \ldots, s_k \in \Sigma^*$ und sei $w : \overline{\Sigma}_0^k \to \mathbb{R}_+$ eine Kostenfunktion für ein Distanzmaß. Es gilt für $\vec{x} \in \bigtimes_{i=1}^{k} [0 : |s_i|]$:

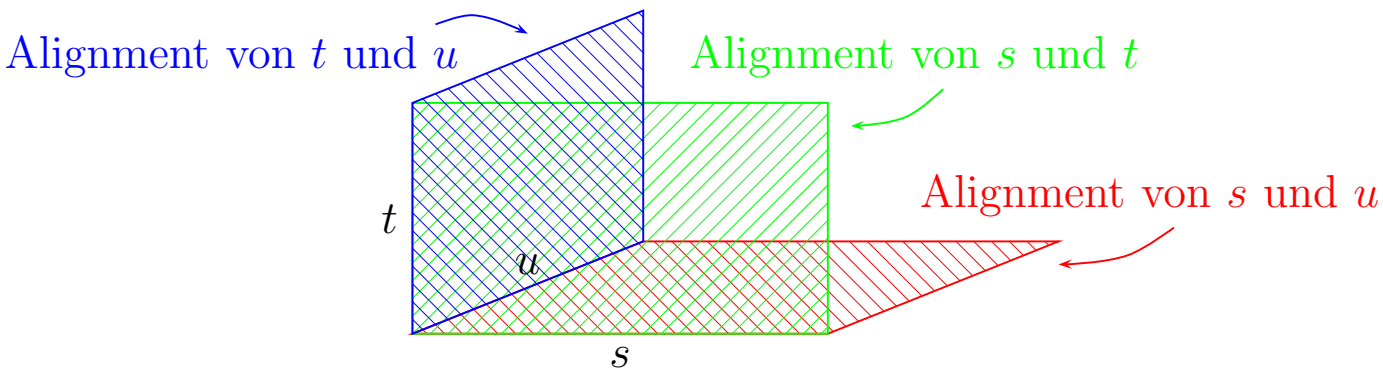
$$D[\vec{x}] := \min \left\{ D[\vec{x} - \vec{\eta}] + w(\vec{x} \bullet \vec{\eta}) \ : \ \vec{\eta} \in [0:1]^k \setminus \vec{0} \wedge \vec{\eta} \leq \vec{x} \right\}.$$

Hierbei ist $\min \emptyset = 0$ *und*
$(x_1, \ldots, x_k) \bullet (\eta_1, \ldots, \eta_k) = (s_{1,x_1} \bullet \eta_1, \ldots, s_{k,x_k} \bullet \eta_k)$ *mit*
$a \bullet 0 := -$ *und* $a \bullet 1 := a$ *für* $a \in \Sigma$.

(Beweis analog wie für das paarweise Sequenzen-Alignment)

Anfangswerte

- Bei der Implementierung kann die Abfrage $\vec{\eta} \geq \vec{x}$ etwas aufwendig werden, so dass man wie im Falle paarweiser Alignments die Randbedingungen oft explizit definiert.
- Wie sehen die Anfangswerte für $\vec{x} \in [0:n]^k \setminus [1:n]^k$ eines solchen mehrfaches Sequenzen-Alignments aus?
- Erklärung am Beispiel von drei Sequenzen:



Anfangswerte für ein 3-faches Sequenzen-Alignment

Anfangswerte

Explizite Rekursionsformeln und Anfangsbedingungen für ein 3-faches Sequenzen-Alignment von $s^{(1)}$, $s^{(2)}$ und $s^{(3)}$ mit $n_i = |s(i)|$ $(i \in [1:3])$:

Es gilt dann für ein globales mehrfaches Sequenzen-Alignment mit $(i,j,k) \in [1:n_1] \times [1:n_2] \times [1:n_3]$:

$$
\begin{aligned}
D[0,0,0] &= 0, \\
D[i,0,0] &= D[i-1,0,0] + w(s_i^{(1)}, -, -), \\
D[0,j,0] &= D[0,j-1,0] + w(-, s_j^{(2)}, -), \\
D[0,0,k] &= D[0,0,k-1] + w(-, -, s_k^{(3)}),
\end{aligned}
$$

Anfangswerte

$$D[i,j,0] = \min \left\{ \begin{array}{c} D[i-1,j,0] + w(s_i^{(1)},-,-), \\ D[i,j-1,0] + w(-,s_j^{(2)},-), \\ D[i-1,j-1,0] + w(s_i^{(1)},s_j^{(2)},-) \end{array} \right\},$$

$$D[i,0,k] = \min \left\{ \begin{array}{c} D[i-1,0,k] + w(s_i^{(1)},-,-), \\ D[i,0,k-1] + w(-,-,s_k^{(3)}), \\ D[i-1,0,k-1] + w(s_i^{(1)},-,s_k^{(3)}) \end{array} \right\},$$

$$D[0,j,k] = \min \left\{ \begin{array}{c} D[0,j-1,k] + w(-,s_j^{(2)},-), \\ D[0,j,k-1] + w(-,-,s_k^{(3)}), \\ D[0,j-1,k-1] + w(-,s_j^{(2)},s_k^{(3)}) \end{array} \right\},$$

Anfangswerte

$$D[i,j,k] \quad = \quad \min \left\{ \begin{array}{c} D[i-1,j,k] + w(s_i^{(1)},-,-), \\ D[i,j-1,k] + w(-,s_j^{(2)},-), \\ D[i,j,k-1] + w(-,-,s_k^{(3)}), \\ D[i-1,j-1,k] + w(s_i^{(1)},s_j^{(2)},-), \\ D[i-1,j,k-1] + w(s_i^{(1)},-,s_k^{(3)}), \\ D[i,j-1,k-1] + w(-,s_j^{(2)},s_k^{(3)}), \\ D[i-1,j-1,k-1] + w(s_i^{(1)},s_j^{(2)},s_k^{(3)}) \end{array} \right\}.$$

Hierbei ist $w : \overline{\Sigma}^3 \to \mathbb{R}_+$ die zugrunde gelegte Kostenfunktion.

Für das Sum-Of-Pairs-Maß gilt dann:
$w(x,y,z) = w'(x,y) + w'(x,z) + w'(y,z)$, wobei $w' : \overline{\Sigma}^2 \to \mathbb{R}_+$ die Standard-Kostenfunktion für Paare ist.

Zeitanalyse

- Für die Zeitanalyse nehmen wir an, dass $|s_i| = \Theta(n)$ für alle $i \in [1 : k]$ gilt.
- Die gesamte Tabelle besitzt $\Theta(n^k)$ viele Einträge.
- Für jeden Eintrag ist eine Minimumsbildung von $2^k - 1$ Elementen durchzuführen, wobei sich jeder Wert in Zeit $\Theta(k^2)$ berechnen lässt (wenn wir dass SP-Maß zugrunde legen).
- Insgesamt ist der Zeitbedarf also $\mathcal{O}(k^2 \cdot 2^k \cdot n^k)$.

- Dies ist leider exponentiell und selbst für moderat große k inakzeptabel.
- Für $k = 3$ ist dies gerade noch verwendbar, für größere k in der Regel unpraktikabel (außer die Sequenzen sind sehr kurz).

Zeitanalyse

- Leider gibt es für die Berechnung eines mehrfachen Sequenzen-Alignment kein effizientes Verfahren.
- Man kann nämlich nachweisen, das die Entscheidung, ob eine gegebene Menge von Sequenzen, ein mehrfaches Alignment besitzt, das eine vorgegebene Distanz unterschreitet (oder Ähnlichkeit überschreitet), $\mathcal{NP}$-hart ist.

- Auch für andere Kostenfunktionen ist die dynamische Programmierung nicht wesentlich schneller, da in jedem Fall $O(\gamma(k) \cdot 2^k \cdot n^k)$ Operationen benötigt werden, wobei $\gamma(k)$ die Anzahl der Operationen beschreibt um die induzierte Kostenfunktion für ein k-Tupel zu ermitteln.

Forward Dynamic Programming

- Heuristik, mit der sich die dynamische Programmierung oft etwas schneller durchführen lässt:
  (wie beim paarweisen Alignment für konkave Lückenstrafen)
  vom backward dynamic programming auf das forward dynamic programming umschalten.
- Wenn also die Zelle $D[\vec{x}]$ vollständig berechnet wurde, werden dann die davon abhängigen Zellen $\vec{x} + \vec{\eta}$ (für $\vec{\eta} \in \{0,1\}^k \setminus \{\vec{0}\}$) mittels der folgenden Gleichung aktualisiert:

$$D[\vec{x} + \vec{\eta}] := \min\{D[\vec{x} + \vec{\eta}], D[\vec{x}] + w((\vec{x} + \vec{\eta}) \bullet \vec{\eta})\}.$$

- Ersetzt man in dieser Gleichung $\vec{x} + \vec{\eta}$ durch $\vec{x}$ sowie $\vec{x}$ durch $\vec{x} - \vec{\eta}$, dann erhält man die übliche Form der Rekursionsgleichung im backward dynamic programming.
- Dieser Trick kann die dynamische Programmierung beschleunigen, wenn auch nicht im worst-case.