

Sequenzen-Alignierung in der Bioinformatik

VO Algorithm Engineering

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering, LS11

23. VO

24.01.2006

Literatur für diese VO

- Volker Heun: Skriptum zur Vorlesung Algorithmische Bioinformatik I/II, SS05 und WS05/06, LMU München, Kap. 5 und 8
- Knut Reinert: Skriptum zur Vorlesung Algorithmische Bioinformatik, WS2005, FU Berlin, Kap. 1
- Folien(-teil): Danke an Gabriele Koller, TU Wien

Überblick

Paarweise Sequenzen Alignierung

Multiple Sequenzen Alignierung

Überblick

Paarweise Sequenzen Alignierung

- Dynamische Programmierung
- Kürzeste-Wege (A* Algorithmus)

- Dyn. Programmierung mit Bandbreite k

Levenshtein 1966: Einf. der Edit-Distanz, aber kein Algorithmus
Needleman & Wunsch 1970: Dyn. Progr. Algorithmus

KBand: DP mit Bandbreite k

- Idee: Betrachte nur die Zellen innerhalb eines Bandes der Höhe $2k$ um die Diagonale von $(0,0)$ nach (m,n) . Dann ist die Laufzeit und Speicherplatz nur noch $O(kn)$.
- Frage: Ist das gefundene Alignment optimal (maximum)?
- Geg. 2 Sequenzen s und t der Länge n . Sei M ein uniformer Match score und p der gap penalty.
- Frage: Sei a_k der beste Zielfunktionswert des *KBand* Algorithmus für das gegebene k . Wann ist a_k gleich dem optimalen Alignmentwert a ?
- Lemma: Falls $a_k \geq M(n-k-1) - 2(k+1)p$, dann gilt $a_k = a$.
Bw: Wenn a_k nicht optimal ist, dann wird das Band überschritten, es ex. also in jeder Sequenz mind. $k+1$ Gaps, damit max. $n - (k+1)$ Matches.

DPlinear: DP mit linearem Platz

Idee: während der Berechnung wird nur die momentane Zeile i und die unmittelbar darüber liegende Zeile $i-1$ benötigt. Speichere immer nur diese beiden relevanten Zeilen!

Input : Sequenzen s und t

Output: Ähnlichkeit zwischen s und t

$m \leftarrow |s|$; $n \leftarrow |t|$;

for $i \leftarrow 0$ **to** n **do** // erste Zeile:
 $D[i] \leftarrow D[i-1] + g$; // t aligniert mit -

for $j \leftarrow 0$ **to** m **do**
 for $i \leftarrow 0$ **to** n **do**

$D'[i] \leftarrow D[i]$

$D[0] \leftarrow D'[0] + g$ // erste Spalte: s aligniert mit -

for $i \leftarrow 1$ **to** n **do**

$D[i] \leftarrow \max($ $D'[i] + g,$ // s aligniert mit -
 $D'[i-1] + p(i,j),$
 $D[i-1] + g);$ // t aligniert mit -

return $D[n]$;

Problem:

Zielfunktionswert bekannt, aber die Lösung selbst nicht (keine Zwischenergebnisse)

Sequenzen-Alignierung mit linearem Platz (Hirschberg, D&C)

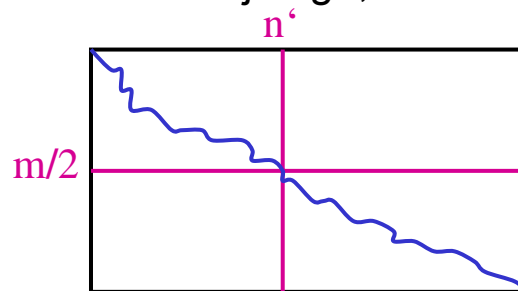
Idee: Divide & Conquer

- Ann.: kennt optimales Alignment zwischen s und t
- Aufteilung in zwei Teile: jeweils $m/2$ Zeichen aus s (Achtung: evtl. nicht eindeutig, da mehrere –)
- Sei n' : Anzahl der Zeichen aus t im ersten Teil
- Seien also: $s=(s' s'')$, $t=(t' t'')$ Aufteilung der Sequenzen und $\underline{s}=(\underline{s}' \underline{s}'')$ bzw. $\underline{t}=(\underline{t}' \underline{t}'')$ des Alignments
- Es gilt: $(\underline{s}', \underline{t}')$ und $(\underline{s}'', \underline{t}'')$ sind optimale Alignments
- D.h.: D&C funktioniert, Conquer: Concatenate
- Problem: Wo liegt n' ?

Sequenzen-Alignierung mit linearem Platz (Hirschberg, D&C)

Idee:

- Berechne alle optimalen Alignment-Distanzen von $s_1 \dots s_{m/2}$ mit $t_1 \dots t_n$ für alle n' aus $[0..n]$: mit Hilfe des Algorithmus *DLinear*
- Berechne alle optimalen Alignment-Distanzen von $s_{m/2+1} \dots s_m$ mit $t_{n'+1} \dots t_n$ für alle n' aus $[0..n]$: mit *DLinear* (durch Drehung der DP Matrix)
- Das korrekte n' ist dasjenige, das die Summe maximiert.



Laufzeit Analyse

- *DPlinear* besucht jeweils $(m/2+1)(n+1)$ Felder
- $T(n,m)=(m+2)(n+1)+T(m/2,k)+T(m/2,n-k)$
- Beh.: Es gilt $T(m,n) \leq 3 m (n+1) + 8 n \log(m)$
- Laufzeit: $O(mn)$
- Speicherplatz: $O(\min\{m,n\})$

Gap-Funktionen

Gap: Folge von $k > 1$ Leerzeichen

bisher: lineare Gap-Funktion $w(k) = g \cdot k$

Alignment 1:

```
GACTGGAATG
G-C-T-A-TG
```

Alignment 2:

```
GACTGCAATG
G-----CTATG
```

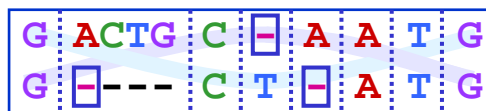
Überlegung:

Mutation, die Gap mit k Leerzeichen einfügt,
ist wahrscheinlicher als Mutation, die k isolierte
Leerzeichen einfügt.

Affine Gap-Funktionen

Etwas weniger allgemein, aber in $O(n^2)$?

Idee: 1. Leerzeichen höher bestrafen ($h+g$)



Affine Funktion: $w(k) = h + gk$, mit $w(0) = 0$.

- Lösung wieder mit Dynamischer Programmierung
- 3 $(m+1) \times (n+1)$ -Matrizen:
- $a(i,j)$ bester Wert bis (i,j) u.d. Bed., dass s_i mit t_j aligniert
- $b(i,j)$ bester Wert bis (i,j) u.d. Bed., dass s_i mit $-$ aligniert
- $c(i,j)$ bester Wert bis (i,j) u.d. Bed., dass t_j mit $-$ aligniert

DP-Algorithmus für affine Gap-Funktionen

Rekursion zur Matrix-Berechnung: ($1 \leq i \leq m, 1 \leq j \leq n$)

$$a[i,j] = p(i,j) + \max\{a[i-1,j-1], b[i-1,j-1], c[i-1,j-1]\}$$

$$b[i,j] = \max\{a[i,j-1] - (h+g), b[i,j-1] - g, c[i,j-1] - (h+g)\}$$

$$c[i,j] = \max\{a[i-1,j] - (h+g), b[i-1,j] - (h+g), c[i-1,j] - g\}$$

Ergebnis: $\max\{a[m,n], b[m,n], c[m,n]\}$

→ Laufzeit: $O(mn)$

Gotoh 1982

→ Speicher: $O(mn)$, aber 3 Matrizen

Überblick

Paarweise Sequenzen Alignierung

- Dynamische Programmierung
- Kürzeste-Wege (A* Algorithmus)
- Dyn. Programmierung mit Bandbreite k
- Alignierung mit linearem Platz
- DP für affine Lückenstrafen (Gap penalties)

jetzt: Multiple Sequenzen Alignierung

Vergleich mehrerer Sequenzen

Ziel: optimales multiples Alignment von $k > 2$ Sequenzen

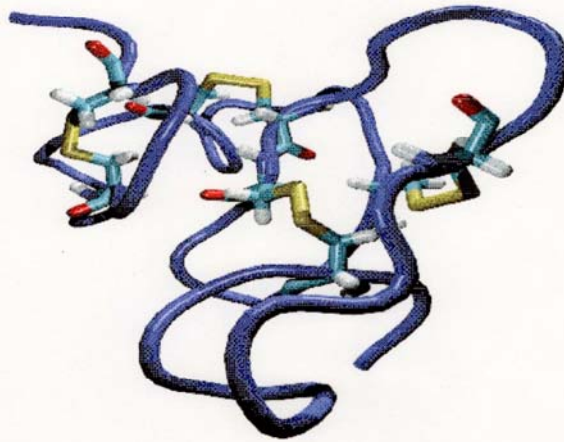
Anwendungsgebiete:

- Phylogenetische Analyse:
Darstellung von Abstammungsbeziehungen auf Basis der Distanz
- Identifizierung von Motiven und Profilen:
durch die Evolution erhalten gebliebene Merkmale
- Struktur- und Funktionsvorhersage

```

AATAHAQRQG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACYT
VAATNAQTQG KQNDGMIAPH NLCCSQFGYC GLGRDYCGTG ..CQSGACCS
VGLVSAQRQG SQGGGGTCPA LWCCSIWGWG GDSEPYCGRT ..CENK.CWS
AATAQAQRQG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
.....QRQG EQGSGMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
SETVKSQNQG .....CAP NLCCSQFGYC GSTDAYCGTG ..CRSGPCRS
RGSAE..QQG RQAGDALCPG GLCCSSYGYC GTTVDYCGIG ..CQSQ.CDG
AGPAAAQNG .....CQP NFCCSKFGYC GTTDAYCGDG ..CQSGPCRS
AGPAAAQNG .....CQP NVCCSKFGYC GTTDEYCGDG ..CQSGPCRS
RGSAE..QQG RQAGDALCPG GLCCSSYGYC GTTADYCGDG ..CQSQ.CDG
RGSAE..QQG RQAGDALCPG GLCCSFYGYC GTTVDYCGDG ..CQSQ.CDG
TGVAIAEQQG RQAGGKLPN NLCCSQWGWG GSTDEYCPD HNCQSN.CK.
.....EQQG RQAGGKLPN NLCCSQYGYC GSSDDYCPSP KNQSN.CK.

```



Berechnung multipler Alignments

Schwieriges Problem:

1. Wahl der Sequenzen:
Homologie (gemeinsamer Vorfahr) vorausgesetzt
→ Aufgabe der Biologen
2. Wahl der Bewertungsfunktion:
Ideal: biologisch korrektes Alignment erzielt
höchsten Wert
3. Wahl des Algorithmus: komplexe Aufgabe
 - exakte Verfahren
 - Heuristik: progressiv, iterativ

Multiple Sequence Alignment (MSA)

Multiples Alignment von $k > 2$ Sequenzen s_1, \dots, s_k :

- alle Sequenzen durch ev. Einfügen von Leerzeichen auf dieselbe Länge erweitern
- keine Spalte darf nur aus Leerzeichen bestehen

Beispiel: Multiples Alignment von Proteinsequenzen

Sequenzen

```
1 PEAAALYGRFTIKS
2 PEALNYGWYSSES
3 PEVIRMQDDNPFSFQS
4 PESLAYNKFSIKS
```



MSA

```
1 PEAAALYGRFT---IKS
2 PEALNYGWY---SSES
3 PEVIRMQDDNPFSFQS
4 PESLAYNKF---SIKS
```

Bewertung eines multiplen Alignments

Beispiel:

Multiplles Alignment

```
1 PEALYGRFT---IKS
2 PEALNYGWY---SSES
3 PEVIRMQDDNPF SFQS
```

Beschränkung auf additive Funktionen

Anforderungen:

- Spalten mit vielen gleichen Symbolen erhalten höheren Wert als solche mit unterschiedlichen oder Leerzeichen
- Reihenfolge der Sequenzen egal

Naheliegend: k -dimensionales Feld für jede Kombination (Platzbedarf steigt exponentiell mit k)

Sum-of-Pairs-Maß

SP-Score: Summe der paarweisen Alignment-Werte einer Spalte, wobei $p(-,-) = 0$

Beispiel:

Multiples Alignment

```
1 PEAALYGRFT--IKS
2 PEALNYGWY--SSES
3 PEVIRMQDDNPFSFQS
```

$$\text{SP-Score(Spalte 3)} = p(\mathbf{A}, \mathbf{A}) + p(\mathbf{A}, \mathbf{V}) + p(\mathbf{A}, \mathbf{V})$$

$$\text{SP-Score(Spalte 12)} = p(-, -) + p(-, \mathbf{F}) + p(-, \mathbf{F})$$

Sequenzen 1 und 2

```
1 PEAAALYGRFT--IKS
2 PEALNYGWY--SSES
```



Projektion

```
1 PEAAALYGRFT-IKS
2 PEALNYGWY-SSES
```

Bewertungsmatrix für den Proteinvergleich

Bewertungsschema soll Ähnlichkeiten von Aminosäuren (Mutationswahrscheinlichkeit) reflektieren

PAM-Matrizen (Dayhoff, 1978):
auf Basis von akzeptierten Mutationen

Palette von PAM-Matrizen

- PAM40: kürzere, starke lokale Ähnlichkeiten
- PAM250: längere, schwächere lokale Ähnlichkeiten

BLOSUM-Serie (Henikoff und Henikoff, 1992)

Affine Gap-Funktion: *GOP* und *GEP*

Beispiel: PAM-Matrix

PAM10-Matrix für die 20 Aminosäuren

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	7	-10	-7	-6	-10	-7	-5	-4	-11	-8	-9	-10	-8	-12	-4	-3	-3	-20	-11	-5
R	-10	9	-9	-17	-11	-4	-15	-13	-4	-8	-12	-2	-7	-12	-7	-6	-10	-5	-14	-11
N	-7	-9	9	-1	-17	-7	-5	-6	-2	-8	-10	-4	-15	-12	-9	-2	-5	-11	-7	-12
D	-6	-17	-1	8	-21	-6	0	-6	-7	-11	-19	-8	-17	-21	-12	-7	-8	-21	-17	-11
C	-10	-11	-17	-21	10	-20	-20	-13	-10	-9	-21	-20	-20	-19	-11	-6	-11	-22	-7	-9
Q	-7	-4	-7	-6	-20	9	-1	-10	-2	-11	-8	-6	-7	-19	-6	-8	-9	-19	-18	-10
E	-5	-15	-5	0	-20	-1	8	-7	-9	-8	-13	-7	-10	-20	-9	-7	-9	-23	-11	-10
G	-4	-13	-6	-6	-13	-10	-7	7	-13	-17	-14	-10	-12	-12	-10	-4	-10	-21	-20	-9
H	-11	-4	-2	-7	-10	-2	-9	-13	10	-13	-9	-10	-17	-9	-7	-9	-11	-10	-6	-9
I	-8	-8	-8	-11	-9	-11	-8	-17	-13	9	-4	-9	-3	-5	-12	-10	-5	-20	-9	-1
L	-9	-12	-10	-19	-21	-8	-13	-14	-9	-4	7	-11	-2	-5	-10	-12	-10	-9	-10	-5
K	-10	-2	-4	-8	-20	-6	-7	-10	-10	-9	-11	7	-4	-20	-10	-7	-6	-18	-12	-13
M	-8	-7	-15	-17	-20	-7	-10	-12	-17	-3	-2	-4	12	-7	-11	-8	-7	-19	-17	-4
F	-12	-12	-12	-21	-19	-19	-20	-12	-9	-5	-5	-20	-7	9	-13	-9	-12	-7	-1	-12
P	-4	-7	-9	-12	-11	-6	-9	-10	-7	-12	-10	-10	-11	-13	8	-4	-7	-20	-20	-9
S	-3	-6	-2	-7	-6	-8	-7	-4	-9	-10	-12	-7	-8	-9	-4	7	-2	-8	-10	-10
T	-3	-10	-5	-8	-11	-9	-9	-10	-11	-5	-10	-6	-7	-12	-7	-2	8	-19	-9	-6
W	-20	-5	-11	-21	-22	-19	-23	-21	-10	-20	-9	-18	-19	-7	-20	-8	-19	13	-8	-22
Y	-11	-14	-7	-17	-7	-18	-11	-20	-6	-9	-10	-12	-17	-1	-20	-10	-9	-8	10	-10
V	-5	-11	-12	-11	-9	-10	-10	-9	-9	-1	-5	-13	-4	-12	-9	-10	-6	-22	-10	8

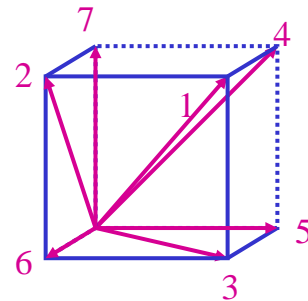
Dynamische Programmierung für MSA

Bei 2 Sequenzen gibt es immer 3 Möglichkeiten, ein Alignment fortzusetzen:

H	--	H
--	P	P

Bei 3 Sequenzen gibt es bereits 7 Möglichkeiten:

H	--	H	H	H	--	--
P	P	--	P	--	--	P
E	E	E	--	--	E	--



Bei N Sequenzen gibt es $2^N - 1$ Möglichkeiten

Dynamische Programmierung für MSA

Variante des DP-Algorithmus

(Vereinfachung: s_1, \dots, s_k mit Länge n)

- k -dimensionales Feld a mit $n+1$ Einträgen pro Dim.
- $a[i_1, \dots, i_k]$ enthält Wert des besten Alignments von $s_1[i_1], \dots, s_k[i_k]$
- Initialisierung: $a[0, \dots, 0]$ mit 0
- Feldeinträge: $a[i_1, \dots, i_k]$ auf Basis der 2^k-1 Vorgänger und dem aktuellen Spaltenwert ermitteln
- Optimales Alignment durch Rückverfolgung

Aufwand für dynamische Programmierung

- Berechnung des kompletten Feldes: n^k Einträge
⇒ Speicherplatz $O(n^k)$, Laufzeit: $\Omega(n^k)$
 - Berechnung eines Eintrags: $2^k - 1$ Vorgänger
(Symbol oder Leerzeichen jeder Sequenz) ⇒ $O(2^k)$
 - Berechnung der Spaltenwerte:
 - einfaches Schema (Symbole zählen) ⇒ $O(k)$
 - SP-Score: $k(k-1)/2$ Paare ⇒ $O(k^2)$
- ⇒ Gesamtlaufzeit: $O(k^2 2^k n^k)$ bzw. $O(k^2 2^k n^k)$

MSA mit SP-Maß ist NP-vollständig!

Ideen für dynamische Programmierung

Verkleinere den Suchraum:

- Nur *relevante* Zellen werden berechnet (Carillo-Lipman)
 - Divide & Conquer Ansatz (Stoye 1997)
 - Kürzeste Wege mittels A* Algorithmus
- Zwar theoretisch keine Ersparnisse, aber praktisch!

Algorithmen für MSA

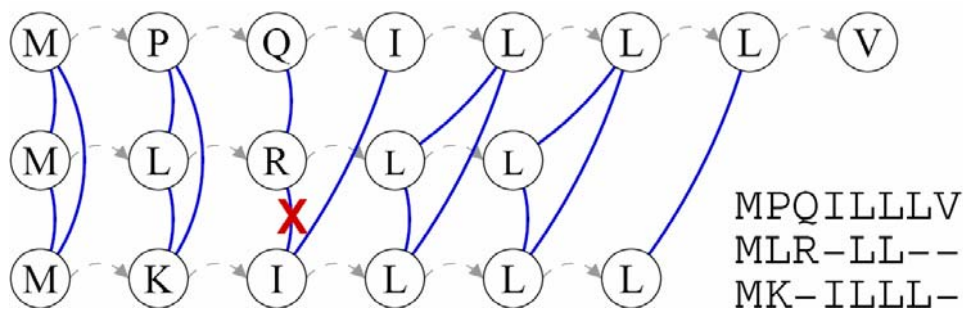
1. Exakte Verfahren
2. Heuristiken
 - Progressive Verfahren
 - Tree Alignments
 - Star Alignments
 - Iterative Verfahren
 - deterministische Methoden
 - stochastische Methoden
3. Konsistenzbasierte Verfahren

Konsistenzbasierte Verfahren

Ziel: multiples Alignment, das am besten zu allen optimalen paarweisen Alignments passt

- Formulierung als Maximum Weight Trace Problem (Kececioglu, 1993): *Trace*, das die Summe des Gewichts der realisierten Kanten maximiert

Alignmentgraph für 3 Sequenzen



ENDE

27