Reflections|Projections 2020

Largest student run tech conference in the midwest, taking place September 21 through September 26
Virtual for the first time in 2020; featuring 20+ speakers and 5+ companies

Opportunity to hear about the bleeding edge on topics including

- Computer science for clinical methods (Tuesday @ 5)
- Ethical artificial intelligence (Wednesday @ 4, Friday @ 7)
- How to be successful in a startup (Monday @ 7, Wednesday @ 5)
- The essentials of product management (Thursday @ 5)
- Protecting privacy at a global scale (Friday @ 4)
- How to build accessible applications (Friday @ 6)
- And more! Check out our social media or website for more details

Network with top companies like Goldman Sachs, Grainger, IMC, Aechelon
Participate in a beginner-friendly, team-based AI competition with prizes
Swag pickup for attendees on campus
Opportunity to win free food!

Why wait?
Register today
at
acmrp.org

Reflections | Projections

Mark Ciaccio

Senior Data Scientist @ AbbVie

Tuesday, September 22 from 6-7PM

reflectionsprojections.org

CS 466
Introduction to Bioinformatics
Lecture 8

Mohammed El-Kebir

September 18, 2020

Course Announcements

Instructor:

- Mohammed El-Kebir (melkebir)
- Office hours: Wednesdays, 3:15-4:15pm

TAs:

- Sarah Christensen (sac2) – Mondays, 3-4pm
- Wesley Wei Qian (weiqian3) – Fridays, 9-10am

Homework 2 released on 9/23 and due 10/1 by 11:59pm (CT)

Midterm will be 10/7, 7-10pm (CT)
Conflict midterm will be 10/8, 8-11am (CT).
You will need to sign up for conflict on Piazza.

Outline

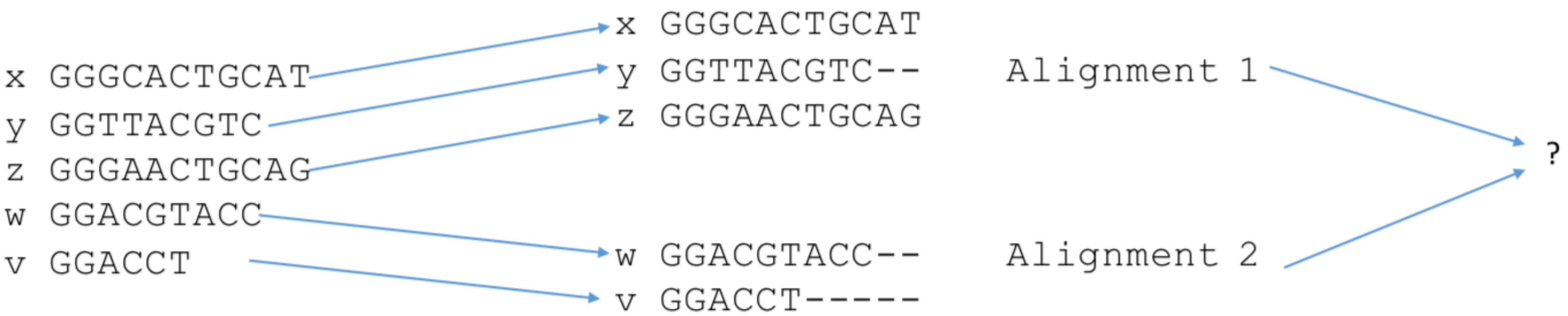
- Progressive alignment | heuristics → no guarantee on opt., or low bnd.
  - Current methods
- Tree and star alignment | approximation algs.

Reading:

- Material based on Chapter 14.6 in book "Algorithms on Strings, Trees and Sequences" by Dan Gusfield
- Lecture notes

Heuristic: Iterative/Progressive Alignment

Iteratively add strings (or alignments) to existing alignment(s).



```
x GGGCACTGCAT
y GGTTACGTC
z GGGAACTGCAG
w GGACGTACC
v GGACCT
```

```
x GGGCACTGCAT
y GGTTACGTC--    Alignment 1
z GGGAACTGCAG
```

```
w GGACGTACC--    Alignment 2
v GGACCT-----
```

Issues:

1. How to merge alignments?
2. What order to use in merging strings/alignments?

Profile Representation of Multiple Alignment

loss of info

-	A	G	G	C	T	A	T	C	A	C	C	T	G
T	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	T	C	A	C	-	G	G
C	A	G	-	C	T	A	T	C	G	C	-	G	G

A		1					1			.8				
C	.6				1			.4	1		.6	.2		
G			1	.2						.2			.4	1
T	.2					1		.6					.2	
-	.2			.8							.4	.8	.4	

A **profile** $P = [p_{i,j}]$ is a $(|\Sigma| + 1) \times l$ matrix,
where $p_{i,j}$ is the frequency of i-th letter in j-th position of alignment

Aligning String to Profile

profile

$$\tau(x, j) = \sum_{y \in \Sigma \cup \{-\}} p_{y,j} \cdot \delta(x, y)$$

'weighted average'

column of profile

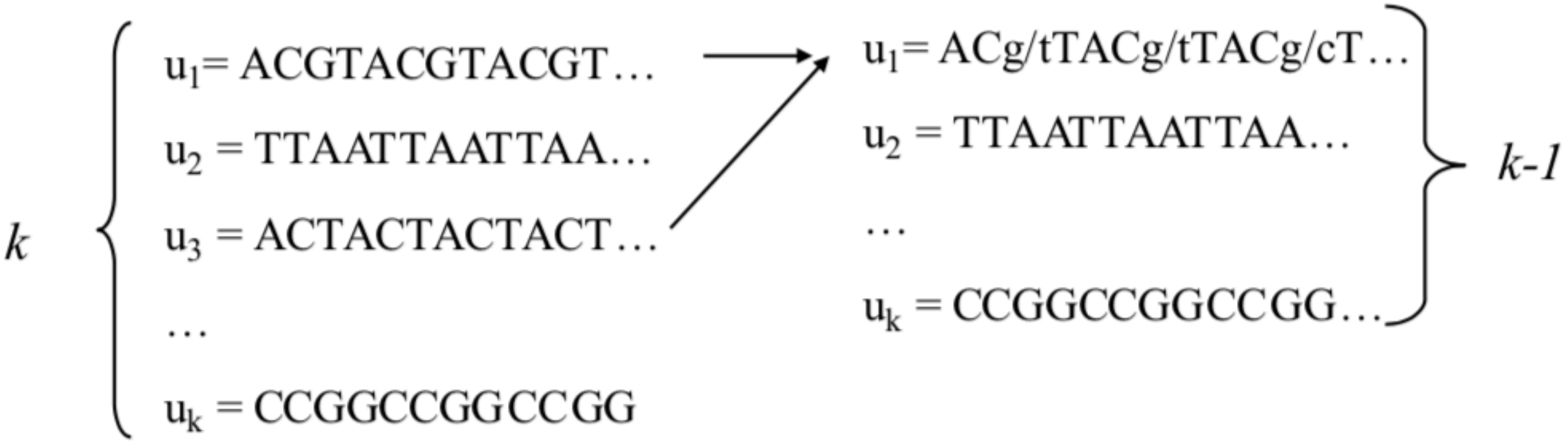
$$s[i, j] = \max \begin{cases} 0, & \text{if } i = 0 \text{ and } j = 0, \\ s[i-1, j] + \delta(v_i, -), & \text{if } i > 0, \quad \text{Insert space in profile} \\ s[i, j-1] + \tau(-, j), & \text{if } j > 0, \quad \text{Insert space in string} \\ s[i-1, j-1] + \tau(v_i, j), & \text{if } i > 0 \text{ and } j > 0. \end{cases}$$

~~column~~ index of string

- $s[i, j]$ is optimal alignment of $v_1, \ldots, v_i$ and first j columns of P
- $\delta(x, y)$ is score for aligning characters x and y
- $\tau(x, j)$ is score for aligning character x and column j of P

Progressive Multiple Alignment: Greedy Algorithm

Choose most similar pair among ***k*** input strings, combine into a profile. This reduces the original problem to alignment of ***k-1*** sequences to a profile. Repeat.



k

u_1= ACGTACGTACGT…

u_2 = TTAATTAATTAA…

u_3 = ACTACTACTACT…

…

u_k = CCGGCCGGCCGG

u_1= ACg/tTACg/tTACg/cT…

u_2 = TTAATTAATTAA…

…

u_k = CCGGCCGGCCGG…

k-1

Outline

- Progressive alignment
  - Current methods
- Tree and star alignment

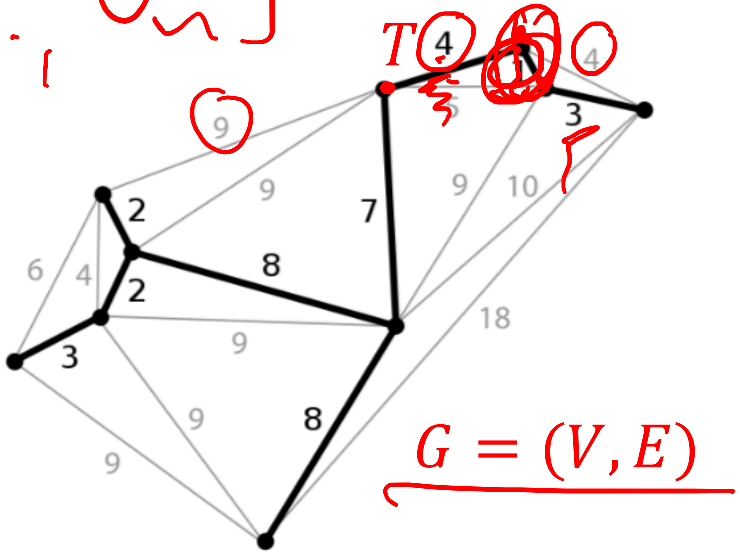
Reading:

- Material based on Chapter 14.6 in book “Algorithms on Strings, Trees and Sequences” by Dan Gusfield
- Lecture notes

Progressive Alignment – Feng and Doolittle (1987)

1. Compute pairwise sequence alignments of n sequences
2. Generate complete graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$
3. Compute a (rooted) minimum spanning tree T of G → guide tree
4. Perform sequence-sequence, sequence-alignment and alignment-alignment alignment to construct MSA according to guide tree T (from most similar to least similar)

$V = \{S_1, \dots, S_n\}$



Minimum spanning tree is a tree T spanning all vertices of G with minimum total weight

‘Once a gap, always a gap’

Progressive Alignment – ClustalW (1994)

weighted

- Widely used alignment method by Thompson, Higgins and Gibson (1994)
- W stands for weighted:
  - Input sequences are weighted to compensate for biased representation
  - Different substitution matrices depending on expected similarity in guide tree (BLOSUM80 for closely related sequences, and BLOSUM50 for distant sequences)
  - Position-specific gap-open and gap-extend penalties depending on context (hydrophobic vs. hydrophilic)

affine gap penalties

Three steps:

1. Construct pairwise alignments
2. Build guide tree T using neighbor joining*
3. Progressive profile alignment guided by T

ClustalW – Step 2: Guide Tree

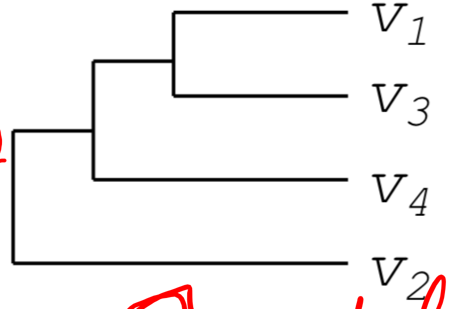
Create Guide Tree using the similarity matrix

("cluster" distances. Details to come...)

distance matrix

	v_1	v_2	v_3	v_4
v_1	–			
v_2	.17	–		
v_3	.87	.28	–	
v_4	.59	.33	.62	–

neighbor joining



additive tree

ClustalW uses the neighbor-joining method

Guide tree roughly reflects evolutionary relationships

Calculate:

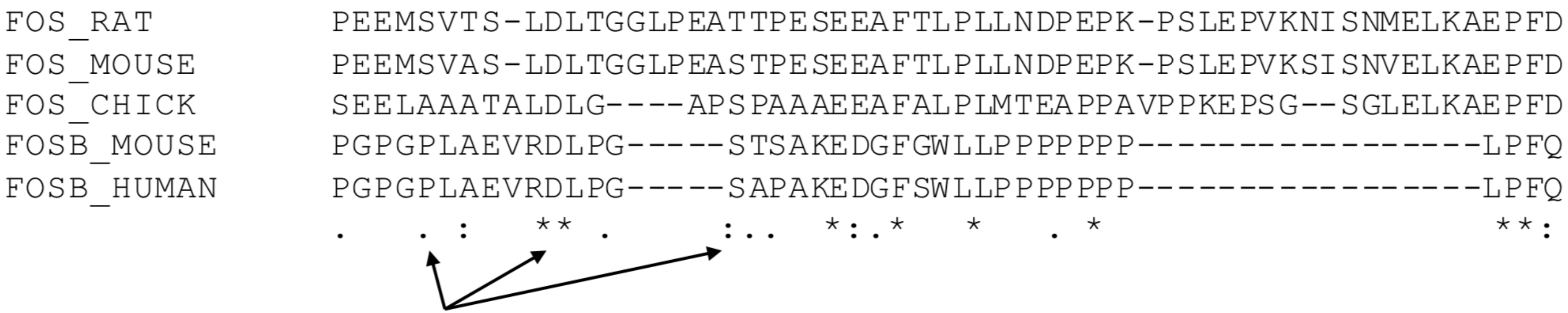
$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment$((v_{1,3}),v_4)$

$v_{1,2,3,4}$ = alignment$((v_{1,3,4}),v_2)$

ClustalW – Step 3: Progressive Alignment

- Start by aligning the two most similar sequences
- Following the guide tree, add in the next sequences, aligning to the existing alignment
- Insert gaps as necessary



```
FOS_RAT         PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE       PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK       SEELAAATALDLG----APSPAAAEEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE      PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----------------LPFQ
FOSB_HUMAN      PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----------------LPFQ
                .   . :   ** .     :..  *:.*   *   . *                   **:
```

Dots and stars show how well-conserved a column is.

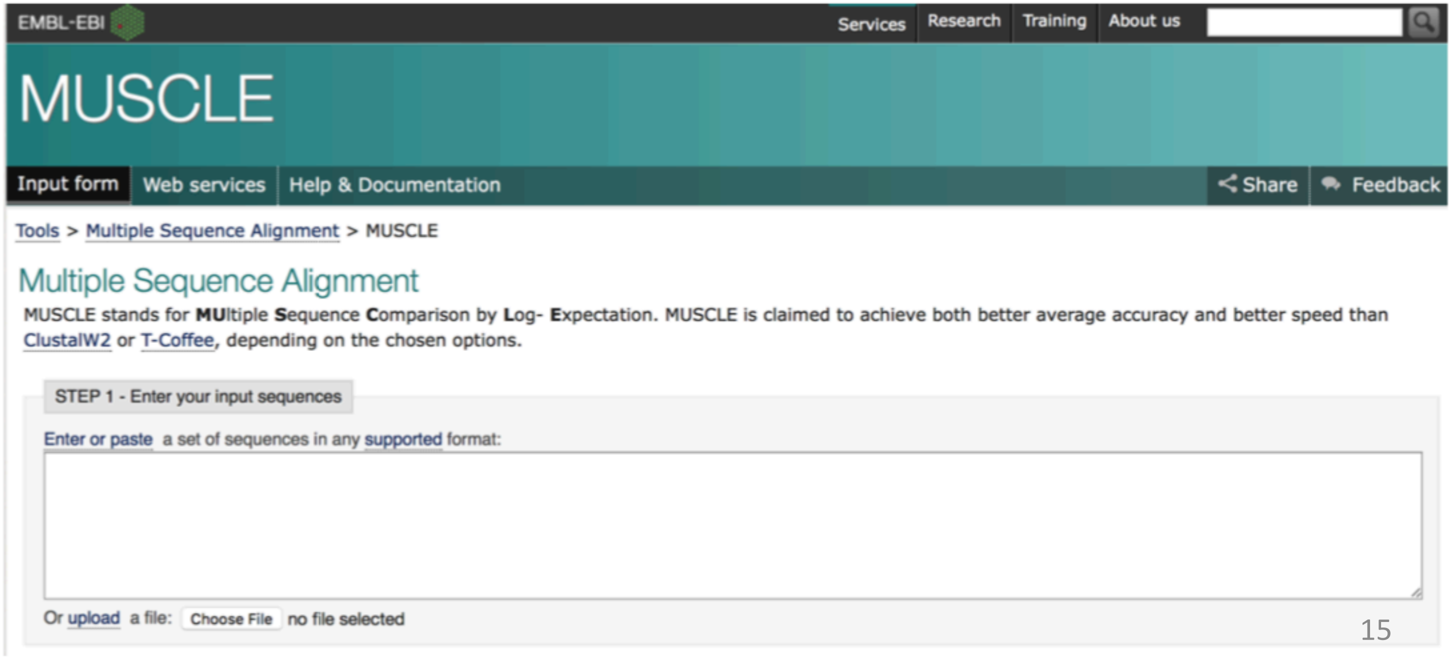
MUSCLE (Edgar, 2004)

(once a gap, always a gap)

Multiple Sequence Comparison by Log-Expectation

Three phases:

1. Draft progressive alignment: fast heuristic
2. Improved progressive: use tree derived in phase 1
3. Refinement of MSA
    - Remove sequence from MSA and realign to profile of remaining sequences
    - Repeat until convergence



Tools > Multiple Sequence Alignment > MUSCLE

Multiple Sequence Alignment

MUSCLE stands for **MU**ltiple **S**equence **C**omparison by **L**og- **E**xpectation. MUSCLE is claimed to achieve both better average accuracy and better speed than ClustalW2 or T-Coffee, depending on the chosen options.

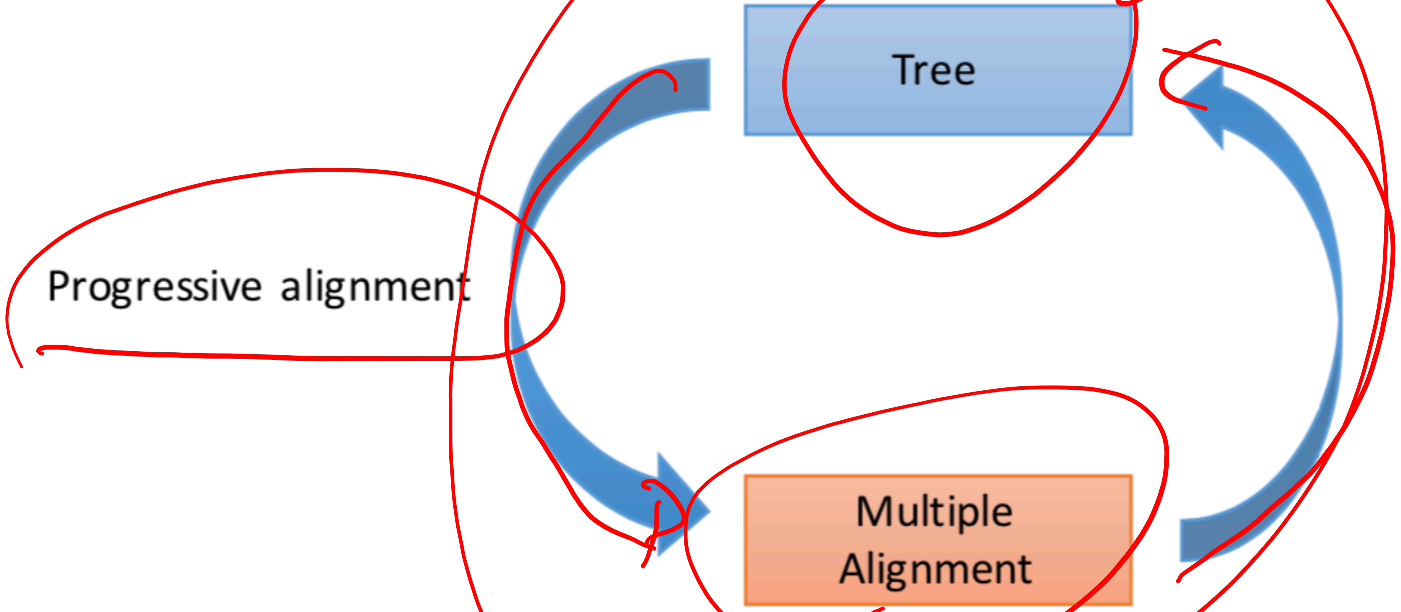
STEP 1 - Enter your input sequences

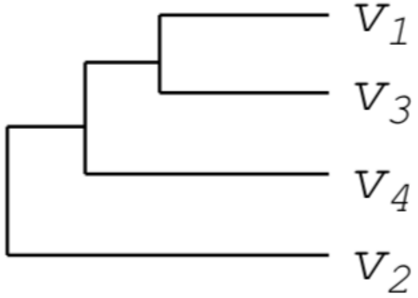
Enter or paste a set of sequences in any supported format:

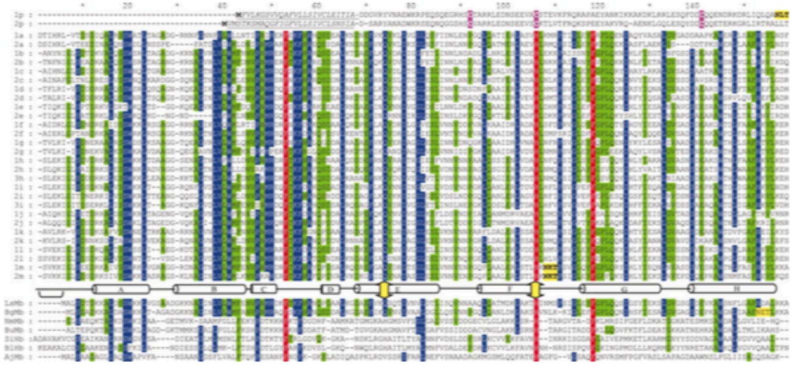
Or upload a file: Choose File no file selected

Progressive MSA

phylogenetic tree







Circularity!

Ideally, want to derive alignment and tree simultaneously → Hard

Outline

- Progressive alignment } messy, heuristics
  - Current methods
- Tree and star alignment ← "clean"

Reading:

- Material based on Chapter 14.6 in book "Algorithms on Strings, Trees and Sequences" by Dan Gusfield
- Lecture notes

Multiple Sequence Alignment Problem w/ SP-Score

MSA-SP problem: Given strings $\mathbf{v}_1, \dots, \mathbf{v}_k$ and scoring function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, find multiple sequence alignment $\mathcal{M}^*$ with **maximum** value of SP-score$(\mathcal{M}^*) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} S(\mathbf{v}_i, \mathbf{v}_j)$ where $S(\mathbf{v}_i, \mathbf{v}_j)$ is the score of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{M}^*$ using δ

Weighted SP-Edit Distance problem: Given strings $\mathbf{v}_1, \dots, \mathbf{v}_k$ and cost function $\delta : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathbb{R}$, find multiple sequence alignment $\mathcal{M}^*$ with **minimum** value of SP-score$(\mathcal{M}^*) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} S(\mathbf{v}_i, \mathbf{v}_j)$ where $S(\mathbf{v}_i, \mathbf{v}_j)$ is the cost of the induced pairwise alignment of $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{M}^*$ using δ

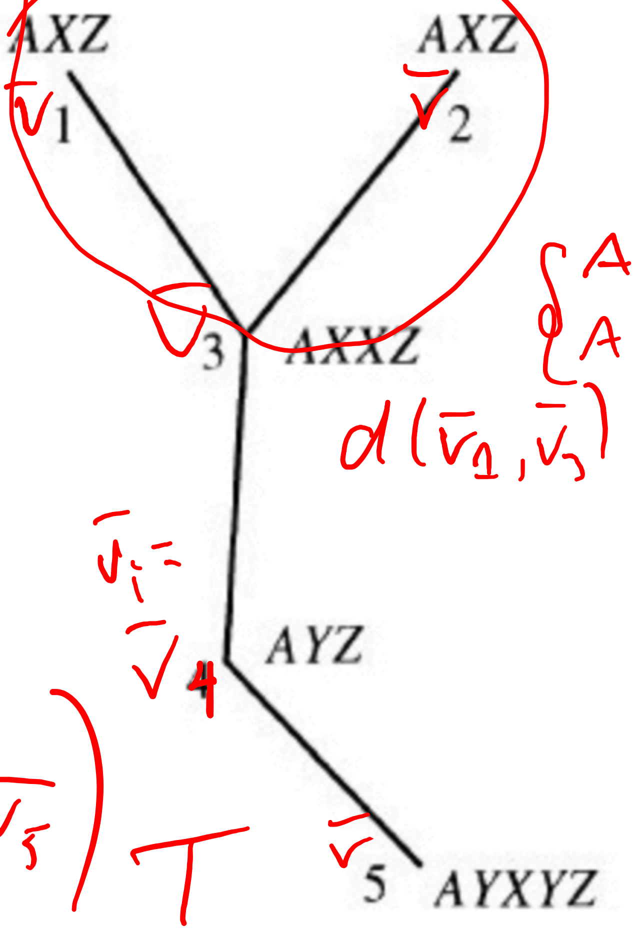
Tree Alignment

Input: $\bar{v}_1, \dots, \bar{v}_k$

$D(\bar{v}_1, \bar{v}_3) = 1$

$D(\bar{v}_i, \bar{v}_j)$ = opt/min cost of aligning $\bar{v}_i$ and $\bar{v}_j$

Let T be a tree with k vertices labeled uniquely by $\{\bar{v}_1, \dots, \bar{v}_k\}$.



$(((\bar{v}_1, \bar{v}_3)\bar{v}_2), \bar{v}_4), \bar{v}_5)$ T

A X X Z
A X - Z

$d(\bar{v}_1, \bar{v}_3) = 1$

b)

3	A	X	X	_	Z
1	A	X	_	_	Z
2	A	_	X	_	Z
4	A	Y	_	_	Z
5	A	Y	X	X	Z

A

A multiple alignment A of $\bar{v}_1, \dots, \bar{v}_k$ is consistent with T if induced pairwise alignment of $\bar{v}_i$ and $\bar{v}_j$ has cost $D(\bar{v}_i, \bar{v}_j)$ for all edges (v_i, v_j) of T

Figure 14.6: a. A tree with its nodes labeled by a (multi)set of strings, b. A multiple alignment of those strings that is consistent with the tree. The pairwise scoring scheme scores a zero for each match and a one for each mismatch or space opposite a character. The reader can verify that each of the four induced alignments specified by an edge of the tree has a score equal to its respective optimal distance. However, the induced alignment of two strings which do not label adjacent nodes may have a score greater than their optimal pairwise distance.

Thm 1 (Gusfield). Let T be a tree whose nodes are labeled uniquely by $\bar{v}_1, ---, \bar{v}_k$. max length is n We can compute an alignment A(T) of $\bar{v}_1, ---, \bar{v}_k$ that is consistent with T in $O(k^2n^2)$ time.

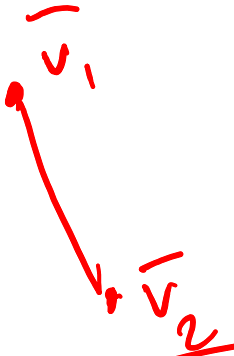
Proof. WLOG, assume that vertices $\bar{v}_1, ---, \bar{v}_k$ are arranged s.t. $\bar{v}_1, \dots, \bar{v}_i$ induces a connected subtree of T.

Prove inductively that adding each string $\bar{v}_i$ while maintaining consistency can be done in $O(in^2)$ time.

$$\sum_{i=1}^{k} O(in^2) = O(n^2) \sum_{i=1}^{k} O(i) = O(n^2) \cdot O(k^2)$$
$$= O(k^2n^2).$$

Base case: $i=2$. $\bar{v}_1$ and $\bar{v}_2$, compute pairwise alignment in $\boxed{O(n^2)}$ time. By definition this alignment will be consistent with

$T[\{\bar{v}_1, \bar{v}_2\}]$



$\boxed{\checkmark O(i n^2)}$

} resulting alignment is consistent with $T[\{\bar{v}_1, \bar{v}_2\}]$ in time $O(n^2)$.

Step: $i \geq 2$. IH: We are given tree T' and $A(T')$ that is consistent with strings $\bar{v}_1, \dots, \bar{v}_{i-1}$.

Consider $\bar{v}_i$. By definition, there is a vertex among $\{\bar{v}_1, \dots, \bar{v}_{i-1}\}$ adjacent to $\bar{v}_i$. Let's call this vertex $\bar{v}_j$. Extract gapped seq. $\tilde{v}_j$ corresponding to $\bar{v}_j$. We align $\tilde{v}_j$ and $\bar{v}_i$ with the added rule that $\boxed{\delta(-,-)=0}$, i.e. Two opposing gaps have a cost of 0.

Let $\tilde{v}_j'$ and $\tilde{v}_i$ be the two resulting gapped sequences. If this new alignment did not insert new gaps into $\tilde{v}_j$ then we add $\tilde{v}_i$ to $A(T')$ $(\tilde{v}_j' = \tilde{v}_j)$. The result is a multiple alignment with one more string, induced cost of $\tilde{v}_j' = \tilde{v}_j$ and $\tilde{v}_i$ $\boxed{\text{equals } D(\bar{v}_j, \bar{v}_i)}$

However, if opt. alignment of $\tilde{v}_j$ and $\bar{v}_i$ inserted a gap in $\tilde{v}_j$ (i.e. $\tilde{v}_j' \neq \tilde{v}_j$), say b/w characters l and $l+1$.

We insert a gap b/w columns l and $l+1$ in each sequence of the previous mult. alignment $A(T')$. Observe that the induced costs of edges of T' remain unchanged.

$\bar{v}_1, \ldots, \bar{v}_{i-1}$.

On the other hand cost of induced pairwize alignment of $\bar{v}_j$ and $\bar{v}_i$ equals $D(\bar{v}_j, \bar{v}_i)$.

In both cases, the new alignment is consistent with the tree T' extended with vertex $\bar{v}_i$.

For the running time, observe that $A(T')$ is composed of $(i-1)$ gapped seqs. each of length at most $(i-1)n$. Worst case $\tilde{v}_j$ has length $(i-1)n = \underline{\underline{O(in)}}$ while $\bar{v}_i$ has length $\overset{\le n}{\underline{O(n)}}$.

We can compute alignment b/w $\tilde{v}_j$ and $\underline{\bar{v}_i}$ in $O(in^2)$ time

Input $\bar{v}_1, \dots, \bar{v}_k \in \Sigma^*$; $\boxed{\delta} : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \mapsto \mathbb{R}$

Goal: Find multiple alignment A with SP score SP(A) that comes with approximation guarantee

$$\begin{tikzcd} A \arrow[d] & A^* \\ SP(A) \geq SP(A^*) & \end{tikzcd}$$

$$\frac{SP(A) = 10}{SP(A^*) = 5} \leq 2$$

$$\boxed{\frac{SP(A)}{SP(A^*)} \leq \underset{?}{2}}$$

Def 2. A cost function $\underline{\delta}$ satisfies the triangle inequality if

$$\underset{1}{\delta(x,z)} \underset{\le}{\le} \underset{1}{\delta(x,y)} \underset{+}{+} \underset{1}{\delta(y,z)}$$

$$\forall\, \underline{x,y,z} \in \Sigma \cup \{-\}$$

	x	y	z
x	0	1	1
y	1	0	1
z	1	1	0

classical edit dist.

$\Rightarrow x = y = z$

$$\delta(x,z) \leq \delta(x,y) + \delta(y,z)$$

$$\delta(x,x) \leq \delta(x,x) + \delta(x,x)$$

$$0 \leq \boxed{0} + \boxed{0}$$

$D(\bar{v}_i, \bar{v}_j)$ opt cost of aligning $\bar{v}_i$ and $\bar{v}_j$ (in isolation).

Given a mult. alignment A

$d(\bar{v}_i(A), \bar{v}_j(A))$ cost of induced by A pairwise alignment of $\bar{v}_i$ and $\bar{v}_j$

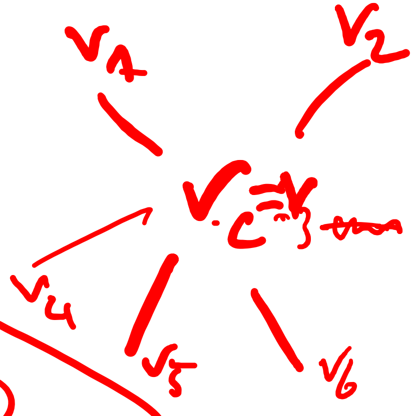
$$D(\bar{v}_i, \bar{v}_j) \le d(\bar{v}_i(A), \bar{v}_j(A))$$

$\forall$ mult. A, $\bar{v}_i$, $\bar{v}_j$.

Def. 3 Given strings $\bar{v}_1, \dots, \bar{v}_k$, the center string $\bar{v}_c$ is the input string ($c \in [k]$) that minimizes

$$\sum_{i=1}^{k} D(\bar{v}_c, \bar{v}_i)$$

$$D(\bar{v}_c, \bar{v}_c) = 0$$



$c = 3$

$k = 6$

The center star is a star tree of k nodes with center node labeled by $\bar{v}_c$ and each of remaining $k-1$ nodes labeled by $\{\bar{v}_1, \dots, \bar{v}_k\} \setminus \{\bar{v}_c\}$.

Construct alignment A_c of $\bar{v}_1, \dots, \bar{v}_k$ consistent with center star. T_c.

① Identify $\bar{v}_c$

- compute all pairwise alignments $O(k^2 n^2)$ time
- identify $\bar{v}_c$ $O(k^2)$ time
- gives you center star T_c

② Construct A_c consistent with T_c

$O(k^2 n^2)$ time.

↳ with SP cost $d(A_c)$

Thm. $$\frac{d(A_c)}{d(A^*)} \le \frac{2(k-1)}{k} \le \boxed{2}$$

holds for any cost function satisfying triangle ineq.

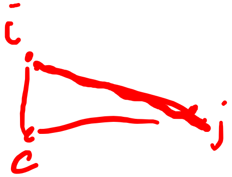
$k=3$ $$\frac{2\cdot(3-1)}{3} = \frac{4}{3}$$

$k=4$ $$\frac{2(4-1)}{4} = \frac{6}{4}$$

$\downarrow$

$k=\infty$ ≤ 2

Lemma 1. Let δ be a cost function that satisfies the triangle ineq. and let $\bar{v}_c$ be the center string of a center star alignment A_c of input $\bar{v}_1, \ldots, \bar{v}_k$. Then, for any input string $\bar{v}_i$ and $\bar{v}_j$ it holds that



$$d(\bar{v}_i(A_c), \bar{v}_j(A_c)) \leq d(\bar{v}_i(A_c), \bar{v}_c(A_c)) + d(\bar{v}_c(A_c), \bar{v}_j(A_c))$$

$$= D(\bar{v}_i, \bar{v}_c) + D(\bar{v}_c, \bar{v}_j)$$

Consider any <u>column</u> of Ac.

$$\begin{matrix} \bar{v}_i(A_c) \\ \\ \bar{v}_l(A_c) \\ \\ v_j(A_c) \end{matrix} \qquad \left.\begin{pmatrix} x \\ \\ y \\ \\ z \end{pmatrix}\right\} \Sigma \cup \{-\}$$

$$\boxed{\delta(x,z) \leq \delta(x,y) + \delta(y,z)}$$

Thm 2.

$$\frac{\boxed{d(A_c)}}{d(A^*)} \le \frac{2(k-1)}{k}$$

$d(A_c)$ SP score of A_c

$$d(A_c) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} d(\bar{v}_i(A_c), \bar{v}_j(A_c))$$

Proof.

$$f(A_c) = \sum_{\substack{(i,j) \in [k]^2 \\ i \neq j}} d(\bar{v}_i(A_c), \bar{v}_j(A_c))$$

$$f(A^*) = \sum_{\substack{(i,j) \in [k]^2 \\ i \neq j}} d(\bar{v}_i(A^*), \tilde{v}_j(A^*))$$

Clearly, $f(A_c) = 2 \cdot d(A_c)$ $\quad f(A^*) = 2 \cdot d(A^*)$

$$F(A_c) = \sum_{\substack{(i,j) \in [k]^2 \\ i \neq j}} d(\bar{v}_i(A_c), \bar{v}_j(A_c))$$

{triangle ineq.} {lemma}

$$\leq \sum_{\substack{(i,j) \in [k]^2 \\ i \neq j}} \left[d(\bar{v}_i(A_c), \bar{v}_c(A_c)) + d(\bar{v}_c(A_c), \bar{v}_j(A_c)) \right]$$

$$= \sum_{\substack{i,j \in [k]^2 \\ i \neq j}} \left[\underline{D(\bar{v}_i, \bar{v}_c)} + \underline{D(\bar{v}_c, \bar{v}_j)} \right]$$

Observe that for any fixed j each of the terms $D(\bar{v}_c, \bar{v}_j)$ and $D(\bar{v}_j, \bar{v}_c)$ show up $u-1$ times. Moreover, $D(\bar{v}_c, \bar{v}_j) = D(\bar{v}_j, \bar{v}_c)$

$$\underline{E(A_c)} \leq \sum_{\substack{i,j \in [k]^2 \\ i \neq j}}^{k} \left[D(\bar{v}_i, \bar{v}_c) + D(\bar{v}_c, \bar{v}_j) \right]$$

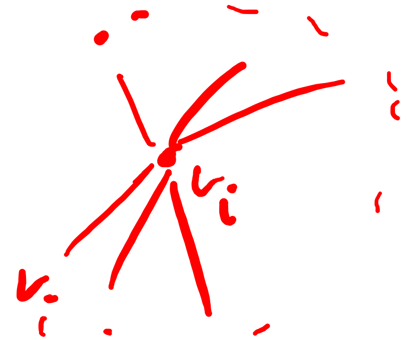
$$= 2(k-1) \sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j) .$$

$$E(A^*) = \sum_{\substack{(i,j)\in[k]^2 \\ i\neq j}} \underbrace{d(\bar{v}_i(A^*), \bar{v}_j(A^*))}$$

$$\geq \sum_{\substack{(i,j)\in[k]^2 \\ i\neq j}} D(\bar{v}_i, \bar{v}_j)$$

$$D(\bar{v}_i, \bar{v}_i) = 0$$

$$= \sum_{i=1}^{k} \underbrace{\sum_{j=1}^{k} D(\bar{v}_i, \bar{v}_j)}_{\text{cost of a star with center } \bar{v}_i}$$



Recall that $\underline{\bar{v}_c}$ corresponds to the star with smallest cost.

$$f^*(A^*) \geq \sum_{i=1}^{k} \underbrace{\boxed{\sum_{j=1}^{k} D(\bar{v}_i, \bar{v}_j)}}_{\text{star centered at } \bar{v}_i}$$

$$\geq k \sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j).$$

$$\sum_{j=1}^{k} D(\bar{v}_i, \bar{v}_j) \geq \sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j)$$

↳ center string

$$\underline{F(A_c)} \le 2(k-1) \sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j)$$

$$\underline{F(A^*)} \ge k \sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j)$$

$$\frac{d(A_c)}{d(A^*)} = \frac{F(A_c)}{F(A^*)} \le \frac{2(k-1) \cancel{\sum_{j=1}^{k} D(\bar{v}_c, \bar{v}_j)}}{k \cancel{\sum_{j=1}^{k} D(\bar{v}_c, v_j)}}$$

$$\le \lim_{k \to \infty} \frac{2(k-1)}{k} = 2$$

Summary

1. Optimal pairwise alignment by dynamic programming in $O(n^2)$ time
2. Optimal multiple alignment with SP-score by dynamic programming in $O(k^2 2^k n^k)$ time
3. Multiple alignment with SP-score is NP-hard (Jiang and Wang, 1994)
4. Carrillo-Lipman enables us to decide whether alignment passes through a vertex (i_1, i_2, i_3) for $k = 3$ sequences (generalizes to $k > 3$)
5. Progressive alignment methods are widely used, but come with no theoretical bounds on alignment quality
6. Star alignment gives 2-approximation algorithm

History

- 1975 Sankoff
  Formulated MSA problem and gave dynamic programming solution
- 1988 Carrillo-Lipman
  Branch and Bound approach for MSA
- 1990 Feng-Doolittle
  Progressive alignment
- 1993 Gusfield
  Star alignment: 2-approximation algorithm
- 1994 Jiang and Wang
  MSA with SP-score is NP-hard
- 1994 Thompson-Higgins-Gibson: ClustalW
  Most popular multiple alignment program
- 2000 Notredam-Higgins-Heringa: T-coffee
  Use library of pairwise alignments
- 2004 Edgar: MUSCLE
  Refinement