Algorithmen der Bioinformatik I
WS 2017/2018

Burkhard Morgenstern
Peter Meinicke

Dept. Bioinformatics
Institute of Microbiology and Genetics (IMG)
University of Göttingen

November 7, 2017

The four-point condition

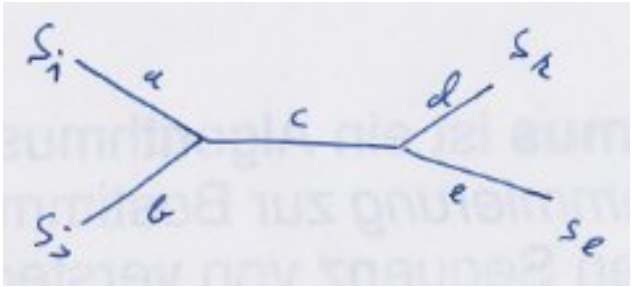
Question: for

$$S_1, \ldots, S_n$$

and a $n \times n$ distance matrix $(d_{i,j})$, is there a tree T representing the distances $d_{i,j}$?

The four-point condition

Consider tree T and S_i, S_j, S_k, S_l arranged in T as:



Then one has:

$$\begin{aligned} d_{i,j} + d_{k,l} &= a + b + d + e \\ d_{i,k} + d_{j,l} &= a + b + 2 \cdot c + d + e \\ d_{i,l} + d_{j,k} &= a + b + 2 \cdot c + d + e \end{aligned}$$

The four-point condition

In general: for each selection of 'species' S_i, S_j, S_k, S_l, two of the sums

$$\begin{array}{lcl} d_{i,j} & + & d_{k,l} \\ d_{i,k} & + & d_{j,l} \\ d_{i,l} & + & d_{j,k} \end{array}$$

are equal and the third one is smaller or equal than these two sums.

Equivalently: for each selection of S_i, S_j, S_k, S_l, one has

$$d_{i,j} + d_{k,l} \leq \max\{d_{i,j} + d_{k,l}, d_{i,j} + d_{k,l}\}$$

('Four-point condition')

The four-point condition

Theorem

For a distance matrix $(d_{i,j})$, there is a tree T representing the distances $d_{i,j}$ if and only if $(d_{i,j})$ meets the four-point condition.

Neighbour Joining

UPGMA can go wrong:

- Distances between root and leaves made equal
- Branching pattern ('topology') can be wrong

Example (UPGMA finds wrong branching pattern)

	A	*B*	*C*	*D*
A	0	6	8	12
B		0	4	8
C			0	6
D				0

Neighbour Joining

Improved distance method for tree reconstruction:

Neighbour Joining

Hierarchical clustering as in *UPGMA*, but: instead of joining sequences S_i, S_j with minimal distance $d_{i,j}$, use 'corrected distances' $D_{i,j}$.

Assumption: sequences evolved from tree T *without* molecular-clock property, i.e. leaves may have different distances from root. Observed distances are 'real' distances in T.

Neighbour Joining

Define:

$$r_i = \frac{1}{n-2} \sum_k d_{i,k}$$

and ‘corrected distances’

$$D_{i,j} = d_{i,j} - r_i - r_j$$

(Attention: values $D_{i,j}$ may be negative!)

Neighbour Joining

It can be proven mathematically:

Theorem

If *there exists an (unknown) tree T with distances $d_{i,j} = d^T_{i,j}$, then:*

Pair of sequences S_i, S_j minimizes $D_{i,j}$ if and only if S_i and S_j are neighbouring leaves in T.

Neighbour Joining

Algorithm:

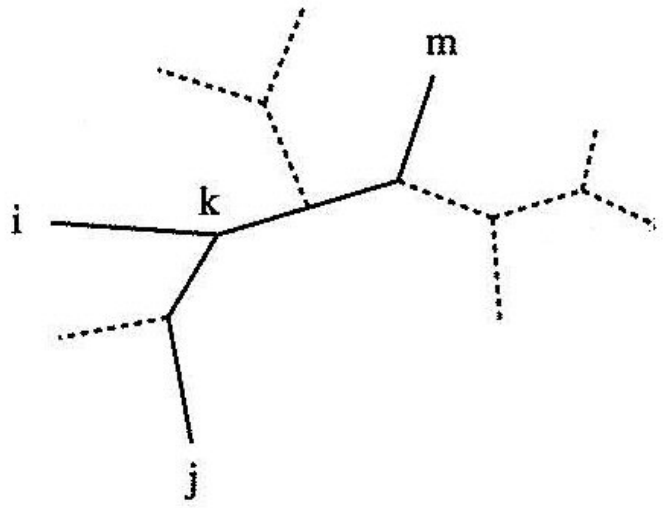
- Calculate 'corrected distances' $D_{i,j}$ for each sequence pair S_i, S_j.
- Join pair of sequences with minimal $D_{i,j}$
- Replace S_i, S_j by new node S_k
- Compute distances between S_k and other sequences.
- Re-calculate 'corrected distances' $D_{i,j}$
- Join pair of sequences with minimal $D_{i,j}$

etc

Neighbour Joining

If nodes S_i and S_j with minimal $D_{i,j}$ connected by new node S_k:

Calculate distance $d_{k,m}$ to remaining sequences S_m.



Easy to see: $$d_{i,m} + d_{j,m} = 2 \cdot d_{k,m} + d_{i,k} + d_{j,k}$$

$$\Rightarrow d_{k,m} = \frac{1}{2}(d_{i,m} + d_{j,m} - d_{i,j})$$

Neighbour Joining

Result:

If there exists an (unknown) tree T, and distances $d_{i,j}$ are derived from T, *i.e.*

$$d_{i,j} = d_{i,j}^T$$

then Neighbour-Joining is guaranteed to find the underlying tree T.

In general:

Neighbour-Joining constructs tree T such that tree distances $d_{i,j}^T$ *approximate* input distances $d_{i,j}$

Estimating distances between DNA sequences

In an ideal world:

'Distance' $d(S_1, S_2)$ between sequences S_1 and S_2 defined as *time* that passed, since S_1 and S_2 evolved from their last common ancester.

Problem: time cannot be inferred by looking at S_1 and S_2

Estimating distances between DNA sequences

Therefore:

Define distance $d(S_1, S_2)$ as (estimated) *number of substitutions per position* that happened since S_1 and S_2 separated in phylogeny.

Estimating distances between DNA sequences

To estimate d:

- Align S_1 and S_2
- If distance between S_1 and S_2 small: number of subsitutions $\approx$ number of mismatches, use relative frequency of mismatches.
- For larger distances, use for example *Jukes-Cantor* formula.

Estimating distances between DNA sequences

S_1 `taaggactgttagaggcaacacatcactgctgccccgtaggtcagtctgatca`

S_2 `taaggagtgtttcagaggcaacacatcactgctgccccgtaggacagtctatca`

How many substitutions have occurred since S_1 and S_2 separated?

Estimating distances between DNA sequences

S_1 `taaggactgtt--agaggcaacacatcactgctgccccgtaggtcagtctgatca`

S_2 `taaggagtgtttcagaggcaacacatcactgctgccccgtaggacagtct-atca`

Align sequences!

Estimating distances between DNA sequences

S_1 `taaggactgtt--agaggcaacacatcactgctgccccgtaggtcagtctgatca`

S_2 `taaggagtgtttcagaggcaacacatcactgctgccccgtaggacagtct-atca`

For close sequences:
number of *mismatches* $\approx$ number of subsitutions

Estimating distances between DNA sequences

S_1 taaggactgtt--agaggcaacacatcactgctgccccgtaggtcagtctgatca

S_3 tagcgagtgcttcatcggtagtacaccacaccttgaccattggatagtgt-acca

For distant sequences:
more *substitutinos* than *mismatches* expected!

Estimating distances between DNA sequences

Jukes-Cantor model for DNA evolution: equal rate for *all* possible substitutios.

Average number d of substitutions per sequence position estimated as

$$d = -\frac{3}{4} \ln \left[1 - \frac{4}{3} \times p \right]$$

where p is (average) number of mismatches per position

Estimating distances between DNA sequences

Example (1):

S_1 taaggactgtt--agaggcaacacatcactgctgccccgtaggtcagtctgatca
S_3 tagcgagtgcttcatcggtagtacaccacaccttgaccattggatagtgt-acca

52 positions in alignment (ignoring gaps), 21 mismatches.

$\Rightarrow p = 21/52 \approx 0.403$ mismatches per position.

With *Jukes-Cantor*: $d = 0.578$ substitution per position estimated.

Estimating distances between DNA sequences

Example (2):

S_1 `taaggactgtt--agaggcaacacatcactgctgccccgtaggtcagtctgatca`
S_2 `taaggagtgtttcagaggcaacacatcactgctgccccgtaggacagtct-atca`

52 positions in alignment (ignoring gaps), 2 mismatches.

$\Rightarrow p = 2/52 \approx 0.0384$ mismatches per position.

With *Jukes-Cantor*: $d = 0.0394$ substitution per position estimated.

Multiple sequence alignment (MSA)

Example (Multiple Alignment)

S_1	N	G	P	E	V	R	E	L	W
S_2	A	R	D	I	W	A			
S_3	Q	A	R	E	S	I	Y	A	
S_4	V	R	E	S	L	W	S		
S_5	W	Y	V	R	D	A	S	L	W	S

Protein Family, not aligned. Homologies not visible.

Multiple sequence alignment (MSA)

Example (Multiple Alignment)

S_1	N	G	P	E	V	R	E	–	–	L	W	–
S_2	–	–	–	–	A	R	D	–	–	I	W	A
S_3	–	–	–	Q	A	R	E	–	S	I	Y	A
S_4	–	–	–	–	V	R	E	–	S	L	W	S
S_5	–	–	W	Y	V	R	D	A	S	L	W	S

Multiple alignment (MSA) highlights (local) similarities

Multiple sequence alignment (MSA)

Goal: calculate best MSA for input sequences

$$S_1, \ldots, S_n$$

automatically

Definition (Projection of Multiple Alignment)

For multiple alignment A, define $P_{i,j}(A)$ *as* projection *of A to sequences* S_i *and* S_j *by omitting all other sequences (and removing double gaps in remaining alignment of* S_i *and* S_j*)*

Multiple sequence alignment (MSA)

Example (Projection of MSA)

S_1	N	G	P	E	V	R	E	–	–	L	W	–
S_2	–	–	–	–	A	R	D	–	–	I	W	A
S_3	–	–	–	Q	A	R	E	–	S	I	Y	A
S_4	–	–	–	–	V	R	E	–	S	L	W	S
S_5	–	–	W	Y	V	R	D	A	S	L	W	S

Multiple alignment A

Multiple sequence alignment (MSA)

Example (Projection of MSA)

S_1	N	G	P	E	V	R	E	–	–	L	W	–
S_2													
S_3													
S_4	–	–	–	–	V	R	E	–	S	L	W	S
S_5													

Projection $P_{1,4}(A)$ of A to S_1 and S_4

Multiple sequence alignment (MSA)

Example (Projection of MSA)

S_1	N	G	P	E	V	R	E	–	–	L	W	–
S_4	–	–	–	–	V	R	E	–	S	L	W	S

Projection $P_{1,4}(A)$ of A to S_1 and S_4

Multiple sequence alignment (MSA)

Example (Projection of MSA)

S_1	N	G	P	E	V	R	E	-	L	W	-
S_4	-	-	-	-	V	R	E	S	L	W	S

Projection $P_{1,4}(A)$ of A to S_1 and S_4

Multiple sequence alignment (MSA)

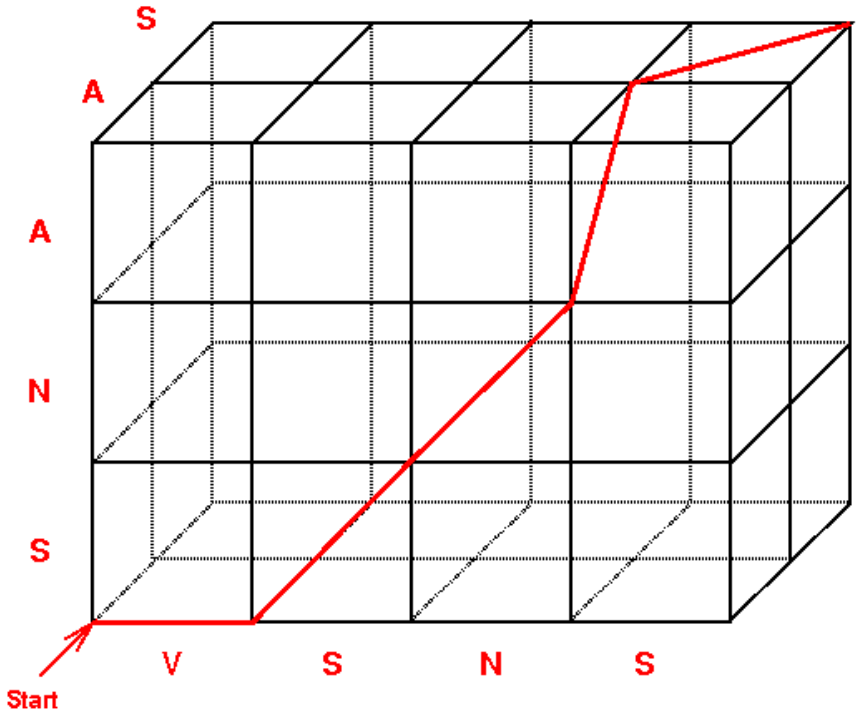
Sum-of-pairs score $Sc(A)$ of MSA A of $S_1, \ldots, S_n$ defined as:

$$Sc(A) = \sum_{i<j} Sc(P_{i,j}(A))$$

Dynamic-programming algorithm to calculate optimal alignments can be generalized to multiple alignment (in theory!)

But: computing time far too long, not practicable

Multiple sequence alignment (MSA)



V S N _ S
_ S N A _
_ _ _ A S

MSA for n sequences corresponds to path through $n-$dimensional space

(`http://www.techfak.uni-bielefeld.de/`)

Multiple sequence alignment (MSA)

Heuristic solutions necessary; most important approach: *progressive* alignment!

Idea for ‘progressive alignment’:

- Align successively sequences and groups of previously aligned sequences, until all sequences are aligned in one MSA
- Two *groups* G_1, G_2 of sequences aligned by A_1, A_2 can be aligned like two *single* sequences if A_1, A_2 remain unchanged

Multiple sequence alignment (MSA)

Example (Alignment of two multiple alignments)

S_1	E	V	R	E	–	V	W	–
S_2	–	A	R	D	–	I	W	A
S_3	Q	A	R	E	S	I	Y	A
S_4	–	–	R	E	S	L	W	S
S_5	R	E	W	S	L	W	S	
S_6	R	E	Y	S	–	–	S	

Multiple sequence alignment (MSA)

Example (Alignment of two multiple alignments)

S_1	E	V	R	E	–	–	V	W	–
S_2	–	A	R	D	–	–	I	W	A
S_3	Q	A	R	E	–	S	I	Y	A
S_4	–	–	R	E	–	S	L	W	S
S_5	–	–	R	E	W	S	L	W	S
S_6	–	–	R	E	Y	S	–	–	S

Score of aligning two columns: sum of scores of amino-acid pairs. For large sets of sequences: represent columns as *profiles*, i.e. frequencies of amino acids.

Multiple sequence alignment (MSA)

Procedure for progressive alignment:

- Construct rooted tree of input sequences $S_1, \ldots, S_n$ ('guide tree').
- Traverse T from leaves to root
- At every inner node, construct profile alignments of sequences corresponding to daughter nodes

Multiple sequence alignment (MSA)

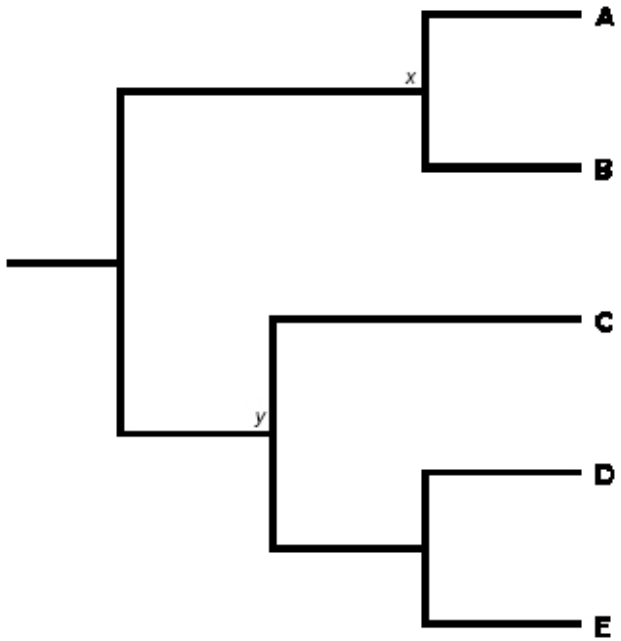


Figure: Tree of 5 sequences,
Topology (branching order, ignoring branch lengths) represented as $((A, B)(C(D, E)))$