Multiple Sequence Alignment

Chin Lung Lu

Computational Biology

Analyses and Applications of Sequences

Multiple sequence alignment

S_1:	R	C	T	L	E	E	S_1:	R	C	T	L	E	E
S_2:	R	C	L	E	E		S_2:	R	C	–	L	E	E
S_3:	C	T	L	E	E		S_3:	–	C	T	L	E	E
S_4:	C	T	E	E			S_4:	–	C	T	–	E	E

4 Sequences ⇨ A Multiple Sequence Alignment

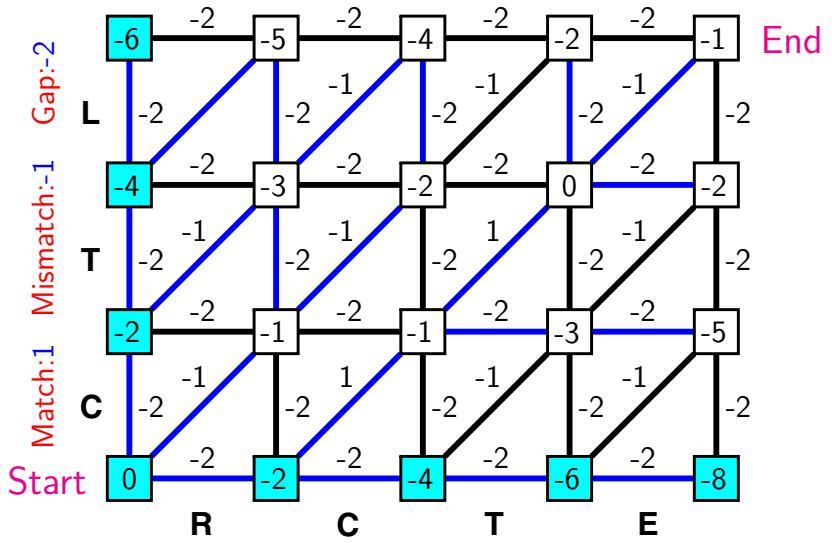
SP-score of an MSA

- SP-score (Sum-of-Pairs) of an MSA: the sum of the SP-scores of all columns
- SP-score of a column: the sum of pairwise scores of all pairs of characters

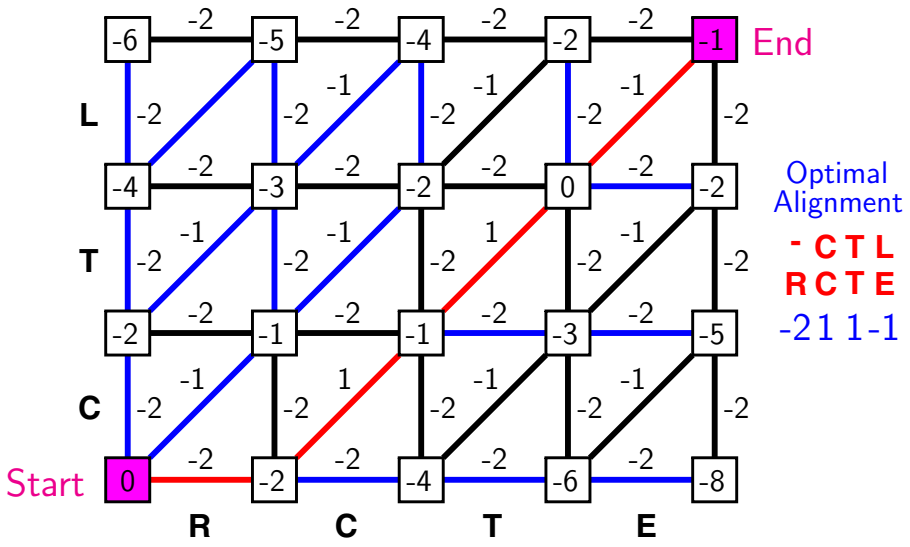
S_1:	R	C	T	L	E	E
S_2:	R	C	–	L	E	E
S_3:	–	C	T	L	E	E
S_4:	–	C	T	–	E	E

The SP-score of the first column is $score(\mathbf{R},\mathbf{R}) + 4 \times score(\mathbf{R},\mathbf{–}) + score(\mathbf{–},\mathbf{–})$, where $score(\mathbf{–},\mathbf{–}) = 0$

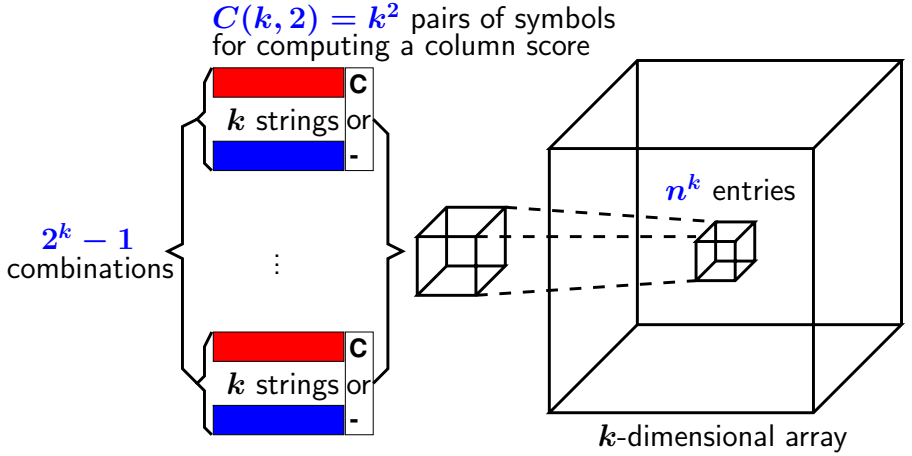
Two sequence alignment



Two sequence alignment



Sum-of-pairs MSA problem



- The total time-complexity is $\mathcal{O}(k^2 2^k n^k)$.

Sum-of-pairs MSA problem

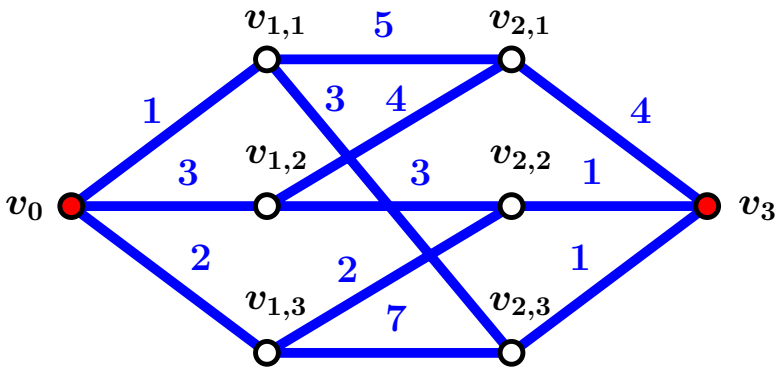
- NP-complete: by Wang and Jiang [WJ94]; Bonizzoni and Vedova [BV01]
- Branch and bound methods: by Carrillo and Lipman [CL88] (improved by [LAK89, GKS95])
- Approximate methods: (k sequences)
  - $(2 - \frac{2}{k})$-approximation, by Gusfield [Gus93]
  - $(2 - \frac{3}{k})$-approximation, by Pevzner [Pev92]
  - $(2 - \frac{l}{k})$-approximation, where $l < k$ by Bafna, Lawler and Pevzner [BLP97]
- Heuristic methods: Progressive algorithms

Branch and bound strategy

1. Branch mechanism: a way of generating the branches of the solution space (tree)
2. Bound mechanism: a way of bounding the branches to avoid the exhaustive search of solution space
   - Find an upper bound of an optimal solution (find a feasible solution using tree searching methods, like breadth-first, depth-first, and best-first searches)
   - Predict a lower bound for a branch
   - If the lower bound of a branch $>$ the upper bound, then the branch is terminated.

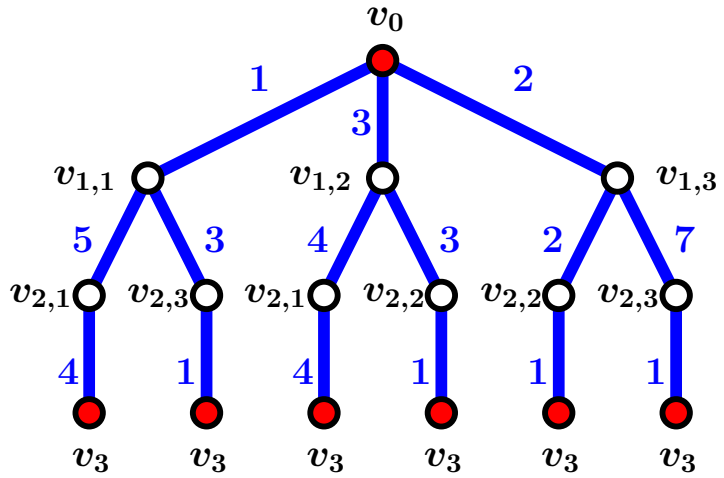
Shortest path problem ❶

- Find a shortest path from v_0 to v_3

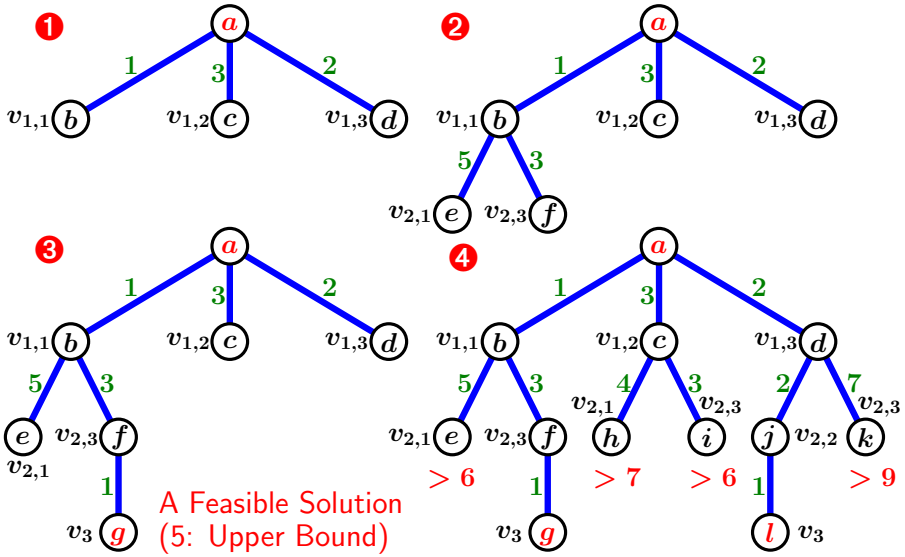


Shortest path problem ❷

- A search tree of the problem (6 feasible solutions):



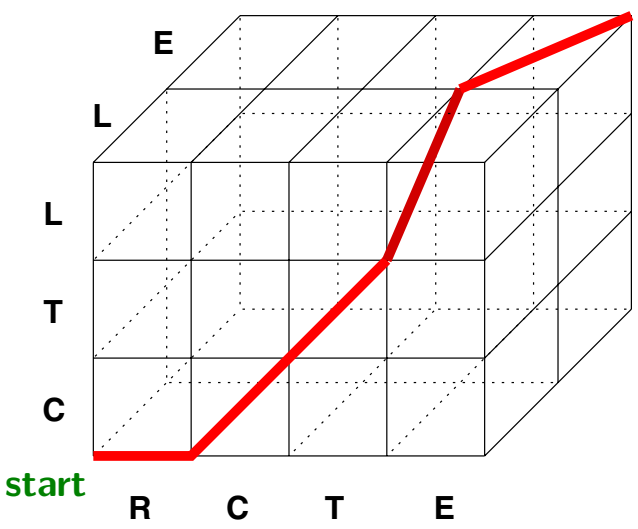
Shortest path problem ❸



Sum-of-pairs MSA problem

- NP-complete: by Wang and Jiang [WJ94]; Bonizzoni and Vedova [BV01]
- Branch and bound methods: by Carrillo and Lipman [CL88] (improved by [LAK89, GKS95])
- Approximate methods: (k sequences)
  - $(2 - \frac{2}{k})$-approximation, by Gusfield [Gus93]
  - $(2 - \frac{3}{k})$-approximation, by Pevzner [Pev92]
  - $(2 - \frac{l}{k})$-approximation, where $l < k$ by Bafna, Lawler and Pevzner [BLP97]
- Heuristic methods: Progressive algorithms

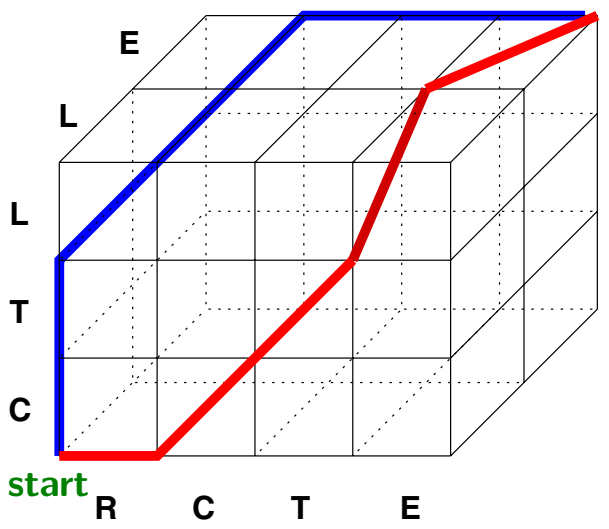
Dynamic programming approach



R C T - E
- C T L -
- - - L E

Optimal MSA

Dynamic programming approach



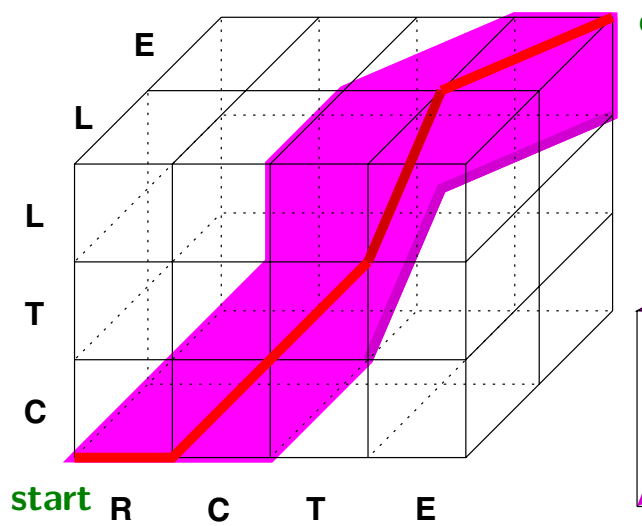
R C T - E
- C T L -
- - - L E

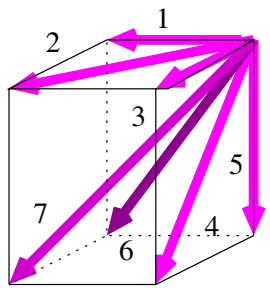
Optimal MSA

- - R - - C T E
C T L - - - - -
- - - L E - - -

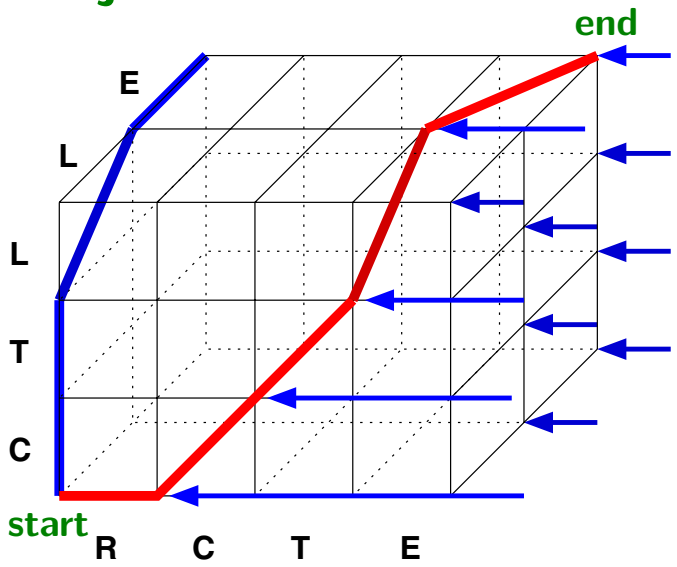
MSA

Polyhedron





Projection of MSA



Induced pairwise alignment

- Let $\mathcal{A}$ be a multiple alignment of $S_1, S_2, \cdots, S_k$.
- Induced pairwise alignment $\mathcal{A}_{i,j}$ ($\mathcal{A}$ w.r.t. S_i and S_j): a pairwise alignment of S_i and S_j obtained by copying rows i and j of $\mathcal{A}$ and removing the columns with a gap in both rows
- $c(\mathcal{A}_{i,j}) \geq d(S_i, S_j)$
  - $c(\mathcal{A}_{i,j})$: the cost of $\mathcal{A}_{i,j}$
  - $d(S_i, S_j)$: the minimum distance of S_i and S_j
- The SP-cost of $\mathcal{A} = c(\mathcal{A}) = \sum_{i<j} c(\mathcal{A}_{i,j})$

Costing function: metric

- A costing function c is a metric if for any x, y and z, it satisfies ❶ $c(x, y) \geq 0$, where $c(x, y) = 0$ if $x = y$, ❷ symmetry: $c(x, y) = c(y, x)$ and ❸ triangle inequality: $c(x, z) \leq c(x, y) + c(y, z)$.
- Given a set S of k sequences $S_1, S_2, \cdots S_k$ and a scoring function c (metric):
  - $\mathcal{A}^o$: an optimal MSA of S with cost $c(\mathcal{A}^o)$
  - $\mathcal{A}^h$: a heuristic MSA of S with cost $c(\mathcal{A}^h)$
  - $\mathcal{A}^o_{i,j}$: the projection of $\mathcal{A}^o$ w.r.t. S_i and S_j
  - $\mathcal{A}^h_{i,j}$: the projection of $\mathcal{A}^h$ w.r.t. S_i and S_j

Carrillo-Lipman bound

- SP-cost$(\mathcal{A}^o) = c(\mathcal{A}^o) = \sum_{1 \leq i < j \leq k} c(\mathcal{A}^o_{i,j})$
- SP-cost$(\mathcal{A}^h) = c(\mathcal{A}^h) = \sum_{1 \leq i < j \leq k} c(\mathcal{A}^h_{i,j})$
- $c(\mathcal{A}^h) \geq c(\mathcal{A}^o)$

  $\Rightarrow \sum_{1 \leq i < j \leq k} c(\mathcal{A}^h_{i,j}) \geq \sum_{1 \leq i < j \leq k} c(\mathcal{A}^o_{i,j})$

  $\Rightarrow \sum_{1 \leq i < j \leq k} \left(c(\mathcal{A}^h_{i,j}) - c(\mathcal{A}^o_{i,j}) \right) \geq 0$

Carrillo-Lipman bound

$\because \sum_{1 \leq i < j \leq k} \left(c(\mathcal{A}^h_{i,j}) - c(\mathcal{A}^o_{i,j}) \right) \geq 0$

$\therefore$ For any arbitrary projection w.r.t. S_p and S_q,

$$\left(\sum_{\substack{1 \leq i < j \leq k \\ (i,j) \neq (p,q)}} \left(c(\mathcal{A}^h_{i,j}) - c(\mathcal{A}^o_{i,j}) \right) \right) + c(\mathcal{A}^h_{p,q}) - c(\mathcal{A}^o_{p,q}) \geq 0$$

$$\Rightarrow \left(\sum_{\substack{1 \leq i < j \leq k \\ (i,j) \neq (p,q)}} \left(c(\mathcal{A}^h_{i,j}) - c(\mathcal{A}^o_{i,j}) \right) \right) + c(\mathcal{A}^h_{p,q}) \geq c(\mathcal{A}^o_{p,q})$$

Carrillo-Lipman bound

- $c(\mathcal{A}^o_{i,j}) \geq d(S_i, S_j)$ for any i and j

$$\because \left(\sum_{\substack{1 \leq i < j \leq k \\ (i,j) \neq (p,q)}} \left(c(\mathcal{A}^h_{i,j}) - c(\mathcal{A}^o_{i,j}) \right) \right) + c(\mathcal{A}^h_{p,q}) \geq c(\mathcal{A}^o_{p,q})$$

$$\therefore \left(\sum_{\substack{1 \leq i < j \leq k \\ (i,j) \neq (p,q)}} \left(c(\mathcal{A}^h_{i,j}) - d(S_i, S_j) \right) \right) + c(\mathcal{A}^h_{p,q}) \geq c(\mathcal{A}^o_{p,q})$$

- Let $U = \sum_{1 \leq i < j \leq k} c(\mathcal{A}^h_{i,j})$.
- Let $L = \sum_{1 \leq i < j \leq k} d(S_i, S_j)$.

Carrillo-Lipman bound

$$\because \left(\sum_{\substack{1 \leq i < j \leq k \\ (i,j) \neq (p,q)}} \left(c(\mathcal{A}^h_{i,j}) - d(S_i, S_j) \right) \right) + c(\mathcal{A}^h_{p,q}) \geq c(\mathcal{A}^o_{p,q})$$

$$\therefore U - L - \left(c(\mathcal{A}^h_{p,q}) - d(S_p, S_q) \right) + c(\mathcal{A}^h_{p,q}) \geq c(\mathcal{A}^o_{p,q})$$

$$\Rightarrow U - L + d(S_p, S_q) \geq c(\mathcal{A}^o_{p,q})$$

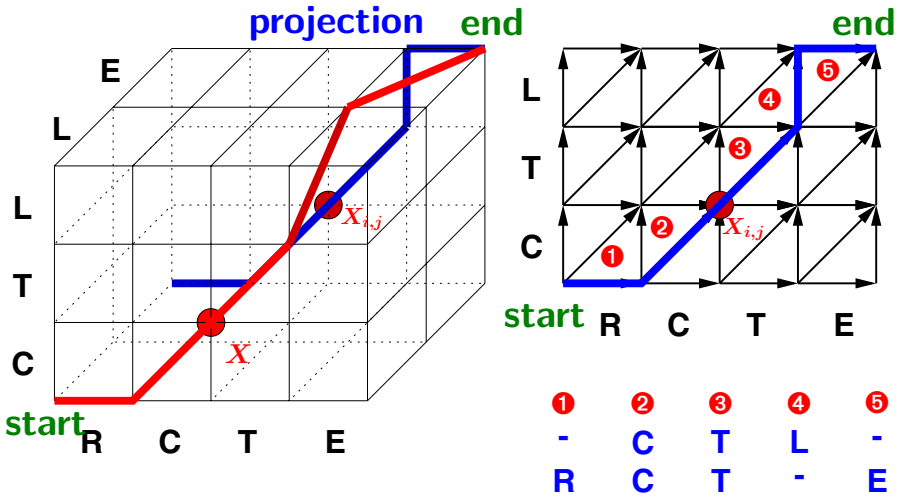
(called Carrillo-Lipman bound, [CL88])

- For an optimal multiple alignment $\mathcal{A}^o$ of S, $c(\mathcal{A}^o_{i,j}) \leq U - L + d(S_i, S_j)$ for all i and j.

Polyhedron

- For an optimal multiple alignment $\mathcal{A}^o$ of S, $c(\mathcal{A}^o_{i,j}) \leq U - L + d(S_i, S_j)$ for all i and j.
- If $\mathcal{A}$ is a candidate of an optimal MSA of S, then $c(\mathcal{A}_{i,j}) \leq U - L + d(S_i, S_j)$ for all i and j.
- Polyhedron: consisting of those nodes that are traversed by least one path which corresponds to an alignment $\mathcal{A}$ such that for all i and j, $c(\mathcal{A}_{i,j}) \leq U - L + d(S_i, S_j)$
- How to determine if a node is in the polyhedron?

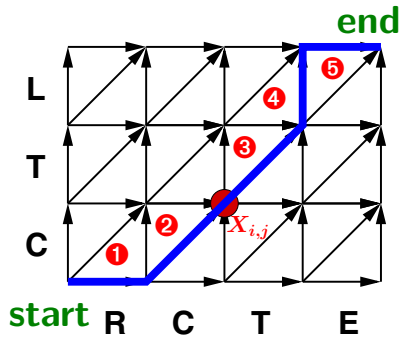
Determine a node of polyhedron



Determine a node of polyhedron

- Let $\mathcal{A}$ be an alignment whose corresponding path P is in the polyhedron.
  - $c(\mathcal{A}_{i,j}) \leq U - L + d(S_i, S_j)$ for any i, j
- Let $X = (x_1, x_2, \cdots, x_k)$ be a node in P.
- Let $X_{i,j} = (x_i, x_j)$ be the projected node of X in the projection plane w.r.t. S_i and S_j.
- For $S = s_1 \cdots s_n$, let $S[i, j] = s_i \cdots s_j$.
- Let $\textit{cut-cost}(x_i, x_j) = d(S_i[1, x_i], S_j[1, x_j])$ $+d(S_i[x_i + 1, n_i], S_j[x_j + 1, n_j])$.

Determine a node of polyhedron



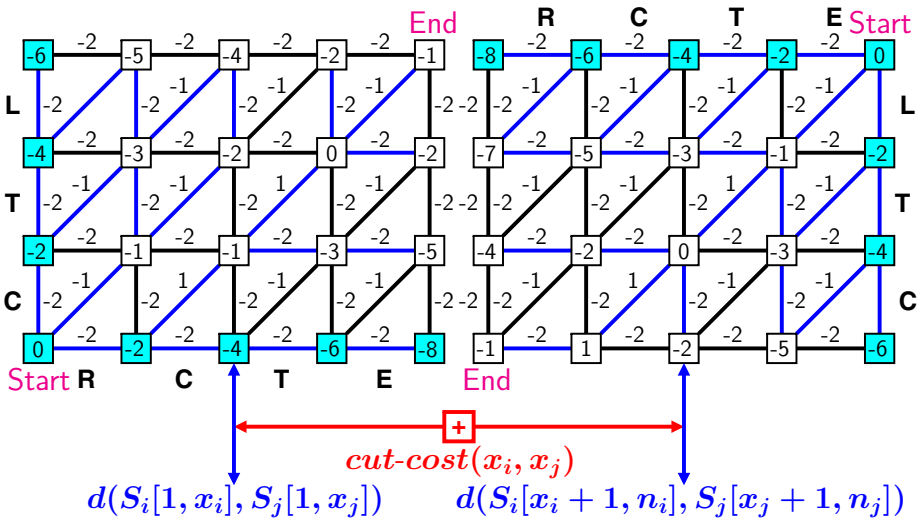
❶	❷	❸	❹	❺
-	C	T	L	-
R	C	T	-	E

$c(\mathcal{A}_{i,j})$

-	C	T	L	-
R	C	T	-	E

$\textit{cut-cost}(x_i, x_j) \leq c(\mathcal{A}_{i,j}) \leq U - L + d(S_i, S_j)$

Computation of $\textit{cut-cost}(x_i, x_j)$



Branch and bound method

- Branch (searching) method: the lexicographic order of the indices of nodes
- Bound rules: eliminate the currently visiting vertex $X = (x_1, x_2, \cdots, x_k)$ if the following condition is satisfied:

  Carrillo-Lipman pruning: violate "for all x_i, x_j, $\textit{cut-cost}(x_i, x_j) \leq U - L + d(S_i, S_j)$"
  - $U = \sum_{1 \leq i < j \leq k} c(\mathcal{A}^h_{i,j})$
  - $L = \sum_{1 \leq i < j \leq k} d(S_i, S_j)$

Sum-of-pairs MSA problem

- NP-complete: by Wang and Jiang [WJ94]; Bonizzoni and Vedova [BV01]
- Branch and bound methods: by Carrillo and Lipman [CL88] (improved by [LAK89, GKS95])
- Approximate methods: (k sequences)
  - $(2 - \frac{2}{k})$-approximation, by Gusfield [Gus93]
  - $(2 - \frac{3}{k})$-approximation, by Pevzner [Pev92]
  - $(2 - \frac{l}{k})$-approximation, where $l < k$ by Bafna, Lawler and Pevzner [BLP97]
- Heuristic methods: Progressive algorithms

2-Approximation algorithm ❶

- Define the scoring function as follows:

$$\sigma(x, y) = \begin{cases} 0, & \text{if } x = y, \\ 1, & \text{otherwise} \end{cases}$$

- $D(S_i, S_j)$: globally aligned distance between S_i and S_j
- Consider the set S of four sequences:

S_1 = ATGCTC
S_2 = AGAGC
S_3 = TTCTG
S_4 = ATTGCATGC

2-Approximation algorithm ❷

S_1 = ATGCTC
S_2 = A-GAGC
$D(S_1, S_2) = 3$

S_2 = AGAGC
S_3 = TTCTG
$D(S_2, S_3) = 5$

S_1 = ATGCTC
S_3 = TT-CTG
$D(S_1, S_3) = 3$

S_2 = A--G-A-GC
S_4 = ATTGCATGC
$D(S_2, S_4) = 4$

S_1 = AT-GC-T-C
S_4 = ATTGCATGC
$D(S_1, S_4) = 3$

S_3 = -TT-C-TG-
S_4 = ATTGCATGC
$D(S_3, S_4) = 4$

2-Approximation algorithm ❸

- Find the center sequence S_i of S which minimizes $\sum_{X \in S \setminus \{S_i\}} D(S_i, X)$.

$$D(S_1, S_2) + D(S_1, S_3) + D(S_1, S_4) = 9$$
$$D(S_2, S_1) + D(S_2, S_3) + D(S_2, S_4) = 12$$
$$D(S_3, S_1) + D(S_3, S_2) + D(S_3, S_4) = 12$$
$$D(S_4, S_1) + D(S_4, S_2) + D(S_4, S_3) = 11$$

Hence, S_1 is the center in this example

2-Approximation algorithm ❹

1. Align S_2 with S_1:

$S_1 =$ ATGCTC
$S_2 =$ A-GAGC

2. Add S_3 by aligning S_3 with S_1:

$S_2 =$ A-GAGC
$S_1 =$ ATGCTC
$S_3 =$ -TTCTG

3. Add S_4 by aligning S_4 with S_1:

$S_3 =$ -T-TC-T-G
$S_2 =$ A--GA-G-C
$S_1 =$ AT-GC-T-C
$S_4 =$ ATTGCATGC

2-Approximation algorithm ❺

- $d(S_i, S_j)$ ($d^\star(S_i, S_j)$): the distance between S_i and S_j induced by this approximation (an optimal) algorithm
- $d(S_i, S_j) + d(S_i, S_k) \geq d(S_j, S_k)$ (triangle inequality)
- $App = \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} d(S_i, S_j)$
- $Opt = \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} d^\star(S_i, S_j)$

2-Approximation algorithm ❻

- Claim that $App \leq 2Opt$
- Since triangle inequality property of $d(S_i, S_j)$ and $d(S_1, S_i) = d(S_i, S_1)$, we have

$$\begin{aligned} App &= \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} d(S_i, S_j) \\ &\leq \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} (d(S_i, S_1) + d(S_1, S_j)) \\ &= (k-1)\sum_{i=2}^{k} d(S_1, S_i) \end{aligned}$$

2-Approximation algorithm ❼

- Since $d(S_1, S_i) = D(S_1, S_i)$ for all i, we have

$$App \leq (k-1)\sum_{i=2}^{k} D(S_1, S_i)$$

- Note that $Opt = \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} d^\star(S_i, S_j)$
- Since $D(S_i, S_j) \leq d^\star(S_i, S_j)$, we have

$$Opt = \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} d^\star(S_i, S_j) \geq \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} D(S_i, S_j)$$

2-Approximation algorithm ⑧

- Since S_1 is the center, we have

$$Opt \geq \frac{1}{2}\sum_{i=1}^{k}\sum_{\substack{j=1\\ j\neq i}}^{k} D(S_i, S_j)$$

$$\geq \frac{1}{2}\sum_{i=1}^{k}\sum_{j=2}^{k} D(S_1, S_j) = \frac{k}{2}\sum_{j=2}^{k} D(S_1, S_j)$$

- $\frac{App}{Opt} \leq \frac{2(k-1)}{k} \Leftrightarrow App \leq (2-\frac{2}{k})Opt < 2 \cdot Opt$

Sum-of-pairs MSA problem

- NP-complete: by Wang and Jiang [WJ94]; Bonizzoni and Vedova [BV01]
- Branch and bound methods: by Carrillo and Lipman [CL88] (improved by [LAK89, GKS95])
- Approximate methods: (k sequences)
  - $(2-\frac{2}{k})$-approximation, by Gusfield [Gus93]
  - $(2-\frac{3}{k})$-approximation, by Pevzner [Pev92]
  - $(2-\frac{l}{k})$-approximation, where $l < k$ by Bafna, Lawler and Pevzner [BLP97]
- Heuristic methods: Progressive algorithms

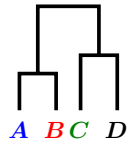
Progressive MSA

1. Construct the guide (evolutionary) tree

(1) Four sequences

	A	B	C	D
A	0	3	5	5
B		0	5	5
C			0	4
D				0

(2) Distance matrix

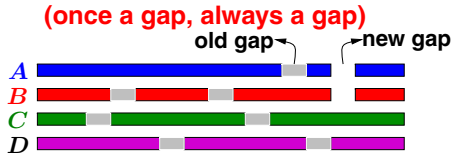


(3) Evolutionary tree

2. Progressively align sequences by the tree

(4) Align most similar pair

(5) Align next similar pair



(6) Align two pre-aligned groups

Progressive MSA

1. Compute the distance matrix by aligning all pairs of sequences (using dynamic programming algorithm)
2. Compute the guide tree from the distance matrix:
   - PILEUP of GCG: UPGMA (Unweighted Pair-Group Method using Arithmetic mean)
   - CLUSTAL W: NJ (Neighbor-Joining)
   - YAMA-MST of Tang's lab: Kruskal MST
3. Progressively align the sequences according to the branching order in the guide tree (once a gap, always a gap)

1. Computation of distance matrix

S_1: RCTLEE
S_2: RCLEE
S_3: CTLEE
S_4: CTEE

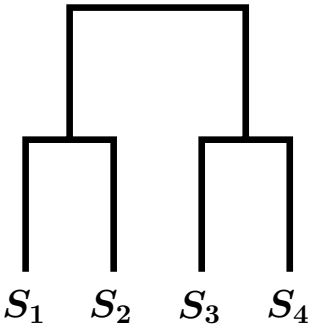
	S_1	S_2	S_3	S_4
S_1	0	1	1	2
S_2		0	2	1
S_3			0	1
S_4				0

4 Sequences ⇨ Distance Matrix

2. Construction of guide tree

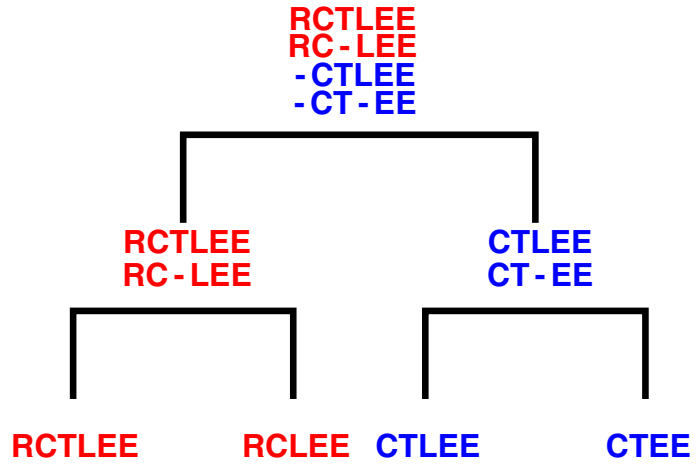
	S_1	S_2	S_3	S_4
S_1	0	1	1	2
S_2		0	2	1
S_3			0	1
S_4				0

Distance Matrix ⇨



Evolutinary Tree

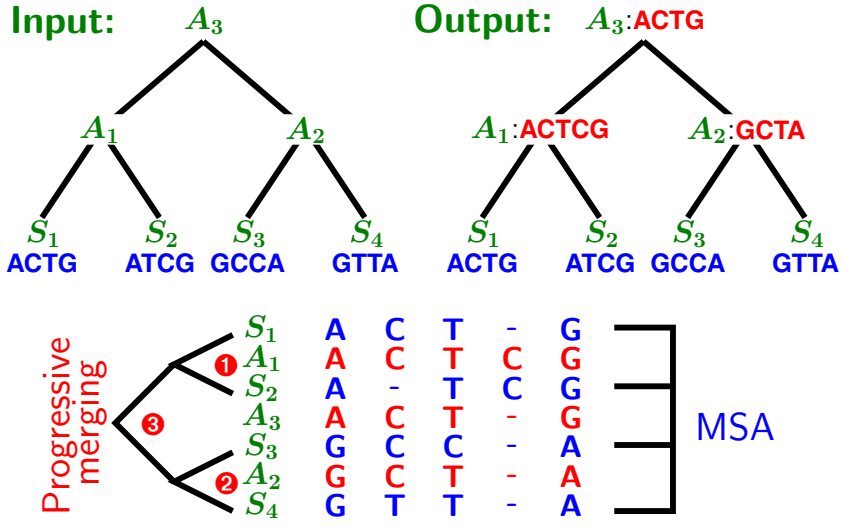
3. Progressive alignment



Tree alignment problem

- **Input**: k sequences and a rooted phylogenetic tree containing k leaves, each of which is labeled with a unique given sequence
- **Output**: Construct a sequence for each internal node of the tree such that the total cost of its edges is minimized, where the cost of an edge is the edit distance between two sequences associated with both ends of the edge
- The given tree represents the evolutionary history for the leaf sequences of extant species, and its internal nodes represent the extinct ancestral species whose sequences are to be determined.

Tree alignment: example

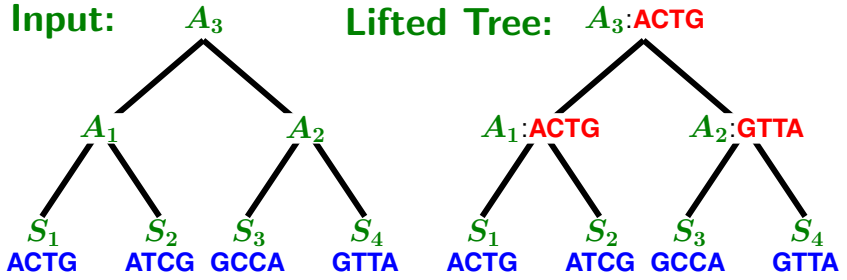


Tree alignment: complexity

- NP-complete: Wang and Jiang, [WJ94]
- 2-approximation: Wang, Jiang and Lawler, [WJL96]
- PTAS: Wang, Jiang and Lawler, [WJL96] (further improved by [WG97, WJG00])
- Polynomial Time Approximation Scheme: a family of algorithms $\{\mathcal{A}_\varepsilon : \varepsilon > 0\}$ each of which takes as **input** both an instance I and an error bound ε and in polynomial time, it outputs an approximate solution $\mathcal{A}(I)$ whose **performance ratio** is $R_{\mathcal{A}}(I, \varepsilon) = \max\{\frac{\mathcal{A}(I)}{OPT(I)}, \frac{OPT(I)}{\mathcal{A}(I)}\} \leq 1 + \varepsilon$

Tree alignment: 2-approximation

- Lifted tree: an evolutionary tree in which the label of each internal node equals to the label of some one of its children



Tree alignment: 2-approximation

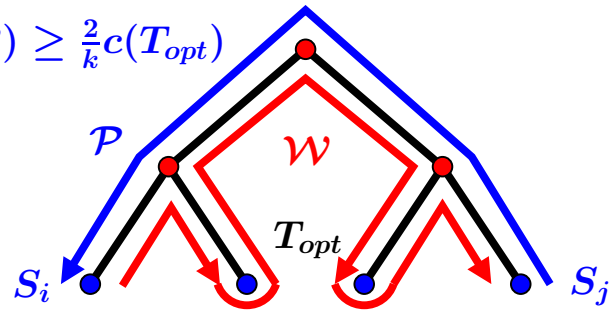
- **Theorem**: There exists a lifted tree with cost at most $2(1 - \frac{1}{k})c(T_{opt})$.
  - k: the number of leaves of the given tree T
  - T_{opt}: an optimal tree with cost $c(T_{opt})$
- Closest descendant leaf $l(v)$ of $v \in T_{opt}$: a descendant leaf of v s.t. the path from v to $l(v)$ is the shortest among all descendant leaves of v
- T_l: a lifted tree of T obtained by assigning the sequence $sl(v)$ of $l(v)$ to each internal node v
- **Lemma**: $c(T_l) \leq 2(1 - \frac{1}{k})c(T_{opt})$

$c(T_l) \le 2(1-\frac{1}{k})c(T_{opt})$ ❶

- $\mathcal{P}$: the longest path among all paths between any two leaves S_i and S_j in T_{opt}
- $\mathcal{W}$: the walk obtained by removing $\mathcal{P}$ from an Euler tour of T_{opt}

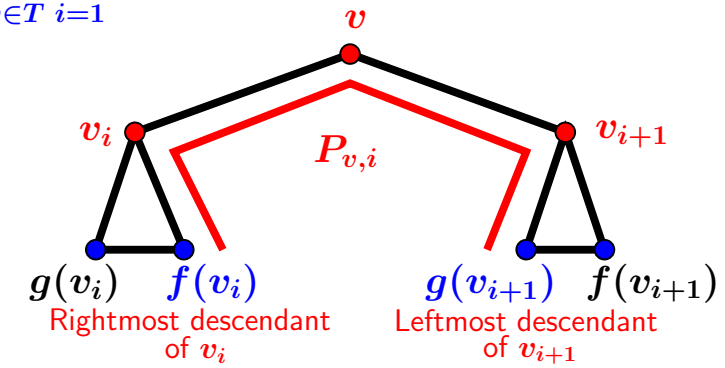
$\therefore c(\mathcal{W}) \le 2(1-\frac{1}{k})c(T_{opt})$

$\because c(\mathcal{P}) \ge \frac{2}{k}c(T_{opt})$



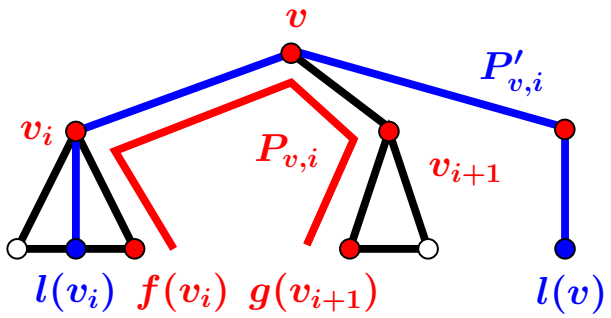
$c(T_l) \le 2(1-\frac{1}{k})c(T_{opt})$ ❷

- $P_{v,i}$: path $f(v_i) \to v_i \to v \to v_{i+1} \to g(v_{i+1})$
- $\sum_{v \in T} \sum_{i=1}^{d-1} c(P_{v,i}) = c(\mathcal{W})$ (d: # of v's children)



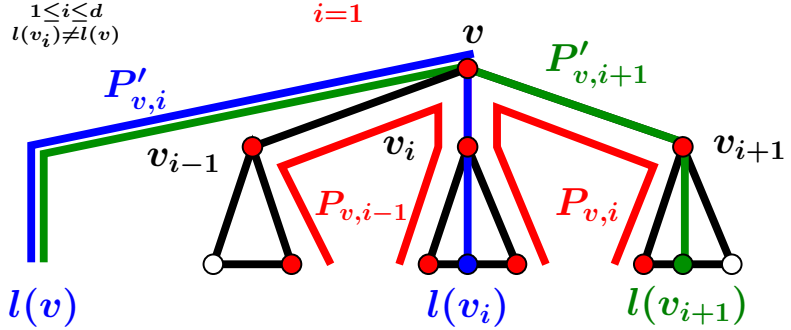
$c(T_l) \le 2(1-\frac{1}{k})c(T_{opt})$ ❸

- $P'_{v,i}$: path $l(v) \to v \to v_i \to l(v_i)$
- $c(P'_{v,i}) \le c(P_{v,i})$ and $c(P'_{v,i+1}) \le c(P_{v,i})$



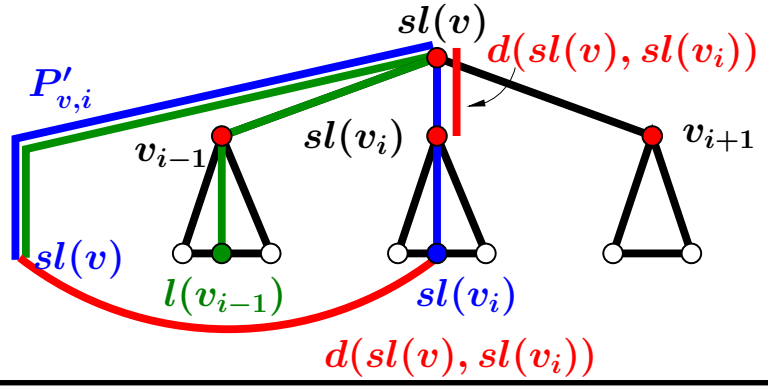
$c(T_l) \le 2(1-\frac{1}{k})c(T_{opt})$ ❹

- $c(P'_{v,i}) \le c(P_{v,i-1})$ and $c(P'_{v,i+1}) \le c(P_{v,i})$
- $\sum_{\substack{1 \le i \le d \\ l(v_i) \ne l(v)}} c(P'_{v,i}) \le \sum_{i=1}^{d-1} c(P_{v,i})$



$c(T_l) \leq 2(1-\frac{1}{k})c(T_{opt})$ ❺

- The cost of edge $(v, v_i) = d(sl(v), sl(v_i))$, where $d(sl(v), sl(v_i)) = 0$ if $l(v) = l(v_i)$
- By triangle inequality, $d(sl(v), sl(v_i)) \leq c(P'_{v,i})$



$c(T_l) \leq 2(1-\frac{1}{k})c(T_{opt})$ ❻

- The cost of T_l is as follows:

$$\begin{aligned} c(T_l) &= \sum_{v \in T} \sum_{i=1}^{d} d(sl(v), sl(v_i)) \\ &\leq \sum_{v \in T} \sum_{\substack{1 \leq i \leq d \\ l(v_i) \neq l(v)}} c(P'_{v,i}) \\ &\leq \sum_{v \in T} \sum_{i=1}^{d-1} c(P_{v,i}) \\ &= c(\mathcal{W}) \leq 2(1-\tfrac{1}{k})c(T_{opt}) \end{aligned}$$

Tree alignment: 2-approximation

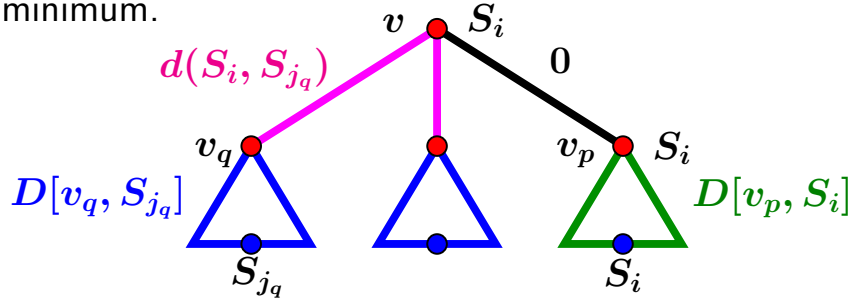
- Computing T_l is hard since it is derived from T_{opt} and the computation of T_{opt} is NP-complete.
- $T^\star$: an optimal one among all the lifted trees of T.
- Clearly, $c(T^\star) \leq c(T_l) \leq 2(1-\frac{1}{k})c(T_{opt})$
- How to compute $T^\star$? (By dynamic programming)
  - T_v: the induced subtree of T rooted at v
  - $S(v)$: the set of leaf sequences of T_v
  - $D[v, S_i]$: the cost of an optimal lifted tree of T_v with v being assigned $S_i \in S(v)$
  - $c(T^\star) = \min\{D[root, S_i] : 1 \leq i \leq k\}$

How to compute $D[v, S_i]$?

Let v be an internal node of T with $v_1, \cdots, v_d$ children, and let $S_i \in S(v_p)$, $1 \leq p \leq d$. Then

$$D[v, S_i] = D[v_p, S_i] + \sum_{\substack{1 \leq q \leq d \\ q \neq p}} (D[v_q, S_{j_q}] + d(S_i, S_{j_q}))$$

where S_{j_q} is in $S(v_q)$ s.t. $D[v_q, S_{j_q}] + d(S_i, S_{j_q})$ is minimum.



[LAK89] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences USA*, 86:4412–4415, 1989.

[Pev92] P. A. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics*, 52(6):1763–1779, 1992.

[WG97] L. Wang and D. Gusfield. Improved approximation algorithms for tree alignment. *Journal of Algorithms*, 25:255–273, 1997.

[WJ94] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

[WJG00] L. Wang, T. Jiang, and D. Gusfield. A more efficient approximation scheme for tree alignment. *SIAM Journal on Computing*, 30(1):283–299, 2000.

[WJL96] L. Wang, T. Jiang, and E. L. Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16(3):302–315, 1996.

References

[BLP97] V. Bafna, E. L. Lawler, and P. A. Pevzner. Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science*, 182(1-2):233–244, 1997.

[BV01] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with SP-score that is a metric. *Theoretical Computer Science*, 259(1–2):63–79, 2001.

[CL88] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48:1073–1082, 1988.

[GKS95] S. K. Gupta, J. Kececioglu, and A. A. Schäffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2:459–472, 1995.

[Gus93] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.