Algorithmen der Sequenzanalyse:

Sequenzalignments

AlgSeq – 09/12/2024

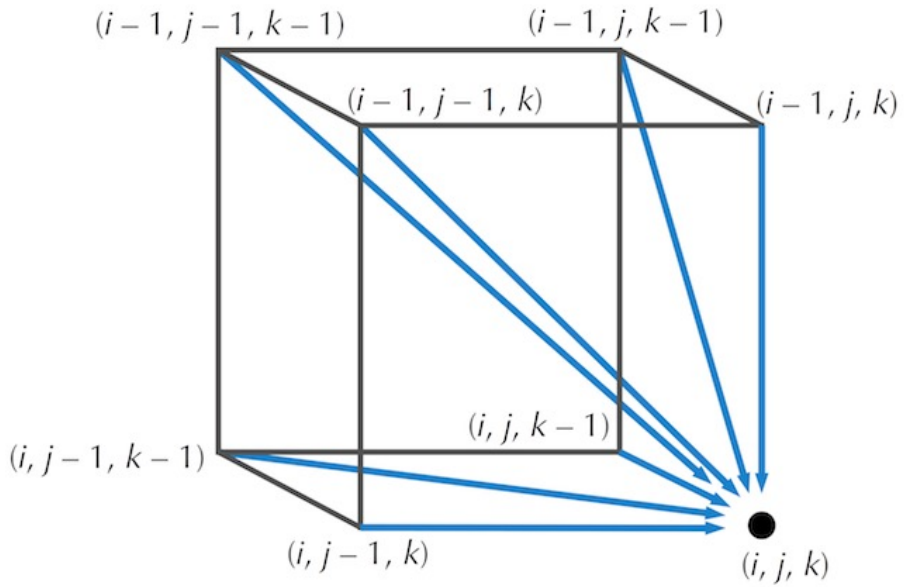
Prof. M. Sammeth

Multiples Sequenz-Aligment (MSA)

Multipel = mehr als 2 Sequenzen

Asp:
YAFDLGYTCMFPVLLGGGELHIVQKETYTAPDEIAHYIKEHGITYIKLTPSLFHTIVNTASFAFDANFESLRLIVLGGEKIIPIDVIAFRKMYGHTE-FINHYGPTEATIGA
Orn:
-AFDVSAGDFARALLTGGQLIVCPNEVKMDPASLYAIIKKYDITIFEATPALVIPLMEYI-YEQKLDISQLQILIVGSDSCSMEDFKTLVSRFGSTIRIVNSYGVTEACIDS
Val:
IAFDASSWEIYAPLLNGGTVVCIDYYTTIDIKALEAVFKQHHIRGAMLPPALLKQCLVSA----PTMISSLEILFAAGDRLSSQDAILARRAVGSGV-Y-NAYGPTENTVLS

Größe des Alignment Graphen



- 2 Sequenzen mit längen $m \sim n$:
  n^2
- 3 Sequenzen: n^3
- k Sequenzen: n^k

Bsp: Proteinfamilie mit 10 Mitgliedern, je 500 Aminosäuren.

➔ $500^{10} \sim 10^{27}$ Knoten (!)

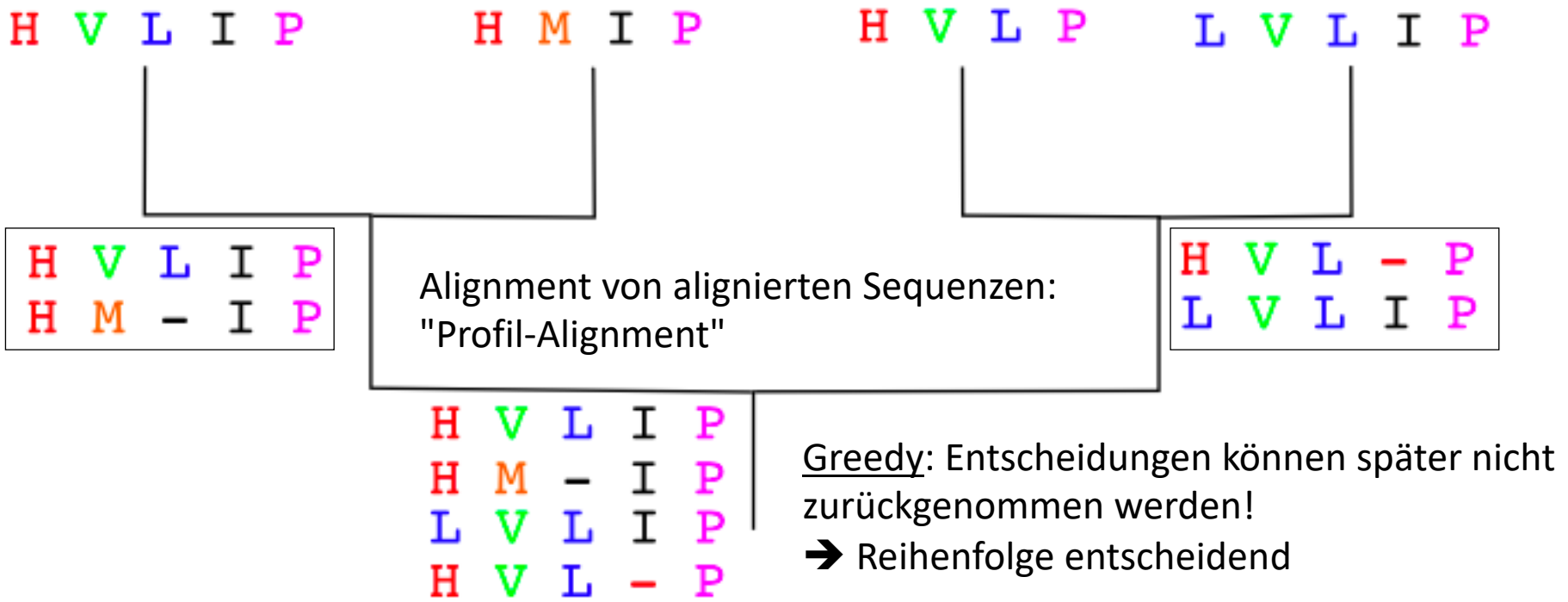
Heuristiken: Progressives Alignment

Heuristik
Abkürzung der Berechnung, effizienterer Algorithmus.

Heuristiken sind meist inexakt, es gibt jedoch auch exakte.

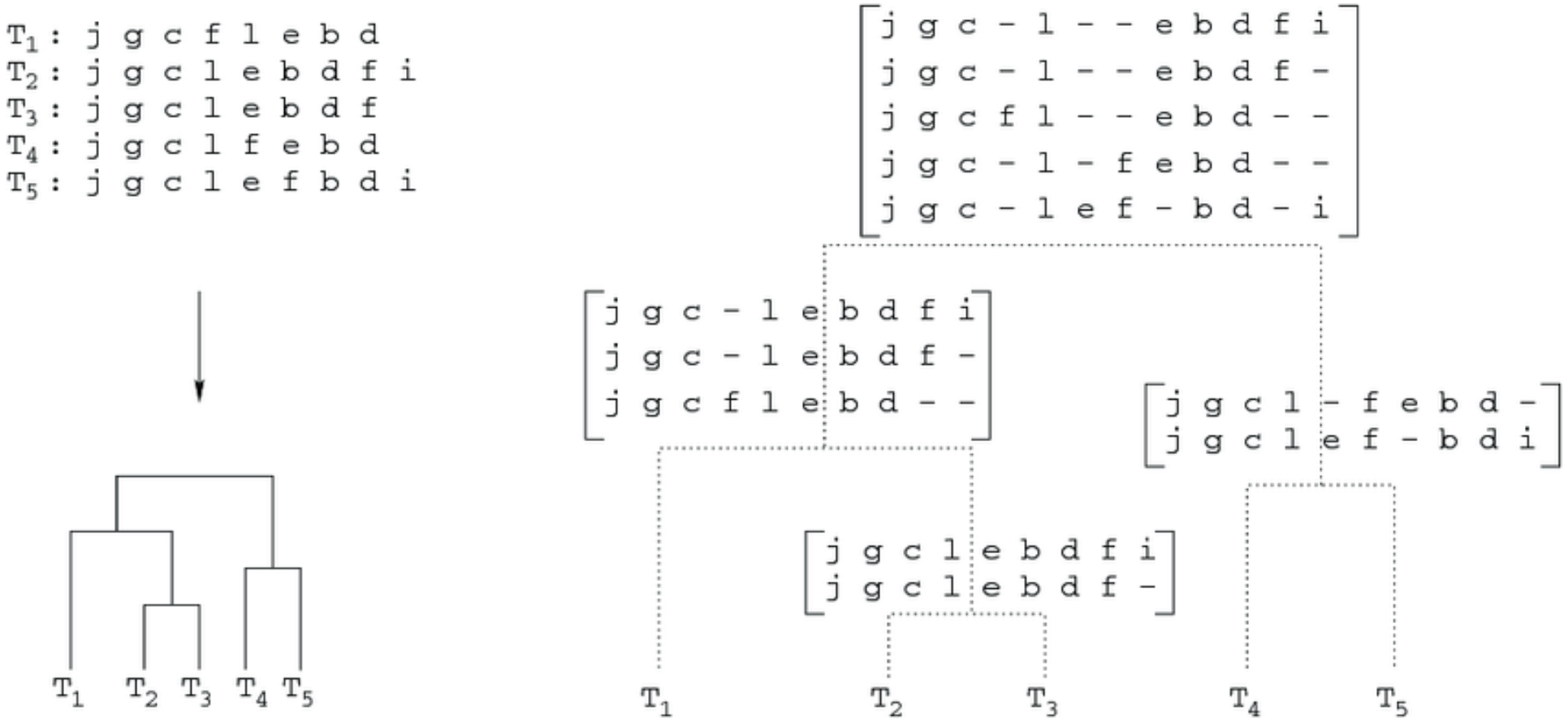
Progressives Alignment: CLUSTALV, CLUSTALW, CLUSTALX

Des Higgins



Greedy: Entscheidungen können später nicht zurückgenommen werden!
➔ Reihenfolge entscheidend

Inexakte Heuristiken: Progressives MSA



Reihenfolge nach Sequenzähnlichkeit (paarweises Alignment)

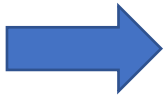
Alignment Profile und Profil Alignments

Alignment

T	**C**	**G**	**G**	**G**	-	g	**T**	**T**	**T**	t	t
c	**C**	-	-	t	**G**	**A**	c	**T**	**T**	a	**C**
a	**C**	**G**	-	**G**	**G**	**A**	**T**	**T**	**T**	t	**C**
T	t	**G**	**G**	**G**	-	**A**	c	**T**	**T**	t	t
a	-	-	-	**G**	-	-	-	**T**	-	**C**	-
T	t	**G**	**G**	**G**	**G**	**A**	c	**T**	**T**	**C**	**C**
T	**C**	**G**	-	-	**G**	**A**	**T**	**T**	c	a	t
-	-	-	**G**	**G**	**G**	**A**	**T**	**T**	c	**C**	-
T	a	**G**	**G**	**G**	**G**	**A**	a	c	-	-	**C**
T	**C**	**G**	**G**	**G**	t	**A**	**T**	a	a	**C**	**C**

Profil

A:	.2	.1	0	0	0	0	**.8**	.1	.1	.1	.2	0
C:	.1	**.5**	0	0	0	0	0	.3	.1	.2	**.4**	**.5**
G:	0	0	**.7**	**.6**	**.8**	.6	.1	0	0	0	0	0
T:	**.6**	.2	0	0	.1	.1	0	**.5**	**.8**	**.6**	.2	.3



z.B. $score(X, v_i) = \sum_{Y \in A}(Profil_i(Y) \frac{score(X,Y)}{|A|}) + (1 - \sum_{Y \in A}(Profil_i(Y))) \cdot score(X, -)$

ClustalW

Valid format for input is: FASTA(Pearson)
max number of sequences = 30
max total length of sequences = 10000

Help page

More information on Clustal home page

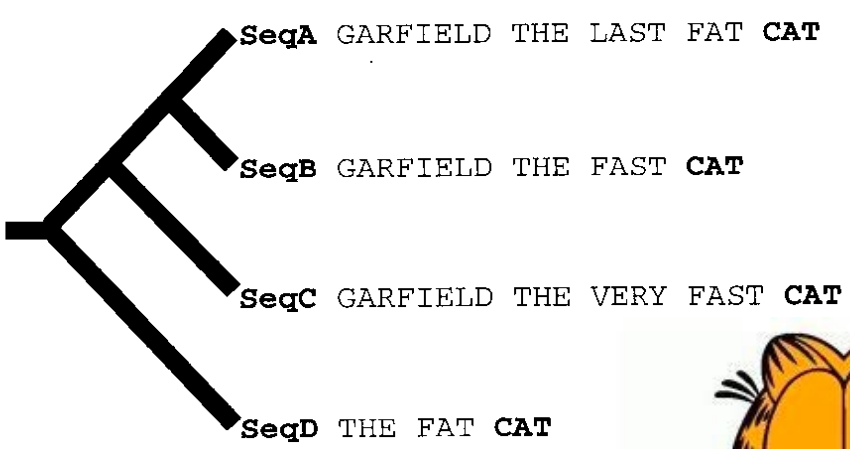
Scoring matrix :	Blosum		
Opening gap penalty :	10	Extending gap penalty :	0.05
End gap penalty :	10	Separation gap penalty :	0.05
Output format :	Clustal	Output order :	Input

Input sequences:
(see above for valid formats)

Run ClustalW | Clear Input

Probleme mit dem Progressiven Alignment

“Once a gap, always a gap”




```
GARFIELD THE LAST FA-T CAT
GARFIELD THE FAST CA-T ---
```

```
GARFIELD THE LAST FA-T CAT
GARFIELD THE FAST CA-T ---
GARFIELD THE VERY FAST CAT
-------- THE ---- FA-T CAT
```

Konsistenz-basiertes Alignment

(1) Vorbereitung: berechne alle paarweisen Alignments.

```
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT ---        Prim. Weight = 88

SeqA GARFIELD THE LAST FA-T CAT
SeqC GARFIELD THE VERY FAST CAT       Prim. Weight = 77

SeqA GARFIELD THE LAST FAT CAT
SeqD ------- THE ---- FAT CAT         Prim. Weight =100

SeqB GARFIELD THE ---- FAST CAT
SeqC GARFIELD THE VERY FAST CAT       Prim Weight = 100

SeqB GARFIELD THE FAST CAT
SeqD ------- THE FA-T CAT             Prim. Weight = 100

SeqC GARFIELD THE VERY FAST CAT
SeqD ------- THE ---- FA-T CAT        Prim. Weight = 100
```

(2) Gewichtung: wie oft wurd jedes Positionspaar in den paarweisen Vergleichen aligniert?

```
SeqA GARFIELD THE LAST FAT CAT
     ||||||||  ||| |||| |||
SeqB GARFIELD THE FAST CAT          Weight = 88

SeqA GARFIELD THE LAST FAT CAT
     ||||||||  ||| |||| || \ \\\
SeqC GARFIELD THE VERY FAST CAT
     ||||||||  |||       |||| ||
SeqB GARFIELD THE       FAST CAT    Weight = 77

Seq1 GARFIELD THE LAST FAT CAT
             |||       ||| |||
SeqD         THE       FAT CAT
             |||       || \ \\\
SeqB GARFIELD THE       FAST CAT    Weight = 100
```

"Konsistenzen"

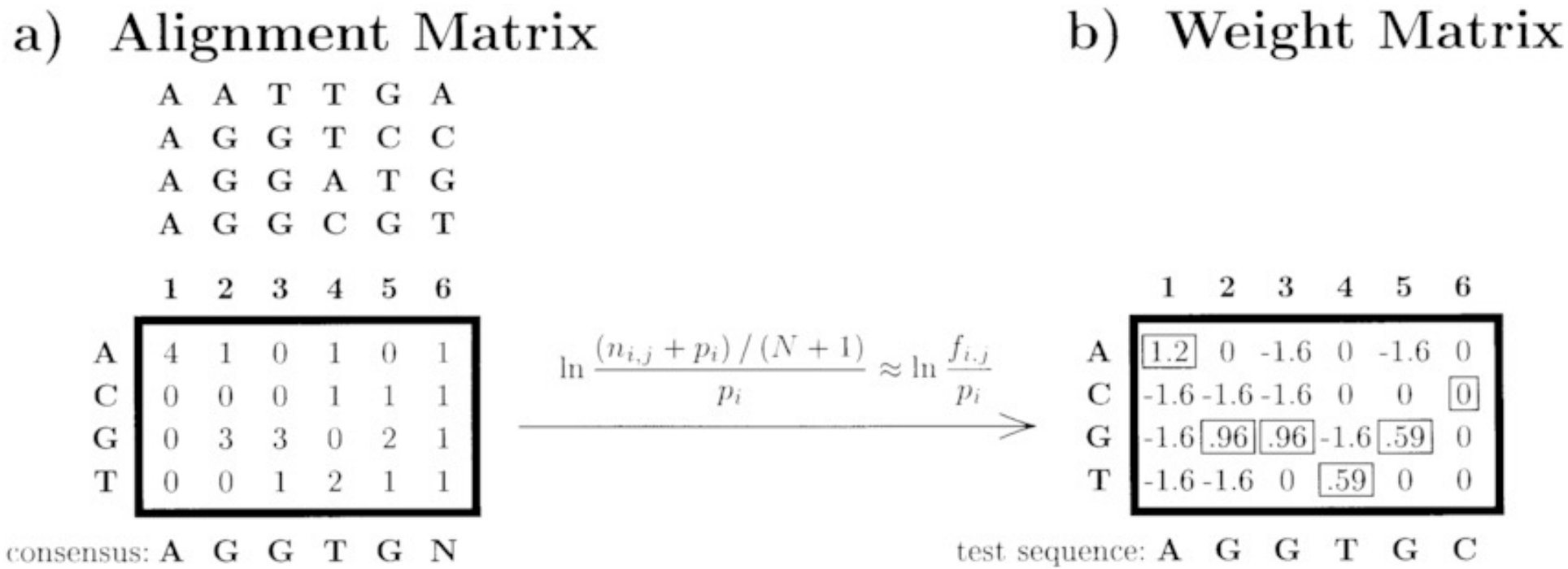
```
SeqA GARFIELD THE LAST FAT  CAT
     |||||||| ||| \\\ ||| |||
SeqB GARFIELD THE      FAST CAT
```

(3) Dynamische Programmierung mit PSSM !

```
GARFIELD THE LAST FA-T CAT
GARFIELD THE ---- FAST CAT
```

(4) Progressives Multiples Alignment

PSSM – Position-Specific Scoring Matrices



T-Coffee

(Tree-based **Consistency Objective Function** for alignment Evaluation)

Cedrik Notredame

"Konsistenz"-Scores können beliebige a priori bekannte Informationen in ein Progressives Alignment einfliessen lassen.

Der COFFEE – Mixer:

M-Coffee – kombiniert unterschiedliche multiple alignments

R-Coffee – RNA Sekundärstruktur Information

3D-Coffee – Protein Strukturinformation

...

T-Coffee

Aligns DNA, RNA or Proteins using the default T-Coffee

Sequences input

Paste or upload your set of sequences in FASTA format

Sequences to align

Click here to use the sample file

- OR - Click here to upload a file

Hide advanced options

Methods

T-Coffee produces an alignment by combining the output of several alignment methods. Use this section to select the individual methods.

Pairwise Structual Methods

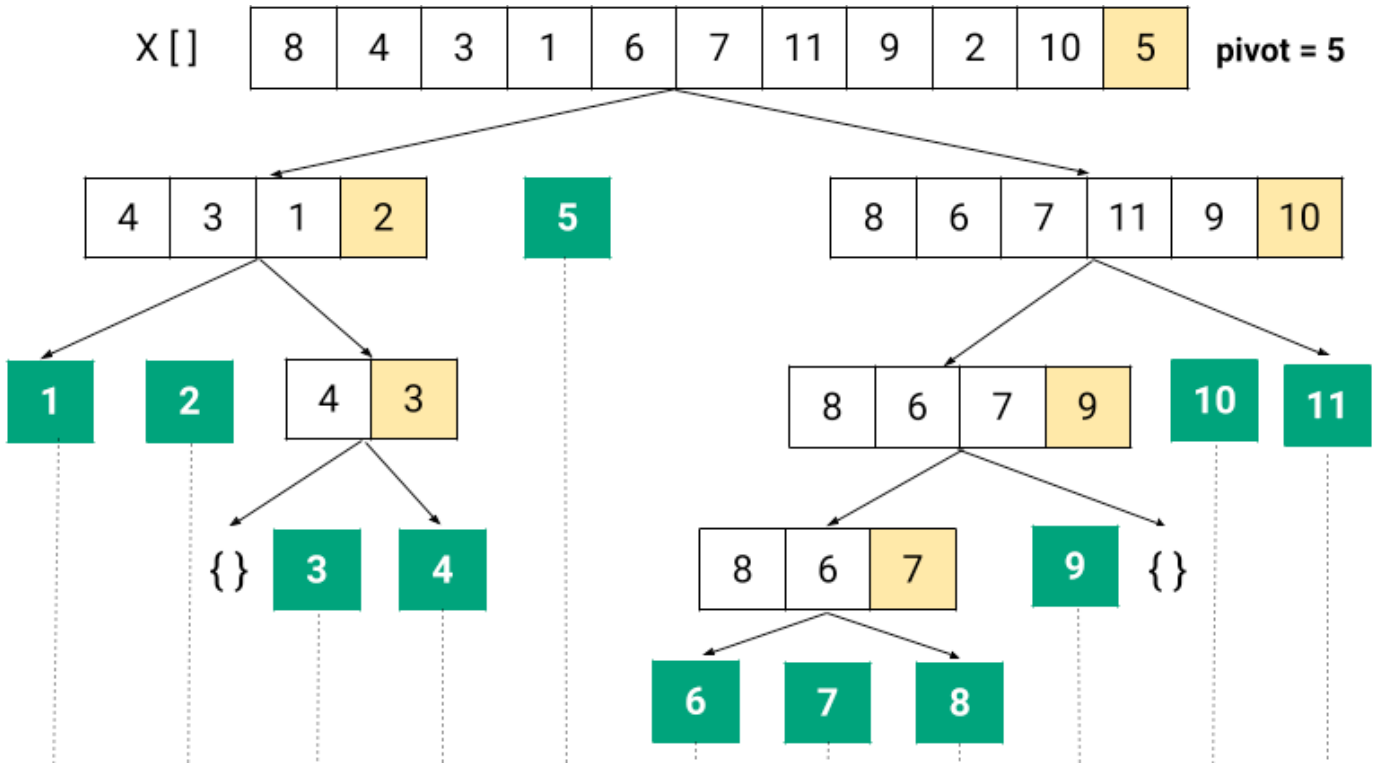
- [ ] sap_pair
- [ ] TMalign_pair
- [ ] mustang_pair

Multiple Methods

- [ ] pcma_msa
- [ ] mafft_msa
- [ ] clustalw_msa
- [ ] dialigntx_msa
- [ ] poa_msa
- [ ] muscle_msa
- [ ] probcons_msa
- [ ] t_coffee_msa
- [ ] amap_msa
- [ ] kalign_msa
- [ ] fsa_msa
- [ ] probconsRNA_msa
- [ ] mus4_msa

Exakte Heuristiken: Divide-and-Conquer MSA

z.B.: Quicksort, Mergesort, binäre Suche...



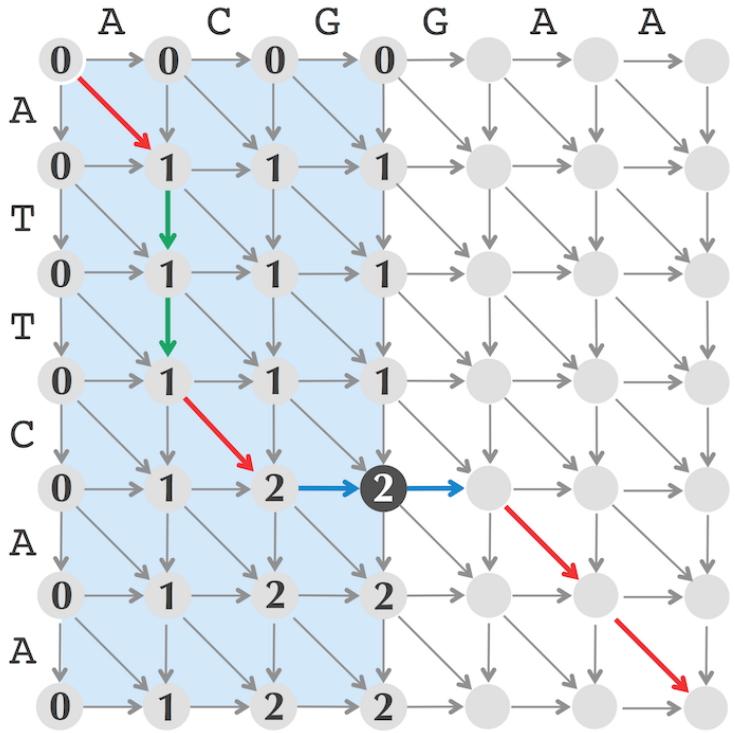
Divide-and-Conquer Ansätze:

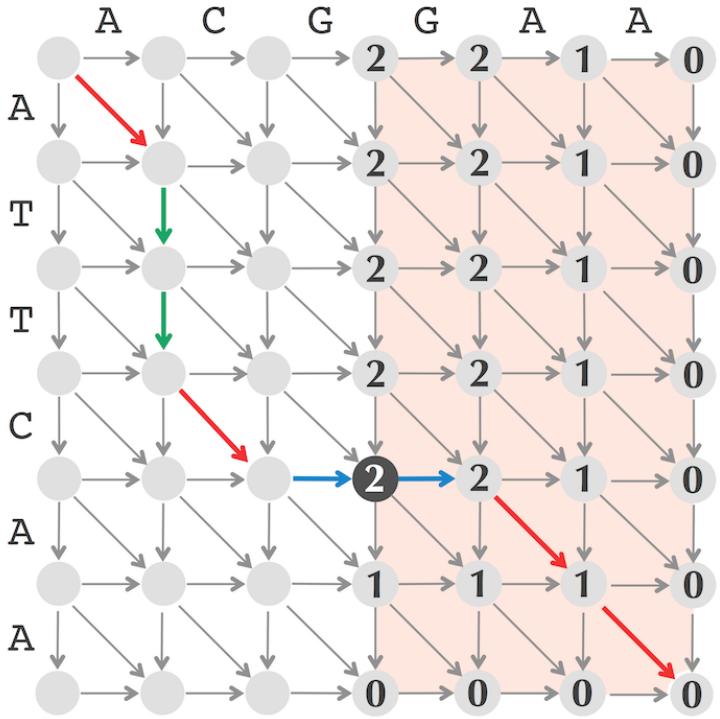
- zerlegen (rekursiv) den Suchraum in kleinere Sub-Suchräume (*Divide*)
- finden für jeden (meist trivialen) Sub-Sub-Suchraum eine Lösung (*Conquer*)
- setzen die Lösung zur Gesamtlösung zusammen (*Merge*)

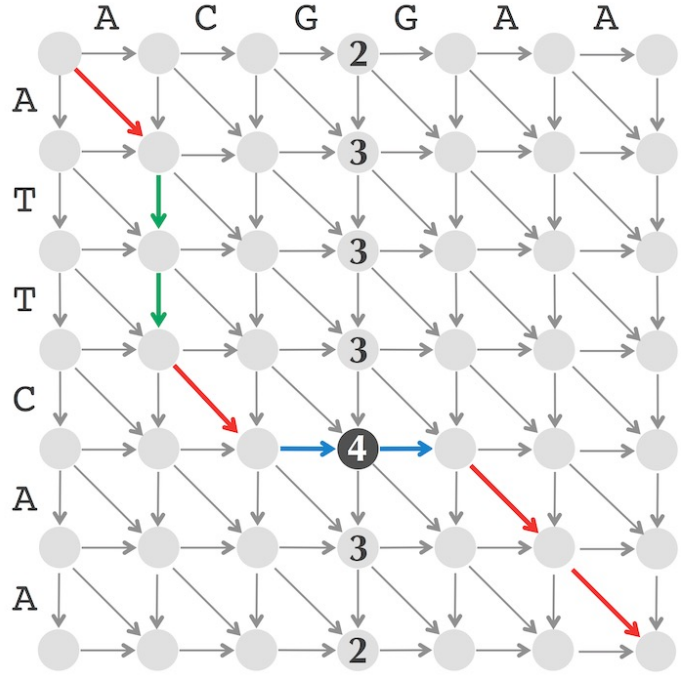
Divide-and-Conquer für Paarweise Alignments

Optimales *Präfix-*, *Suffix-* und *Totales* Alignment Score
Bsp.: LCS-Scoring (Match= +1)

optimaels Präfix-Alignment Score (*i,j*) + optimales Suffix-Alignment Score (*i,j*) = Score des optimalen Alignments, d.h. längster Pfad durch (*i,j*)



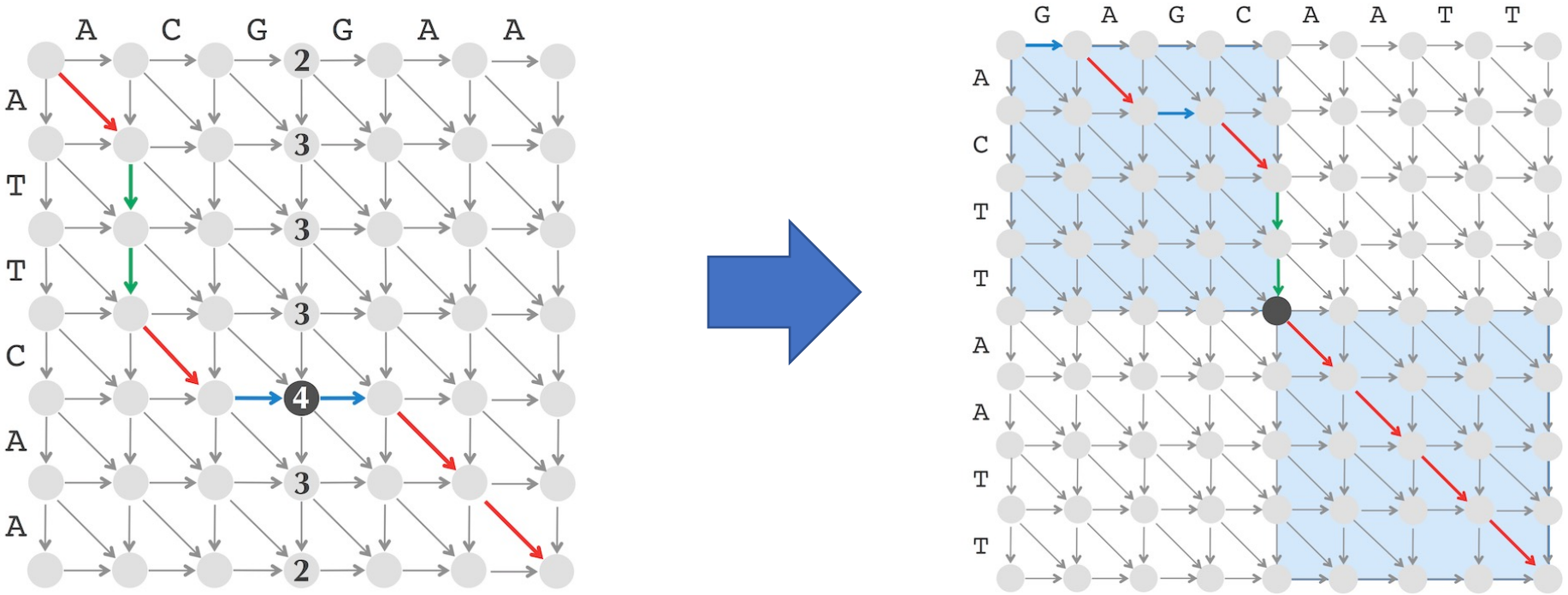




Pfadlänge(*i,j*) = *vonSource*(*i,j*) + *zurSink*(*i,j*)

Der “Mittlere” Knoten

Mittlere Spalte := Spalte, die die Dynamische Programmier-Matrix halbiert
Mittlerer Knoten := (ein) Knoten in der mittleren Spalte des Alignments mit maximalem Score.



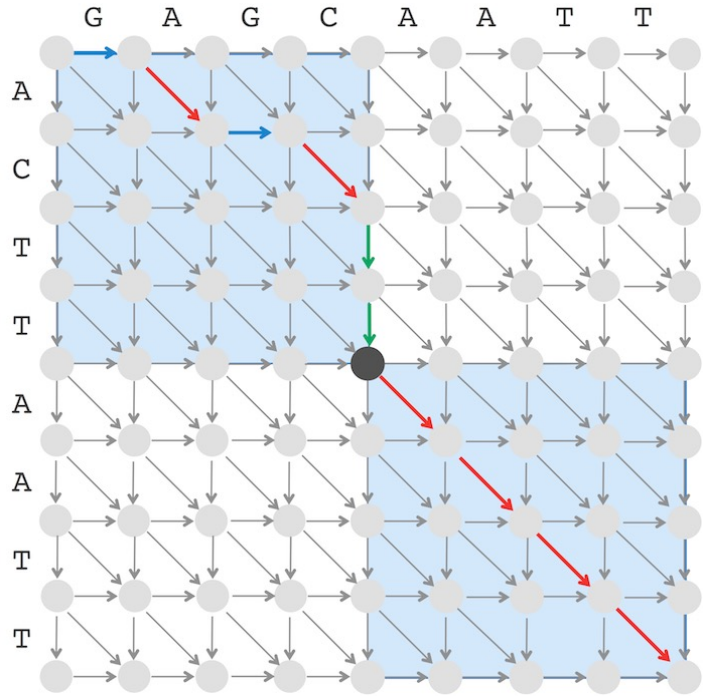
Ein Mittlerer Knoten (*i,j*) teilt die DP-Matrix in die zwei Submatrizen [0;*i*] x [0;j] und [*i*+1;*n*] x [*j*+1;*m*]

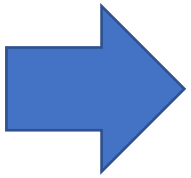
Divide-and-Conquer mit mittleren Knoten

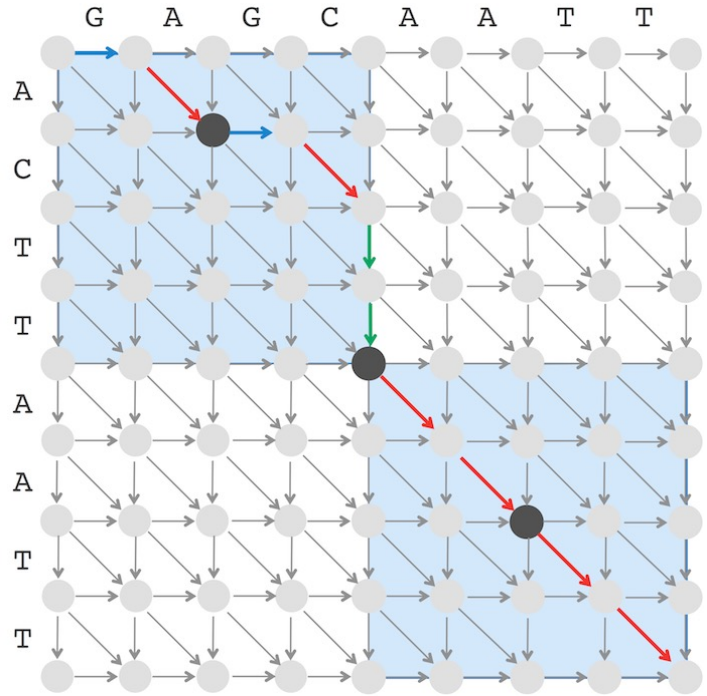
Idee:

- Der mittlere Knoten teilt das Problem in zweil +/- gleich große Subprobleme (*Divide*).
- Wenn links/oben und rechts/unten vom *mittleren Knoten* alle Knoten des längsten Pfades bekannt sind (*Conquer*),
- dann kann die Lösung einfach zusammengesetzt werden (*Merge*) als:

  Knoten des längsten Pfades links/oben + mittlerer Knoten + Knotendes längsten Pfades rechts/unten





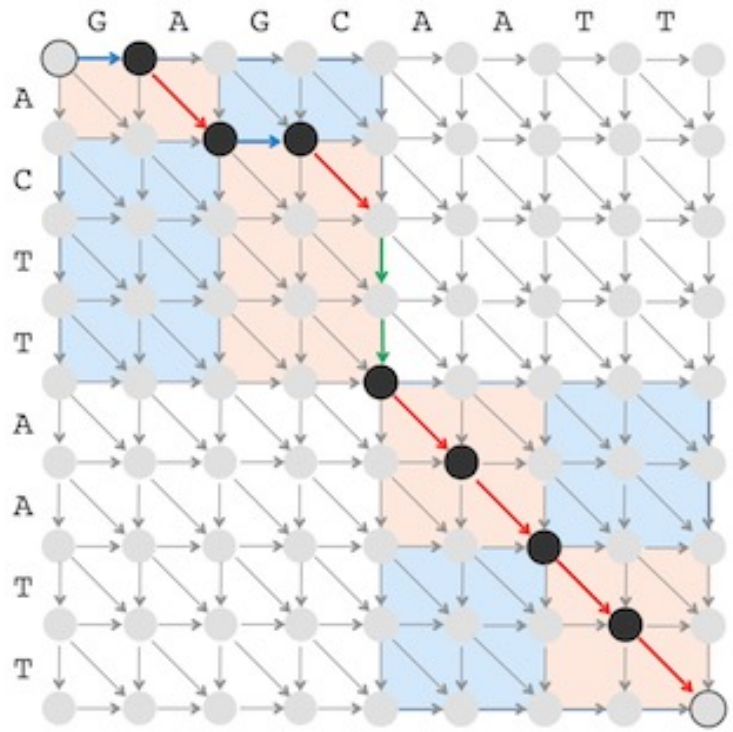


erster mittlerer Knoten: teilt DP Matrix in zwei Submatrizen

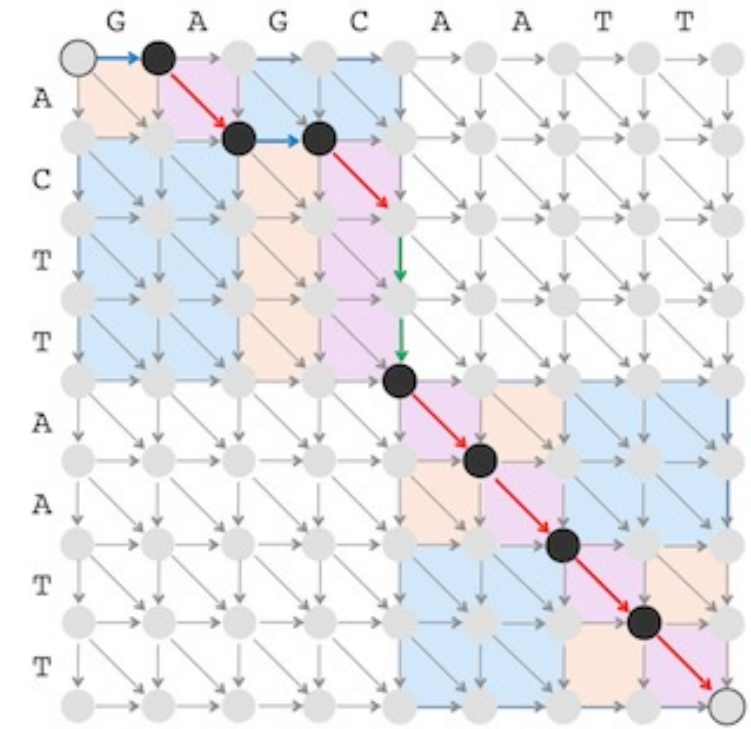
zweite mittlere Knoten, je einer pro Submatrix

Paarweises Divide-and-Conquer Alignment

Rekursive Fortsetzung führt zu



dritte mittlere Knoten

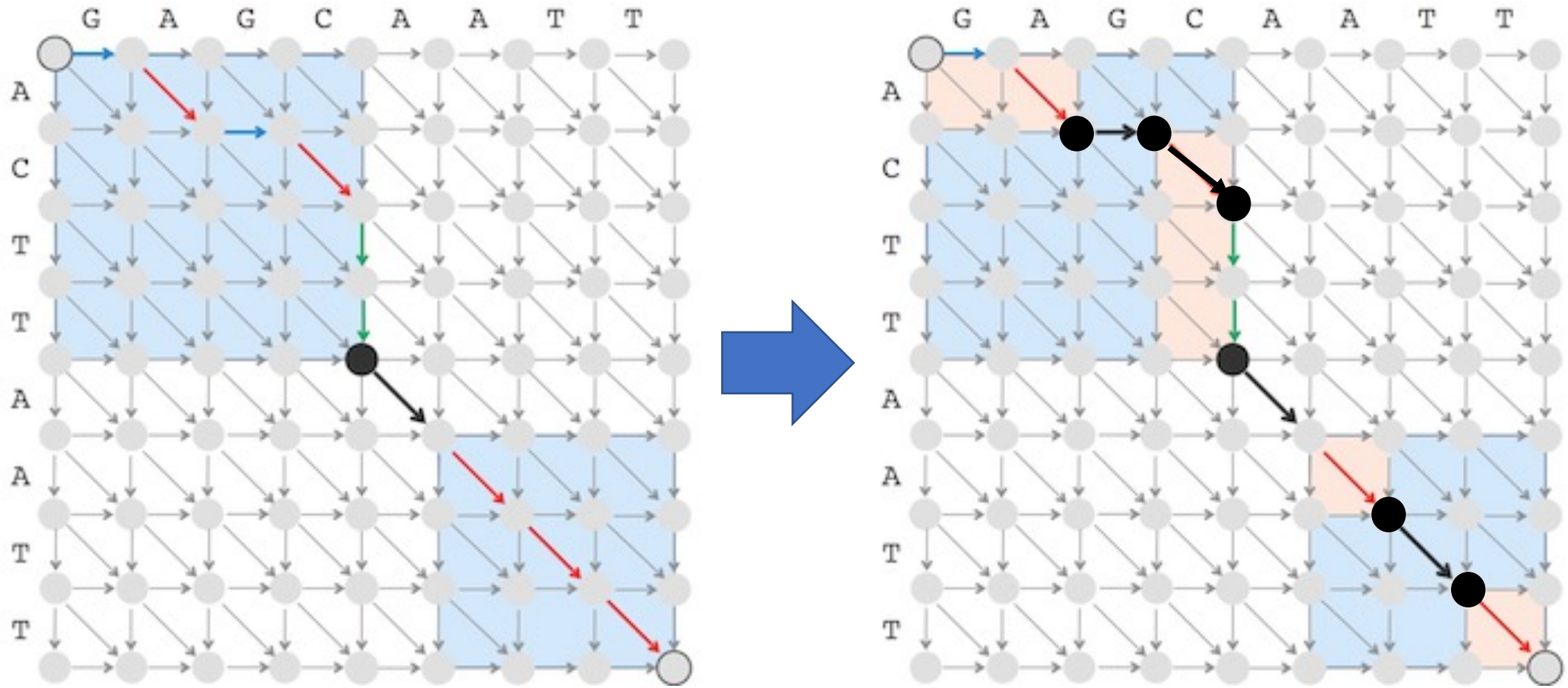


(fast) alle Knoten des längsten Pfades bekannt

paarweises Alignment in $\boldsymbol{O}(n)$ Speicher möglich!

Mittlere Kanten statt Mittlere Knoten

Mittlere Kante := ausgehende Kante des/eines mittleren Knotens, die Teil des/eines optimalen Alignments ist.



Rekursive Teilung über “mittlere Kanten” rekonstruiert intrinsisch den optimalen Alignement Pfad.

Algorithmus Outline:

Algorithm: DCAlignment($oben$, $unten$, $links$, $rechts$)

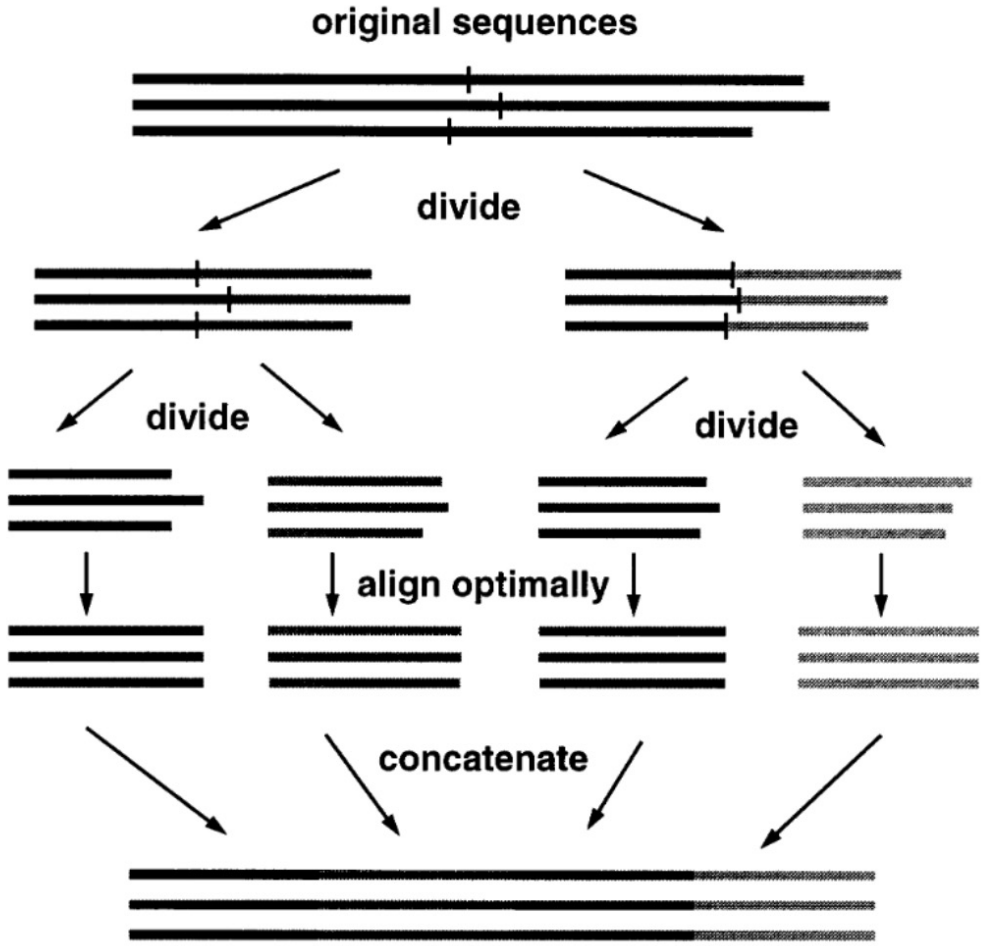
if $links = rechts$ **then**
  return Alignment aus $(unten - oben)$ "↓" Kanten
if $oben = unten$ **then**
  return Alignment aus $(rechts - links)$ "→" Kanten
$mitllereSpalte = \lfloor (links + rechts)/2 \rfloor$
$mitllereZeile =$ MittlererKnoten($oben$, $unten$, $links$, $rechts$)
$mitllereKante =$ MittlereKante($oben$, $unten$, $links$, $rechts$)
output DCAlignment($oben$, $mittlereZeile$, $links$, $mittlereSpalte$)
output $mittlereKante$
if $mittlereKante =$ "→" or $mittlereKante =$ "↘" **then**
  $mittlereSpalte = mittlereSpalte + 1$
if $mittlereKante =$ "↓" or $mittlereKante =$ "↘" **then**
  $mittlereZeile = mittlereZeile + 1$
output DCAlignment($mittlereZeile$, $unten$, $mittlereSpalte$, $rechts$)

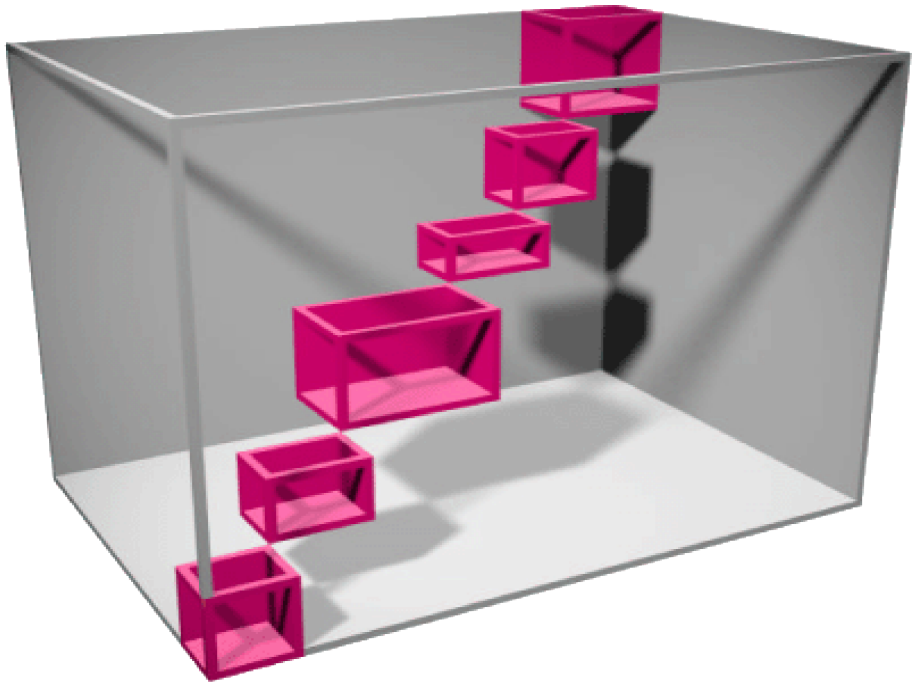
MittlererKnoten(...): gibt Koordinate *i* des mittleren Knotens der Substrings v_{oben+1} ... v_{unten} und $w_{links+1}$...w_{rechts} zurück

MittlereKante(...): gibt die Richtung der ausgehenden Kante des mittleren Knotens (*i*, *mittlereSpalte*) zurück

Divide-and-Conquer für MSA

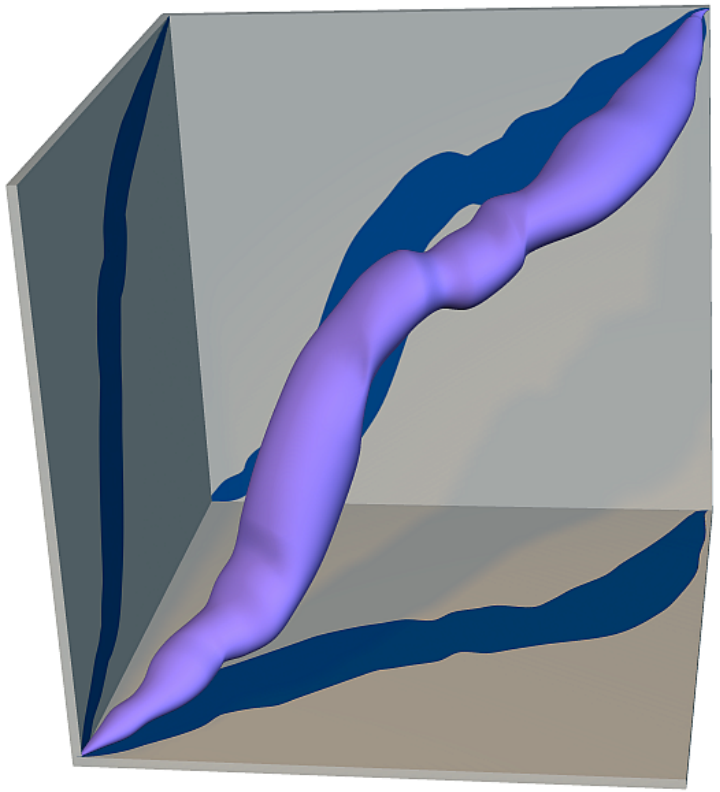
Jens Stoye





Stoye 1998, *Multiple sequence alignment with the divide-and-conquer method.* https://doi.org/10.1016/S0378-1119(98)00097-3 Bielefeld

Exake MSA Heuristiken: Branch-and-Bound



Branch-and-Bound Optimierung:

- exploriert systematisch den Ergebnisraum (*branching* step)
- berücksichtigt aber suboptimale Lösungen fühzeitig nicht weiter (*bounding step*)
- benötigt Annahme über das Score des optimalen Alignments und monotone Scoring Funktion.
- im *worst case* muss der ganze Datenraum exploriert werden.