Multiple alignments

Plan

(1) Today: What is a multiple alignment of a set of sequences, how to compute an optimal multiple alignment using SP-score (and similar column-based score function) by dynamic programming.

(2) Today and next week: How to approximate an optimal SP-score multiple alignment efficiently

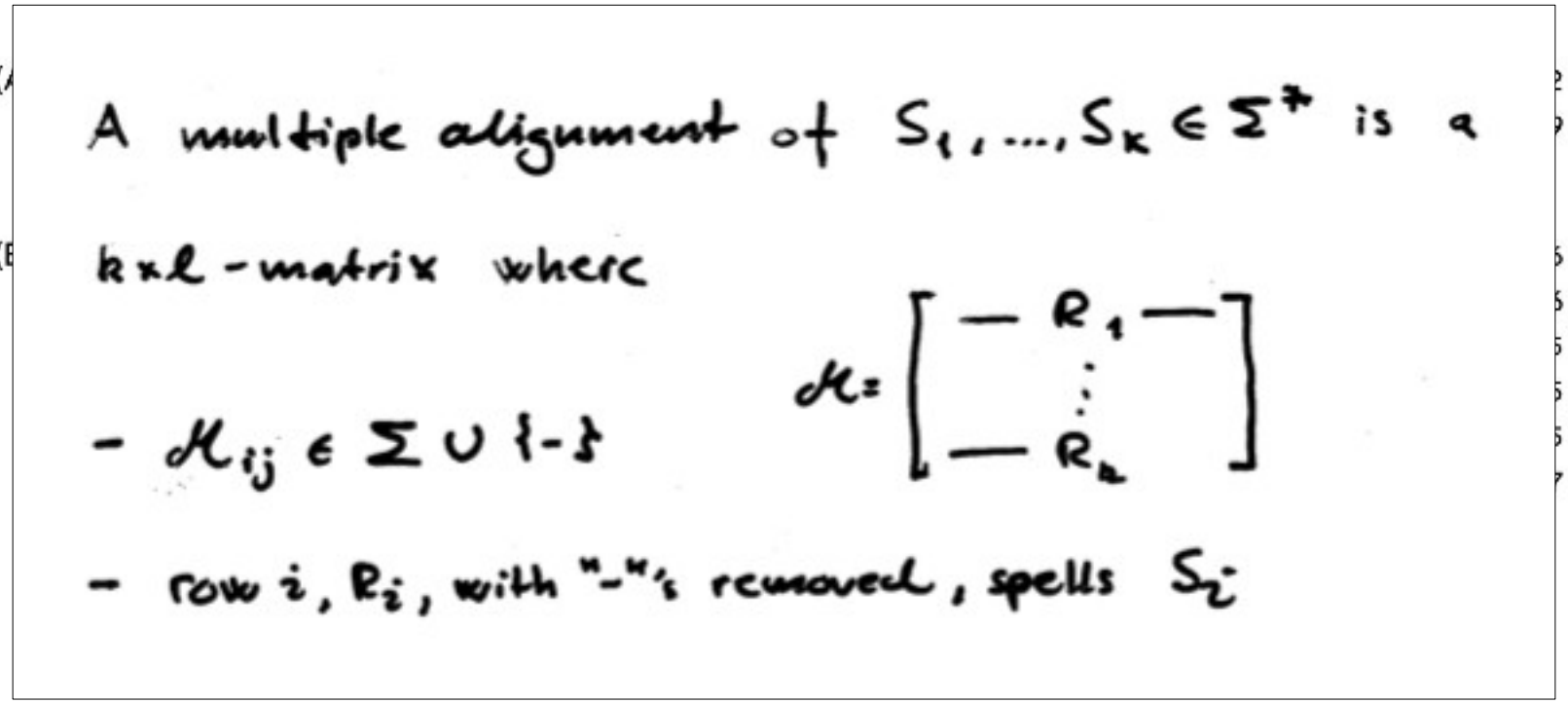
(3) In a couple of weeks: Speeding up the implementation of MSA using forward dynamic programming.

Multiple Alignment

A generalization of pairwise alignment, comparison of multiple sequence which makes it possible to identify weaker similarities

Multiple Alignment

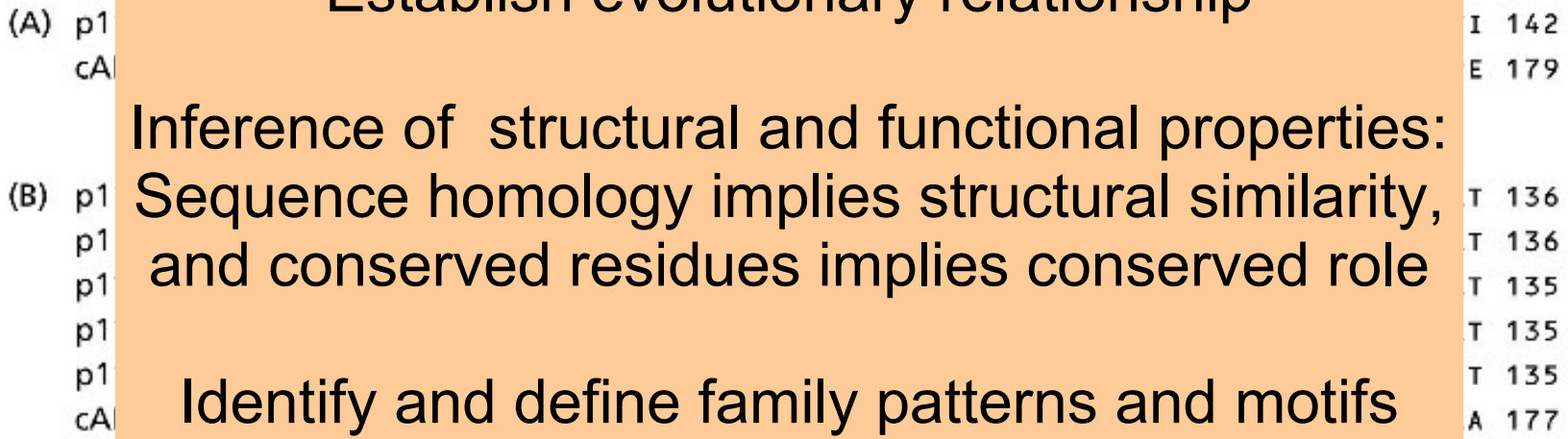
A generalization of pairwise alignment, comparison of multiple sequence which makes it possible to identify weaker similarities



The starting point of many bioinformatics workflows

Multiple Alignment

A generalization of pairwise alignment, comparison of multiple sequence which makes it possible to identify weaker similarities



Establish evolutionary relationship

Inference of structural and functional properties: Sequence homology implies structural similarity, and conserved residues implies conserved role

Identify and define family patterns and motifs

Sequence assembly

The starting point of many bioinformatics workflows

Multiple Alignment

A generalization of pairwise alignment, comparison of multiple sequence which makes it possible to identify weaker similarities

Establish evolutionary relationship

(A) p1 ... I 142
cA ... E 179

To computationally find an optimal multiple alignment, we must:

Define the cost of an alignment

Define an optimal alignment

Construct an efficient algorithm

Sequence assembly

The starti

$$\mathcal{M}^* = \underset{\mathcal{M}}{\operatorname{argmax}} \; \text{SCORE}(\mathcal{M})$$

workflows

The cost of an multiple alignment

Idea: Define cost such that ideas from pairwise alignment can be reused

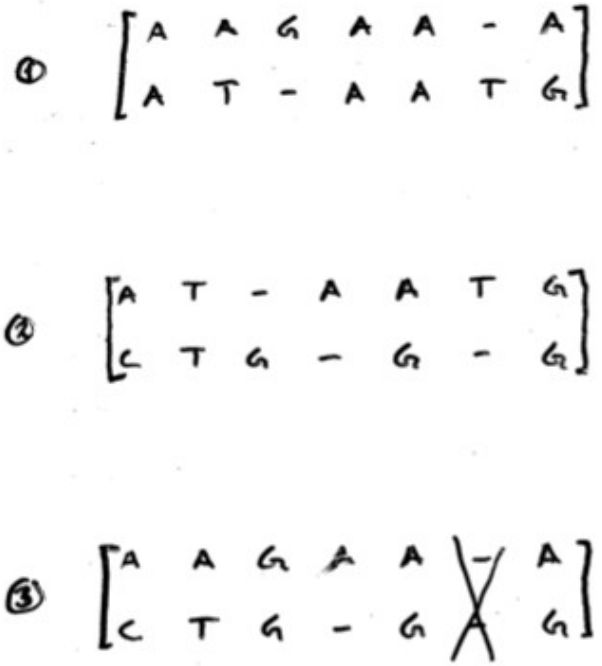
The cost of an multiple alignment

Idea: Define cost such that ideas from pairwise alignment can be reused

Induced pairwise alignments

Sum-of-pairs score

The SP score of an multiple alignment is the sum of the cost of the induced pairwise alignments.



Sum-of-pairs score

The SP-score $SP(\mathcal{M})$ is the sum of the scores of all induced pairwise alignments ...

$$\mathcal{M} = \begin{bmatrix} — R_1 — \\ \vdots \\ — R_k — \end{bmatrix} \qquad SP(\mathcal{M}) = \sum_{1 \le i < j \le k} \text{PAIRSCORE}(R_i, R_j)$$

Is SP-score column based, i.e. is the cost of an multiple alignment the sum of the cost of each column?

If PAIRSCORE is column-based (e.g. global alignment with linear gap cost) then ...

$$SP(\mathcal{M}) = \sum_{1 \le i < j \le k} \text{PAIRSCORE}(R_i, R_j)$$

$$= \sum_{1 \le i < j \le k} \sum_{1 \le c \le l} d(R_i[c], R_j[c])$$

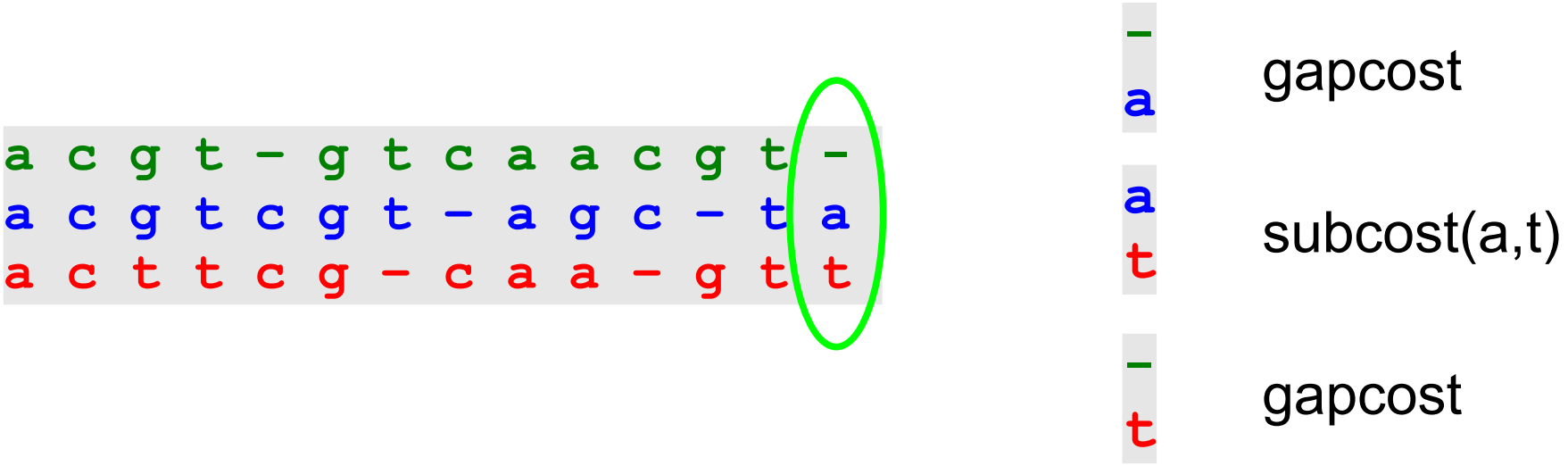
$$= \sum_{1 \le c \le l} \sum_{1 \le i < j \le k} d(\mathcal{M}_{ic}, \mathcal{M}_{jc})$$

$$= \sum_{1 \le c \le l} SP\left(\begin{bmatrix} \mathcal{M}_{1c} \\ \vdots \\ \mathcal{M}_{kc} \end{bmatrix}\right)$$

↑ column c in $\mathcal{M}$

... SP-score is column-based

Sum-of-pairs score



Cost of alignment = "sum of the cost of each column"

Problem: Given a set of sequences $S_1, S_2, S_3, \ldots$, a score matrix and a gap cost, find a multiple alignment of optimal sum-of-pairs cost

Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all possible last columns of such an alignment.

$$\begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \begin{bmatrix} - \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \ldots, \begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \ldots$$

What is number of possible last columns?

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} \left[D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \right]$$

Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all possible last columns of such an alignment.

$$\begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \begin{bmatrix} - \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \ldots, \begin{bmatrix} S_1[i_1] \\ S_2[i_2] \\ \vdots \\ S_k[i_k] \end{bmatrix}, \ldots$$

※ possible last.columns $= 2^k - 1$

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} \left[D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \right]$$

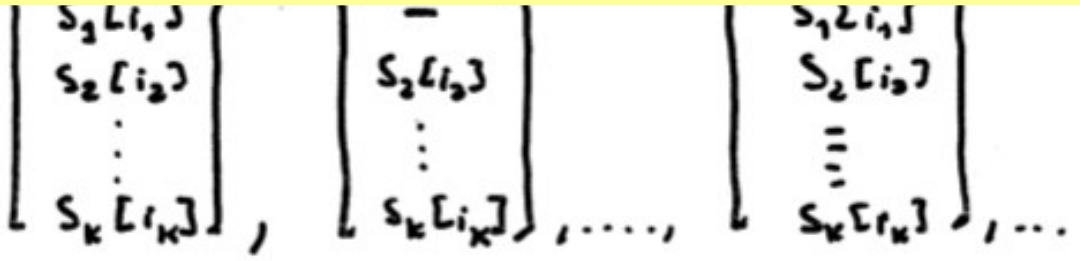
Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all

Time:

Space:



※ possible last. columns $= 2^k - 1$

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} \left[D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \right]$$

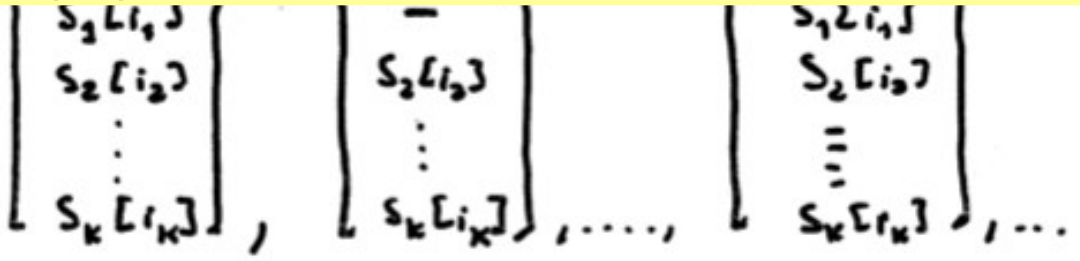
Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all

Time: $O(n^k \cdot 2^k \cdot$ "time to compute cost of last column")

Space: $O(n^k)$



* possible last. columns $= 2^k - 1$

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} [\, D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \,]$$

Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all

Time: $O(n^k \cdot 2^k \cdot$ "time to compute cost of last column")

Space: $O(n^k)$

$S_1[i_1]$ $S_2[i_2]$ | $-$ $S_2[i_2]$ | $S_1[i_1]$ $S_2[i_2]$

Comment: An optimal alignment can be found by backtracking (in time?), and space can be reduced by a factor *n* using "Hirschberg's trick"

* possible last.columns = $2^k - 1$

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} [\, D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \,]$$

Computing an optimal MSA using a column based score

* Let $D(i_1, \ldots, i_k)$ be the cost of an optimal mult. align of $S_1[1..i_1], \ldots, S_k[1..i_k]$

* Compute $D(i_1, \ldots, i_k)$ by optimizing over all

Time: $O(n^k \cdot 2^k \cdot$ "time to compute cost of last column")

Space: $O(n^k)$

Is it useful?

Aligning 10 sequences of length 100 takes:

100^{10} op / 100.000.000 op/sec = 32.000 years !!!

A huge need for fast and useful alternatives.

$$D(i_1, \ldots, i_k) = \min_{\text{last.col}} [\, D(i'_1, \ldots, i'_k) + \text{cost}(\text{last.col}) \,]$$

Recursive formulation for 3 seqs

Input: Three sequences A[1..n], B[1..n'], C[1..n'']

Recursion:

D(0,0,0) = 0

$$
D(i,j,k) = \min \begin{cases}
D(i-1, j-1, k-1) + SP(A[i], B[j], C[k]) & \text{if } i>0,\ j>0,\ k>0, \\
D(i-1, j-1, k) + SP(A[i], B[j], -) & \text{if } i>0,\ j>0,\ k\geq 0, \\
D(i-1, j, k-1) + SP(A[i], -, C[k]) & \text{if } i>0,\ j\geq 0,\ k>0, \\
D(i, j-1, k-1) + SP(-, B[j], C[k]) & \text{if } i\geq 0,\ j>0,\ k>0. \\
D(i-1, j, k) + SP(A[i], -, -) & \text{if } i>0,\ j\geq 0,\ k\geq 0, \\
D(i, j-1, k) + SP(-, B[j], -) & \text{if } i\geq 0,\ j>0,\ k\geq 0, \\
D(i, j, k-1) + SP(-, -, C[k]) & \text{if } i\geq 0,\ j\geq 0,\ k>0
\end{cases}
$$

D(n, n', n'') can be computed in time O(n*n'*n'') using memorization

Iterative formulation for 3 seqs

```
T[0..n][0..n'][0..n''] = undef

for i = 0 to n:
    for j = 0 to n':
        for k = 0 to n'':

            v0 = v1 = v2 = v3 = v4 = v5 = v6 = v7 = undef

            if i=0 and j=0 and k=0 then
                v0 = 0

            if i>0 and j>0 and k>0 then
                v1 = T[i-1][j-1][k-1] + SP(A[i], B[j], C[k])

            if i>0 and j>0 and k≥0 then
                v2 = T[i-1][j-1][k] + SP(A[i], B[j], - )

            if i>0 and j≥0 and k>0 then
                v3 = T[i-1][j][k-1] + SP(A[i], -, C[k])

            if i≥0 and j>0 and k>0 then
                v4 = T[i][j-1][k-1] + SP(-, B[j], C[k])

            if i>0 and j≥0 and k≥0 then
                v5 = T[i-1][j][k] + SP(A[i], -, - )

            if i≥0 and j>0 and k≥0 then
                v6 = T[i][j-1][k] + SP(-, B[j], - )

            if i≥0 and j≥0 and k>0 then
                v7 = T[i][j][k-1] + SP(-, -, C[k])

            T[i][j][k] = min(v0, v1, v2, v3, v4, v5, v6, v7)

print T[n][n'][n'']
```

SP-cost of a column of 3 symbols

Let **sub(x,y)** be the substitution cost and **gap** be the gapcost

SP(A[i], B[j], C[k]) = sub(A[i], B[j]) + sub(B[j], C[k]) + sub(A[i], C[k])

SP(A[i], B[j], -) = sub(A[i], B[j]) + gap + gap

SP(A[i], -, C[k]) = gap + sub(A[i], C[k]) + gap

SP(-, B[j], C[k]) = gap + gap + sub(B[j], C[k])

SP(A[i], -, -) = gap + gap

SP(-, B[j], -) = gap + gap

SP(-, -, C[k]) = gap + gap

Other column based score functions?

Consensus score

Tree score

...

Consensus score

A consensus seq., or row, of $\mathcal{M}$ is a seq. that summarize $\mathcal{M}$.

The consensus score $CS(\cdot)$ measures to what extend this is possible ...

$$\mathcal{M} = \begin{bmatrix} R_1 \\ \vdots \\ R_K \end{bmatrix}$$

R_{K+1}

$$CS(\mathcal{M}) = \sum_{1 \le i \le K} \text{Pair Score}(R_i, R_{K+1})$$

CS alignment problem

Find $\mathcal{M}^*$ such that $CS(\mathcal{M}^*)$ is minimized (or maximized)

.... complexity depends on definition of R_{K+1}.

Consensus score

A typical definition of R_{cc} is:

$R_M = c_1 c_2 \ldots c_L$, $c_j \in \Sigma \cup \{-\}$ where

$$c_j = \underset{c \in \Sigma \cup \{-\}}{\operatorname{argmin}} \sum_{i=1}^{k} d(c, M_{ij})$$

Using this definition and assuming PairScore is col-based yields:

$$CS(M) = \sum_{1 \le i \le k} \underbrace{\sum_{1 \le j \le L} d(c_j, M_{ij})}_{\text{PairScore}(R_i, R_{cc})}$$

$$= \sum_{1 \le j \le L} \underbrace{\sum_{1 \le i \le k} d(c_j, M_{ij})}_{\text{Score of col. } j}$$

Consensus score

A typical definition of R_{OC} is:

$CS(\cdot)$ is turned into col. based score function, i.e. opt. score $CSCol(\pi)$

can be computed as:

$$D(i_1, i_2, \ldots, i_k) = \min_{c,\ \text{lastcol}} \left[D(i_1', \ldots, i_k') + \sum_{1 \le i \le k} d(c, \text{lastcol}[i]) \right]$$

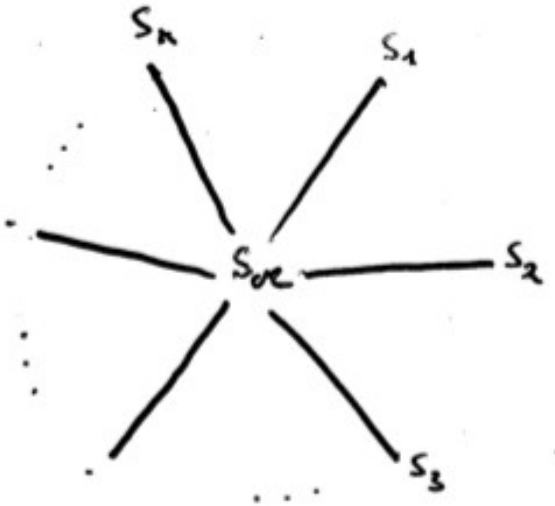
Time: $O(|\Sigma| \cdot 2^k \cdot k \cdot n^k)$, space: $O(n^k)$

$$= \sum_{1 \le j \le L} \underbrace{\sum_{1 \le i \le k} d(c_j, d_{ij})}_{\text{Score of col. } j}$$

Consensus score – Alternative form

Find S_{ac} such that $\sum_{i=1}^{k} D(S_i, S_{ac})$ is minimized,

where $D(\cdot,\cdot)$ is the score of an opt. pairwise alignment



A "Steiner Problem

The optimal alignment $\mathcal{A}^*$ is a combination of the

k pairwise alg. s.t. $\mathcal{A}^*(R_i, R_j) = \mathcal{A}(S_i, S_j)$.

This is possible for any tree cf. Thm 14.6.1. in [Gusfield].

Tree score

Generalized consensus score. Say seq's $S_1, S_2, \ldots, S_k$ are related by a tree:

$$\mathcal{M} = \begin{bmatrix} - & R_1 & - \\ & \vdots & \\ - & R_k & - \end{bmatrix}$$

Given an assignment of rows to internal nodes, then:

$$TS(\mathcal{M}) = \sum_{\text{edge}(X,Y)} \text{PairScore}(X,Y)$$

Tree score

Alignment problem

Find $\mathcal{M}^*$, i.e. $R_1, \ldots, R_K$, and assignments to internal nodes s.t.

$TS(\mathcal{M}^*)$ is minimized

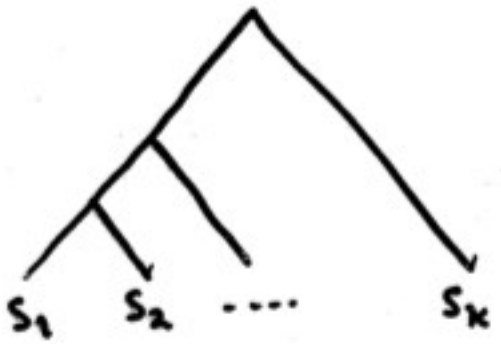
If PairScore$(\cdot,\cdot)$ is col. based, then $TS(\cdot)$ becomes col. based similar to $CS(\cdot)$, we can compute an optimal TS-alignment in

Time: $O(|\Sigma|^K \cdot 2^K \cdot k \cdot n^K)$, Space: $O(n^K)$

↑ all possible assignments.

Tree score – Alternative form

Say $S_1, \ldots, S_k$ are related by a tree:



Find assignment of seq's $\in \Sigma^*$ to internal nodes s.t.

$$\sum_{\text{edge}(X,Y)} D(X,Y)$$

is minimized, the optimal alignment all^* is a combination

of the optimal pairwise alignments.

Tree score – Alternative form

How do we get T ?

The "big" tree alignment problem:

Find tree T^* relating $S_1, \ldots, S_k$ and mult. align. $\mathcal{A}^*$ of $S_1, \ldots, S_k$

such $TS(\mathcal{A}^*)$ given $\mathcal{T}^*$ is minimized over all possible

choices of $\mathcal{A}^*$ and $\mathcal{T}^*$ ....