

# Bioinformatik

Multiple Sequence Alignment  
Sum-of-pairs Score



Ulf Leser

Wissensmanagement in der  
Bioinformatik



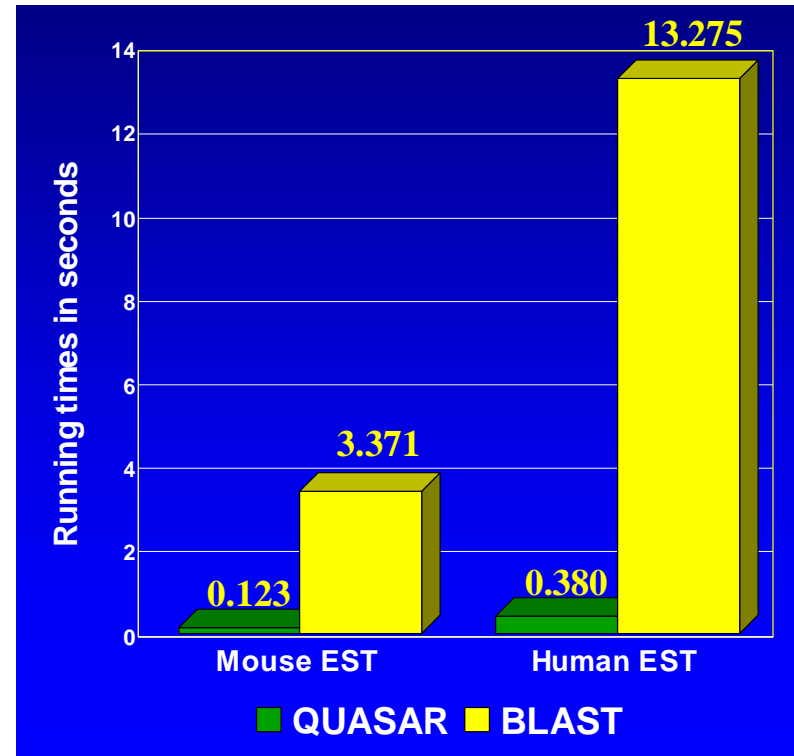
# Quasar Grundidee

---

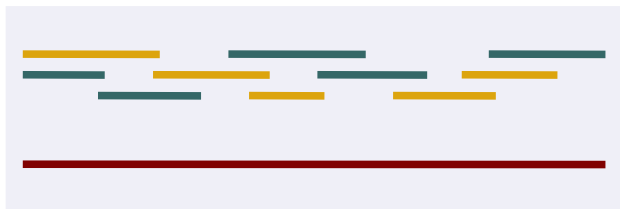
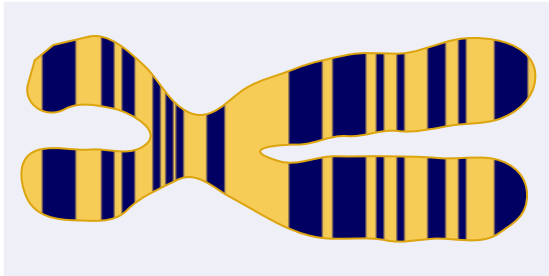
- Search-Phase sucht Regionen mit Länge  $w$  und hoher Ähnlichkeit
  - D.h. Regionen mit Editabstand  $< k$
- $q$ -Gramme sind Substrings der Länge  $q$ 
  - $q$ -Gramme =  $q$ -mere
- Dann gilt das  **$q$ -Gramm Lemma**
  - Gegeben zwei Sequenzen  $X, Y$  der Länge  $w$  mit *Editabstand*  $< k$ . Dann haben  $X$  und  $Y$  *mindestens*  $t = (w + 1) - (k + 1) * q$   $q$ -Gramme gemeinsam.
- Beweis
  - $X$  und  $Y$  besitzen jeweils  $(w - q + 1)$   $q$ -Gramme der Länge  $q$
  - Wenn in einem Substring  $Z$  in  $X$  ein Mismatch mit  $Y$  auftaucht, dann ist dieser Mismatch in  $q$   $q$ -Grammen enthalten
  - $X$  enthält höchstens  $k$  Mismatches zu  $Y$
  - Diese können höchstens  $k * q$   $q$ -Gramm-Matches „zerstören“
  - Damit bleiben mindestens  $t = (w - q + 1) - k * q = (w + 1) - (k + 1) * q$  perfekt matchende  $q$ -Gramme übrig
  - qed.

# Ergebnisse

- Test Parameters
  - 6% Error
  - $w = 50$ ,  $q = 11$
  - scan with BLAST
  - averaged for 1000 queries
- ~30 times faster than BLAST
- Bei etwa gleichbleibender Sensitivität
- Klappt nur für sehr ähnliche Sequenzen
- Nur schnell, wenn der  $q$ -Gramm Index komplett in Hauptspeicher passt

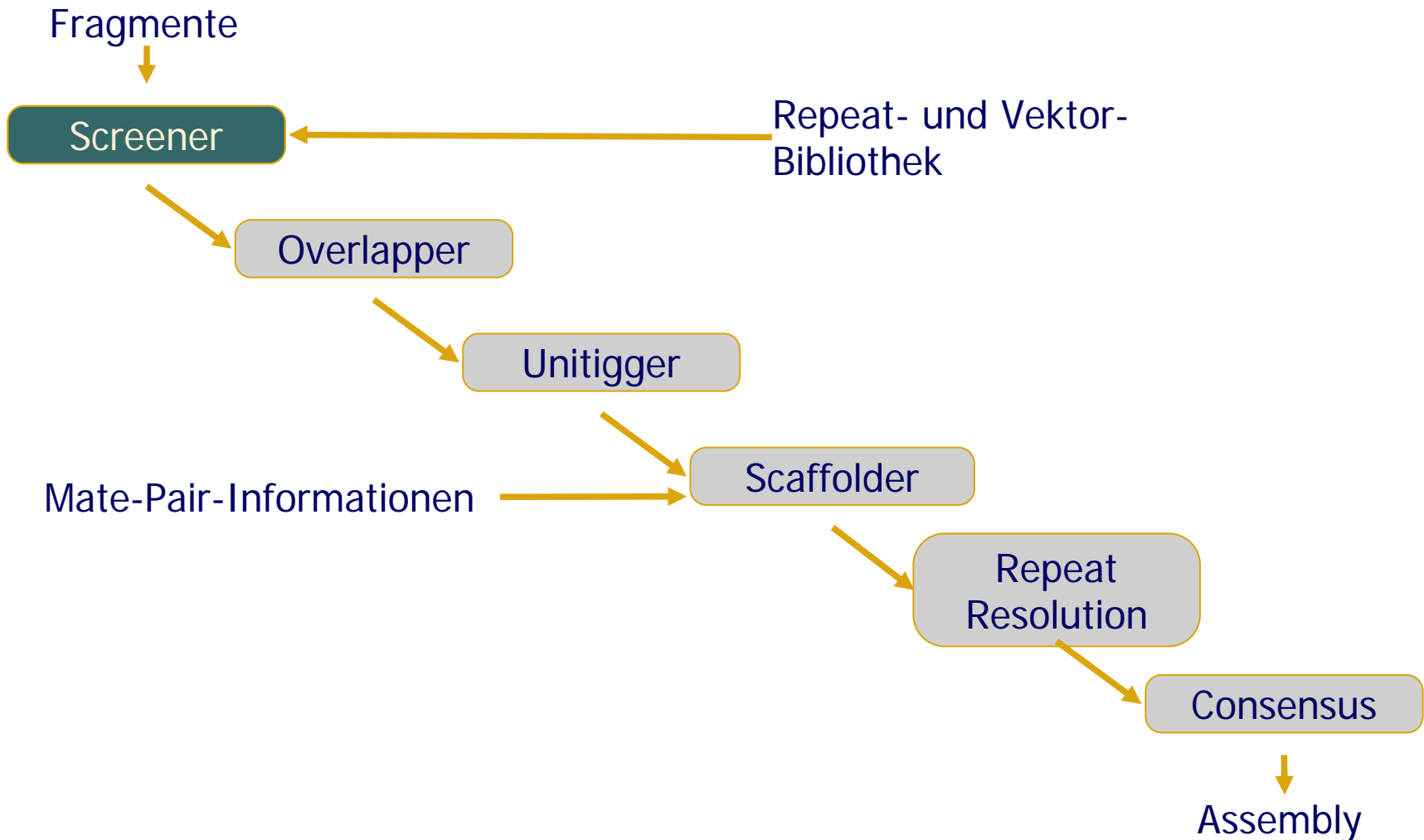


# Whole Genome Shotgun



- Zerschneiden eines kompletten Chromosoms in Einzelstücke
- (An)sequenzierung jedes Bruchstücks
- **Assembly** der Einzelsequenzen zur Konsensussequenz
- Konkret
  - Clone der Länge 2kb, 10kb, 50kb und 150kb
  - "Double Barrel" Shotgun
    - **Ansequenzieren** von beiden Seiten

# Assembler - Überblick



# Overlapper

---

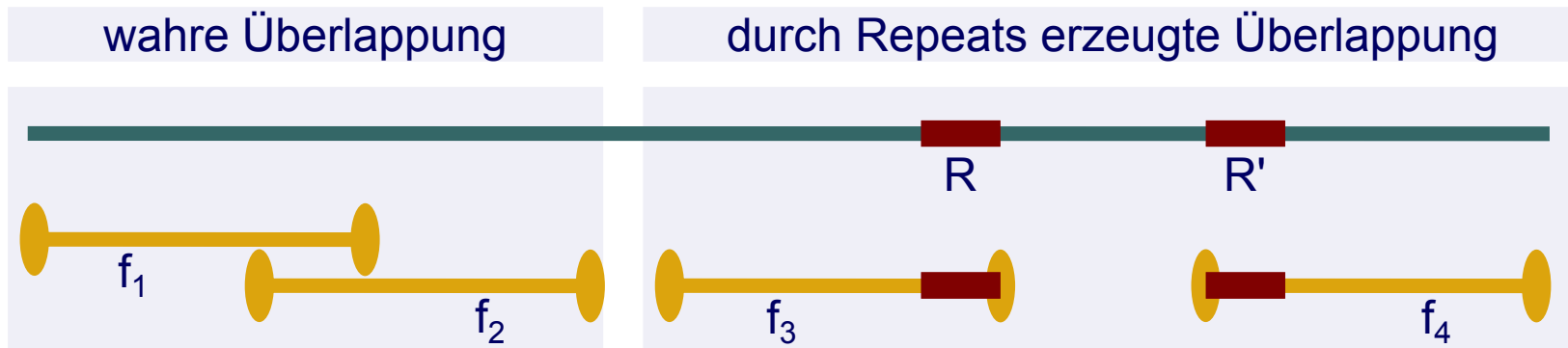
- Suche nach Überlappungen zwischen Fragmenten
- Möglichkeiten wie Fragmente überlappen können:



- Dynamische Programmierung?
  - Zu langsam
  - Nur high-scoring Overlaps relevant

# Repeats

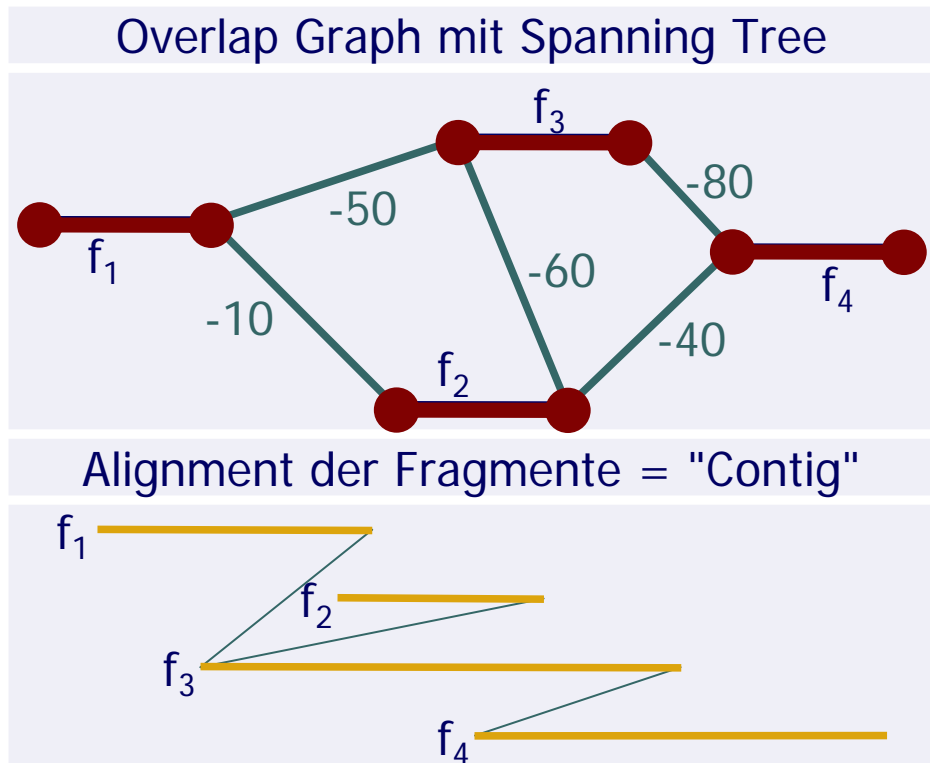
- Repeat-induced Overlaps



- Vermeiden durch
  - Screening bekannter Repeats
  - Sehr häufige k-mere ignorieren
- **Repeats bleiben das Hauptproblem des Assemblies**
  - Führen zu falsch-positiven Überlappungen
  - Bilden Verbindungen über das gesamte Chromosom
  - Zerstören/Überlagern die wahre Anordnung

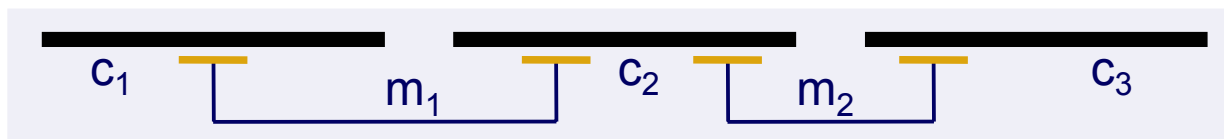
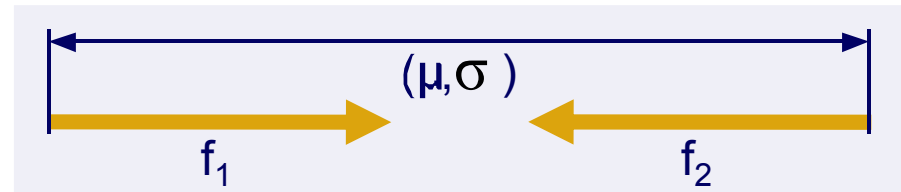
# Unitigs

- Verbindet Fragmente zu Unitigs
  - „Unique Contigs“
- Einfache Heuristik:  
**Spannender Wald**
  - Greedy Verfahren
  - „Leichte“ Kanten zuerst
  - Kanten, die Zyklen erzeugen, ignorieren
  - So lange, bis jede Zusammenhangskomponente abgedeckt ist



# Scaffolder

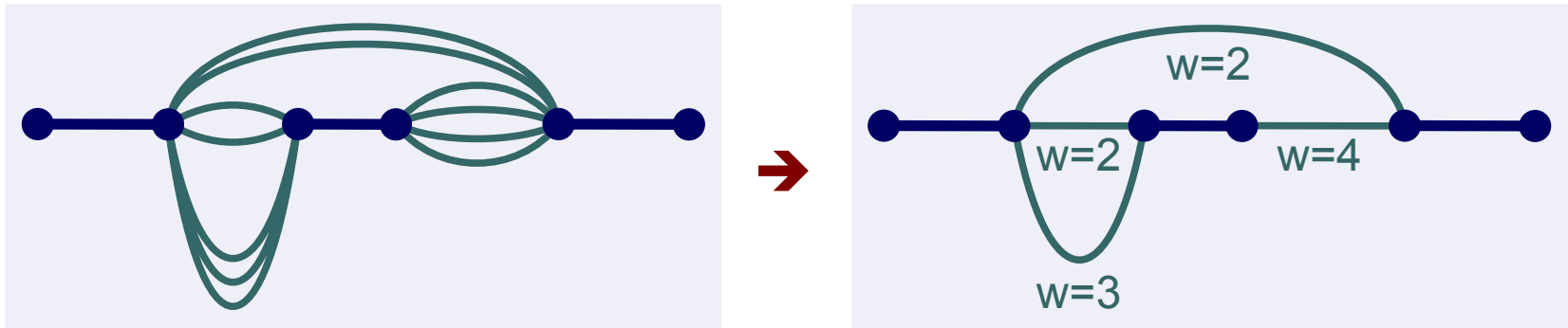
- Verbindet Unitigs zu Scaffolds
  - Gerüst für ein komplettes Chromosom
- Verwendet Unitigs und **Mate-Pairs**
  - "Double Barrel" Shotgun:  
Paare von Fragmenten mit bekannter Orientierung, Distanz  $\mu$  und Standardabweichung  $\sigma$
- Scaffold
  - Durch Mate-Pairs verbundene Contigs



# Vereinfachung

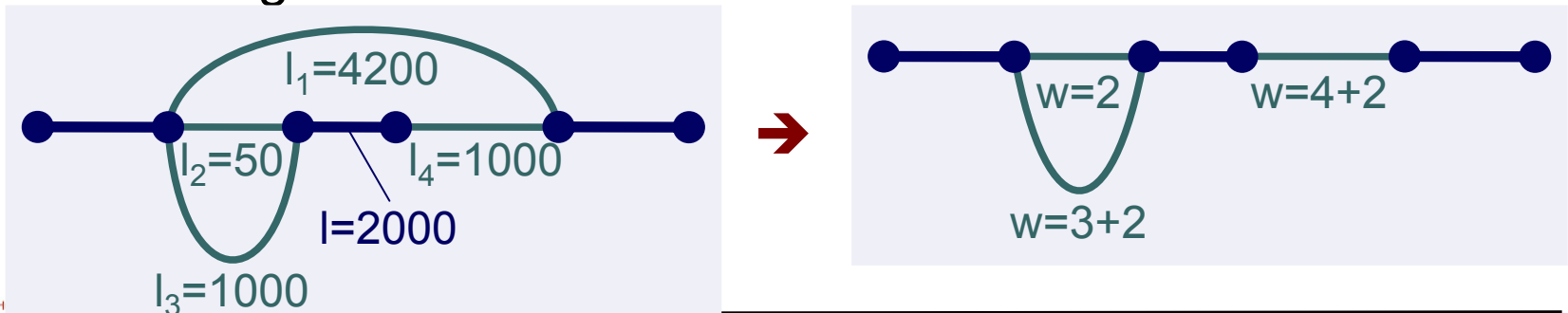
- Edge-Bundling

- Heuristisch sich bestätigende Mate-Kanten verschmelzen



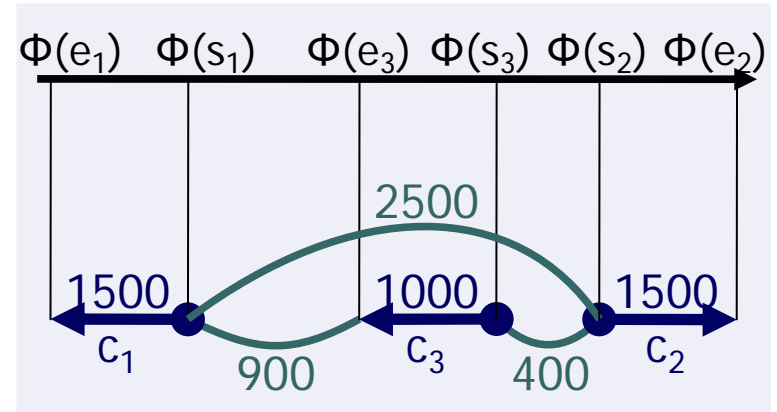
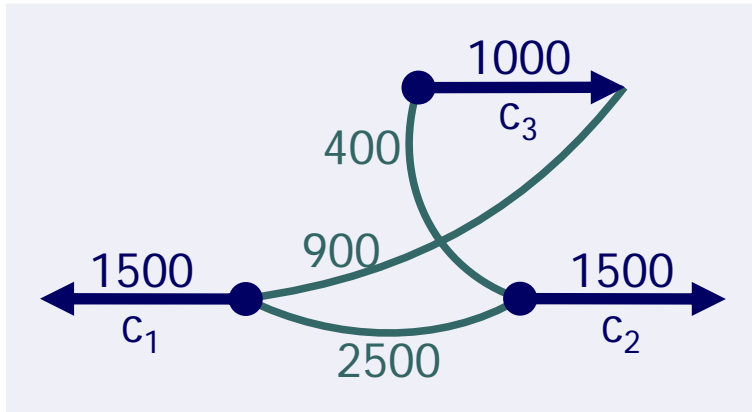
- Transitive **Kantenreduktion**

- Lange Mate-Kanten auf Pfade reduzieren



# Happy Mate-pairs

- Koordinatenberechnung für Knoten
  - Reihenfolge und Orientierung der Contigs im Graph
  - Erhaltung der Längen und Abstände



## Happy Mate-Pairs

- Richtige Orientierung, Länge, Abstand bezogen auf  $\Phi$
- Wir suchen Anordnung  $\Phi$  so, dass **möglichst viele „wichtige“ Mate-pairs happy sind**

# Greedy Path-Merging Algorithmus

---

- Verwendung einer anderen Heuristik
- Pfad: Zusammenhänge Menge von Contigs
- Greedy Path-Merging Algorithmus
  - Sortiere Mate-Kanten absteigend nach Gewicht
  - Für alle Kanten  $e = (v,w)$ 
    - Seien  $P_1 / P_2$  die durch  $e$  neu verbundenen Pfade
    - Versuche,  $P_1$  und  $P_2$  zu mergen (nächste Folie)
    - Wenn dabei **mehr Happy- als Unhappy-Edges** hinzukommen
      - Ersetze  $P_1$  &  $P_2$  durch neuen Pfad  $P$
    - Entferne  $e$
- Ergebnis:  $\Phi$ 
  - Enthält unhappy Kanten, aber hoffentlich nicht allzu viele

# Consensus

---

- Ermittelt Konsensussequenz für einzelne Contigs
- Input:
  - Scaffolds
- Output:
  - Konsensussequenz
- Multialignment (Heuristik)

$f_1$	CAACTACAAGA-CTTCATGGC
$f_2$	TCCAAGAACTTAATGGCAAAT-TTC
$f_3$	ATTC-ACTAC
$f_4$	TAATAGCAAATCTTCGAAAAACC
Consensus	ATTCAACTACAAGAACTTAATGGCAAATCTTCGAAAAACC

# Menschliches Genom

---

- 27 Mio Fragmente, Ø-Länge: 550b, 70% Mate-Pairs

	CPU-Stunden		Max. Speicher
Screeener	4800	2- 3 Tage auf 10- 20 Computer	2 GB
Overlapper	12000	10 Tage auf 10- 20 Computern	4 GB
Unitigger	120	4- 5 Tage auf 1 Computer	32 GB
Scaffolder	120	4- 5 Tage auf 1 Computer	32 GB
Repeat Res	50	2 Tage auf 1 Computer	32 GB
Consensus	160	1 Tag auf 10- 20 Computern	2 GB
Total	18000		

- Ergebnis: **Assembly besteht aus 6500 Scaffolds**
  - Umfasst 2,8 Gb Sequenz
  - **150.000 Gaps** (insgesamt 148 Mb)

# Inhalt dieser Vorlesung

---

- Multiples Sequenzalignment
- Motivation
- Sum-Of-Pair Zielfunktion
- Suchen mit MSAs
  - Profile und deren Bewertung

# Definition

---

- Bisher
  - Immer Vergleich zweier Strings
- Jetzt
  - Multipler Stringvergleich: Vergleich von  $k > 2$  Strings
- Definition
  - *Ein **multiple Sequenzalignment (MSA)** von  $k$  Strings  $S_i$ ,  $1 \leq i \leq k$ , ist eine Tabelle mit  $k$  Zeilen und  $l$  Spalten, so dass*
    - *In Zeile  $i$  steht String  $S_i$ , mit beliebig eingefügten Leerzeichen*
    - *Jedes Zeichen jedes  $S_i$  steht in exakt einer Spalte*
    - *In keiner Spalte stehen nur Leerzeichen*
- Bemerkungen
  - Direkte Generalisierung des Alignment zweier Strings
  - Es folgt, dass  $l = |\text{MSA}| \leq \sum(|S_i|)$ 
    - Warum?



# Motivation

---

- Alignment sucht **ähnliche Sequenzen**
  - Weil: ähnliche Sequenz – ähnliche Struktur – ähnliche Funktion
- MSA sucht „**das Ähnliche**“ in vielen Sequenzen
  - Argumentationsrichtung ist umgekehrt
  - Auch beim paarweisen Alignment findet man Regionen guter Übereinstimmung – aber nur für diese zwei Sequenzen
  - MSA startet mit vielen Sequenzen, bei denen man ähnliche Funktion / Struktur vermutet
  - MSA stellt fest, was das Gemeinsame dieser Sequenzen ist – Domänen, Motive, Signaturen, Profile, ...
  - These: dieses Gemeinsame ist **biologisch relevant**

# Motivation II

---

- Proteine (und damit auch DNA) setzen sich aus **funktionalen Blöcken** und „Zwischenraum“ zusammen
  - Die Blöcke findet man nicht, wenn man Sequenzen nur paarweise vergleicht
    - Bzw. man kann sie nicht vom Rauschen unterscheiden
- Trennung des eventuell zufällig Gemeinsamen (Alignment) vom bedeutungsvoll Gemeinsamen (MSA)

```
AAC__GTG__AT__T__GAC__  
_TCGAGTGC_TTTACA_GT
```

```
AAC__GTG__AT__T__GAC__  
_TCGAGTGC_TTTACA_GT  
GCCG__TGC__TA__GTCG_  
TTC__AGTGGACGTG__GTA  
G____GTGCA__TGACC__
```

# Konservierte Domänen

---

- Gedankengang
  - Gegeben: Proteine  $S_1, \dots, S_k$  mit ähnlicher Funktion
    - Z.B.: Können durch die Zellwand tunneln, an DNA binden, bestimmte Proteine aktivieren, etc.
  - Annahme: identischer evolutionärer Ursprung
    - Es gab einmal das „Mutterprotein“  $S$
  - $S$  unterliegt Evolution
    - Mutation, Rekombination und Selektion
  - Abschnitte in  $S_i$ , die trotz **Evolution** gleich blieben (**konserviert** sind), müssen für die gemeinsame Funktion wichtig sein
    - Andere Abschnitte dagegen sind nicht oder weniger wichtig
- Also: Um **funktionale Blöcke** zu finden, muss man viele Sequenzen vergleichen
  - Sequenzieren bleibt wichtig

# Blöcke, Domänen, Sites

---

- Proteine
  - Bindungsstelle für andere Proteine / Liganden / Moleküle
  - Bindungsstelle an DNA
  - Phosphorylierung / Dephosphorylierungsstelle
  - Signal zum Transport des Proteins
  - Signal zum Abbau des Proteins
  - ...
- DNA
  - Bindungsstellen für Proteine
  - Promotoren und Inhibitoren
  - Strukturtragende Bereiche
  - Start- und Stoppcodons
  - Signal für differenzielles Splicen
  - ...

# Proteinfamilien


---

- Man unterteilt Proteine in Familien, Superfamilien, ...
  - Diverse Klassifikationen vorhanden (CATH, SCOP, ...)
- Idee
  - X00.000 Proteine zerfallen in X.000 Klassen ähnlicher Funktion?, Struktur?, Substruktur?
  - Untersuchung von Vertretern von Familien statt aller Proteine
  - Finden familienspezifischer Domänen
  - Benutzung familienspezifischer Substitutionsmatrizen
- Verwendung
  - Starte mit Proteinen gleicher/ähnlicher Funktion
  - **Finde das Gemeinsame durch MSA**
  - Suche nur mit hochkonservierten Blöcken nach weiteren Vertretern
  - Modifiziere Familie entsprechend
  - Iteriere, bis Zufriedenheit eintritt

# Beispiel: SCOP

- Structural Classification of Proteins
- Hierarchische Anordnung
  - *Fold*: Major structural similarity
    - All Alpha, All Beta, Membrane proteins, ...
  - *Superfamily*: Probable common evolutionary origin
    - Nucleotide-binding domain, Neurotransmitter-gated ion-channel transmembrane pore, ...
  - *Family*: Clear evolutionarily relationship
    - Globins, Death Domain, 4 families of Immunoglobulin ...
  - Protein
  - Spezies

Structural Classification of Proteins



**Root: scop**

**Classes:**

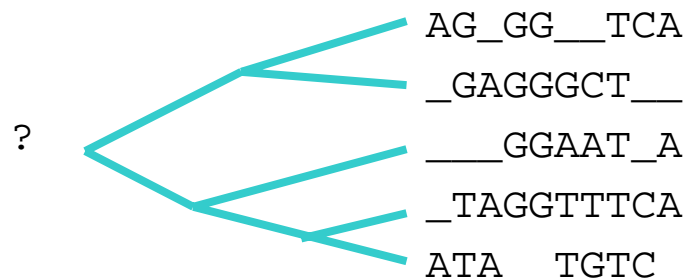
1. [All alpha proteins](#) [46456] (179)
2. [All beta proteins](#) [48724] (126)
3. [Alpha and beta proteins \(a/b\)](#) [51349] (121)   
*Mainly parallel beta sheets (beta-alpha-beta units)*
4. [Alpha and beta proteins \(a+b\)](#) [53931] (234)   
*Mainly antiparallel beta sheets (segregated alpha and beta regions)*
5. [Multi-domain proteins \(alpha and beta\)](#) [56572] (38)   
*Folds consisting of two or more domains belonging to different classes*
6. [Membrane and cell surface proteins and peptides](#) [56835] (36)   
*Does not include proteins in the immune system*
7. [Small proteins](#) [56992] (66)   
*Usually dominated by metal ligand, heme, and/or disulfide bridges*
8. [Coiled coil proteins](#) [57942] (6)   
*Not a true class*
9. [Low resolution protein structures](#) [58117] (18)   
*Not a true class*
10. [Peptides](#) [58231] (105)   
*Peptides and fragments. Not a true class*
11. [Designed proteins](#) [58788] (39)   
*Experimental structures of proteins with essentially non-natural sequences. Not a true class*

Enter [search](#) key:

# MSA Zielfunktion

---

- **Zielfunktion** beim einfachen Alignment war klar
  - Möglichst wenig I,R,D
  - Eventuell mit Substitutionsmatrix
  - Eventuell mit spezieller Behandlung von Gaps
- Zielfunktion für MSA ist nicht so klar
  - Score einer Spalte mit 2 T, zwei G und einem Leerzeichen?
  - Angabe einer Substitutionsmatrix für k Sequenzen über Alphabet  $\Sigma$  würde  $O(|\Sigma|^{k+1})$  Werte erfordern
  - Nicht machbar und biologisch nicht begründbar



# MSA Überblick

---

- Weg über Substitutionsmatrizen nicht gangbar
- Verschiedene alternative Vorschläge für Zielfunktionen existieren
  - Maximiere die Summe aller paarweisen Alignments
  - Maximiere die Summe der Alignments jeder Sequenz zu einer Consensussequenz (Center-Star)
  - Maximiere die Summe der Alignments folgend dem phylogenetischen Baum der Sequenzen

# Definitionen

---

- Definition

- Gegeben ein MSA  $M$  für Sequenzen  $S_1, \dots, S_k$ . Das *durch  $M$  induziertes Alignment für zwei Sequenzen  $S_i$  und  $S_j$*  ist das folgende:
  - Entferne aus  $M$  alle Zeilen außer  $i$  und  $j$
  - Entferne alle Spalten, die in  $i$  und  $j$  ein Leerzeichen enthalten
- Gegeben ein MSA  $M$  für Sequenzen  $S_1, \dots, S_k$ . Der *Sum-Of-Pairs Score für  $M$  (SP-Score)* ist die Summe aller Alignmentsscores der durch  $M$  induzierten paarweisen Alignments
- Das *SP-Alignment Problem für Sequenzen  $S_1, \dots, S_k$*  sucht das MSA  $M$  mit minimalem SP-Score

- Bemerkung

- Vergleich aller Sequenzen mit allen anderen Sequenzen – aber entsprechend dem vorgegebenen MSA

# Beispiel

d/i	=	1
r	=	1
m	=	0

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{AT\_AATG} \\ \text{CTG\_G\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 5 \} 14$$

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{\_ATAATG} \\ \text{C\_TGG\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 7 \} 16$$

- Die Berechnung des SP-Scores für ein gegebenes MSA über k Sequenzen ist einfach
  - Nämlich?

# Beispiel

d/i	=	1
r	=	1
m	=	0

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{AT\_AATG} \\ \text{CTG\_G\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 5 \} 14$$

$$\begin{array}{l} \text{AAGAA\_A} \\ \text{\_ATAATG} \\ \text{C\_TGG\_G} \end{array} \begin{array}{l} \rangle 4 \\ \rangle 5 \end{array} \rangle 7 \} 16$$

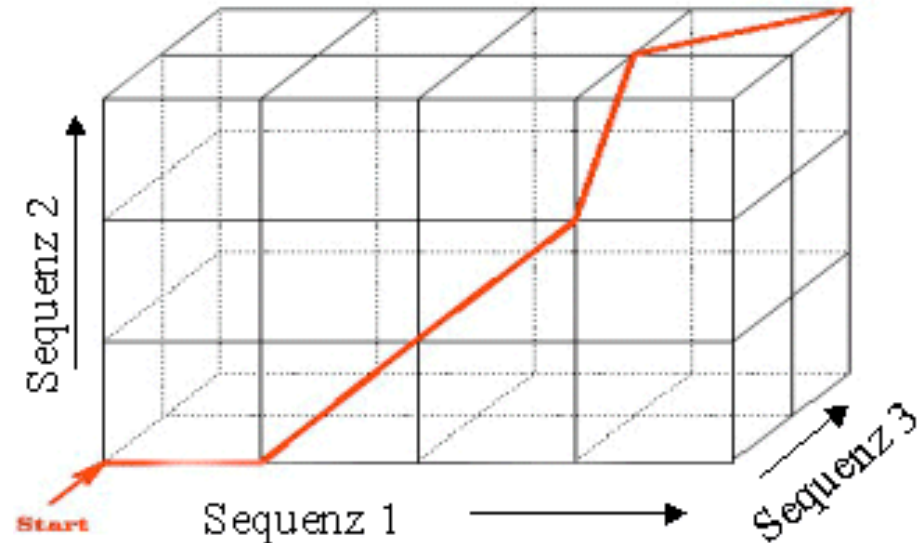
- Die Berechnung des SP-Scores für ein gegebenes MSA über  $k$  Sequenzen ist einfach
  - Komplexität  $O(k^2 * \max(|S_i|))$  (über alle  $i$ )
- Aber wie findet man das MSA mit **minimalem SP-Score**?

# Dynamische Programmierung in k Dimensionen

- $k = 2$ 
  - 2-dimensionale Matrix

		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	7
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4	3	4	5	6
5	n	5	5	5	5	4	4	5	6
6	e	6	6	6	6	5	4	5	6
7	r	7	7	6	7	6	5	4	5

- $k = 3$ 
  - 3-dimensionale Matrix



# Erinnerung

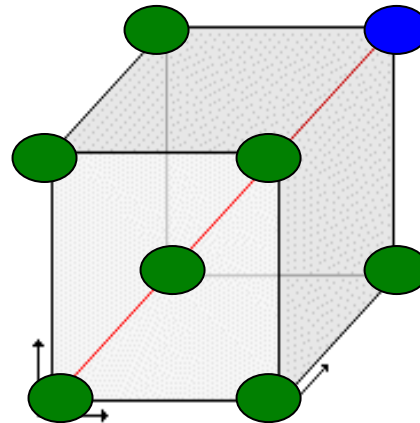
---

- Grundidee der dynamischen Programmierung für zwei Sequenzen  $S_1, S_2$ 
  - Berechnung des Alignment  $d(i,j)$  von  $S_1[1..i]$  und  $S_2[1..j]$  für steigende Werte  $(i, j)$  bis  $i=|S_1|$  und  $j=|S_2|$
  - Berechnung von  $d(i,j)$  aus  $d(i-1,j-1), d(i,j-1), d(i-1,j)$ 
    - Man verlängert  $d(i-1,j-1)$  um Match oder Mismatch
    - ... oder man verlängert  $d(i,j-1)$  um ein Insert
    - ... oder man verlängert  $d(i-1,j)$  um eine Deletion
  - Statische Initialisierung der Werte  $d(i,0)$  und  $d(0,j)$
- Wir betrachten im Folgenden nur den Fall  $k=3$

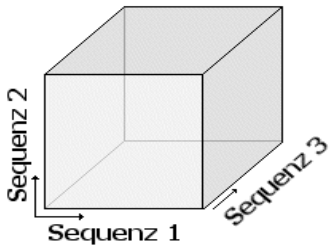
# Dyn Prog. für SP-MSA

---

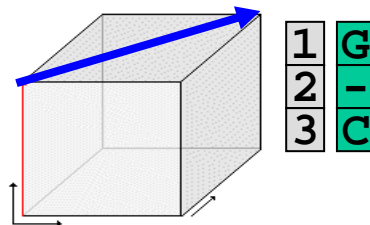
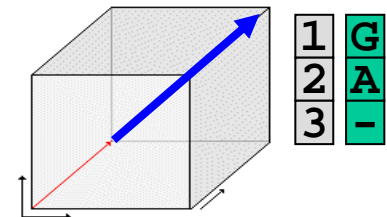
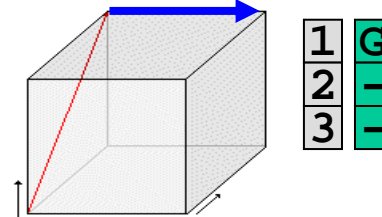
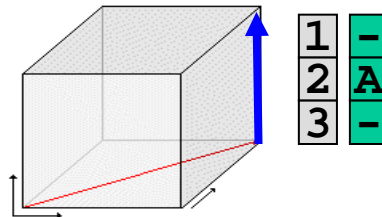
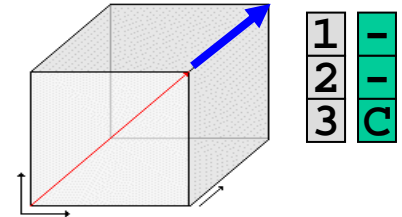
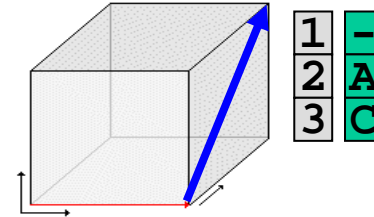
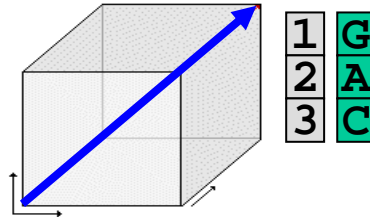
- Übertragung auf MSA
- Berechnung von  $d(i,j,k)$  aus
  - $d(i-1,j-1,k-1)$
  - $d(i,j-1,k-1)$
  - $d(i,j,k-1)$
  - $d(i,j-1,k)$
  - $d(i-1,j,k)$
  - $d(i-1,j-1,k)$
  - $d(i-1,j,k-1)$

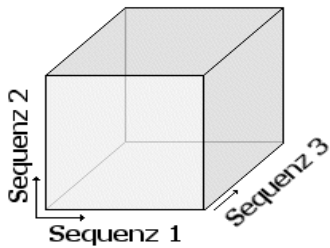


# Dyn Prog. für SP-MSA



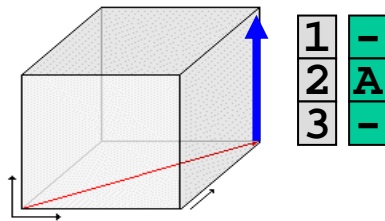
- $d(i-1, j-1, k-1)$
- $d(i, j-1, k-1)$
- $d(i, j, k-1)$
- $d(i, j-1, k)$
- $d(i-1, j, k)$
- $d(i-1, j-1, k)$
- $d(i-1, j, k-1)$





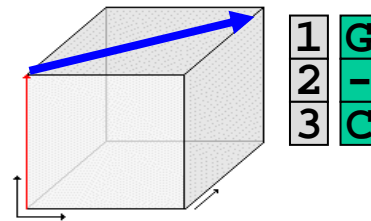
# Analogie

$d(i, j-1, k)$



- SP-Alignment von  $d(i, j-1, k)$  ist bekannt
- Wir erweitern dieses zu  $d(i, j, k)$
- Dazu alignieren wir  $S_2[j]$  zweimal mit Leerzeichen (Inserts)

$d(i-1, j, k-1)$



- SP-Alignment von  $d(i-1, j, k-1)$  ist bekannt
- Wir erweitern dieses zu  $d(i, j, k)$
- Dazu alignieren wir ein Leerzeichen mit  $S_1[i-1]$  und mit  $S_3[k-1]$

# Formaler

- Wir nehmen ein einfaches Kostenmodell (I/D/R=1, M=0)
- Theorem
  - Gegeben Sequenzen  $S_1, S_2, S_3$ .
    - Sei  $d(i,j,k)$  der Score des SP-optimalen Alignments der Strings  $S_1[1..i], S_2[1..j], S_3[1..k]$
    - Sei  $c_{ij} = 0$ , wenn  $S_1(i) = S_2(j)$ , sonst 1
    - Sei  $c_{ik} = 0$ , wenn  $S_1(i) = S_3(k)$ , sonst 1
    - Sei  $c_{jk} = 0$ , wenn  $S_2(j) = S_3(k)$ , sonst 1
  - Dann berechnet sich  $d(i,j,k)$  als

$$d(i, j, k) = \min \left\{ \begin{array}{llll} d(i-1, j-1, k-1) + c_{ij} & + c_{ik} & & + c_{jk} \\ d(i-1, j-1, k) & + c_{ij} & + 2 & \\ d(i-1, j, k-1) & + c_{ik} & + 2 & \\ d(i, j-1, k-1) & + c_{jk} & + 2 & \\ d(i-1, j, k) & & + 2 & \\ d(i, j-1, k) & & + 2 & \\ d(i, j, k-1) & & + 2 & \end{array} \right.$$

# Formaler 2

---

- Theorem Fortsetzung

- ...

- mit Initialisierung

- Sei  $d_{a,b}(i,j)$  der optimale Alignmentsscore von  $S_a[1..i]$  mit  $S_b[1..j]$
    - $D(0, 0, 0) = 0$
    - $D(i, j, 0) = D_{1,2}(i, j) + (i+j)$
    - $D(i, 0, k) = D_{1,3}(i, k) + (i+k)$
    - $D(0, j, k) = D_{2,3}(j, k) + (j+k)$

- Bemerkung

- Beweis analog zum paarweisen Alignment
  - Alignment eines Leerzeichen mit einem Leerzeichen ist im induzierten paarweisen Alignment nicht enthalten

# Algorithmus

---

```
initialize matrix d;
for i := 1 to |S1|
  for j := 1 to |S2|
    for k := 1 to |S3|
      if (S1(i) = S2(j)) then cij := 0; else cij := 1;
      if (S1(i) = S3(k)) then cik := 0; else cik := 1;
      if (S2(j) = S3(k)) then cjk := 0; else cjk := 1;

      d1 := d[i - 1, j - 1, k - 1] + cij + cik + cjk;
      d2 := d[i - 1, j - 1, k] + cij + 2;
      d3 := d[i - 1, j, k - 1] + cik + 2;
      d4 := d[i, j - 1, k - 1] + cjk + 2;
      d5 := d[i - 1, j, k) + 2;
      d6 := d[i, j - 1, k) + 2;
      d7 := d[i, j, k - 1) + 2;

      d[i, j, k] := min(d1, d2, d3, d4, d5, d6, d7);
    end for;
  end for;
end for;
```

# Komplexität

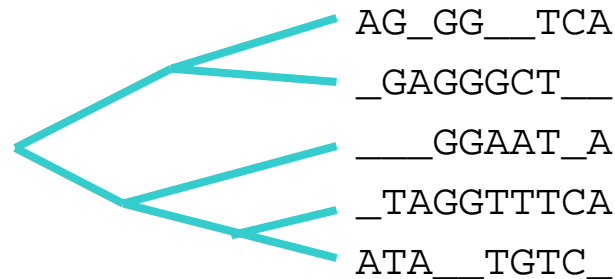
---

- ?
- Für drei Sequenzen der Länge  $n$ 
  - Würfel hat  $n^3$  Zellen
  - Für jede Zelle sind 7 Berechnungen notwendig
  - Zusammen  $O(7 * n^3)$
- Allgemeiner Fall:  $k$  Sequenzen der Länge  $n$ 
  - Hyperwürfel hat  $n^k$  Zellen
  - Für jede Zelle sind  $2^k - 1$  Berechnungen notwendig
    - Alle Ecken eines  $k$ -dimensionalen Würfels minus eins (Das ist die Ecke die gerade berechnet wird)
  - Zusammen  $O(2^k * n^k)$
- Tatsächlich gilt
  - *Das SP-Alignment Problem ist NP vollständig*

# MSA also praktisch unlösbar?

---

- SP-Score für mehr als eine Handvoll Sequenzen nennenswerter Länge nicht berechenbar
- Aber
  - SP berechnet **nicht die Menge an notwendiger Evolution**



- SP ist nur eine mögliche Zielfunktion
  - Center-Star, MSA entlang des phylogenetischen Baums
- Viele Heuristiken (Branch&Bound, iterative, lokal-Greedy, ...)

# Suche mit MSA

---

- Erinnerung: Erzeugung von Proteinfamilien
  - Starte mit Proteinen gleicher/ähnlicher Funktion
  - Finde das Gemeinsame durch MSA
  - Suche „damit“ nach weiteren Vertretern
  - Modifiziere Familie entsprechend
  - Iteriere, bis Zufriedenheit eintritt
- Wie sucht man mit einem MSA?
  - Wir müssen entscheiden, wie gut eine (neue) Sequenz  $S$  zu einem MSA  $M$  passt
  - Profiles