

# Algorithmen auf Sequenzen

## Paarweiser Sequenzvergleich: Alignments

Sven Rahmann

Genominformatik  
Universitätsklinikum Essen  
Universität Duisburg-Essen  
Universitätsallianz Ruhr

- Bisher:
  - Berechnung von Abstands- oder Ähnlichkeitsmaßen von zwei Sequenzen, z.B. Edit-Distanz, längste gemeinsame Teilsequenz, ...
  - fehlertolerante Suche eines Musters in einem Text (Edit-Distanz)

- Bisher:
  - Berechnung von Abstands- oder Ähnlichkeitsmaßen von zwei Sequenzen, z.B. Edit-Distanz, längste gemeinsame Teilsequenz, ...
  - fehlertolerante Suche eines Musters in einem Text (Edit-Distanz)
- Jetzt:
  - vergleichende Darstellung der Sequenzen (“Alignment”)
  - flexible Ähnlichkeitsmaße (“Alignment-Scores”)
  - verschiedene Arten des Alignments und ihre Anwendungen

- Vergleich von biologischen Sequenzen (z.B. Proteinsequenzen) erfordert differenzierte Betrachtung von Ähnlichkeiten (statt nur Gleichheit oder Ungleichheit von Zeichen).  
Beispiele: Leucin und Isoleucin (L und I) sind physikalisch und chemisch ähnlich. Tryptophan (Y) ist in seinen Eigenschaften sehr verschieden von den meisten anderen Aminosäuren.

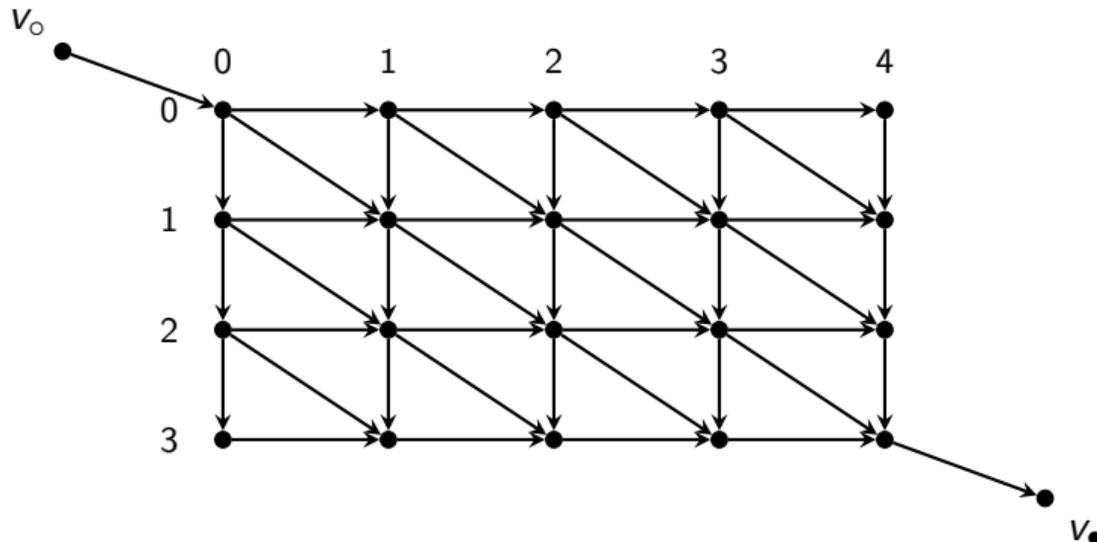
- Vergleich von biologischen Sequenzen (z.B. Proteinsequenzen) erfordert differenzierte Betrachtung von Ähnlichkeiten (statt nur Gleichheit oder Ungleichheit von Zeichen).  
Beispiele: Leucin und Isoleucin (L und I) sind physikalisch und chemisch ähnlich. Tryptophan (Y) ist in seinen Eigenschaften sehr verschieden von den meisten anderen Aminosäuren.
- Paradigmenwechsel: Bewerte Ähnlichkeit (positiv und negativ) statt Kosten.
- Ähnlichkeit von 0 heißt “neutral”, positiv “ähnlich”, negativ “unähnlich”.

- Vergleich von biologischen Sequenzen (z.B. Proteinsequenzen) erfordert differenzierte Betrachtung von Ähnlichkeiten (statt nur Gleichheit oder Ungleichheit von Zeichen).  
Beispiele: Leucin und Isoleucin (L und I) sind physikalisch und chemisch ähnlich. Tryptophan (Y) ist in seinen Eigenschaften sehr verschieden von den meisten anderen Aminosäuren.
- Paradigmenwechsel: Bewerte Ähnlichkeit (positiv und negativ) statt Kosten.
- Ähnlichkeit von 0 heißt “neutral”, positiv “ähnlich”, negativ “unähnlich”.
- Daher: Beliebige Scorematrix  $s = s(a, b)$  für alle  $a, b \in \Sigma$ ,  
(negative) Ähnlichkeitswerte für Insertionen und Deletionen

# Universeller Alignment-Algorithmus

Gegeben:

- Sequenzen  $s, t$
- Scoring-Schema
- Alignment-Graph-Topologie (z.B. für globales Alignment)



# Universeller Alignment-Algorithmus

- Gegeben: Sequenzen  $s, t$ , Scoring-Schema, Graph-Topologie
- Gesucht:
  - Maximaler Score aller Pfade  $v_o \rightarrow v_\bullet$  (**optimaler Alignment-Score**)
  - Pfad, der den Score maximiert (**optimales Alignment**)
- Sei  $S(v)$  der maximale Score aller Pfade  $v_o \rightarrow v$  und  $S(v_o) := 0$ .
- Sei  $T(v)$  der Vorgängerknoten von  $v$ , über den der maximale Score erreicht wird.

# Universeller Alignment-Algorithmus

- Gegeben: Sequenzen  $s, t$ , Scoring-Schema, Graph-Topologie
- Gesucht:
  - Maximaler Score aller Pfade  $v_o \rightarrow v_\bullet$  (**optimaler Alignment-Score**)
  - Pfad, der den Score maximiert (**optimales Alignment**)
- Sei  $S(v)$  der maximale Score aller Pfade  $v_o \rightarrow v$  und  $S(v_o) := 0$ .
- Sei  $T(v)$  der Vorgängerknoten von  $v$ , über den der maximale Score erreicht wird.
- Berechnung für  $v \neq v_o$ :

$$S(v) = \max_{w: w \rightarrow v \in E} \{S(w) + \text{score}(w \rightarrow v)\},$$

$$T(v) = \arg \max_{w: w \rightarrow v \in E} \{S(w) + \text{score}(w \rightarrow v)\}.$$

- Berechnung in topologischer Sortierung (Graph ist azyklisch!)
- Den optimalen Score erhalten wir als  $S(v_\bullet)$ .
- Den optimalen Pfad erhalten wir durch **Traceback** von  $v_\bullet$  aus:
 
$$v_\bullet \rightarrow T(v_\bullet) \rightarrow T(T(v_\bullet)) \rightarrow \dots \rightarrow T^k(v_\bullet) \rightarrow \dots \rightarrow v_o.$$

- Traceback (auch Backtracing, nicht verwechseln mit Backtracking!):  
Rekonstruktion des optimalen Pfades durch Zurückverfolgen der Vorgängerknoten, die jeweils zum maximalen Score führen

- Traceback (auch Backtracing, nicht verwechseln mit Backtracking!):  
Rekonstruktion des optimalen Pfades durch Zurückverfolgen der Vorgängerknoten, die jeweils zum maximalen Score führen
- Rekonstruiert den optimalen Pfad rückwärts
- optimales Alignment:  
Kantenlabel (Buchstabenpaare) des optimalen Pfades ablesen

- Traceback (auch Backtracing, nicht verwechseln mit Backtracking!):  
Rekonstruktion des optimalen Pfades durch Zurückverfolgen der Vorgängerknoten, die jeweils zum maximalen Score führen
- Rekonstruiert den optimalen Pfad rückwärts
- optimales Alignment:  
Kantenlabel (Buchstabenpaare) des optimalen Pfades ablesen
- Speicherplatz:
  - Für  $S(v)$  wird (hier) stets nur die aktuelle und die vorangehende Zeile oder Spalte benötigt:  $\mathcal{O}(\min(n, m))$

- Traceback (auch Backtracing, nicht verwechseln mit Backtracking!):  
Rekonstruktion des optimalen Pfades durch Zurückverfolgen der Vorgängerknoten, die jeweils zum maximalen Score führen
- Rekonstruiert den optimalen Pfad rückwärts
- optimales Alignment:  
Kantenlabel (Buchstabenpaare) des optimalen Pfads ablesen
- Speicherplatz:
  - Für  $S(v)$  wird (hier) stets nur die aktuelle und die vorangehende Zeile oder Spalte benötigt:  $\mathcal{O}(\min(n, m))$
  - Für  $T(v)$ :  $\mathcal{O}(mn)$  benötigt, damit man den ganzen Pfad rekonstruieren kann (Verbesserung gleich!)

Zu zwei Sequenzen kann man mit geeigneten Scoring-Schemas folgende optimale paarweise Alignments berechnen:

- 1 Globales Alignment (Ähnlichkeit)
- 2 Semiglobales Alignment (Mustersuche)
- 3 Free-end-gaps Alignment (Überlapp-Test)
- 4 Lokales Alignment (Regionen großer Ähnlichkeit)

Zu zwei Sequenzen kann man mit geeigneten Scoring-Schemas folgende optimale paarweise Alignments berechnen:

- 1 Globales Alignment (Ähnlichkeit)
- 2 Semiglobales Alignment (Mustersuche)
- 3 Free-end-gaps Alignment (Überlapp-Test)
- 4 Lokales Alignment (Regionen großer Ähnlichkeit)

Wir zeigen im folgenden die entsprechenden Alignment-Graph-Topologien. Spezielle Algorithmen ergeben sich durch Spezialisierung des universellen Algorithmus.

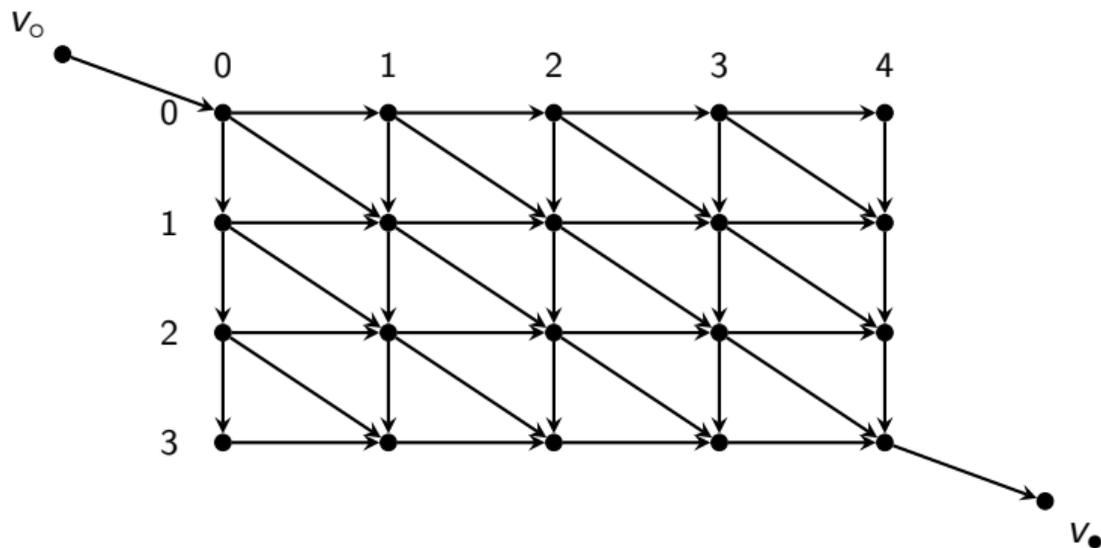
## Definition (globaler Alignment-Graph)

- Knotenmenge  $V := \{(i, j) : 0 \leq i \leq m, 0 \leq j \leq n\} \cup \{v_o, v_\bullet\}$
- Kanten:

	Kante	Label	Score
horizontal	$(i, j) \rightarrow (i, j + 1)$	$\begin{bmatrix} - \\ t_j \end{bmatrix}$	$< 0$ (*)
vertikal	$(i, j) \rightarrow (i + 1, j)$	$\begin{bmatrix} s_i \\ - \end{bmatrix}$	$< 0$ (*)
diagonal	$(i, j) \rightarrow (i + 1, j + 1)$	$\begin{bmatrix} s_i \\ t_j \end{bmatrix}$	beliebig (*)
Initialisierung	$v_o \rightarrow (0, 0)$	$\epsilon$	0
Finalisierung	$(m, n) \rightarrow v_\bullet$	$\epsilon$	0

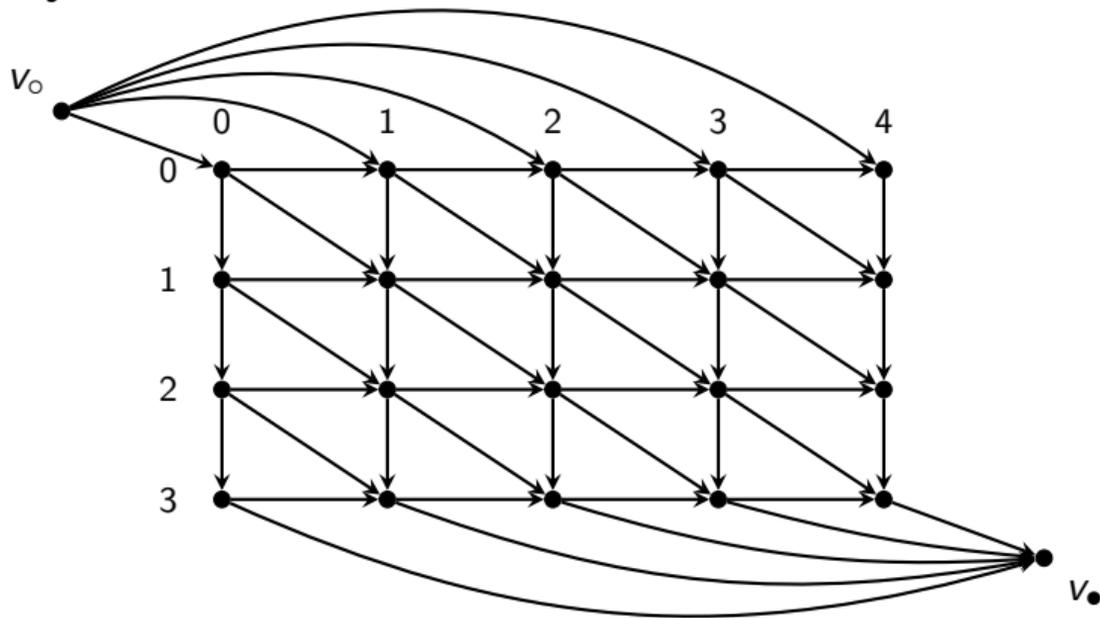
(\*): Sinnvolle Scoring-Schemas erfüllen i.d.R., dass der Score für Gaps und Substitutionen negativ, für Übereinstimmungen aber positiv ist.

# Globales Alignment



# Semiglobales Alignment (Mustersuche)

zusätzliche Initialisierungskanten  $v_o \rightarrow (0, j)$  und Finalisierungskanten  $(m, j) \rightarrow v_e$  mit Score 0 für alle  $j$ .



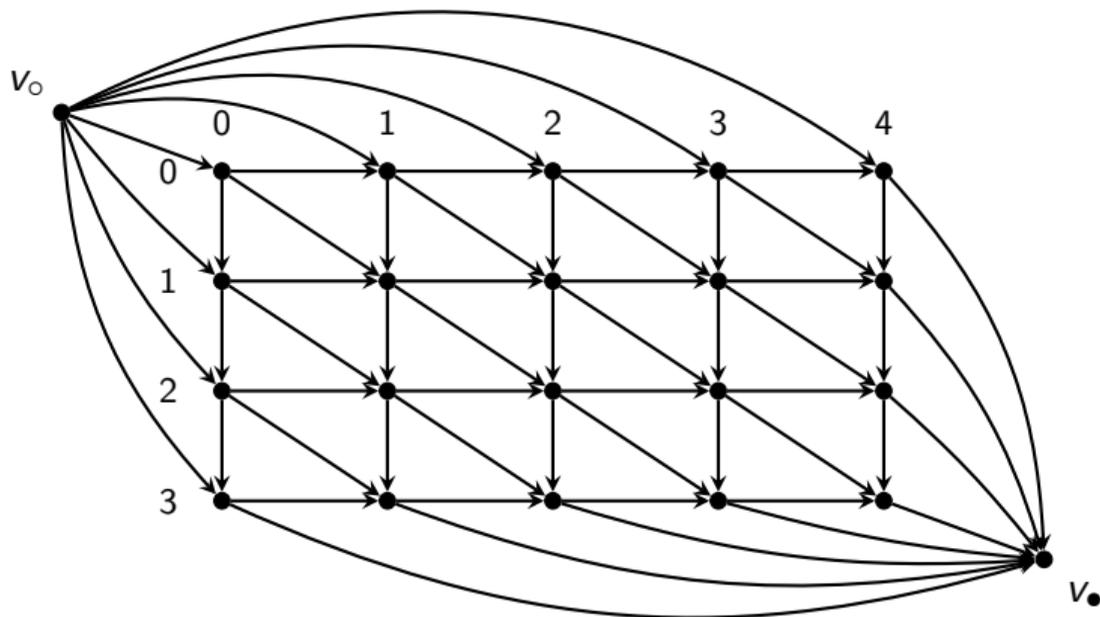
# “Free End Gaps”-Alignment

- Fragestellung: überlappen sich zwei Sequenzen?



- Gaps am Anfang bzw. Ende des Alignments sollen nicht bestraft werden.
- Zusätzliche Initialisierungskanten  $v_{\circ} \rightarrow (i, 0)$  und  $v_{\circ} \rightarrow (0, j)$  und Finalisierungskanten  $(i, n) \rightarrow v_{\bullet}$  und  $(m, j) \rightarrow v_{\bullet}$ , alle jeweils mit Score 0

# “Free End Gaps”-Alignment



- Gesucht ist Alignment mit maximalem Score zwischen allen Teilstrings  $s'$  (von  $s$ ) und  $t'$  (von  $t$ ).



# Lokales Alignment

- Gesucht ist Alignment mit maximalem Score zwischen allen Teilstrings  $s'$  (von  $s$ ) und  $t'$  (von  $t$ ).



- Initialisierungskanten:  $v_{\circ} \rightarrow (i, j)$  für alle  $i \leq m, j \leq n$ ,
- Finalisierungskanten:  $(i, j) \rightarrow v_{\bullet}$  für alle  $i \leq m, j \leq n$ .
- Visualisierung nicht sinnvoll (zu viele Kanten!)

# Lokales Alignment

- Gesucht ist Alignment mit maximalem Score zwischen allen Teilstrings  $s'$  (von  $s$ ) und  $t'$  (von  $t$ ).



- Initialisierungskanten:  $v_{\circ} \rightarrow (i, j)$  für alle  $i \leq m, j \leq n$ ,
- Finalisierungskanten:  $(i, j) \rightarrow v_{\bullet}$  für alle  $i \leq m, j \leq n$ .
- Visualisierung nicht sinnvoll (zu viele Kanten!)
- Überlegungen:
  - Das leere Alignment (Score 0) ist stets eine Option
  - optimale lokale Alignments sind nur sinnvoll, wenn der erwartete Score zufälliger Sequenzen negativ ist

Für jede Alignment-Variante:

- Welche konkrete Bedeutung hat der Score  $S(v)$  für  $v = (i, j)$ ?  
("Score eines optimalen Alignments mit ...")
- Wie sieht der Algorithmus in "Matrix-Schreibweise" konkret aus?  
 $S[i, j] = \dots$
- Ändert sich die Laufzeit oder der Speicherbedarf im Vergleich zum globalen Alignment?