Multiple Alignment

Guan-Shieng Huang

Mar. 18, 2003

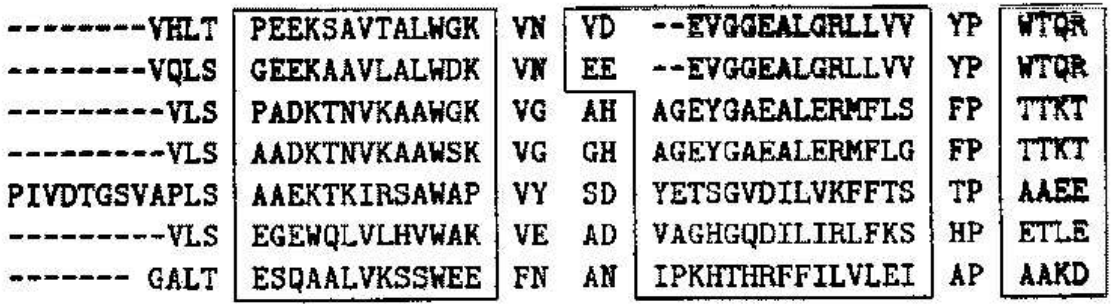
What is a multiple alignment

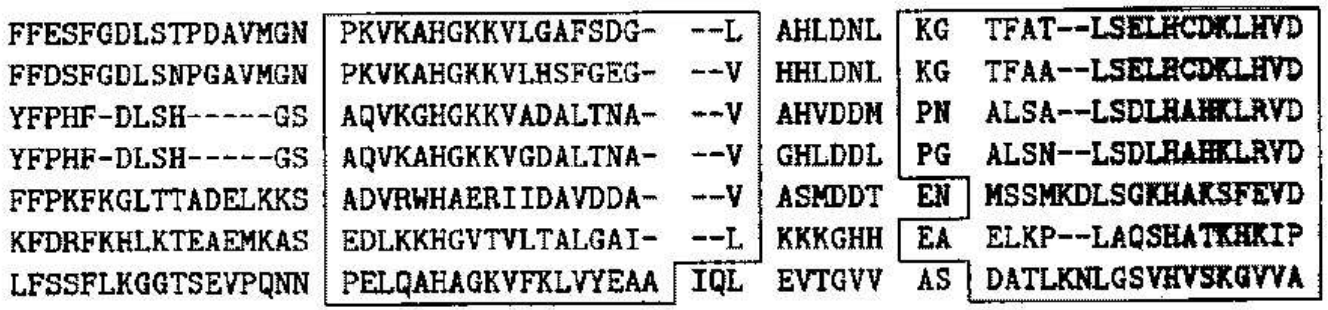
A C . . B C D B

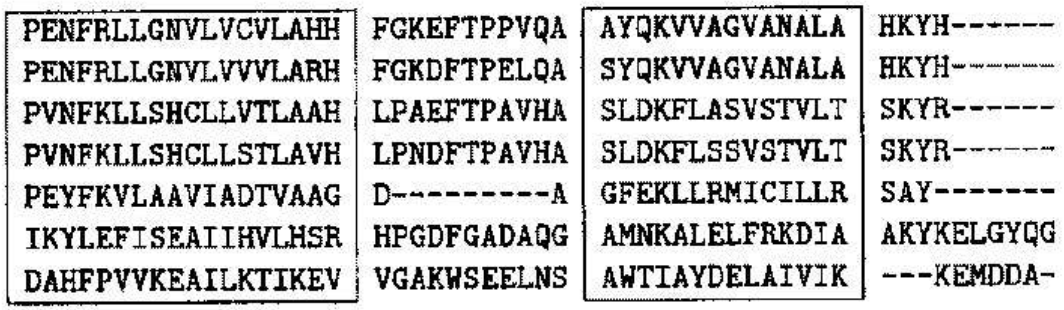
. C A D B . D .

A C A . B C D .

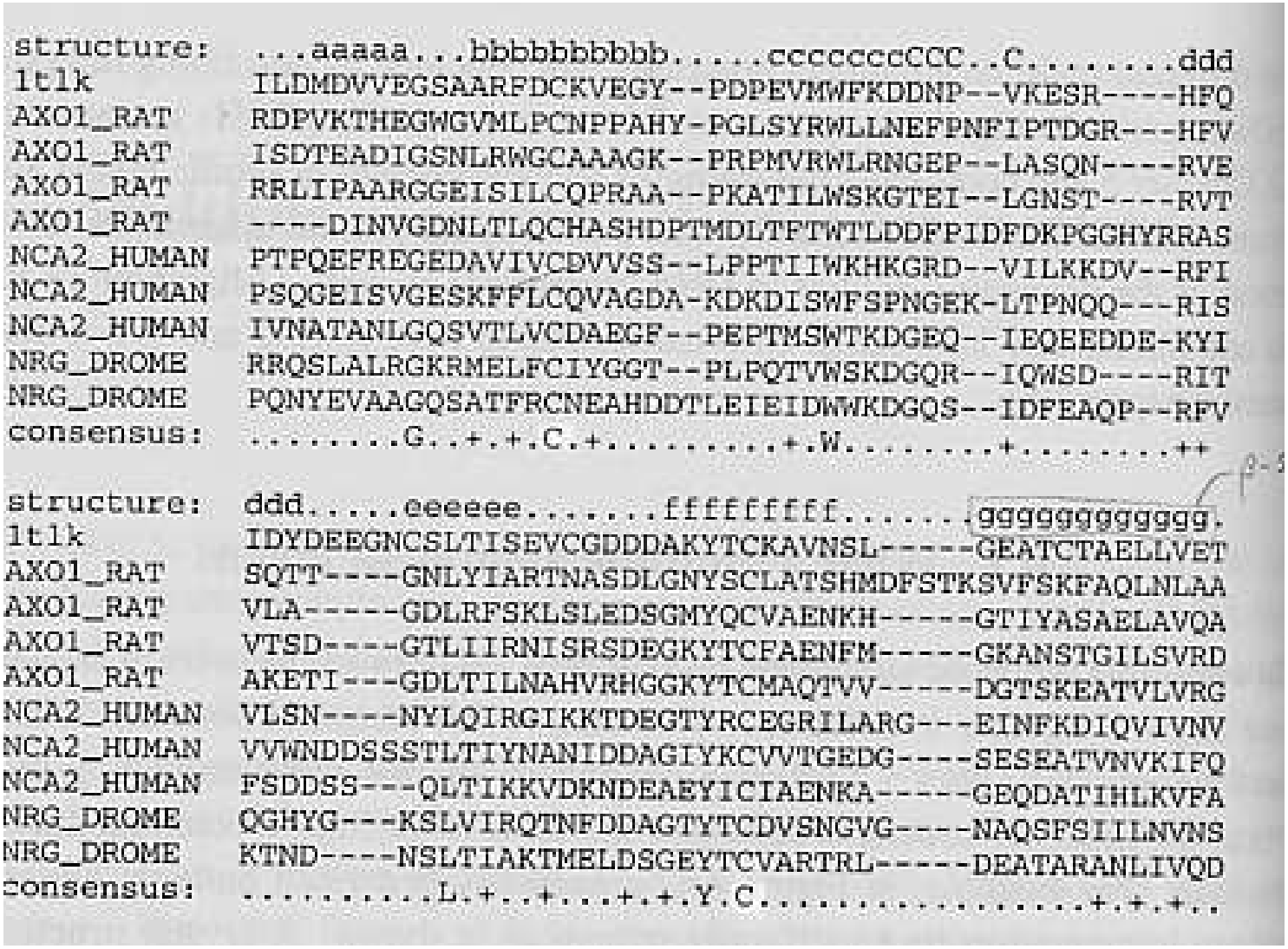
An alignment of globins produced by CLUSTAL







An alignment of ten I-set immunoglobin superfamily



```
structure:   ...aaaaa...bbbbbbbbbb.....ccccccccCC..C.........ddd
1tlk         ILDMDVVEGSAARFDCKVEGY--PDPEVMWFKDDNP--VKESR----HFQ
AXO1_RAT     RDPVKTHEGWGVMLPCNPPAHY-PGLSYRWLLNEFPNFIPTDGR---HFV
AXO1_RAT     ISDTEADIGSNLRWGCAAAGK--PRPMVRWLRNGEP--LASQN----RVE
AXO1_RAT     RRLIPAARGGEISILCQPRAA--PKATILWSKGTEI--LGNST----RVT
AXO1_RAT     ----DINVGDNLTLQCHASHDPTMDLTFTWTLDDFPIDFDKPGGHYRRAS
NCA2_HUMAN   PTPQEFREGEDAVIVCDVVSS--LPPTIIWKHKGRD--VILKKDV--RFI
NCA2_HUMAN   PSQGEISVGESKFFLCQVAGDA-KDKDISWFSPNGEK-LTPNQQ---RIS
NCA2_HUMAN   IVNATANLGQSVTLVCDAEGF--PEPTMSWTKDGEQ--IEQEEDDE-KYI
NRG_DROME    RRQSLALRGKRMELFCIYGGT--PLPQTVWSKDGQR--IQWSD----RIT
NRG_DROME    PQNYEVAAGQSATFRCNEAHDDTLEIEIDWWKDGQS--IDFEAQP--RFV
consensus:   ........G..+.+.C.+.........+.W........+........++

structure:   ddd.....eeeeee.......fffffffff.......ggggggggggggg.
1tlk         IDYDEEGNCSLTISEVCGDDDAKYTCKAVNSL-----GEATCTAELLVET
AXO1_RAT     SQTT----GNLYIARTNASDLGNYSCLATSHMDFSTKSVFSKFAQLNLAA
AXO1_RAT     VLA-----GDLRFSKLSLEDSGMYQCVAENKH-----GTIYASAELAVQA
AXO1_RAT     VTSD----GTLIIRNISRSDEGKYTCFAENFM-----GKANSTGILSVRD
AXO1_RAT     AKETI---GDLTILNAHVRHGGKYTCMAQTVV-----DGTSKEATVLVRG
NCA2_HUMAN   VLSN----NYLQIRGIKKTDEGTYRCEGRILARG---EINFKDIQVIVNV
NCA2_HUMAN   VVWNDDSSSTLTIYNANIDDAGIYKCVVTGEDG----SESEATVNVKIFQ
NCA2_HUMAN   FSDDSS---QLTIKKVDKNDEAEYICIAENKA-----GEQDATIHLKVFA
NRG_DROME    QGHYG---KSLVIRQTNFDDAGTYTCDVSNGVG----NAQSFSIILNVNS
NRG_DROME    KTND----NSLTIAKTMELDSGEYTCVARTRL-----DEATARANLIVQD
consensus:   ..........L.+..+...+.+.Y.C.................+.+.+..
```

Motivation

1. **Protein databases are often categorized by protein families.**
   How can we identify a newly sequenced protein?
2. ***Alu* repeat is approximately 300 bps and appears over 600000 times in the human genome.**
3. **A multiple alignment may suggest**
   - **a common structure of the protein products**
   - **a common function**
   - **a common evolutionary source.**

Issues

- **How to define meaningful scoring function for an alignment?**
  1. **evolutionary correct alignment — more difficult!**
  2. **structure alignment**
- **How to find the best alignment? By algorithms.**

Three types of alignment problems

1. **DNA**
2. **protein (joined by disulfide bond)**
3. **RNA — more difficult due to long-range correlation**

In this lecture, we focus on alignment problems of sequences of DNAs or proteins.

Formulation

1. **Input:** k **sequences** $S_1, S_2, \ldots, S_k$
2. **Output:** $S'_1, S'_2, \ldots, S'_k$ **of equal length such that** S'_i **is obtained from** S_i **by inserting spaces.**
3. **Goal: For each column of the alignment, we have a score. Our goal is to optimize the overall score.**

Example:

S'_1:	A	C	.	.	B	C	D	B
S'_2:	.	C	A	D	B	.	D	.
S'_3:	A	C	A	.	B	C	D	.

Here we assume that individual columns are statistically independent, and hence the overall score is the sum of scores in individual columns.

Algorithm — for $k = 3$

1. $S_1 = x_1 x_2 \ldots x_{n_1}$

   $S_2 = y_1 y_2 \ldots y_{n_2}$

   $S_3 = z_1 z_2 \ldots z_{n_3}$

2. $D(i_1, i_2, i_3)$**: the best score for aligning the prefixes of length** i_1, i_2, i_3 **of** S_1, S_2, S_3**, respectively.**

3. $\sigma(x_{i_1}, y_{i_2}, z_{i_3})$ **the distance to align** $x_{i_1}, y_{i_2}, z_{i_3}$ **in one column.**

4. We have

$$D(i_1,i_2,i_3)=\min\begin{cases} D(i_1-1,i_2-1,i_3-1)+\sigma(x_{i_1},y_{i_2},z_{i_3}) \\ D(i_1-1,i_2-1,i_3)+\sigma(x_{i_1},y_{i_2},-) \\ D(i_1-1,i_2,i_3-1)+\sigma(x_{i_1},-,z_{i_3}) \\ D(i_1,i_2-1,i_3-1)+\sigma(-,y_{i_2},z_{i_3}) \\ D(i_1-1,i_2,i_3)+\sigma(x_{i_1},-,-) \\ D(i_1,i_2-1,i_3)+\sigma(-,y_{i_2},-) \\ D(i_1,i_2,i_3-1)+\sigma(-,-,z_{i_3}) \end{cases}$$

and $D(0,0,0)=0$**.**

Note that we minimize the total distance of the alignments.

Complexity

- **time** $= O(n_1 n_2 n_3 \cdot (2^3 - 1))$
  time $= O(2^k \cdot \prod_{j=1}^{k} n_j)$ **for any** k
  Hence the DP algorithm for MSA is in fact a pseudo-polynomial algorithm.

- **space** $= O(n_1 n_2 n_3)$
  space $= O(\prod_{j=1}^{k} n_j)$ **for any** k**.**

- **the problem is NP-complete even for the Sum-of-pairs (Wang & Jiang, 1994)**

To prove that a computational problem is NP-hard, we need

- **to reduce an NP-complete (hard) problem to this problem.**

When a computational problem is NP-hard, we deal with it by

- **heuristic: convince other people by experiments**
- **approximation: how to analyze the performance**
- **randomization: how to design a reasonable algorithm.**

Famous NP-complete problems

1. **Satisfiability: Test whether a logical formula given in conjunctive normal form is satisfiable.**

   $$(x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1) \wedge (x_1 \vee \neg x_2 \vee x_3) \cdots$$

2. **Traveling Salesman Problem: Find the shortest tour in a graph.**
3. **3-coloring: Ask whether a graph can be colored by three colors.**
4. **Max-clique: Find a maximum clique in arbitrary graph.**
5. **Max-Cut**
6. **Knapsack problem**

Notations

$D(X, Y)$: the pariwise distance from X to Y

$d_{\mathcal{M}}(X, Y)$: the distance from X to Y in some alignment $\mathcal{M}$

We usually write $d_{\mathcal{M}}(X, Y)$ as $d(X, Y)$ when the alignment $\mathcal{M}$ is implicitly assumed.

Note that $D(X, Y) \leq d(X, Y)$ always holds.

Suppose we have an alignment :

S_1' :	**a**	**c**	**a**	**c**	**a**	**c**	**a**	**c**	
S_2' :	**a**	**b**	**a**	**b**	**a**	**b**	**a**	**b**	
S_3' :	**c**	**b**	**c**	**b**	**c**	**b**	**c**	**b**	
	2	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**distance = 16**

However, the pairwise distance between S_1 and S_2 is

S_1 :	**a**	**c**	**a**	**c**	**a**	**c**	**a**	**c**	
S_2 :	**a**	**b**	**a**	**b**	**a**	**b**	**a**	**b**	
	0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**distance = 4**

For S_1, S_3, it is

S_1 :	**a**	**c**	**a**	**c**	**a**	**c**	**a**	**c**	**-**	
S_3 :	**-**	**c**	**b**	**c**	**b**	**c**	**b**	**c**	**b**	
	1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**distance = 5**

And for S_2, S_3, it is

S_2 :	**a**	**b**	**a**	**b**	**a**	**b**	**a**	**b**	
S_3 :	**c**	**b**	**c**	**b**	**c**	**b**	**c**	**b**	
	1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**distance = 4**

We can see that $D(S_1, S_2) + D(S_1, S_3) + D(S_2, S_3) = 4 + 5 + 4 = 13 < 16 = d(S'_1, S'_2, S'_3)$**.**

Multiple alignment problems

1. **Distance from consensus:**
   Find a center string C such that $\Sigma D(S_i, C)$ is minimum.

2. **Sum-of-pairs:**
   Minimize $\Sigma_{i<j} d_{\mathcal{M}}(S_i, S_j)$ for all possible alignment $\mathcal{M}$.
   $(\sigma(x_{i_1}, x_{i_2}, x_{i_3}) = d(x_{i_1}, x_{i_2}) + d(x_{i_1}, x_{i_3}) + d(x_{i_2}, x_{i_3}))$

3. **Evolutionary tree alignment:**
   Optimize the cost of an evolutionary tree.
   (E.g. minimum spanning tree)

Sum-of-pairs problem

- **Input: a set of sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$**
- **Output: compute a global multiple alignment $\mathcal{M}$ with minimum sum-of-pairs score**

Example: $\mathcal{M} =$

S_1':	**A**	**C**	**.**	**.**	**B**	**C**	**D**	**B**
S_2':	**.**	**C**	**A**	**D**	**B**	**.**	**D**	**.**
S_3':	**A**	**C**	**A**	**.**	**B**	**C**	**D**	**.**

for $k = 3$**.**

$d(S_1', S_2') + d(S_1', S_3') + d(S_2', S_3')) = 5 + 2 + 3 = 10$**.**

(equal=0, not equal=1)

Branch & bound heuristic for the DP algorithm of the Sum-of-pairs:

- **Carrillo & Lipman (1988)**
- **The idea was implemented in the famous problem MSA**
  Lipman, Altshul, Kececiogly, 1989
- **MSA can align 6 sequences of length ~ 200 in reasonable time.**

Let $\mathcal{M}$ be any multiple alignment of $\mathcal{S}$.

Let $\widehat{\mathcal{M}}$ be an optimal multiple alignment of $\mathcal{S}$.

(Hence, $\Sigma_{i<j} d_{\widehat{\mathcal{M}}}(S_i, S_j) \leq \Sigma_{i<j} d_{\mathcal{M}}(S_i, S_j)$.)

And also, $D(S_i, S_j) \leq d_{\mathcal{M}}(S_i, S_j)$ for all $\mathcal{M}$.

Suppose an upper bound $\sigma(\mathcal{S})$ of the best alignment is given. For any s, t, we have

$$\begin{aligned} \sigma(\mathcal{S}) &\geq \Sigma_{i<j} d_{\widehat{\mathcal{M}}}(S_i, S_j) \\ &\geq \Sigma_{i<j} D(S_i, S_j) - D(S_s, S_t) + d_{\widehat{\mathcal{M}}}(S_s, S_t). \end{aligned}$$

$D(S_i, S_j)$ can be evaluated by the optimal pairwise alignment algorithm. Thus, $d_{\widehat{\mathcal{M}}}(S_s, S_t)$ has an upper bound on the s, t-plane.

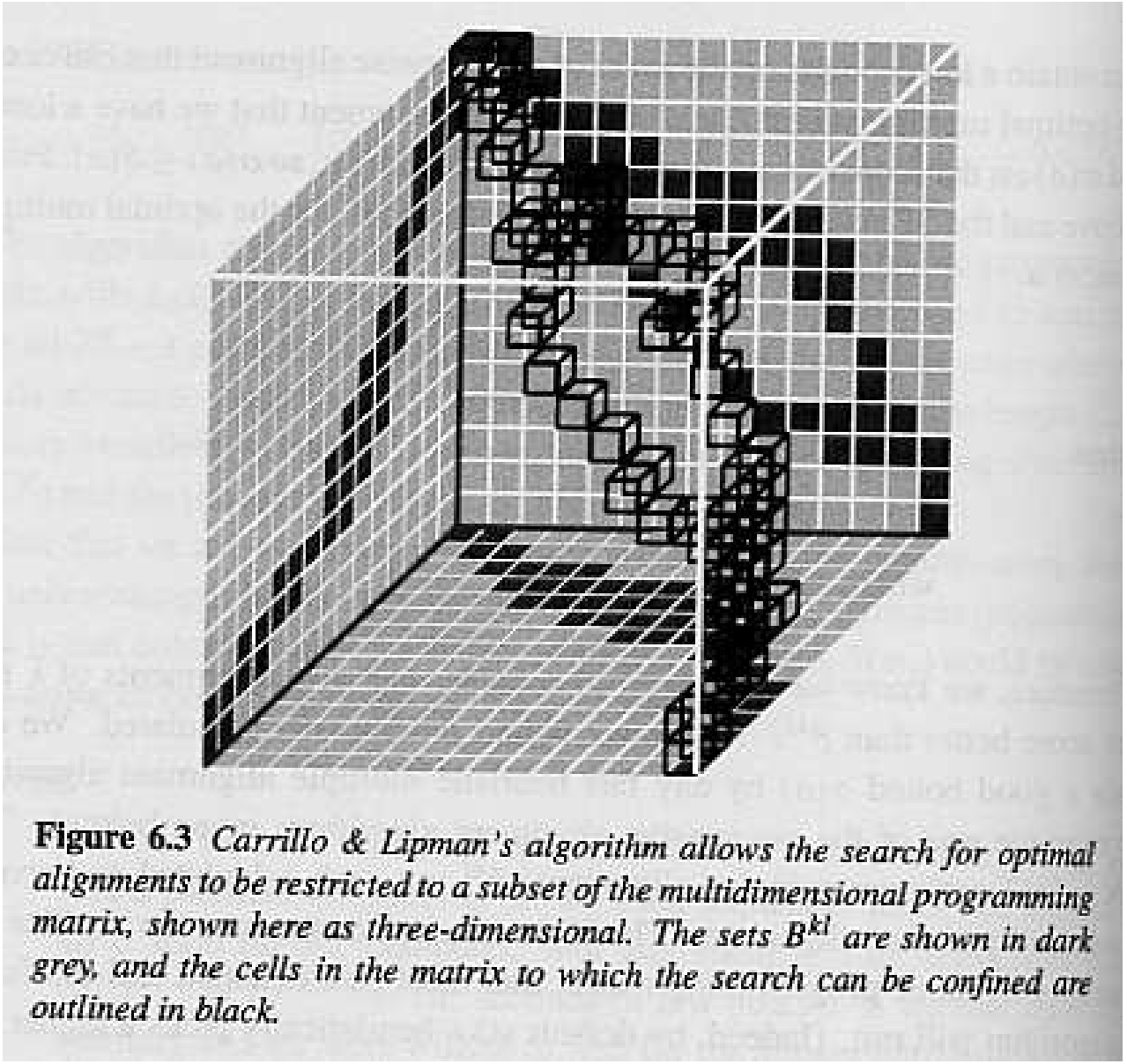


Figure 6.3 *Carrillo & Lipman's algorithm allows the search for optimal alignments to be restricted to a subset of the multidimensional programming matrix, shown here as three-dimensional. The sets B^{kl} are shown in dark grey, and the cells in the matrix to which the search can be confined are outlined in black.*

Metric Space

In the following, we assume

- $d(x, x) = 0$ **for all characters (including blank** $-$**)**
- $d(x, y) = d(y, x)$ **(symmetry)**
- $d(x, y) \leq d(x, z) + d(z, y)$ **holds (triangle inequality).**

We discuss approximation algorithms for the Sum-of-Pairs and for the distance-from-consensus.

2-Approximation algorithm for the Sum-of-pairs

(the Center Star Method)

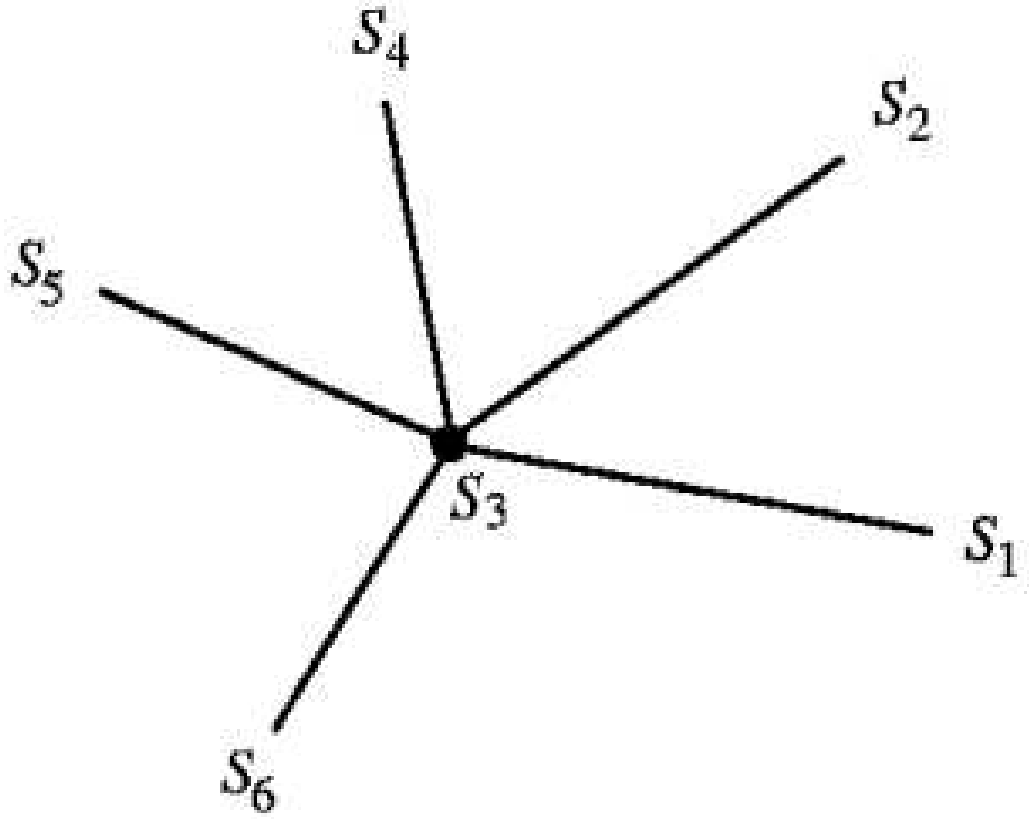
1. **Find $S_t \in \mathcal{S}$ minimizing $\sum_{i \neq t} D(S_i, S_t)$ and let $\mathcal{M} = \{S_t\}$.**
   Call S_t the the center of $\mathcal{S}$.

2. **Add the sequences in $\mathcal{S} - \{S_t\}$ to $\mathcal{M}$ one by one so that the alignment of every newly added sequence with S_t is optimal.**

Running time:

1. $O(k^2n^2)$ **for Step 1.**
2. $\sum_{i=1}^{k-1} O((i \cdot n) \cdot n) = O(k^2 \cdot n^2)$ **for Step 2.**

The produced answer is at most twice of the minimum solution.


S_4
S_2
S_5
S_3
S_1
S_6

Example

$\{S_1 = \texttt{ATGCTC}, S_2 = \texttt{AGAGC}, S_3 = \texttt{TTCTG}, S_4 = \texttt{ATTGCATGC}\}$

S_1=ATGCTC S_2=A-GAGC $D(S_1, S_2) = 3$	S_1=ATGCTC S_3=TT-CTG $D(S_1, S_3) = 3$	S_1=AT-GC-T-C S_4=ATTGCATGC $D(S_1, S_4) = 3$
S_2=AGAGC S_3=TTCTG $D(S_2, S_3) = 5$	S_2=A--G-A-GC S_4=ATTGCATGC $D(S_2, S_4) = 4$	S_3=-TT-C-TG- S_4=ATTGCATGC $D(S_3, S_4) = 4$

Hence $D(S_1, S_2) + D(S_1, S_3) + D(S_1, S_4) = 9$**,**

$D(S_2, S_1) + D(S_2, S_3) + D(S_2, S_4) = 12$**,**

$D(S_3, S_1) + D(S_3, S_2) + D(S_3, S_4) = 12$**,**

$D(S_4, S_1) + D(S_4, S_2) + D(S_4, S_3) = 11$**.** $\therefore$ **The center is** S_1**.**

The alignment of S_1, S_2, S_3 is

S_1= ATGCTC

S_2= A-GAGC

S_3= TT-CTG.

And after adding S_4, we have

S_1= AT-GC-T-C

S_2= A--GA-G-C

S_3= TT--C-T-G

S_4= ATTGCATGC.

Performance Analysis

$App = \sum_{1 \le i \ne j \le k} d_{\mathcal{M}}(S_i, S_j)$

$Opt = \sum_{1 \le i \ne j \le k} d_{\widehat{\mathcal{M}}}(S_i, S_j)$

WLOG, we assume S_1 is the center.

$$
\begin{aligned}
App &= \sum_{1 \le i \ne j \le k} d_{\mathcal{M}}(S_i, S_j) \\
&\le \sum_{1 \le i \ne j \le k} [D(S_i, S_1) + D(S_1, S_j)] \\
&= \sum_{1 \le i \ne j \le k} D(S_1, S_i) + \sum_{1 \le i \ne j \le k} D(S_1, S_j) \\
&= 2(k-1) \sum_{1 < i \le k} D(S_1, S_i)
\end{aligned}
$$

$$
\begin{aligned}
Opt &= \sum_{1 \le i \ne j \le k} d_{\widehat{\mathcal{M}}}(S_i, S_j) \\
&\ge \sum_{1 \le i \ne j \le k} D(S_i, S_j) \\
&= \sum_{1 \le i \le k} \sum_{1 \le j \ne i \le k} D(S_i, S_j) \\
&\ge k \sum_{1 < j \le k} D(S_1, S_j)
\end{aligned}
$$

$$
\therefore \frac{App}{Opt} \le \frac{2(k-1)}{k} < 2.
$$

Distance from consensus:

- **Input: a set of sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$**
- **Output: a center string C such that $\Sigma D(S_i, C)$ is minimum.**
  (C may not be in $\mathcal{S}$.)

Fact: The center $S_t \in \mathcal{S}$ minimizing $\sum_{i \neq t} D(S_i.S_t)$ is a 2-approximation.

Performance Analysis

WLOG, we assume S_1 is the center.

$App = \sum_{1<i\leq k} D(S_1, S_i)$

$Opt = \sum_{1\leq i\leq k} D(C, S_i)$

$$\begin{aligned}
\sum_{1\leq i\neq j\leq k} D(S_i, S_j) &\leq \sum_{1\leq i\neq j\leq k} [D(S_i, C) + D(C, S_j)] \\
&= \sum_{1\leq i\neq j\leq k} D(S_i, C) + \sum_{1\leq i\neq j\leq k} D(C, S_j) \\
&= 2 \sum_{1\leq i\neq j\leq k} D(C, S_i) \\
&= 2(k-1) \sum_{1\leq i\leq k} D(C, S_i) = 2(k-1) \cdot Opt.
\end{aligned}$$

$$\begin{aligned}\sum_{1\le i\neq j\le k} D(S_i,S_j) &= \sum_{1\le i\le k}\sum_{1\le j\neq i\le k} D(S_i,S_j)\\ &\ge k\sum_{1<j\le k} D(S_1,S_j) = k\cdot App.\end{aligned}$$

$$\therefore \frac{App}{Opt} \le \frac{2(k-1)}{k} < 2.$$

A better solution can be produced by choosing the most common characters in each column.

Example:

A	**B**	**-**	**C**	**-**	**D**
A	**B**	**C**	**C**	**-**	**D**
A	**B**	**C**	**D**	**E**	**-**
A	**B**	**C**	**C**	**-**	**D**

Phylogenetic Tree