Maximum Likelihood Methods in Molecular Phylogeny

Heiko A. Schmidt

Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories (MFPL)
Vienna, Austria
heiko.schmidt@univie.ac.at

September 2007

Main Types of Phylogenetic Methods

Data	Method	Evaluation Criterion
Characters (Alignment)	**Maximum Parsimony**	Parsimony
	Statistical Approaches: Likelihood, Bayesian	Evolutionary Models
Distances	**Distance Methods**	Evolutionary Models

Introduction: ML on Coin Tossing

Given a box with 3 coins of different fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

Introduction: ML on Coin Tossing

Given a box with 3 coins of different fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin an toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

Introduction: ML on Coin Tossing

Given a box with 3 coins of different fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin an toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

Probability

$$p(k \text{ heads in } n \text{ tosses}|\theta)$$

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Hence, *"likelihood flips the probability around."*

Introduction: ML on Coin Tossing

Given a box with 3 coins of different fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin an toss 20 times:

$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

	Probability		**Likelihood**
$p(k$ heads in n tosses$\|\theta)$	$\equiv$	$L(\theta\|k$ heads in n tosses$)$	

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Hence, *"likelihood flips the probability around."*

Introduction: ML on Coin Tossing

Given a box with 3 coins of different fairness $\left(\frac{1}{3}, \frac{1}{2}, \frac{2}{3} \text{ heads}\right)$

We take out one coin an toss 20 times:

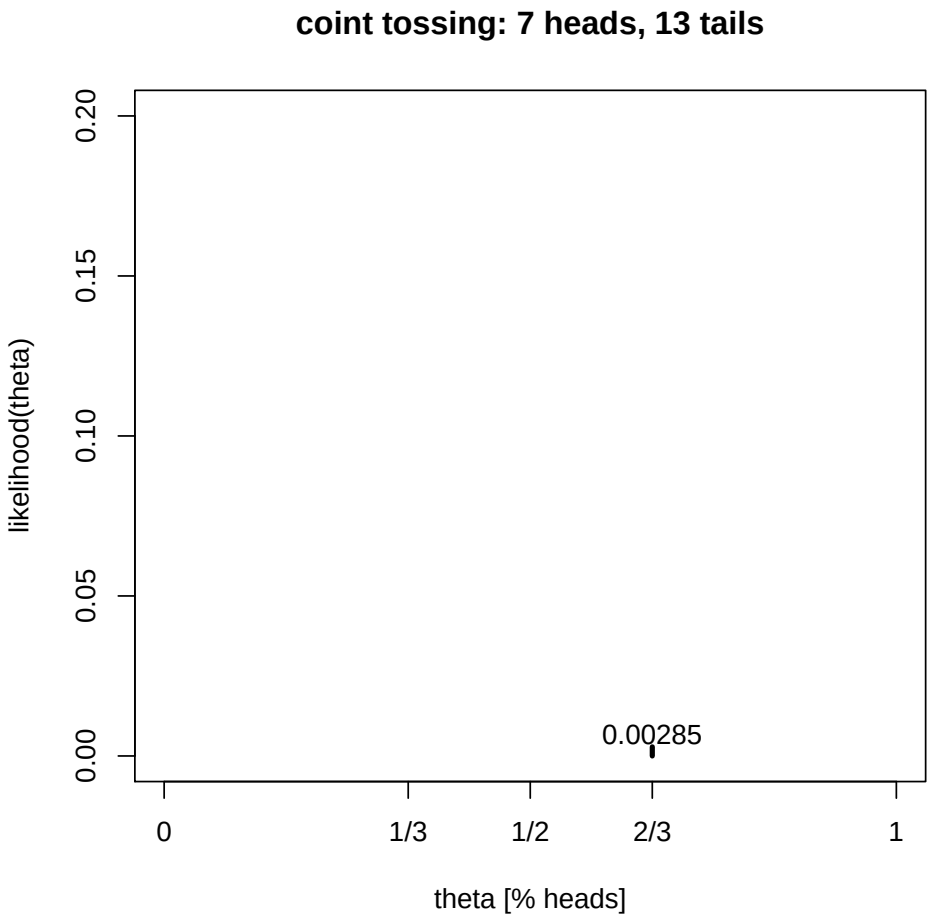
$$H, T, T, H, H, T, T, T, T, H, T, T, H, T, H, T, T, H, T, T$$

	Probability		**Likelihood**
$p(k \text{ heads in } n \text{ tosses}\|\theta)$	$\equiv$	$L(\theta\|k \text{ heads in } n \text{ tosses})$	
	$=$	$\binom{n}{k}\theta^k(1-\theta)^{n-k}$	
		(here binomial distribution)	

Aim: The ML approach seaches for that parameter set θ for the generating process which maximizes the probability of our given data.

Hence, *"likelihood flips the probability around."*

Introduction: ML on Coin Tossing (Estimate)

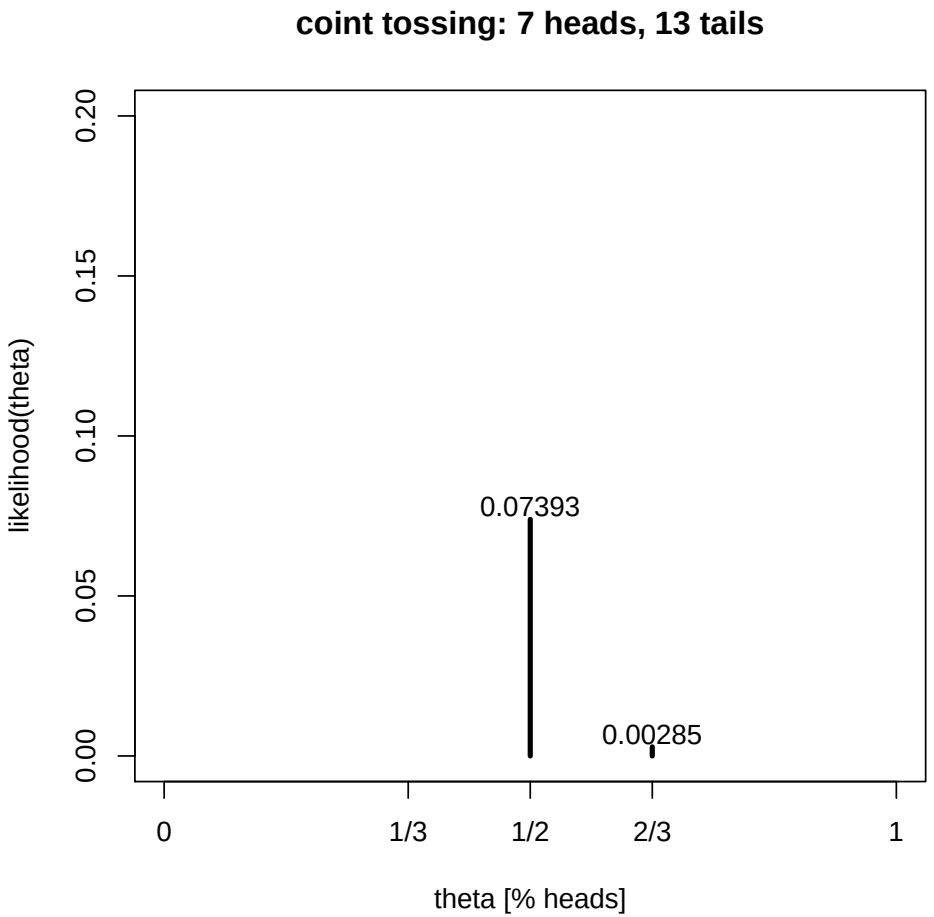


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)

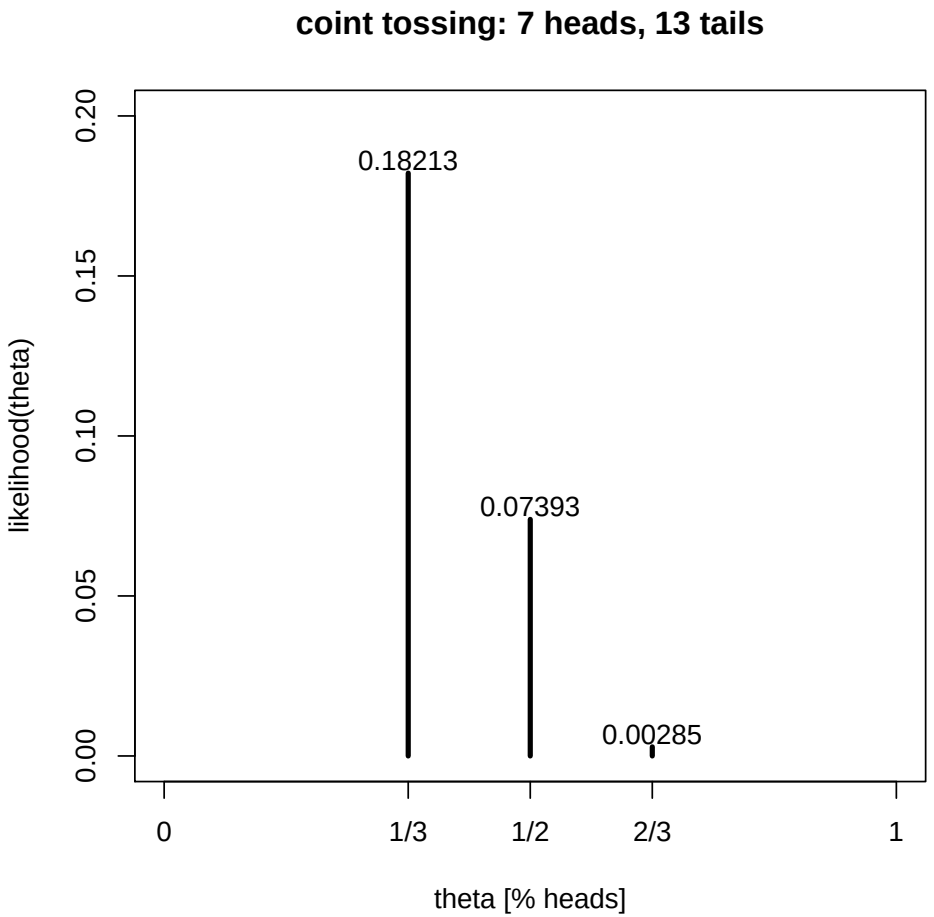


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)

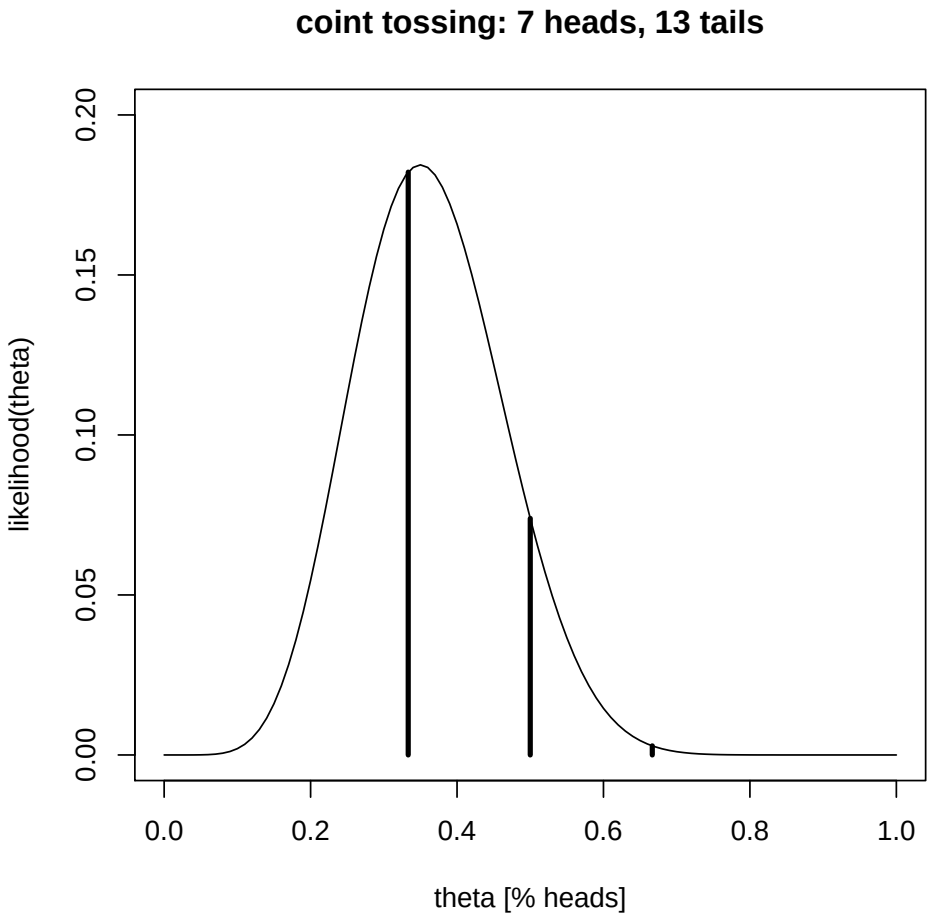


Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

Introduction: ML on Coin Tossing (Estimate)



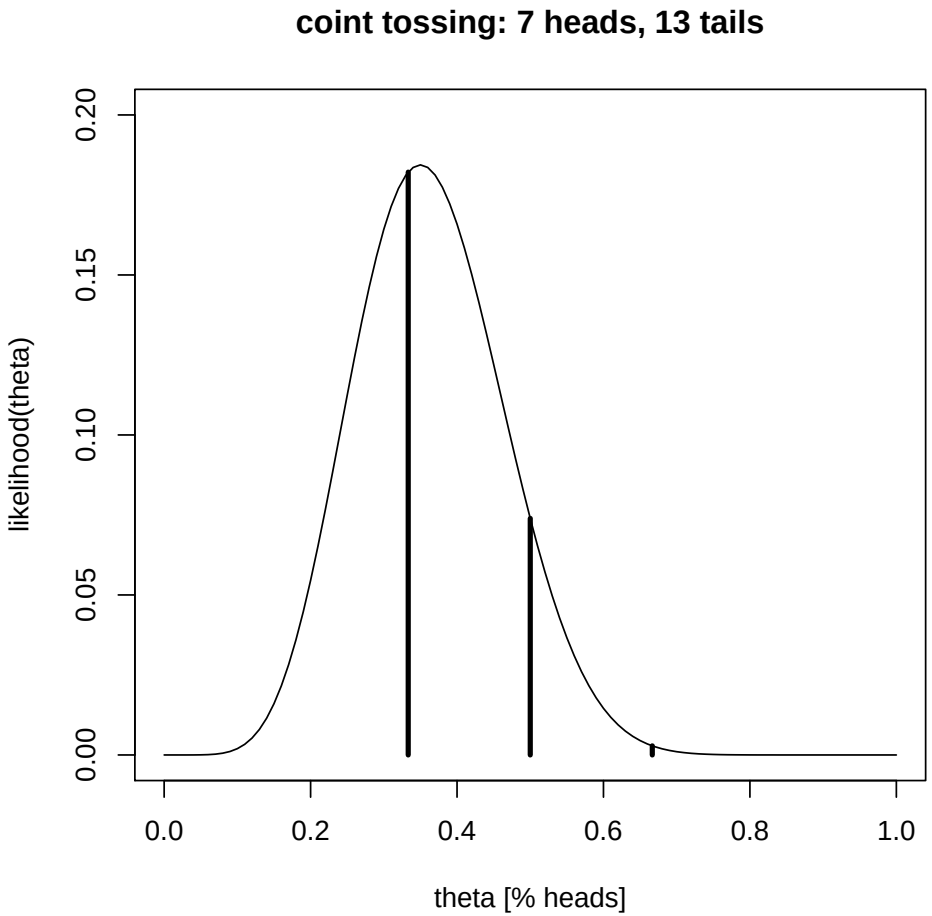
Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

For infinitely many coins
$\theta = (0...1)$

Introduction: ML on Coin Tossing (Estimate)



Three coin case

$$L(\theta|7 \text{ heads in } 20) = \binom{20}{7}\theta^7(1-\theta)^{13}$$

for each coin $\theta \in \left\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\right\}$

For infinitely many coins
$\theta = (0...1)$

ML estimate: $L(\hat{\theta}) = 0.1844$ where coin shows $\hat{\theta} = 0.35$ heads

From Coins to Phylogenies?

While the coin tossing example might look easy, in phylogenetic analysis, the parameter (set) θ comprises:

- evolutionary model
- its parameters
- tree topology
- its branch lengths

That means, a high dimensional optimization problem.
Hence, some parameters are often estimated/set separately.

Modeling Evolution

- Evolution is usually modeled as a

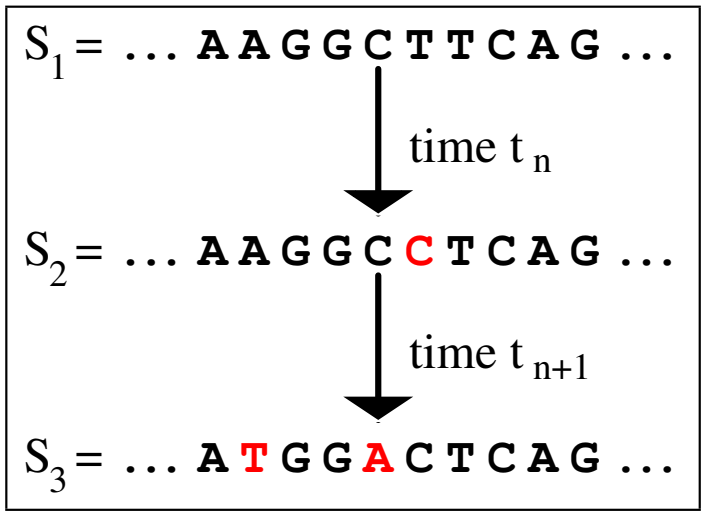
  stationary, time-reversible Markov process.

- What does that mean?

Assumptions on Evolution

Markov Process

The (evolutionary) process evolves without memory, i.e. sequence S_2 mutates to S_3 during time t_{n+1} independent of state of S_1.



Assumptions on Evolution

Stationary:
The overall character frequencies π_j of the nucleotides or amino acids are in an equilibrium and remain constant.

Time-Reversible:
Mutations in either direction are equally likely

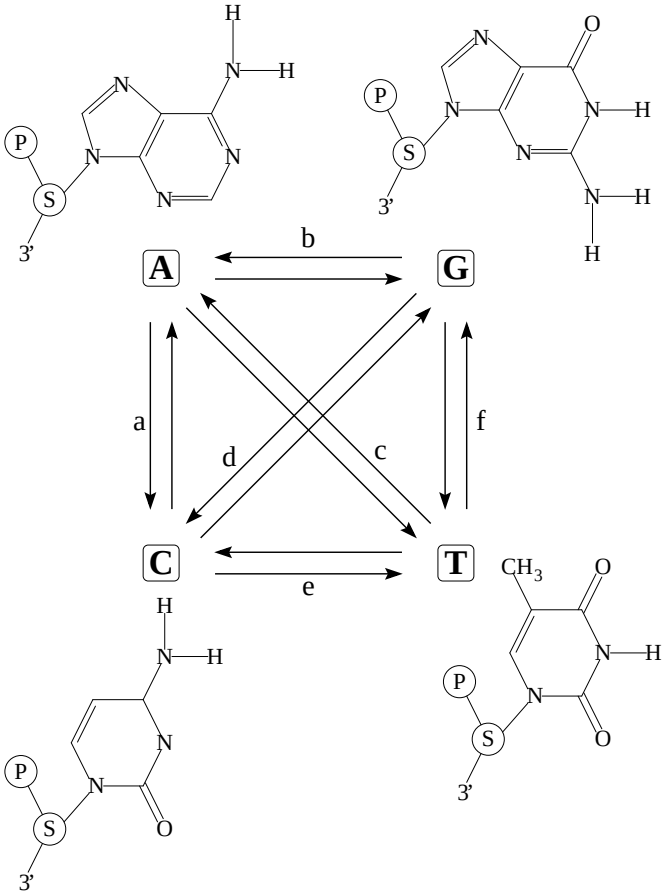
$$\pi_i \cdot P_{ij}(t) = P_{ji}(t) \cdot \pi_j$$

This means a mutation is as likely as its back mutation.

$$P(i \rightarrow j) = P(i \leftarrow j) \qquad \text{(JC69)}$$

Substitution Models

Evolutionary models are often described using a substitution rate matrix R and character frequencies Π. Here, 4×4 matrix for DNA models:



$$R = \begin{pmatrix} A & C & G & T \\ - & a & b & c \\ a & - & d & e \\ b & d & - & f \\ c & e & f & - \end{pmatrix}$$

$$\Pi = (\pi_A, \pi_C, \pi_G, \pi_T)$$

From Substitution rates to probabilities

. . . R and Π are combined into the instantaneous rate matrix Q

$$Q = \begin{pmatrix} \bullet_A & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & \bullet_C & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & \bullet_G & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & \bullet_T \end{pmatrix} \qquad \begin{aligned} \bullet_A &= -(a\pi_C + b\pi_G + c\pi_T) \\ \bullet_C &= -(a\pi_A + d\pi_G + e\pi_T) \\ \bullet_G &= -(b\pi_A + d\pi_C + f\pi_T) \\ \bullet_T &= -(c\pi_A + e\pi_C + f\pi_G) \end{aligned}$$

(where the row sums are zero).

From Substitution rates to probabilities

. . . R and Π are combined into the instantaneous rate matrix Q

$$Q = \begin{pmatrix} \bullet_A & a\pi_C & b\pi_G & c\pi_T \\ a\pi_A & \bullet_C & d\pi_G & e\pi_T \\ b\pi_A & d\pi_C & \bullet_G & f\pi_T \\ c\pi_A & e\pi_C & f\pi_G & \bullet_T \end{pmatrix} \qquad \begin{aligned} \bullet_A &= -(a\pi_C + b\pi_G + c\pi_T) \\ \bullet_C &= -(a\pi_A + d\pi_G + e\pi_T) \\ \bullet_G &= -(b\pi_A + d\pi_C + f\pi_T) \\ \bullet_T &= -(c\pi_A + e\pi_C + f\pi_G) \end{aligned}$$

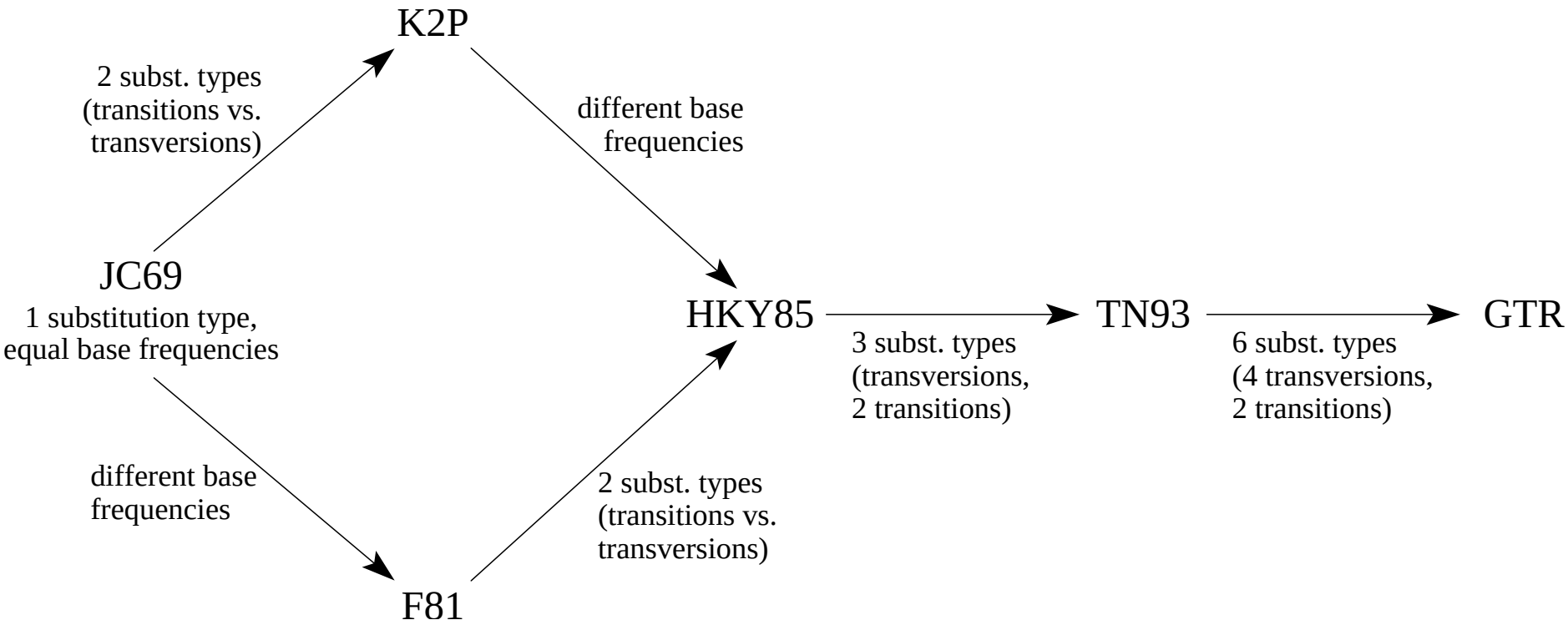
(where the row sums are zero).

Given now the instantaneous rate matrix Q, we can compute a substitution probability matrix P

$$P(t) = e^{Qt}$$

With this matrix P we can compute the probability $P_{ij}(t)$ of a change $i \longrightarrow j$ over a time t.

Relations between DNA models



Protein Models

Generally this is the same for protein sequences, but with 20×20 matrices. Some protein models are:

- Poisson model ("JC69" for proteins)
- Dayhoff (Dayhoff *et al.*, 1978)
- JTT (Jones *et al.*, 1992)
- mtREV (Adachi & Hasegawa, 1996)
- cpREV (Adachi *et al.*, 2000)
- VT (Müller & Vingron, 2000)
- WAG (Whelan & Goldman, 2000)
- BLOSUM 62 (Henikoff & Henikoff, 1992)

Computing ML Distances Using $\mathbf{P_{ij}(t)}$

The Likelihood of sequence s evolving to s' in time t:

$$L(t|s \rightarrow s') = \prod_{i=1}^{m} \Big(\Pi(s_i) \cdot P_{s_i s'_i}(t) \Big)$$

Likelihood surface for two sequences under JC69:

GATCCTGAGAGAAATAAAC
GGTCCTGACAGAAATAAAC

Computing ML Distances Using $\mathbf{P_{ij}(t)}$

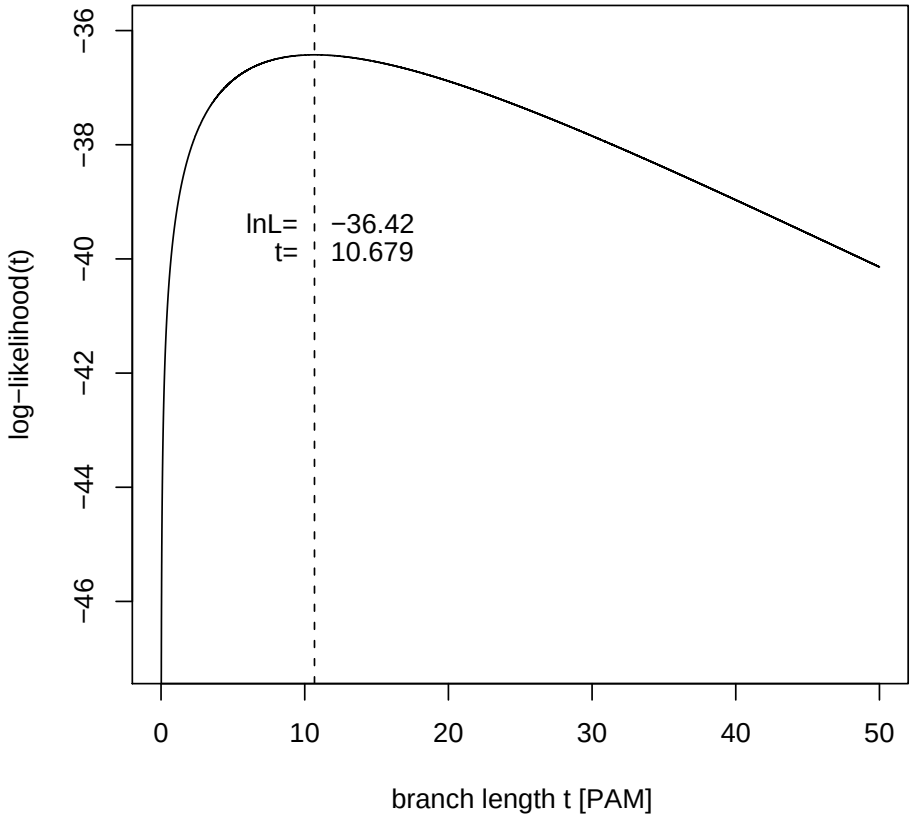
The Likelihood of sequence s evolving to s' in time t:

$$L(t|s \rightarrow s') = \prod_{i=1}^{m} \left(\Pi(s_i) \cdot P_{s_i s'_i}(t) \right)$$

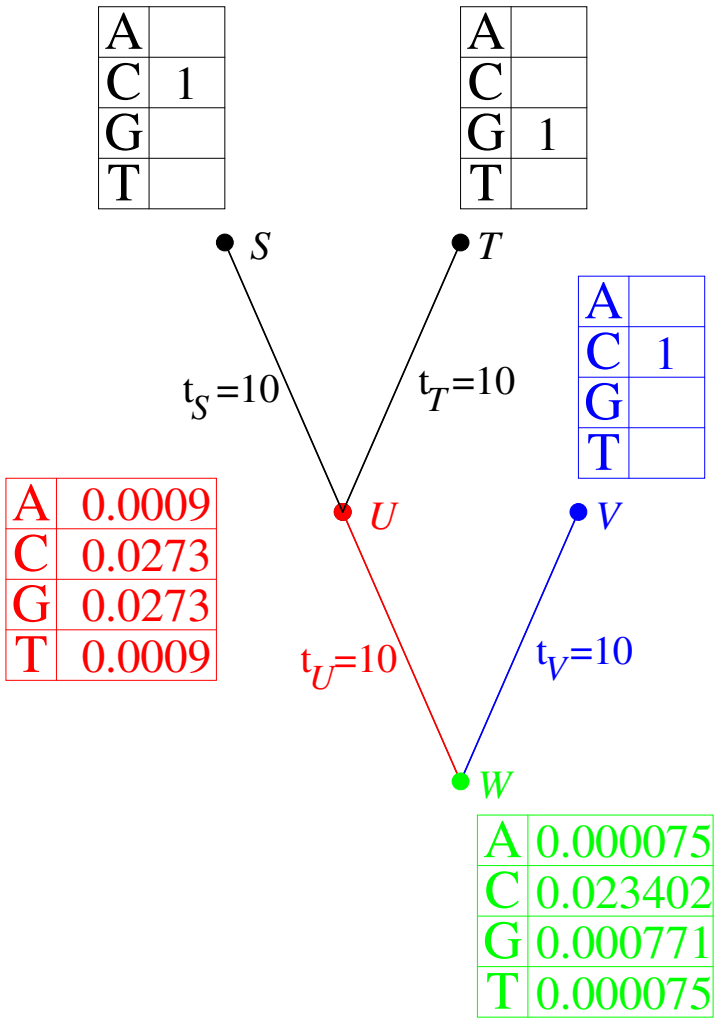
Likelihood surface for two sequences under JC69:

GATCCTGAGAGAAATAAAC
GGTCCTGACAGAAATAAAC

Note: we do not compute the probability of the distance t but that of the data $D = \{s, s'\}$.



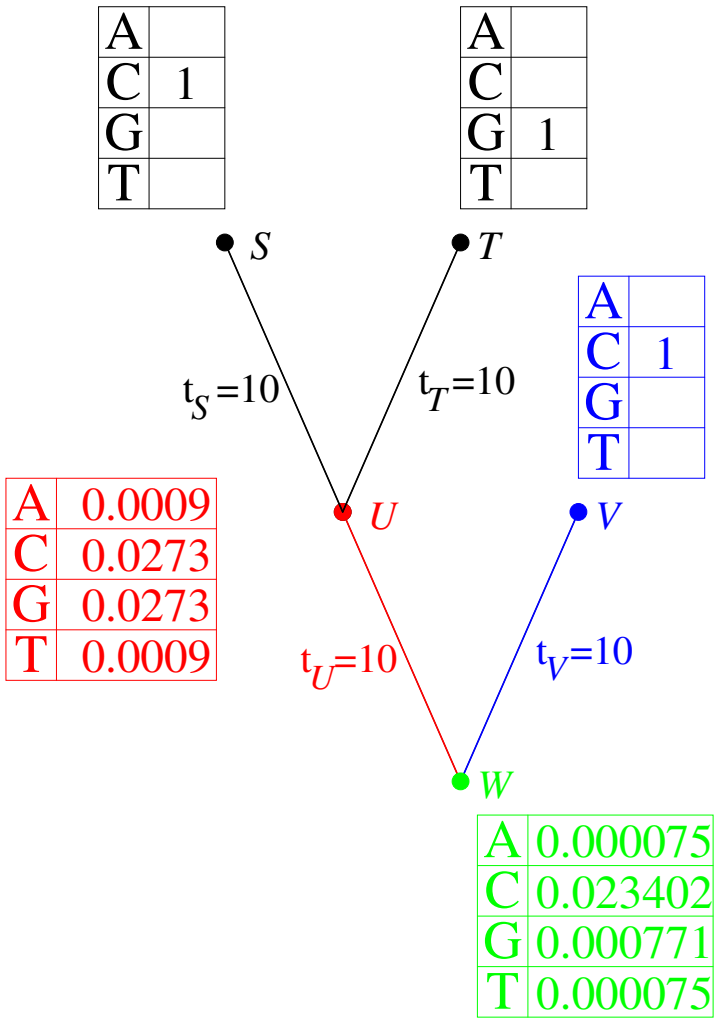
Likelihoods of Trees (Single column $\begin{matrix} C \\ G \\ C \end{matrix}$, given tree)



Likelihoods of nucleotides at inner nodes:

$$L_U(i) = [P_{iC}(10) \cdot L(C)] \cdot [P_{iG}(10) \cdot L(G)]$$

Likelihoods of Trees (Single column $\begin{matrix} C \\ G \\ C \end{matrix}$, given tree)



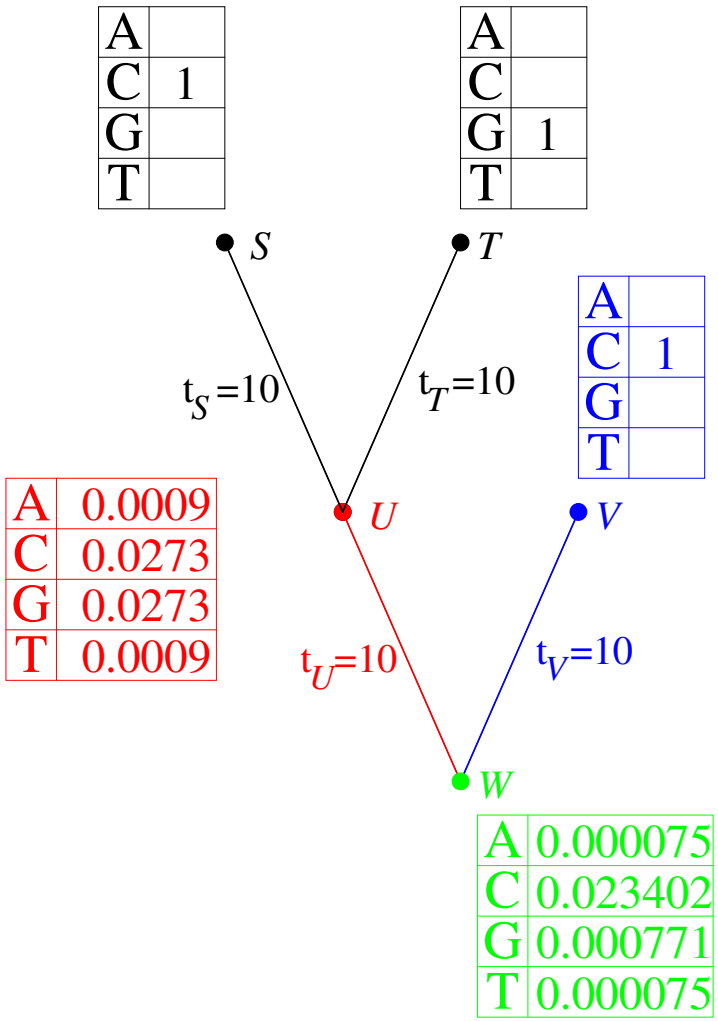
Likelihoods of nucleotides at inner nodes:

$$L_U(i) = [P_{iC}(10) \cdot L(C)] \cdot [P_{iG}(10) \cdot L(G)]$$

$$L_W(i) = \left[\sum_{\substack{u= \\ ACGT}} P_{iu}(t_U) \cdot L_U(u)\right] \cdot$$

$$\left[\sum_{\substack{v= \\ ACGT}} P_{iv}(t_V) \cdot L_V(v)\right]$$

Likelihoods of Trees (Single column $\begin{matrix} C \\ G \\ C \end{matrix}$, given tree)



Likelihoods of nucleotides at inner nodes:

$$L_U(i) = [P_{iC}(10) \cdot L(C)] \cdot [P_{iG}(10) \cdot L(G)]$$

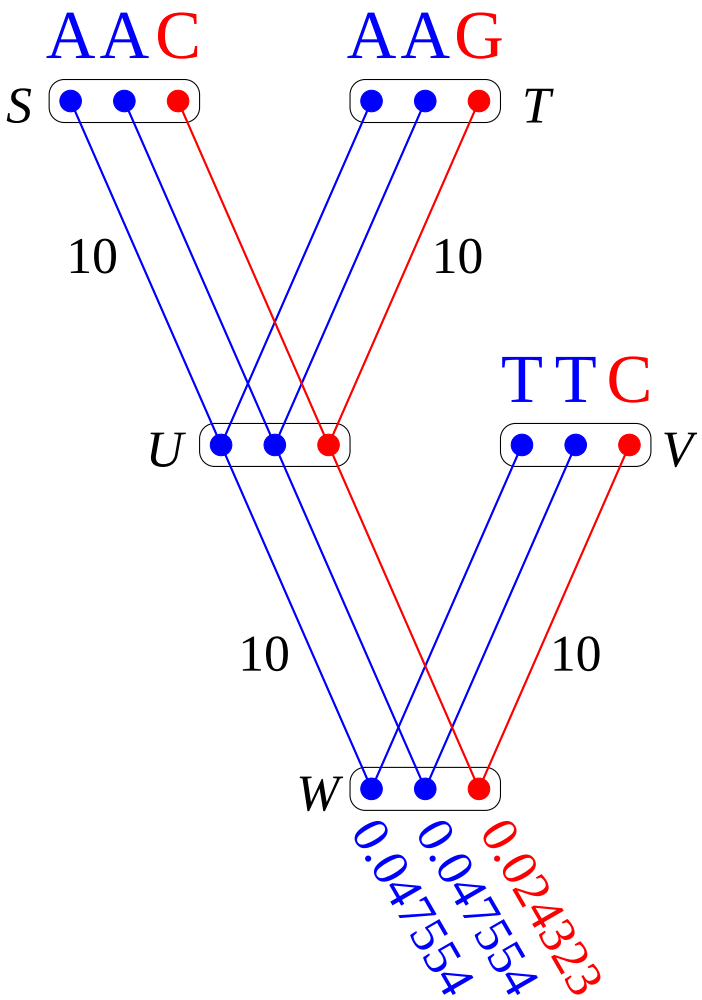
$$L_W(i) = \left[\sum_{\substack{u= \\ ACGT}} P_{iu}(t_U) \cdot L_U(u)\right] \cdot$$

$$\left[\sum_{\substack{v= \\ ACGT}} P_{iv}(t_V) \cdot L_V(v)\right]$$

Site-Likelihood of an alignment column k:

$$L^{(k)} = \sum_{\substack{i= \\ ACGT}} \pi_i \cdot L_W(i) = 0.024323$$

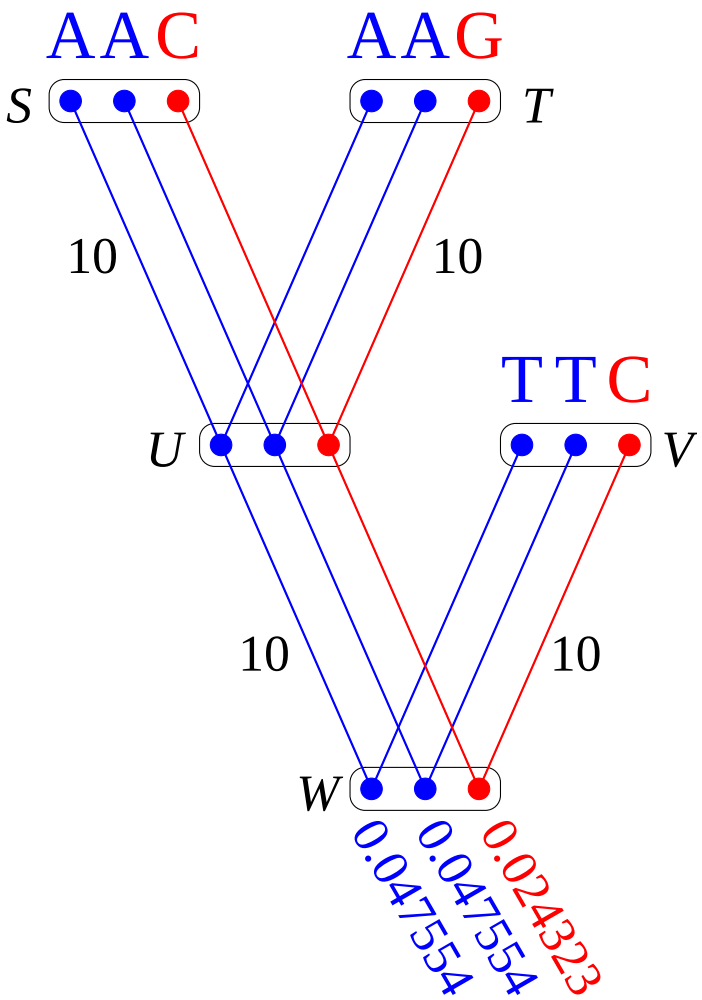
Likelihoods of Trees (multiple columns)



Considering this tree with $n = 3$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)}$$

Likelihoods of Trees (multiple columns)

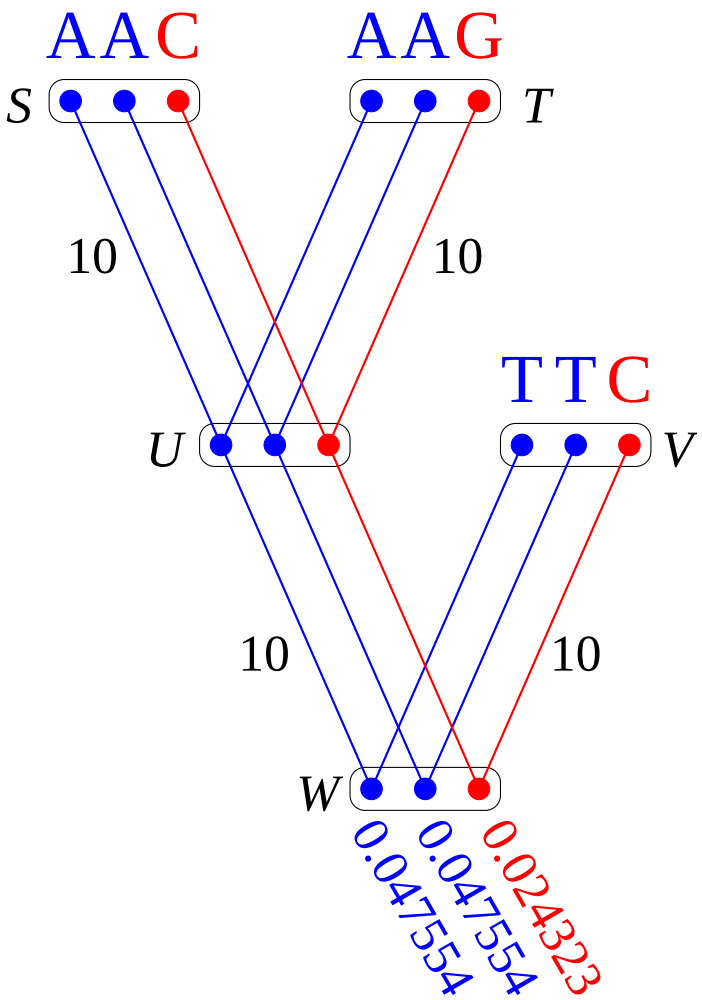


Considering this tree with $n = 3$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)} = 0.047554^2 \cdot 0.024323$$

$$= 0.000055$$

Likelihoods of Trees (multiple columns)



Considering this tree with $n = 3$ sequences of length $m = 3$ the tree likelihood of this tree is

$$\mathcal{L}(T) = \prod_{k=1}^{m} L^{(k)} = 0.047554^2 \cdot 0.024323$$

$$= 0.000055$$

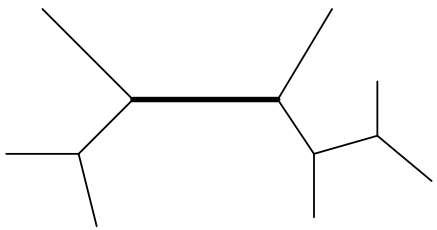
or the log-likelihood

$$\ln \mathcal{L}(T) = \sum_{k=1}^{m} \ln L^{(k)} = -9.80811$$

Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
Choose a branch (A.). Move the virtual root to an adjacent node (B.).
Compute all partial likelihoods recursively (C.). Adjust the branch length to maximize the likelihood value (D.).

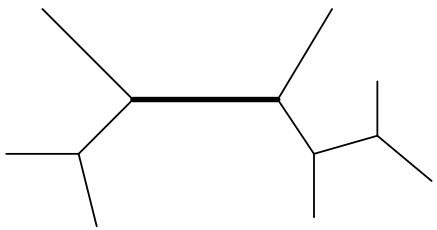
A.



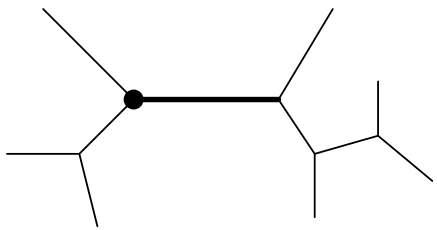
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
Choose a branch (A.). Move the virtual root to an adjacent node (B.).
Compute all partial likelihoods recursively (C.). Adjust the branch length to maximize the likelihood value (D.).

A.



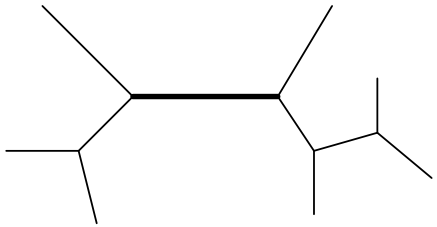
B.



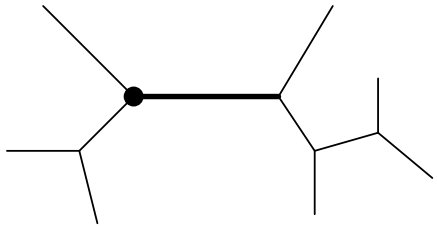
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
Choose a branch (A.). Move the virtual root to an adjacent node (B.).
Compute all partial likelihoods recursively (C.). Adjust the branch length to maximize the likelihood value (D.).

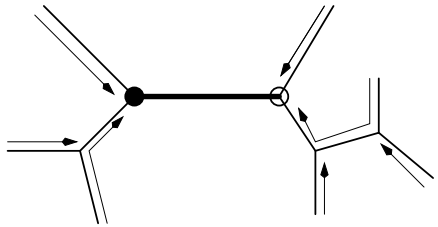
A.



B.

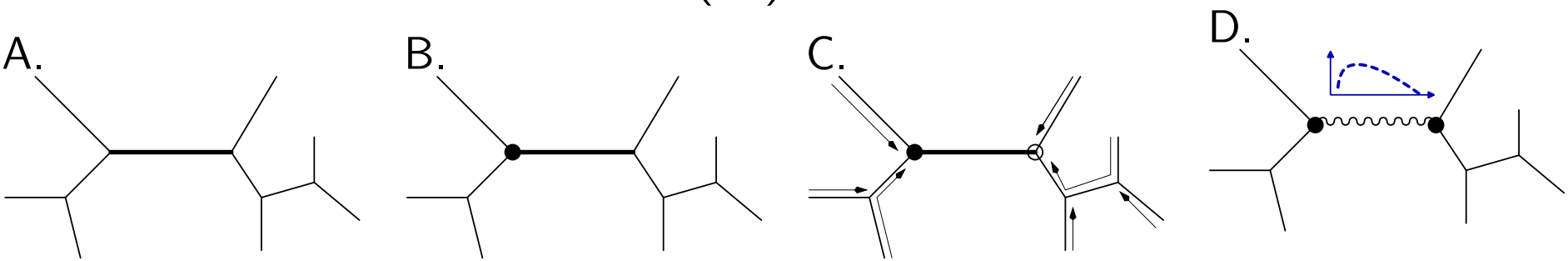


C.



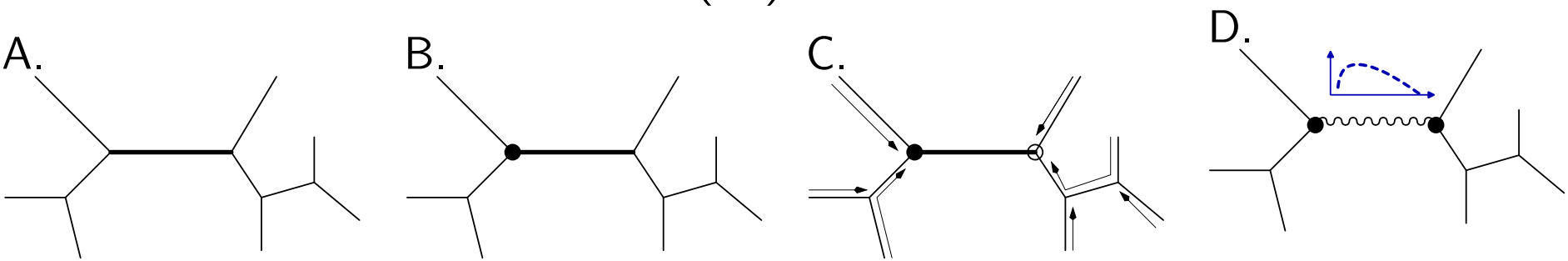
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
Choose a branch (A.). Move the virtual root to an adjacent node (B.).
Compute all partial likelihoods recursively (C.). Adjust the branch length to maximize the likelihood value (D.).



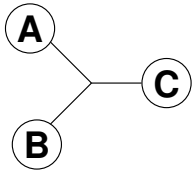
Adjusting Branch Lengths Step-By-Step

To compute optimal branch lengths do the following. Initialize the branch lengths.
Choose a branch (A.). Move the virtual root to an adjacent node (B.).
Compute all partial likelihoods recursively (C.). Adjust the branch length to maximize the likelihood value (D.).

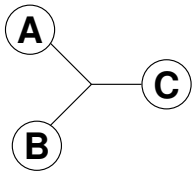


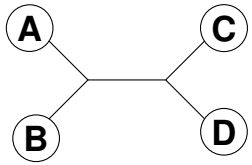
Repeat this for every branch until no better likelihood is gained.

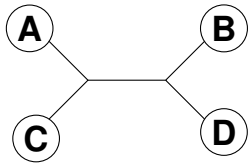
Number of Trees to Examine...

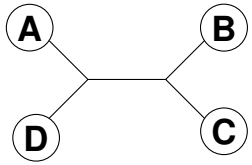


Number of Trees to Examine...

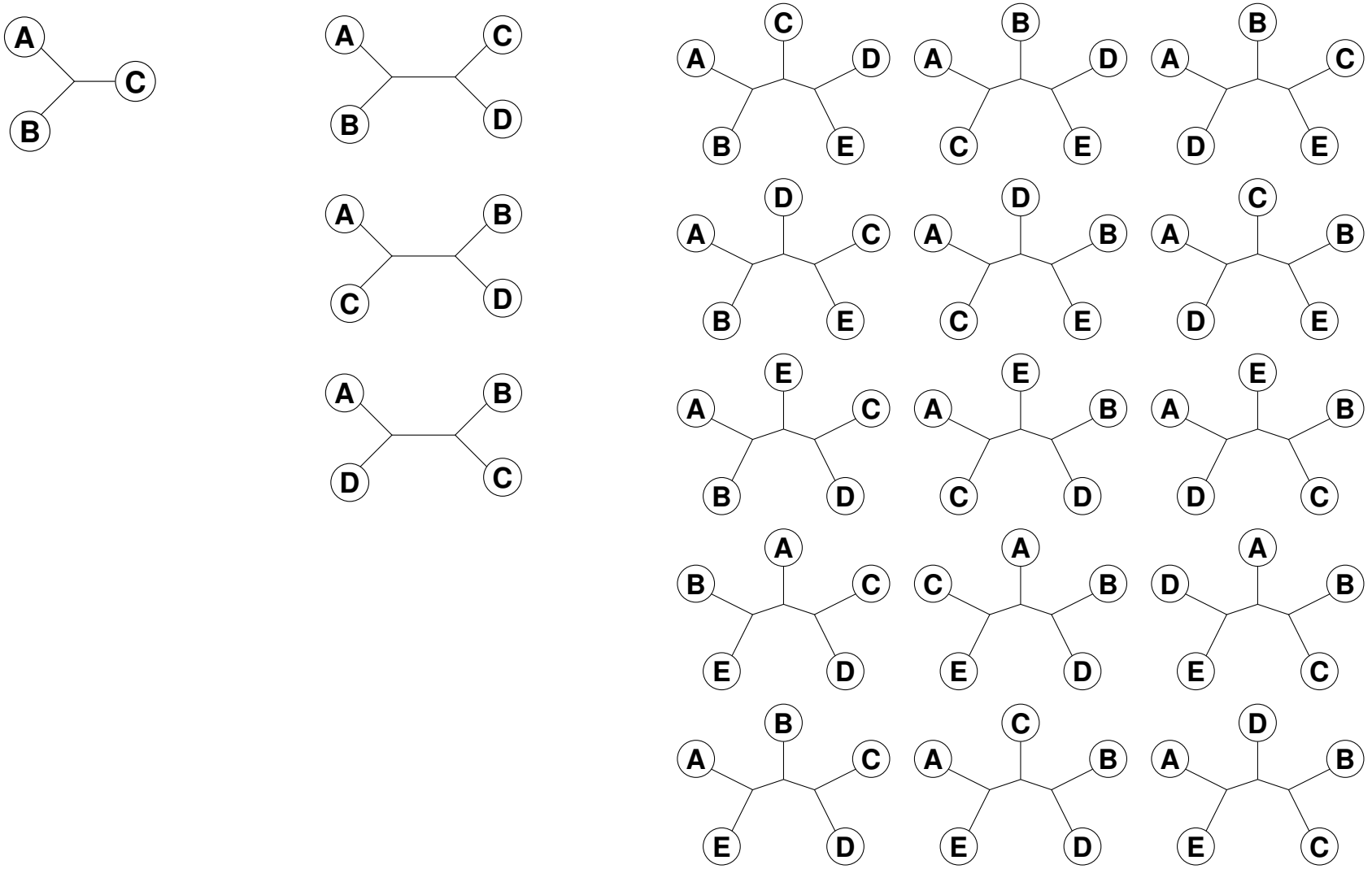




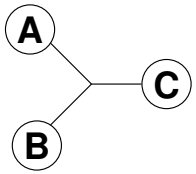


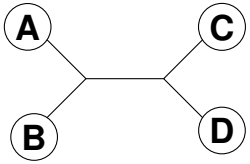


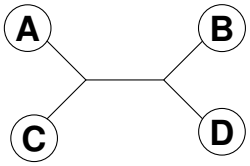
Number of Trees to Examine...

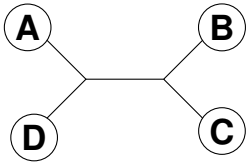


Number of Trees to Examine...

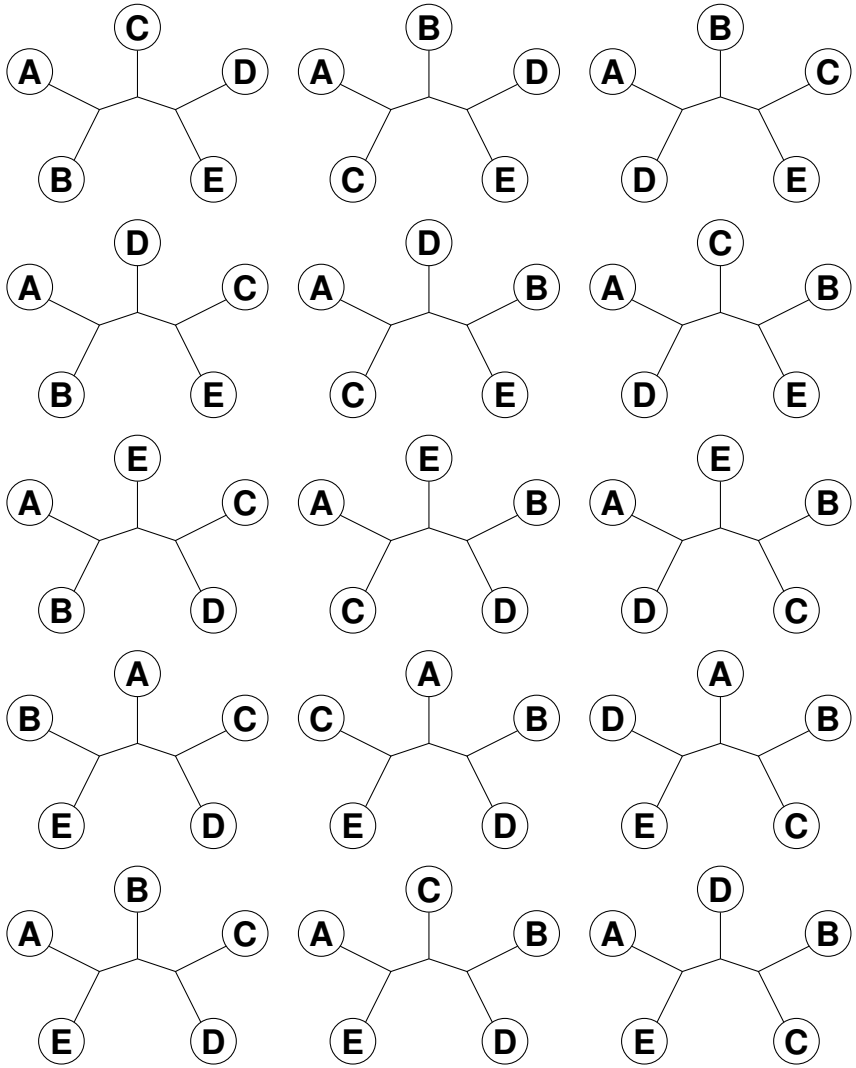




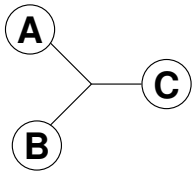


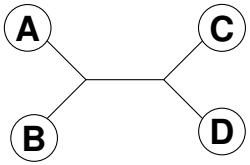


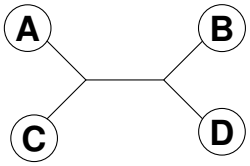
$$B(n) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

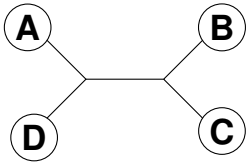


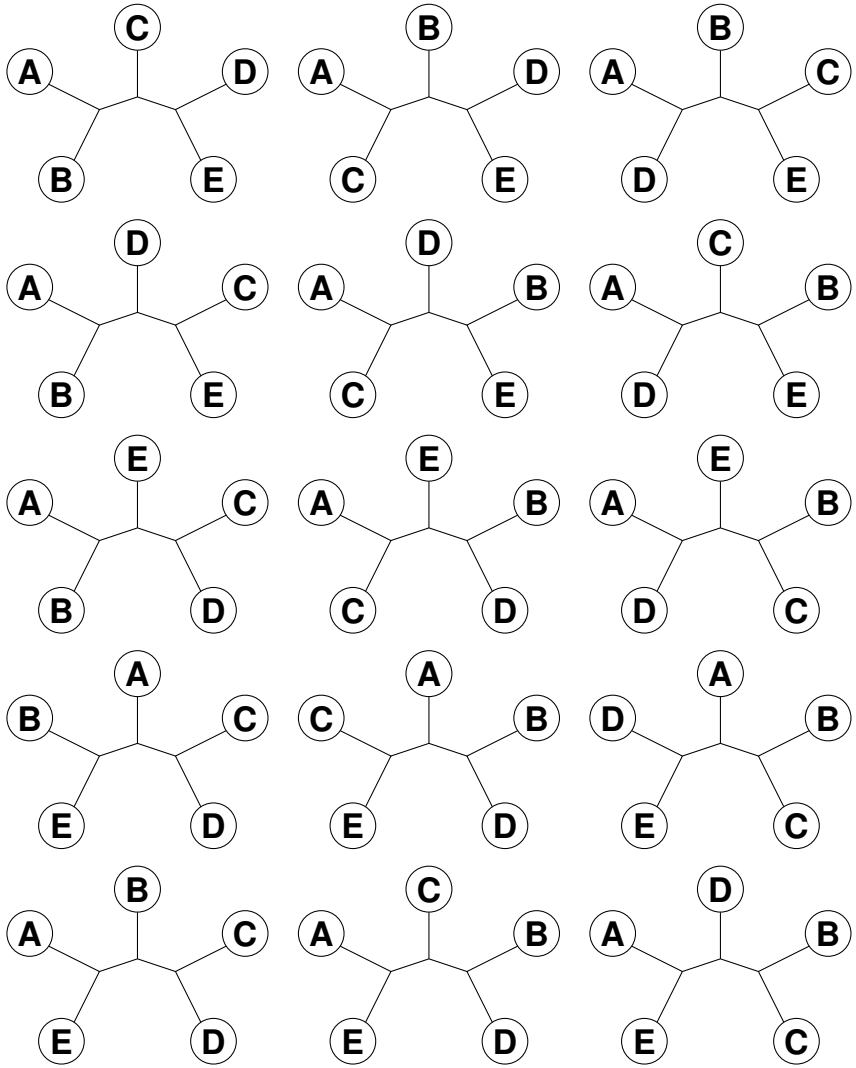
Number of Trees to Examine...











$$B(n) = \frac{(2n-5)!}{2^{n-3}(n-3)!}$$

$$B(10) = 2027025$$

$$B(55) = 2.98 \cdot 10^{84}$$

$$B(100) = 1.70 \cdot 10^{182}$$

Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

Branch and Bound: guarantees to find the optimal tree, without searching certain parts of the tree space – can run on more sequences, but often not for current-day datasets.

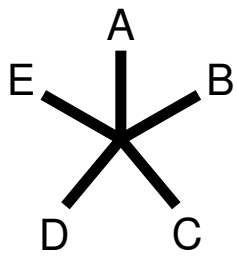
Finding the ML Tree

Exhaustive Search: guarantees to find the optimal tree, because all trees are evaluated, but not feasible for more than 10-12 taxa.

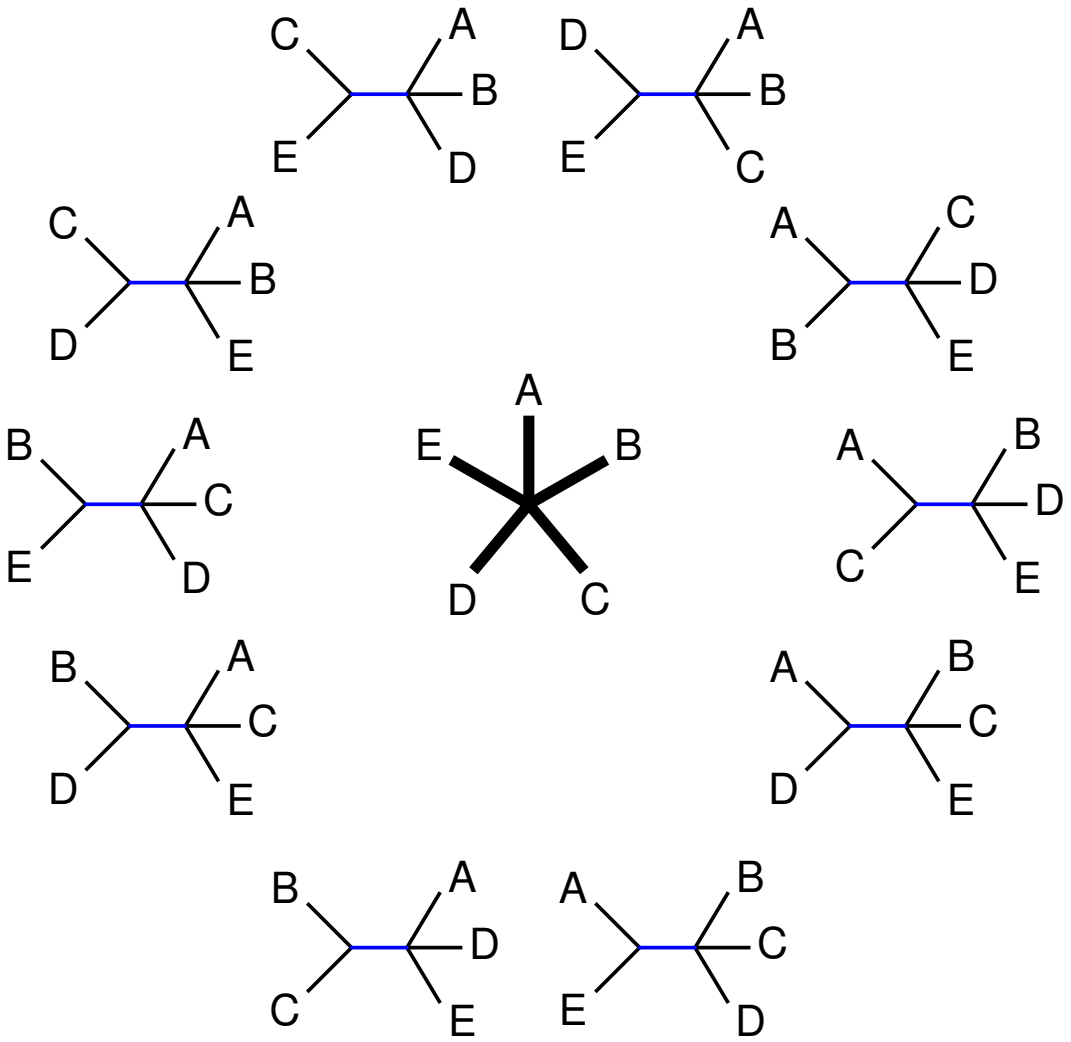
Branch and Bound: guarantees to find the optimal tree, without searching certain parts of the tree space – can run on more sequences, but often not for current-day datasets.

Heuristics: cannot guarantee to find the optimal tree, but are at least able to analyze large datasets.

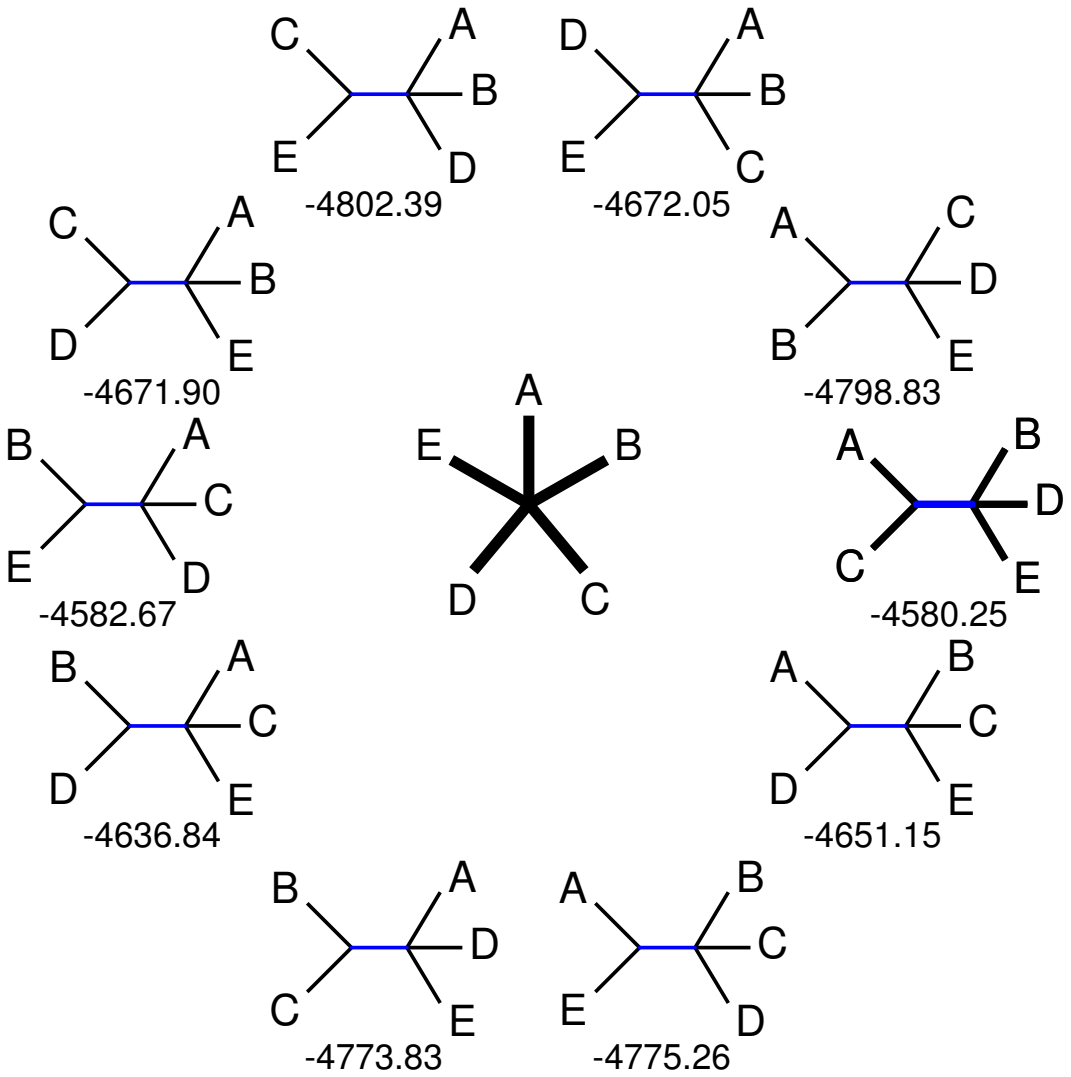
Build up a tree: Star Decomposition



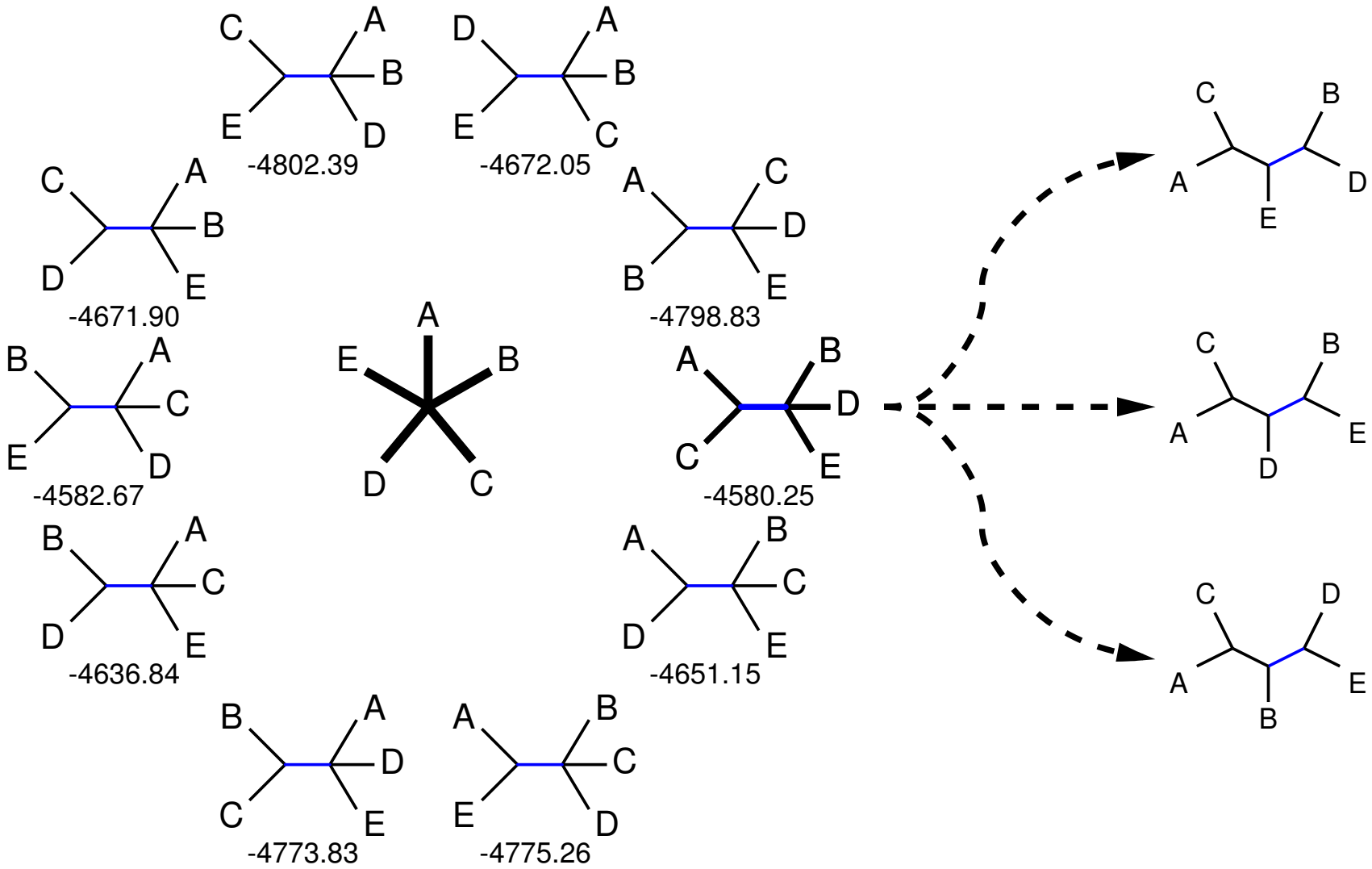
Build up a tree: Star Decomposition



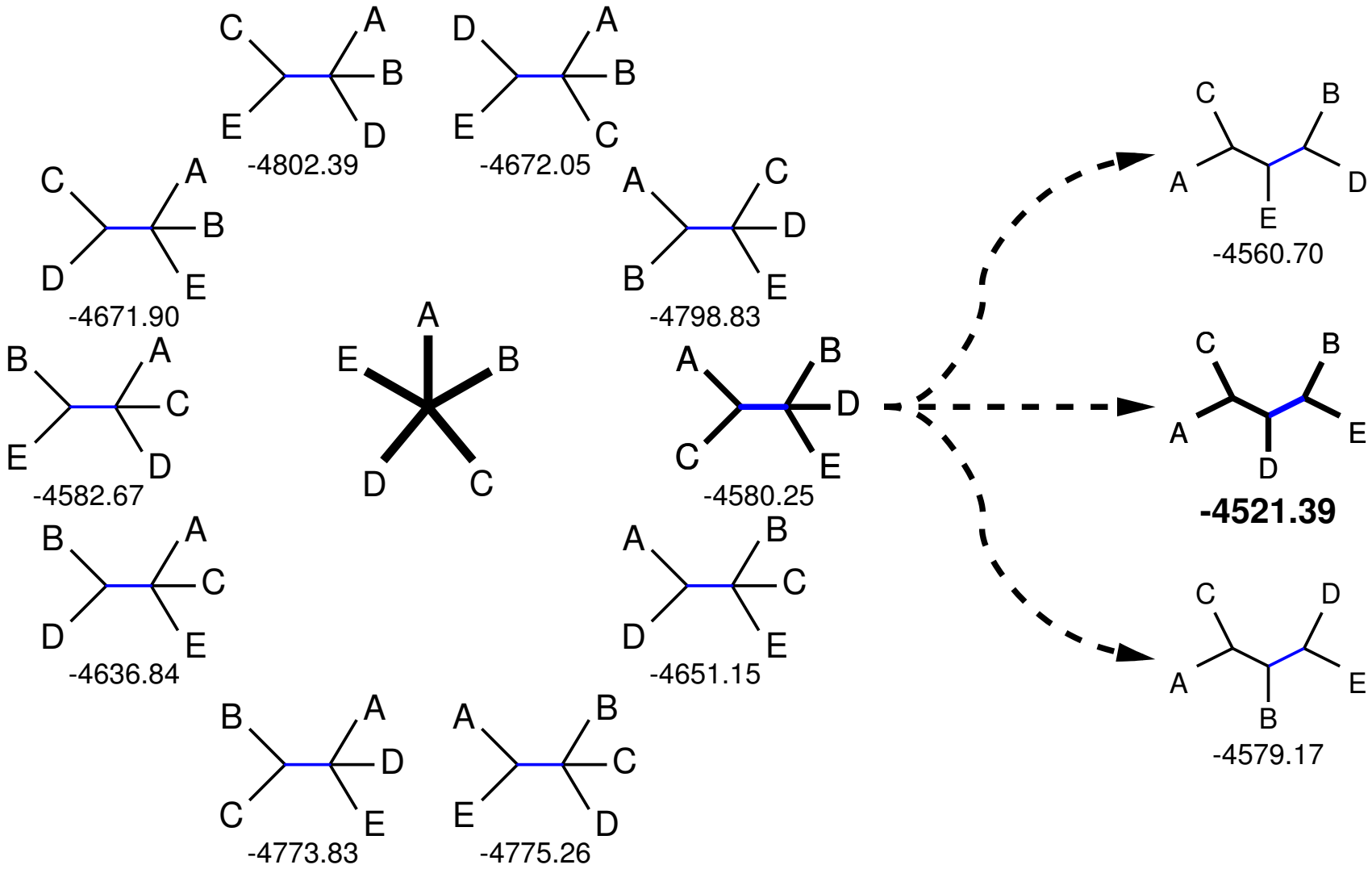
Build up a tree: Star Decomposition



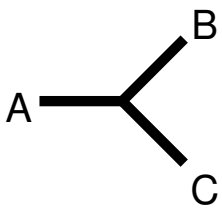
Build up a tree: Star Decomposition



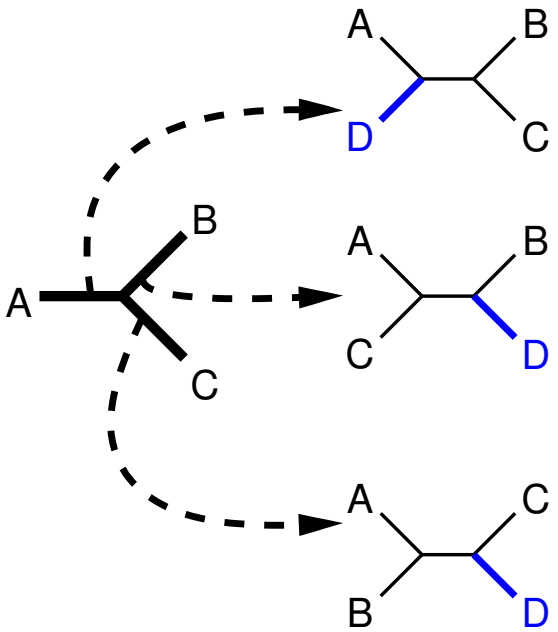
Build up a tree: Star Decomposition



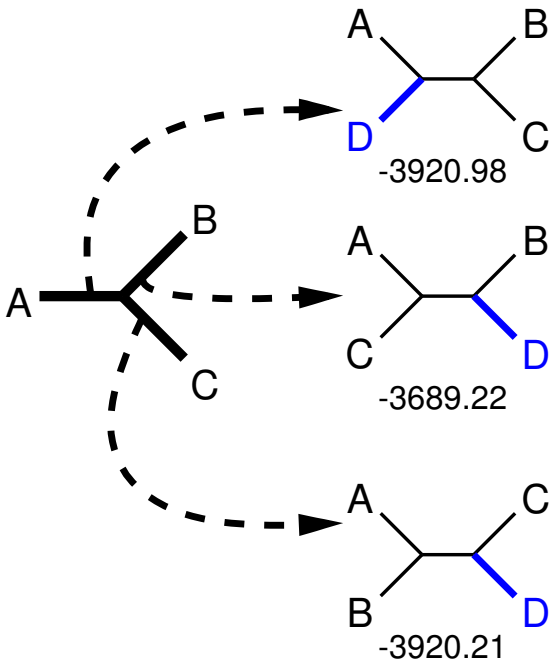
Build up a tree: Stepwise Insertion



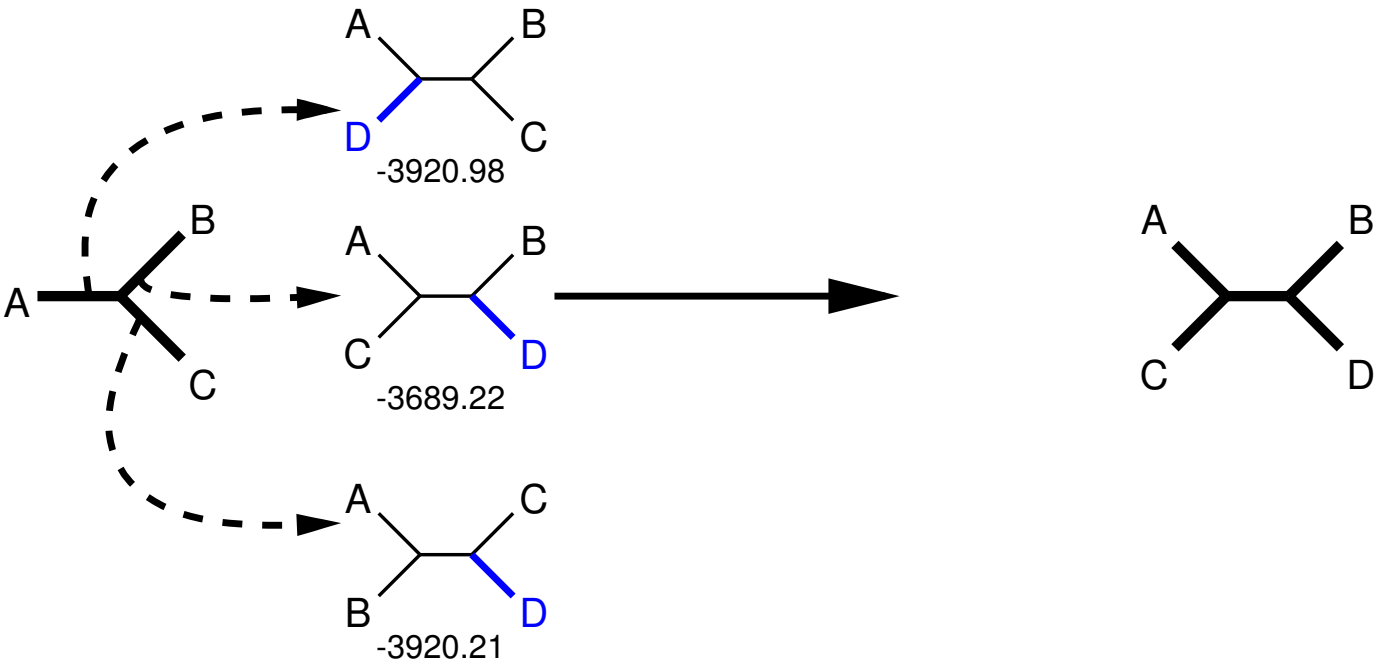
Build up a tree: Stepwise Insertion



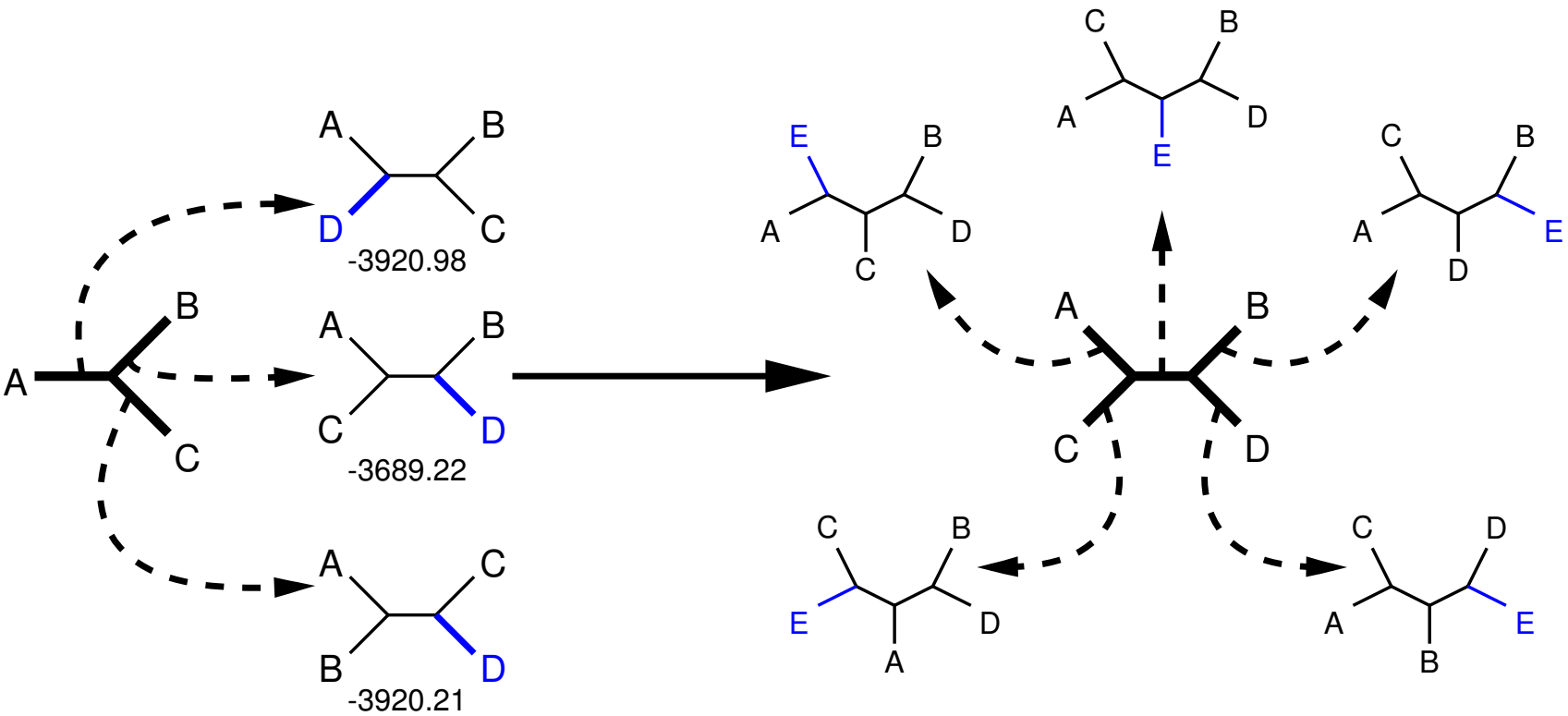
Build up a tree: Stepwise Insertion



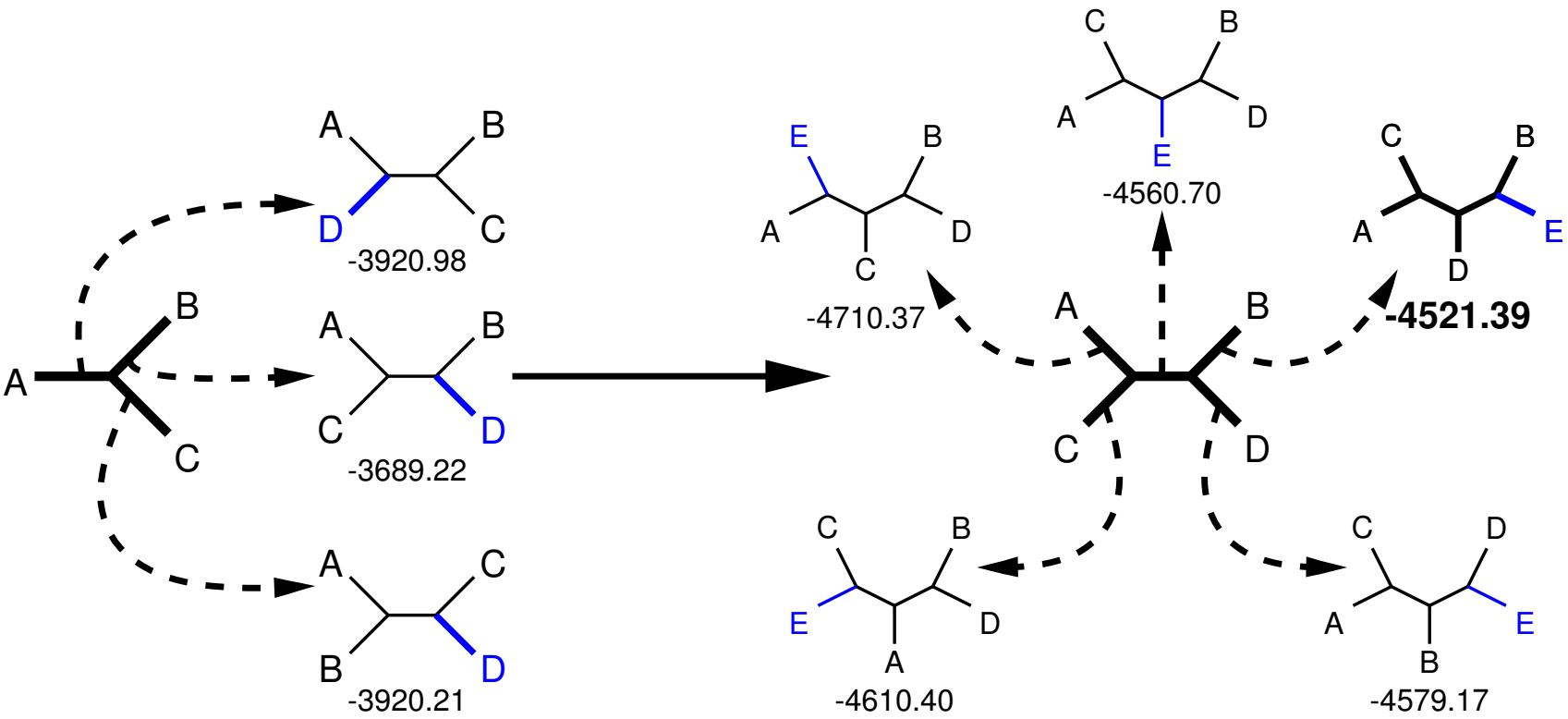
Build up a tree: Stepwise Insertion



Build up a tree: Stepwise Insertion

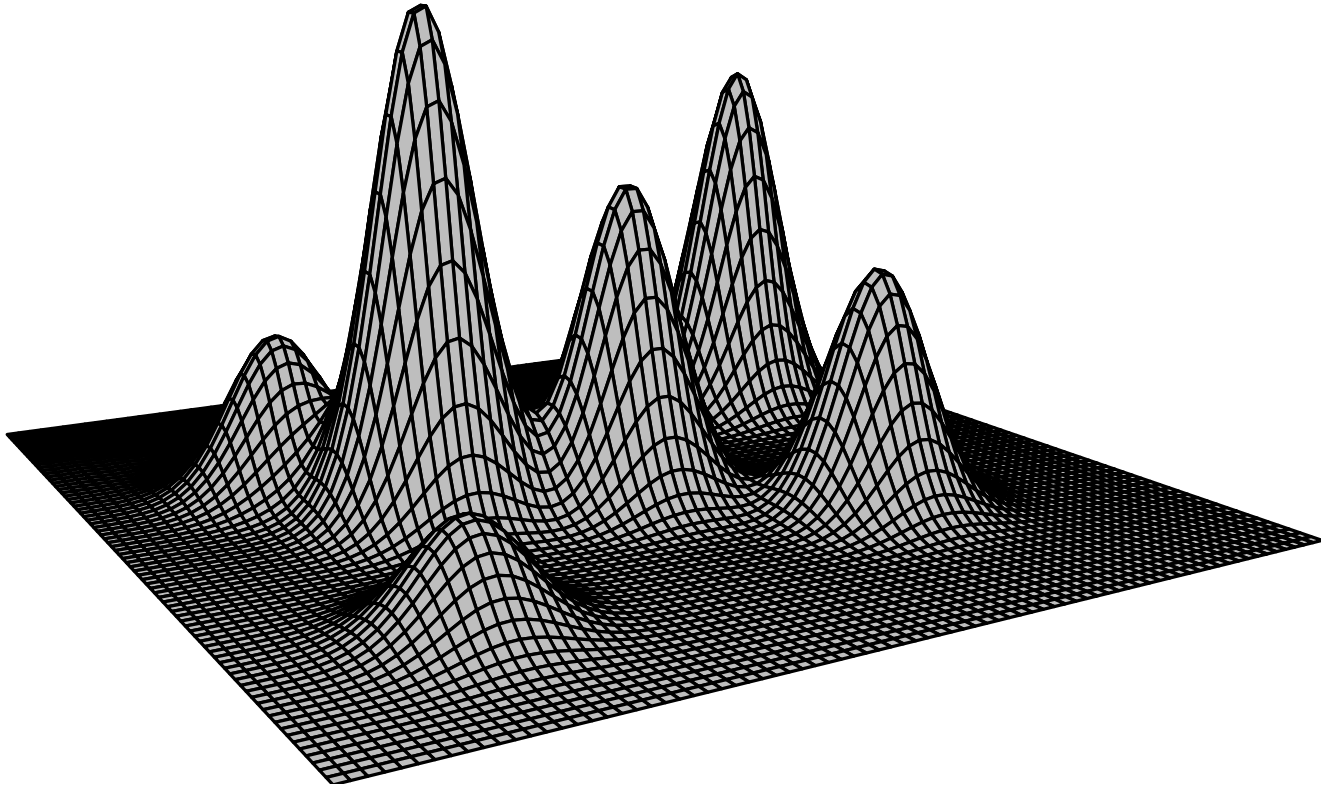


Build up a tree: Stepwise Insertion



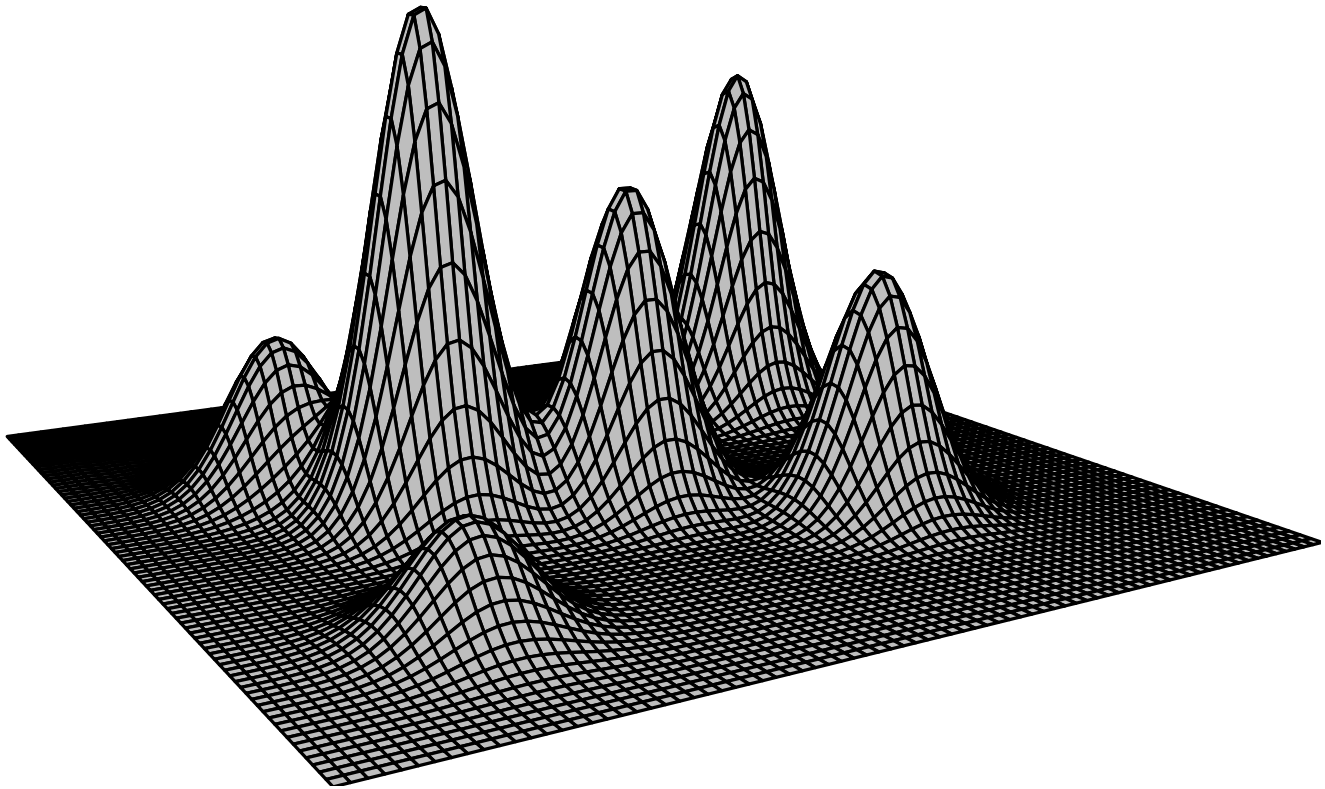
Local Maxima

What if we have multiple maxima in the likelihood surface?



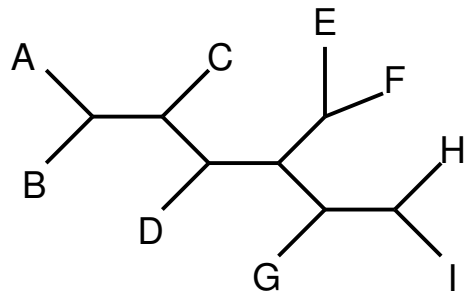
Local Maxima

What if we have multiple maxima in the likelihood surface?

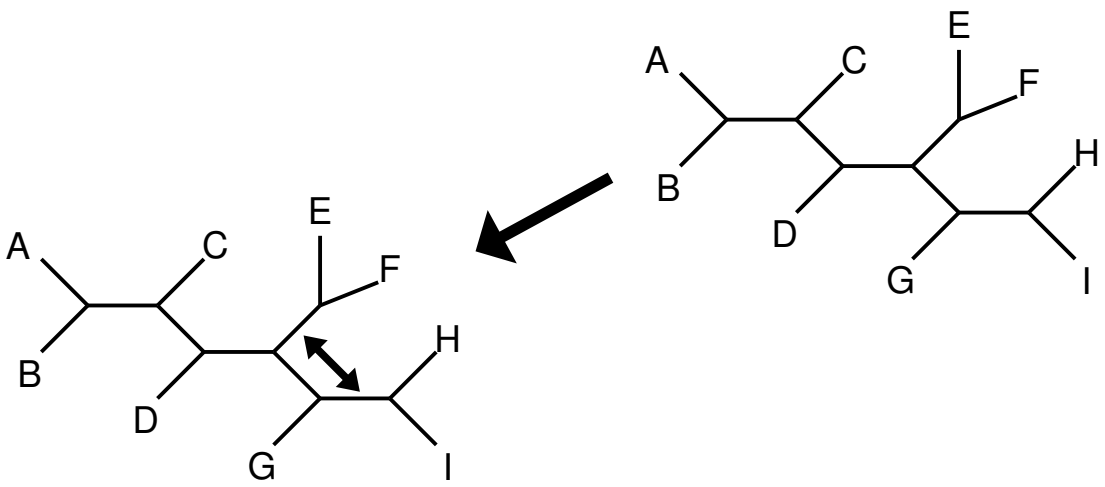


Tree rearrangements to escape local maxima.

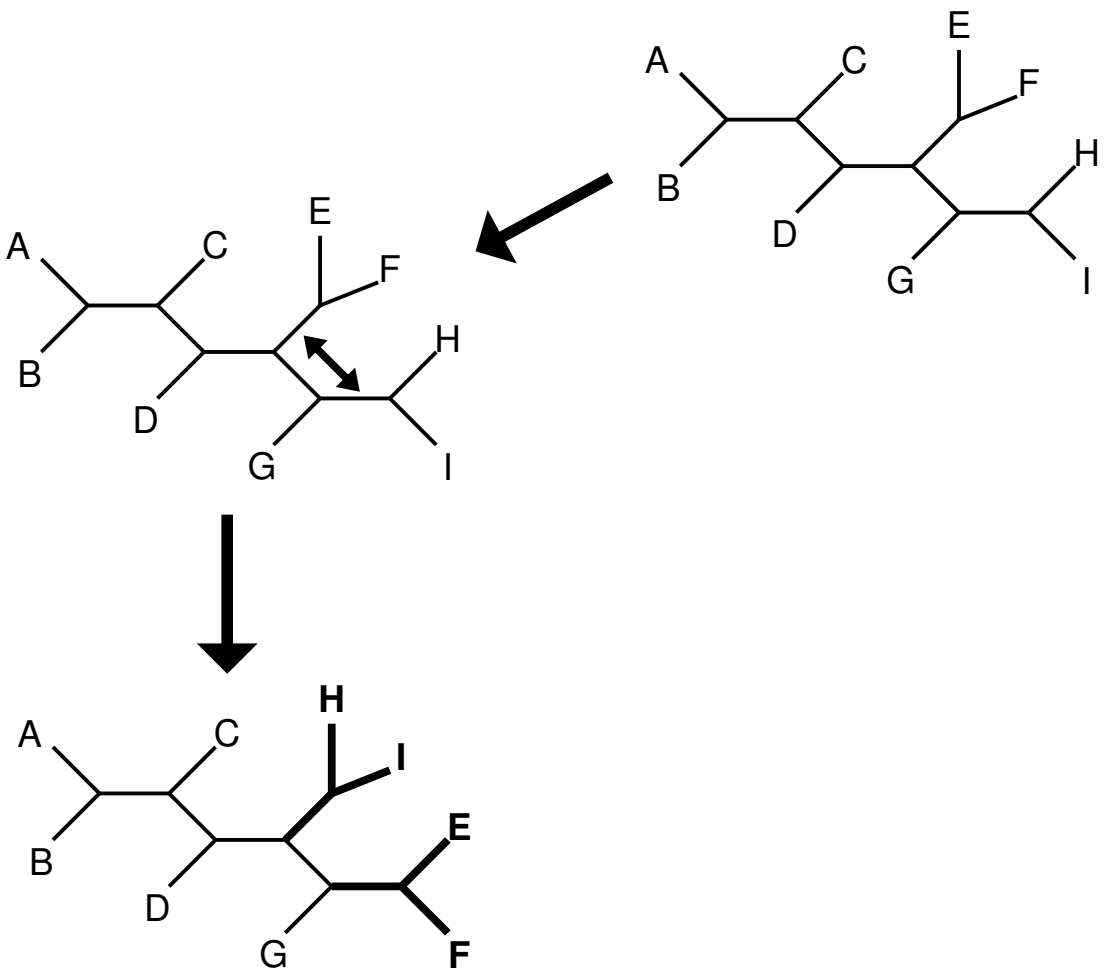
Tree Rearrangements: Scanning a Tree's Neighborhood



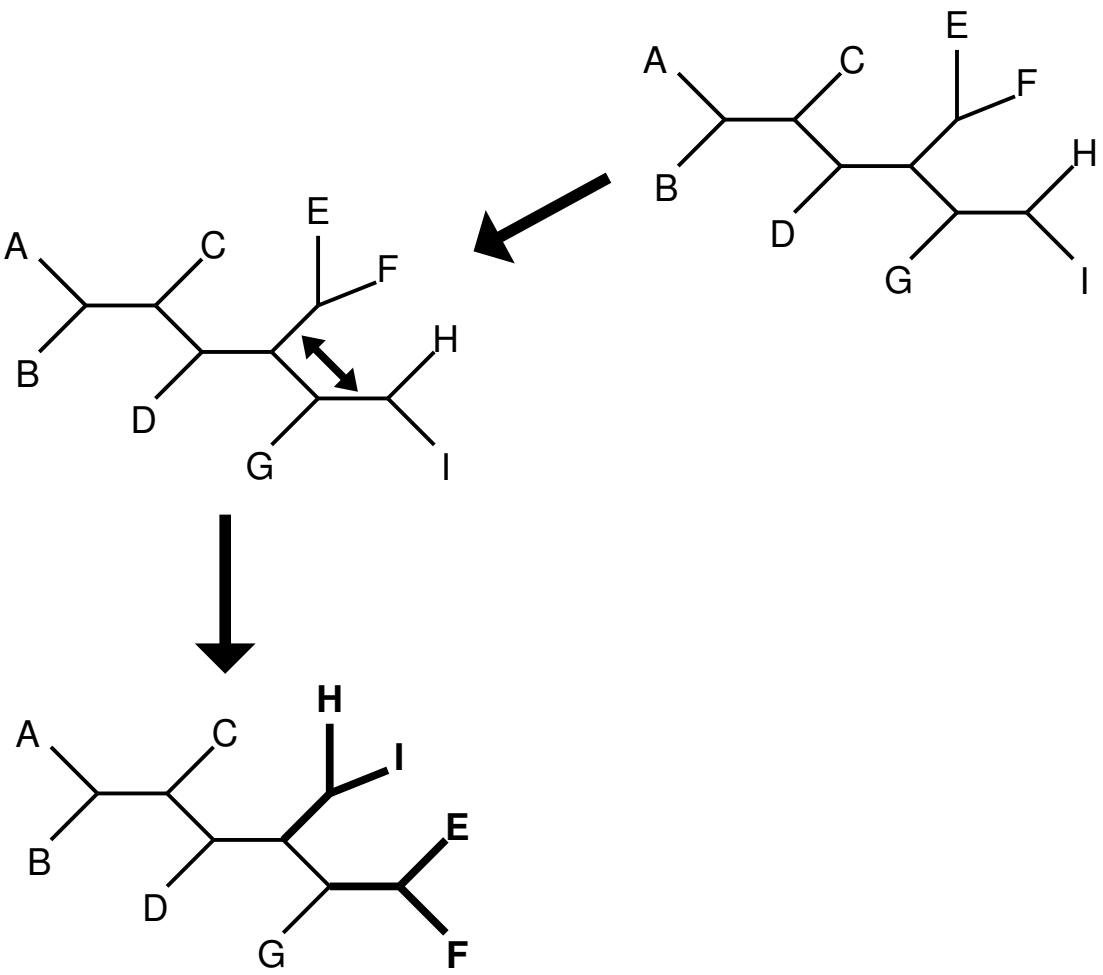
Tree Rearrangements: Scanning a Tree's Neighborhood



Tree Rearrangements: Scanning a Tree's Neighborhood



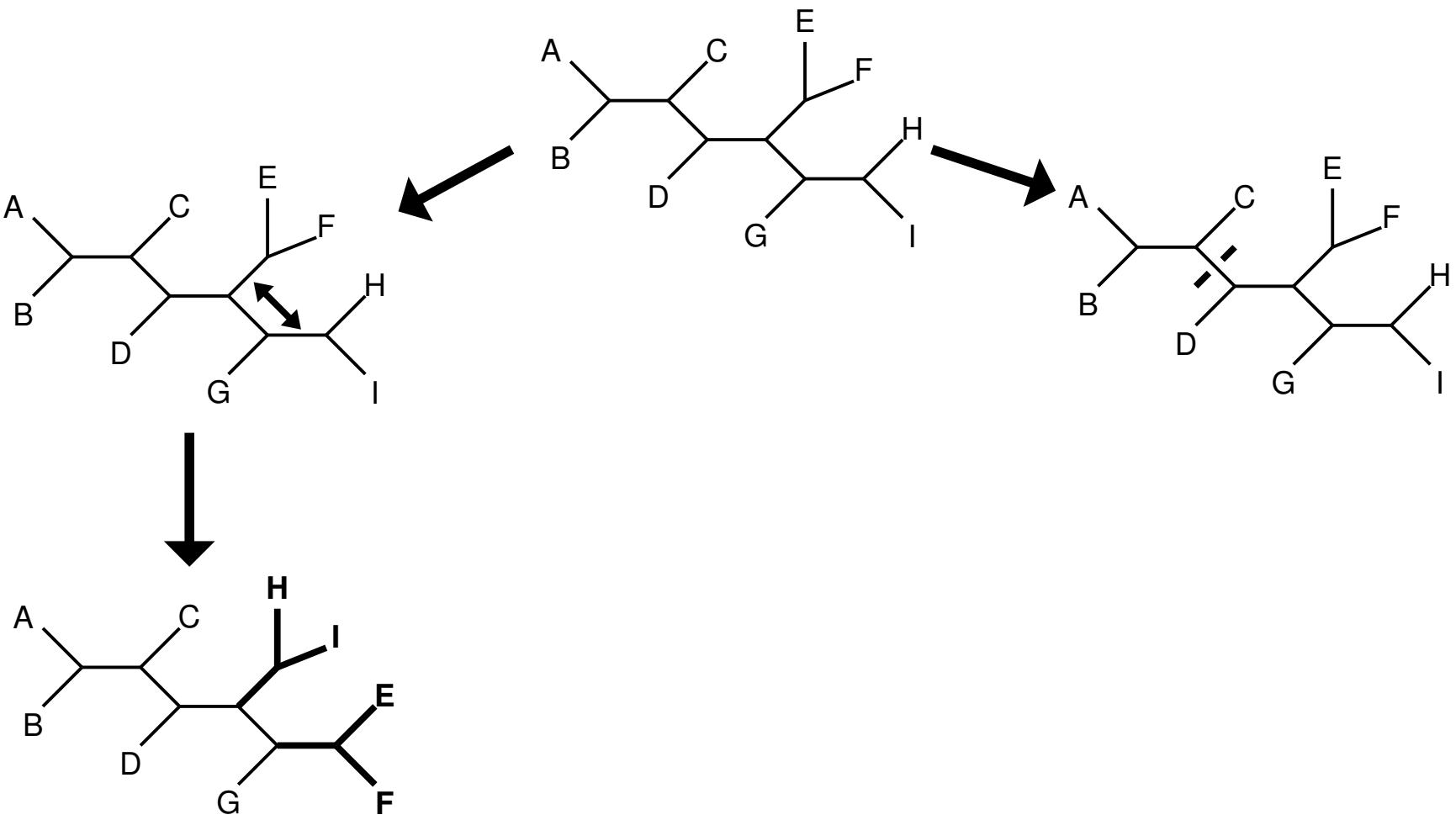
Tree Rearrangements: Scanning a Tree's Neighborhood



Nearest Neighbor Interchange

Possible NNI trees = O(n)

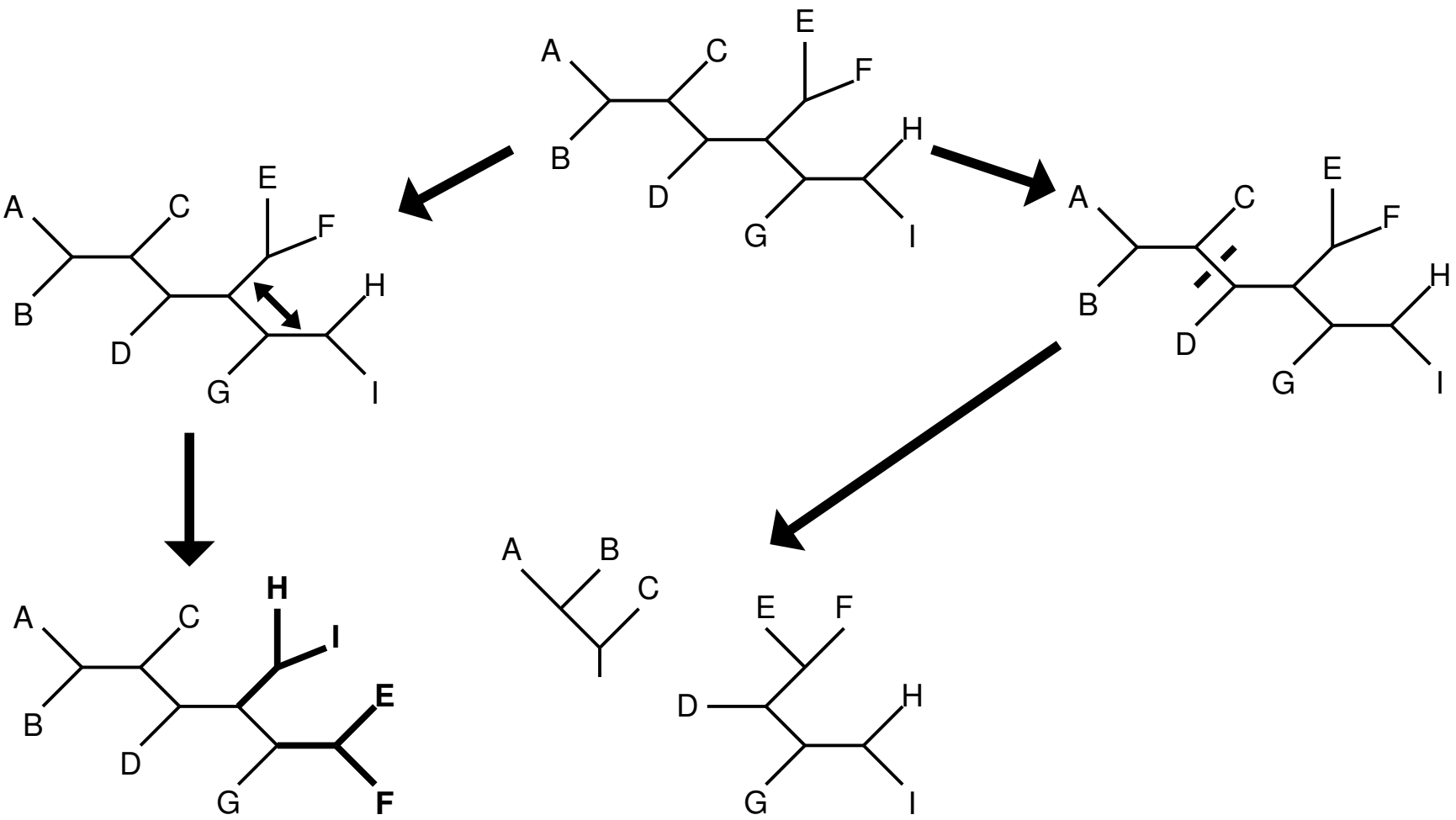
Tree Rearrangements: Scanning a Tree's Neighborhood



Nearest Neighbor Interchange

Possible NNI trees = O(n)

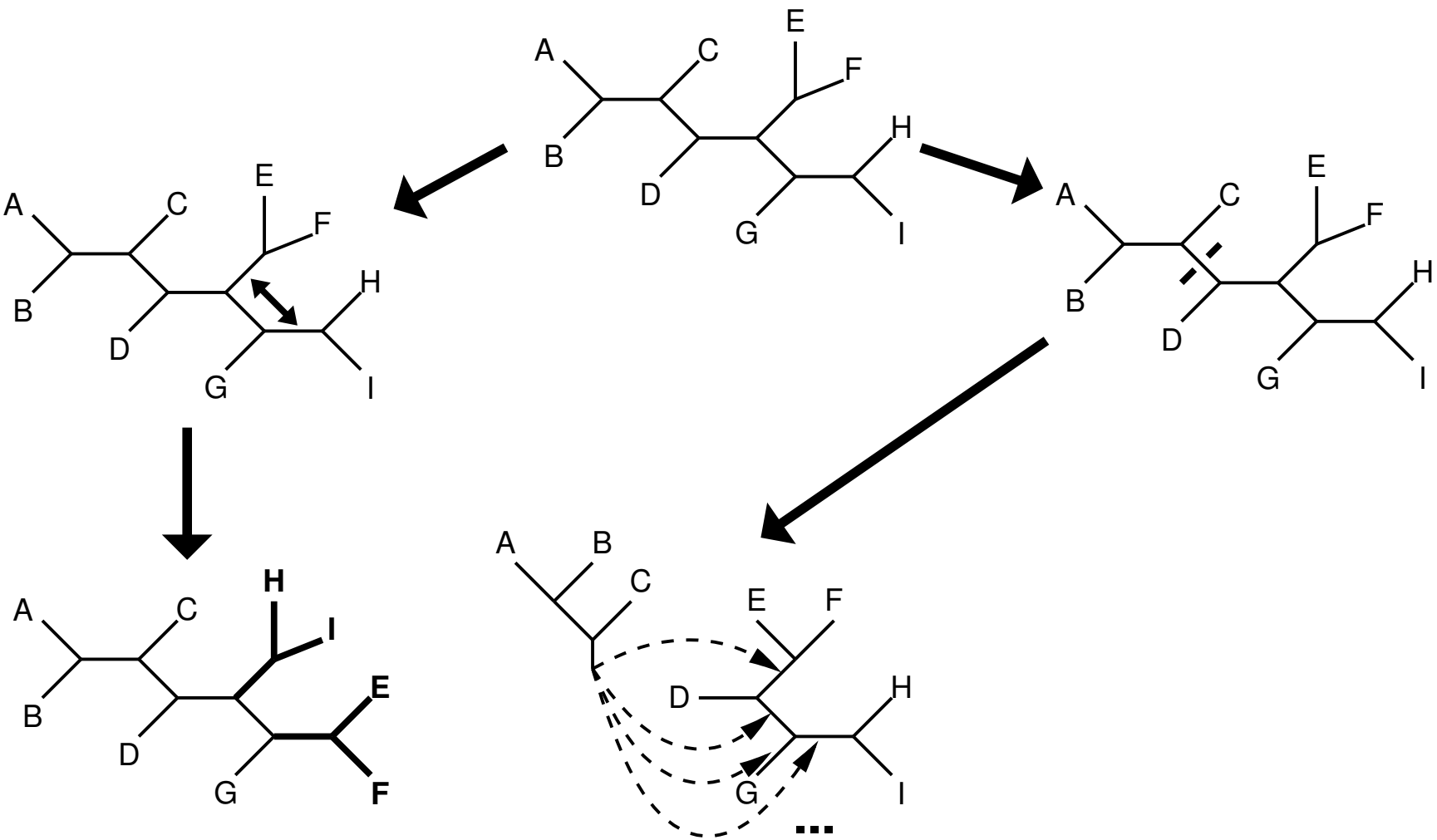
Tree Rearrangements: Scanning a Tree's Neighborhood



Nearest Neighbor Interchange

Possible NNI trees = O(n)

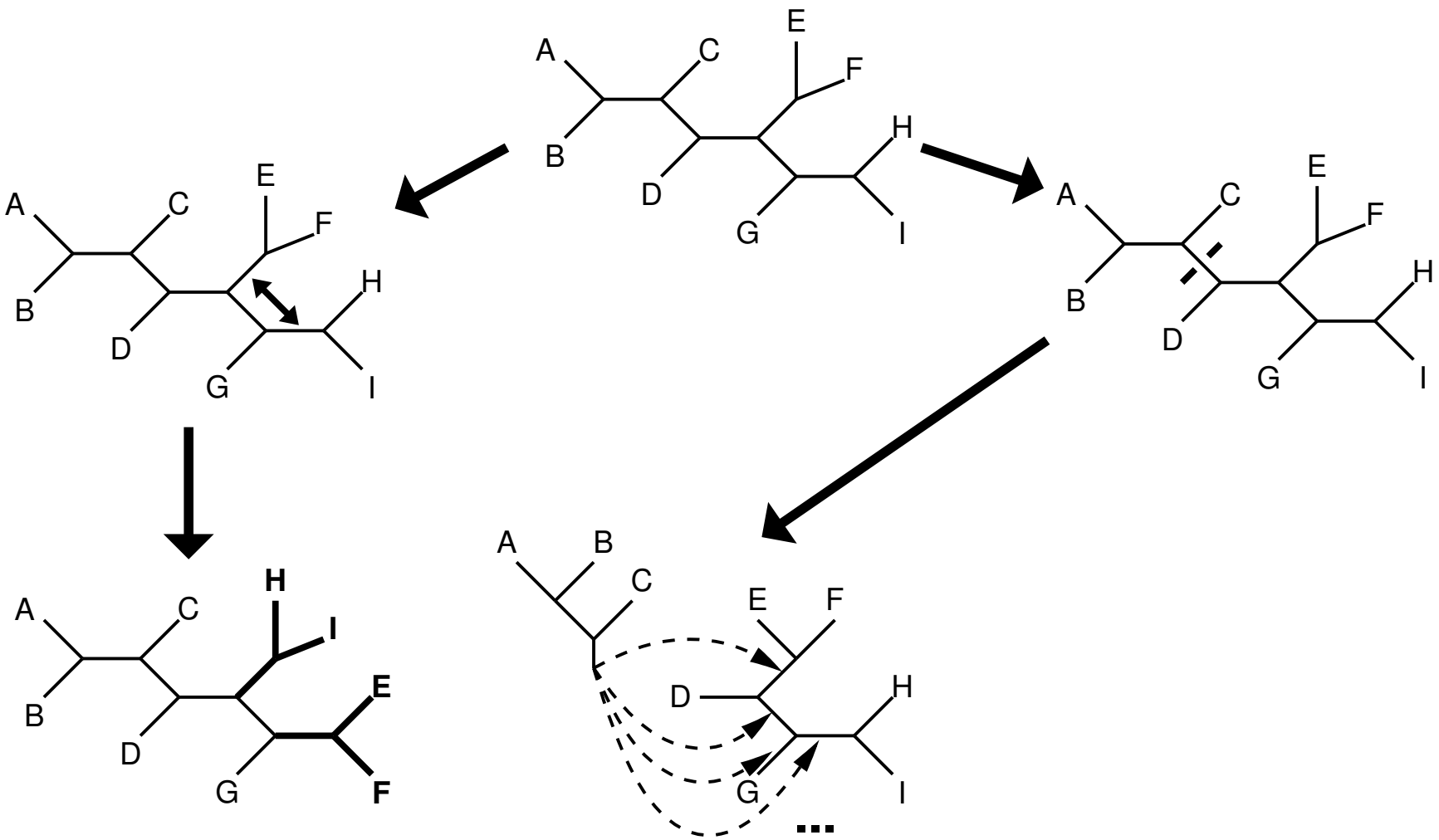
Tree Rearrangements: Scanning a Tree's Neighborhood



Nearest Neighbor Interchange

Possible NNI trees = O(n)

Tree Rearrangements: Scanning a Tree's Neighborhood



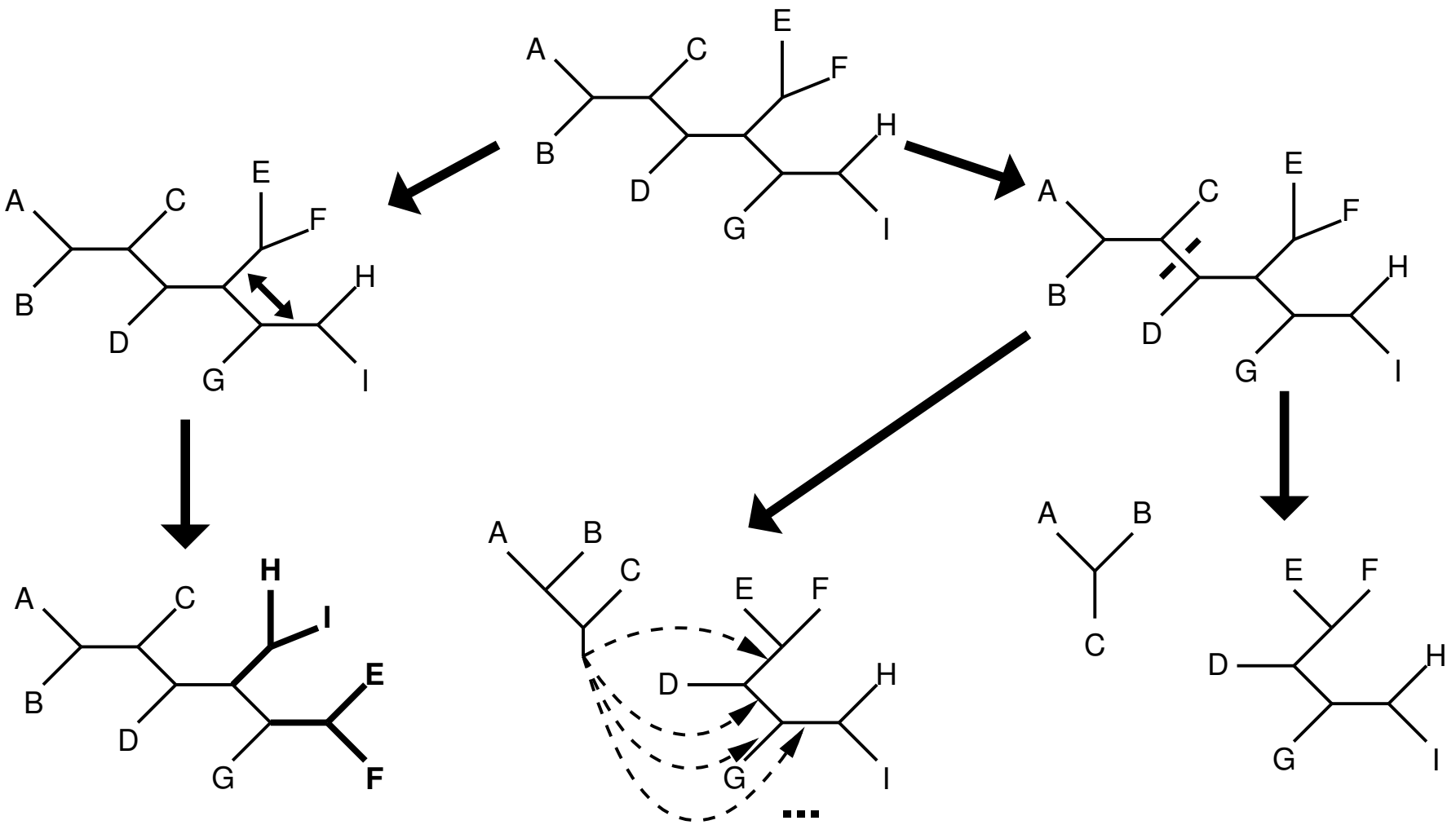
Nearest Neighbor Interchange

Possible NNI trees = O(n)

subtree pruning + regrafting

Possible SPR trees = O(n*n)

Tree Rearrangements: Scanning a Tree's Neighborhood



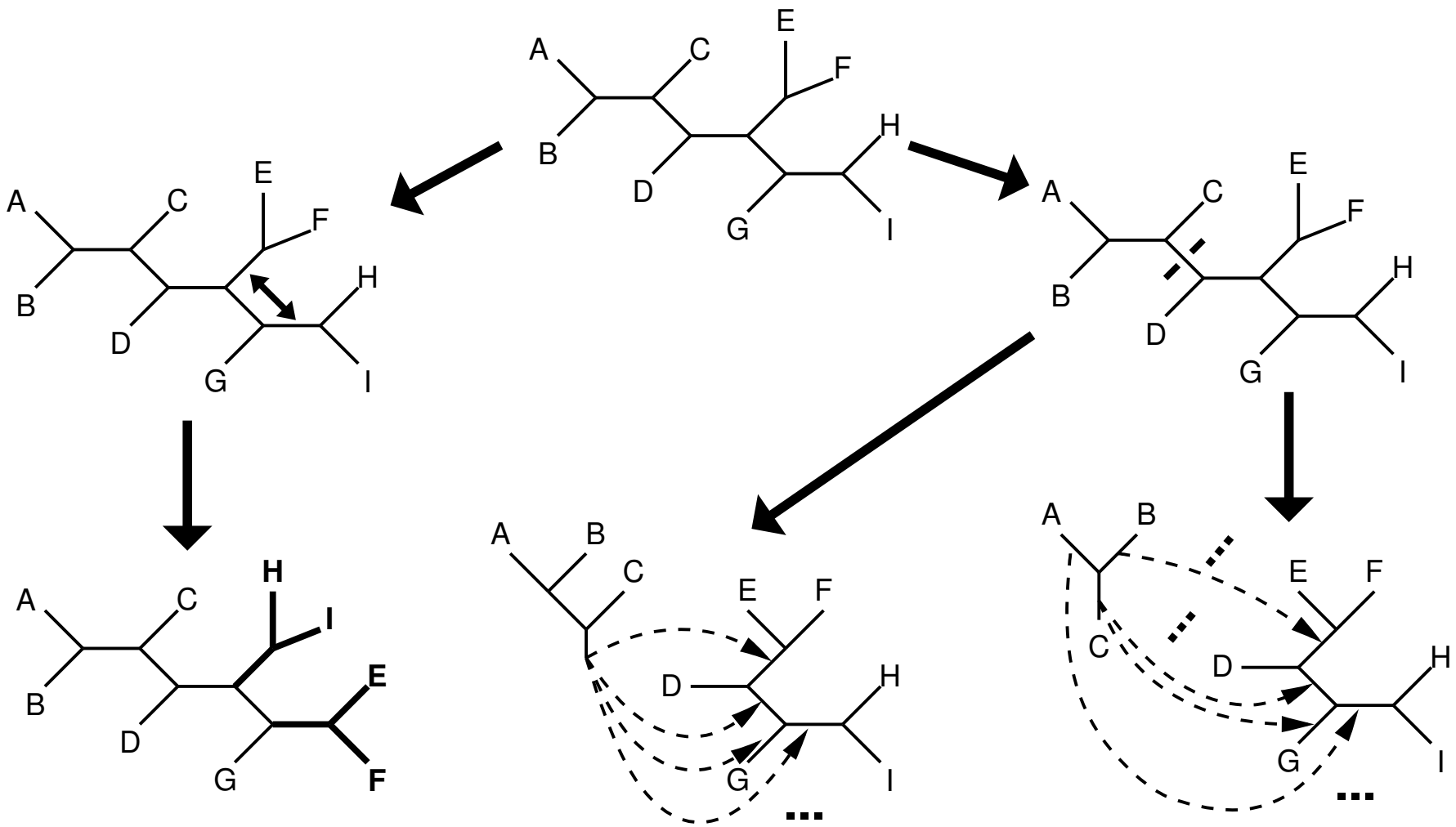
Nearest Neighbor Interchange

Possible NNI trees = O(n)

subtree pruning + regrafting

Possible SPR trees = O(n*n)

Tree Rearrangements: Scanning a Tree's Neighborhood



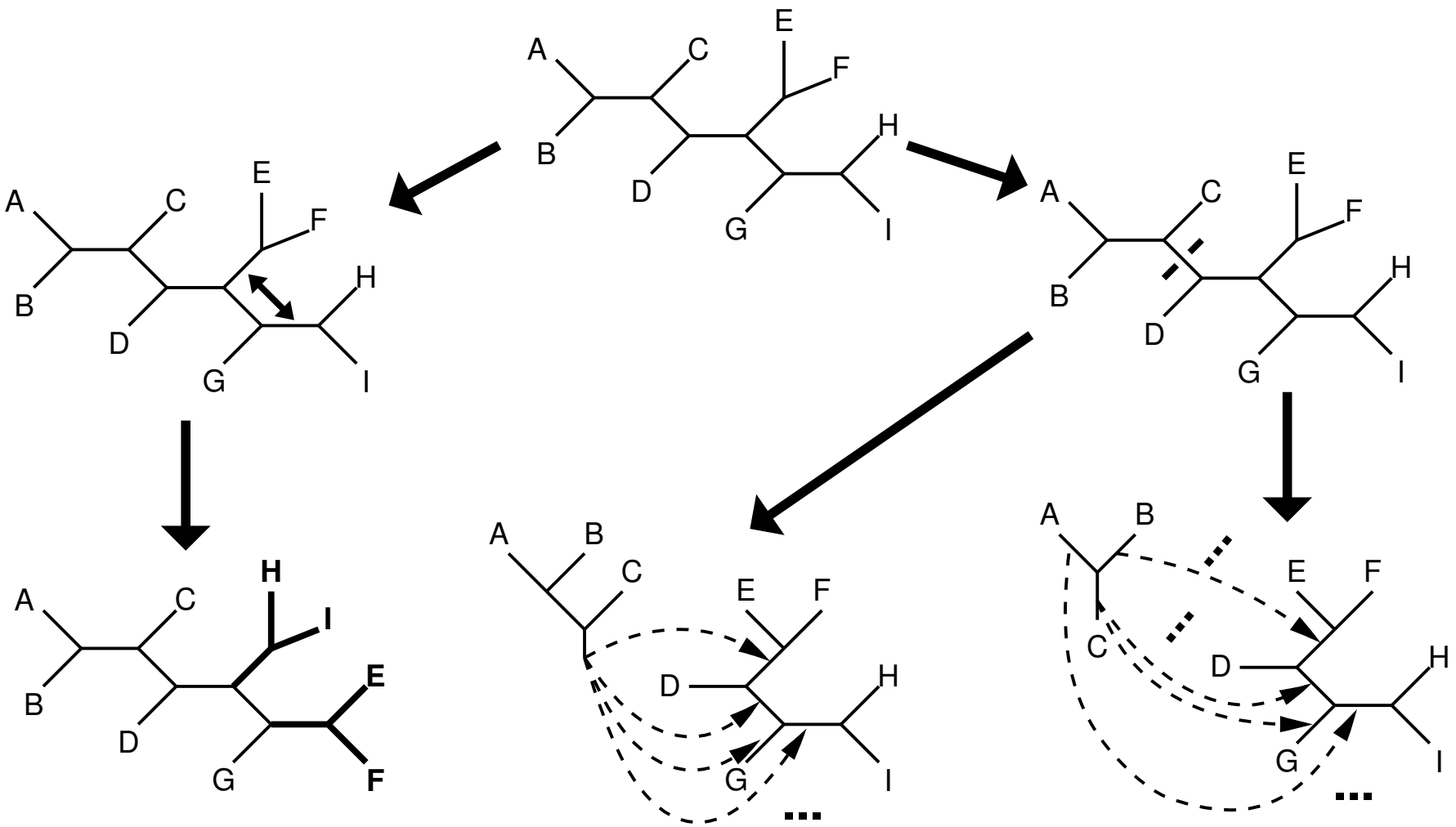
Nearest Neighbor Interchange

Possible NNI trees = O(n)

subtree pruning + regrafting

Possible SPR trees = O(n*n)

Tree Rearrangements: Scanning a Tree's Neighborhood



Nearest Neighbor Interchange

Possible NNI trees = O(n)

subtree pruning + regrafting

Possible SPR trees = O(n*n)

tree-bisection + reconnection

Possible TBR trees = O(n*n*n)

ML programs: DNAML (PHYLIP), fastDNAml

Concept: Stepwise insertion + NNI/SPR

1. Build tree with stepwise insertion
   (a) after each insertion optimize using NNI/local rearrangement (default, but user-adjustable gradually up to SPR; only fastDNAml)
   (b) repeat (a) rearrangements until no better tree found.

ML programs: DNAML (PHYLIP), fastDNAml

Concept: Stepwise insertion + NNI/SPR

1. Build tree with stepwise insertion
   (a) after each insertion optimize using NNI/local rearrangement (default, but user-adjustable gradually up to SPR; only fastDNAml)
   (b) repeat (a) rearrangements until no better tree found.
2. after the last insertion optimize using SPR/global rearrangement (in DNAML; in fastDNAml user-adjustable gradually down to NNI)
3. repeat (2) rearrangements until no better tree found.

ML programs: DNAML (PHYLIP), fastDNAml

Concept: Stepwise insertion + NNI/SPR

1. Build tree with stepwise insertion
   (a) after each insertion optimize using NNI/local rearrangement (default, but user-adjustable gradually up to SPR; only fastDNAml)
   (b) repeat (a) rearrangements until no better tree found.
2. after the last insertion optimize using SPR/global rearrangement (in DNAML; in fastDNAml user-adjustable gradually down to NNI)
3. repeat (2) rearrangements until no better tree found.

Pro: Evaluating large neighborhood with SPR.

Con: Slow.

ML programs: DNAML (PHYLIP), fastDNAml

Concept: Stepwise insertion + NNI/SPR

1. Build tree with stepwise insertion
   (a) after each insertion optimize using NNI/local rearrangement (default, but user-adjustable gradually up to SPR; only fastDNAml)
   (b) repeat (a) rearrangements until no better tree found.
2. after the last insertion optimize using SPR/global rearrangement (in DNAML; in fastDNAml user-adjustable gradually down to NNI)
3. repeat (2) rearrangements until no better tree found.

Pro: Evaluating large neighborhood with SPR.

Con: Slow.

Note: To save time, in other methods steps (1) and (2) are usually substituted by swiftly computed trees (e.g., BioNJ).

ML programs: MOLPHY

Concept: Star decomposition + NNI

1. Build tree with star decomposition
2. after the last insertion optimize using NNI/local rearrangement
3. repeat rearrangements (2) until no better tree found.

Con: Slow, not maintained anymore.

ML programs: MOLPHY

Concept: Star decomposition + NNI

1. Build tree with star decomposition
2. after the last insertion optimize using NNI/local rearrangement
3. repeat rearrangements (2) until no better tree found.

Con: Slow, not maintained anymore.

ML programs: RAxML

Concept: MP tree + LSR

Descendant on fastDNAml, but ...

1. Starting with MP tree.
2. Uses *lazy subtree rearrangements* (only the 3 insertion branches are optimized), collecting candidates.
3. Candidates are evaluated.
4. Iterating (2)-(4).

ML programs: RAxML

Concept: MP tree + LSR

Descendant on fastDNAml, but ...

1. Starting with MP tree.
2. Uses *lazy subtree rearrangements* (only the 3 insertion branches are optimized), collecting candidates.
3. Candidates are evaluated.
4. Iterating (2)-(4).

Pro: Fast,
smart algorithmic and numerical optimized ML computation.

Con: Only few trees fully evaluated trees.

ML programs: PHYML

Concept: BioNJ tree + fastNNI

1. Start with BioNJ tree.
2. Do fastNNIs to optimize trees, i.e., evaluate all NNIs simultaneously and then merge all best ones which are non-conflicting.
3. Repeat (1) until no better tree found anymore.

ML programs: PHYML

Concept: BioNJ tree + fastNNI

1. Start with BioNJ tree.
2. Do fastNNIs to optimize trees, i.e., evaluate all NNIs simultaneously and then merge all best ones which are non-conflicting.
3. Repeat (1) until no better tree found anymore.

Pro: Fast

Con: Prone to get stuck on local optima due to NNI-only.
(SPR-based version PhyML-SPR has not been released yet.)

ML programs: PHYML-SPR

1. Start with BioNJ tree.
2. Evaluate SPR by fast non-ML criterion to find best candidates.
3. Evaluate the candidate(s) more rigorously with ML and fastNNI.
4. Repeat until no better tree found anymore.

Pro: less prone to get stuck on local optima.

Con: software not released yet.

ML programs: IQPNNI

Concept: BioNJ tree + randomization + fastNNI

1. Start with BioNJ tree.
2. Do fastNNIs to optimize trees, i.e., evaluate all NNIs simultaneously and then accept all best ones which are non-conflicting. (after first round, identical to PHYML).
3. Remove randomly a certain amount of taxa and re-insert them by a fast and rough quartet-based method. (some randomization)
4. Repeat (2)-(3) until stop criterion is met.

ML programs: IQPNNI

Concept: BioNJ tree + randomization + fastNNI

1. Start with BioNJ tree.
2. Do fastNNIs to optimize trees, i.e., evaluate all NNIs simultaneously and then accept all best ones which are non-conflicting. (after first round, identical to PHYML).
3. Remove randomly a certain amount of taxa and re-insert them by a fast and rough quartet-based method. (some randomization)
4. Repeat (2)-(3) until stop criterion is met.

Pro: Can evade local optima,
offers automatic stopping criterion,
hints when search didn't run enough,
numerically optimized ML computation,
offers codon models

Con: slower than PhyML/RAxML

ML programs: Genetic Algorithms (GARLI, MetaPIGA)

1. Start with some (random) tree.
2. View tree topology, branch lengths, and model parameter as part of a 'genome'.
3. Evolve the 'genome' by mutating (slightly changing) its parts.
4. Accept or reject new tree topologies from a pool of suggested trees according to their likelihood.
5. Iterate (3) and (4)

ML programs: Simulated Annealing

- Start with some (random) tree.
- Start a '*hot chain*' to suggest tree topologies (being far away).
- Accept proposals according to their likelihood.
- Cool down the chain, until the suggestions end up in some (local) optimum.

How reliable is the reconstructed tree:

- Usually programs deliver a single tree, but without confidence values for the subtrees.
- How can we assess reliability for the subtree?

Quartet Puzzling

The Quartet Puzzling algorithm implemented in the TREE-PUZZLE program is a three step procedure:

maximum-likelihood step: compute ML trees for all quartets of an alignment.

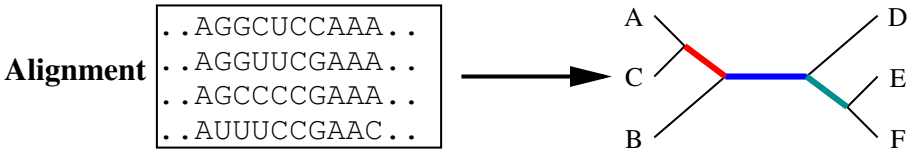
puzzling step: compose intermediate tree from quartet trees (this is done multiple times).

consensus step: construct a majority rule consensus tree from the intermediate trees and evaluate the branch lengths.

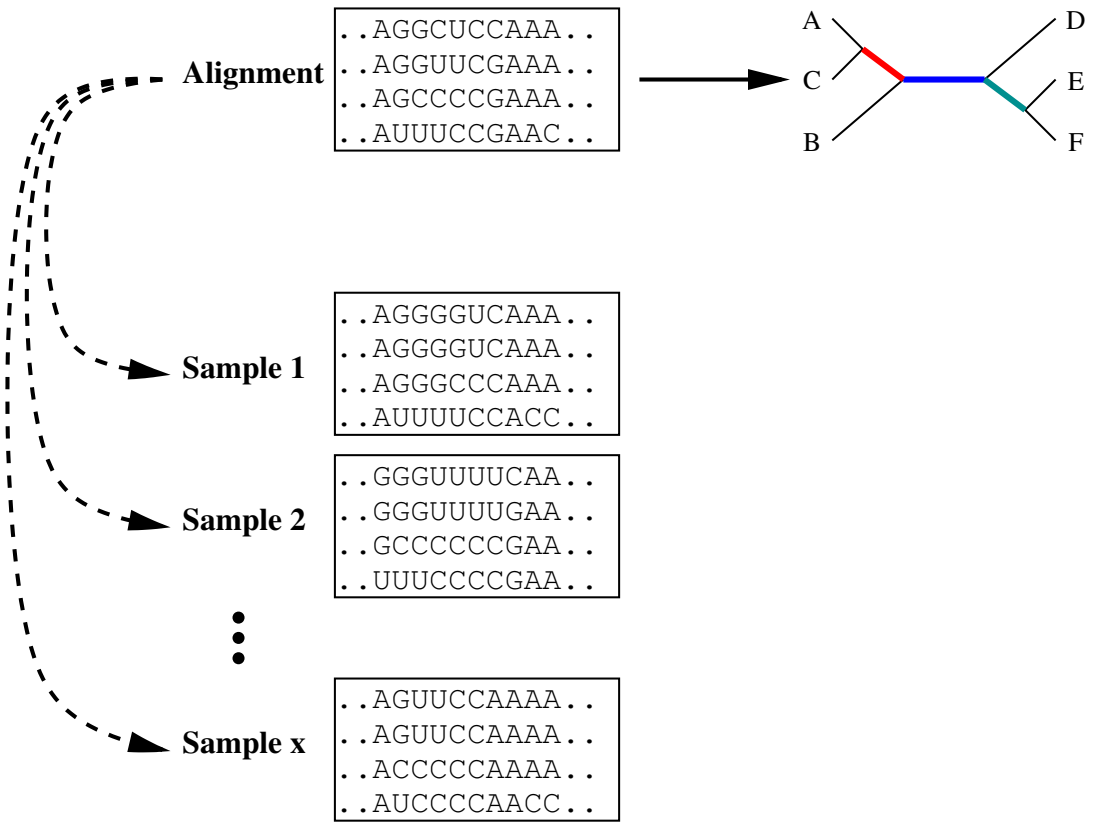
Branch Support

- We can now reconstruct ML trees, but how comparable are the likelihoods, how reliable the groupings?
- Branch reliability can be checked, support values computed using:
  - Randomizing input orders in stepwise insertions (TREE-PUZZLE).
  - Jackknifing alignment columns + consensus.
  - Bootstrapping alignment columns + consensus.
  - Trees from Bayesian MCMC sampling + consensus.

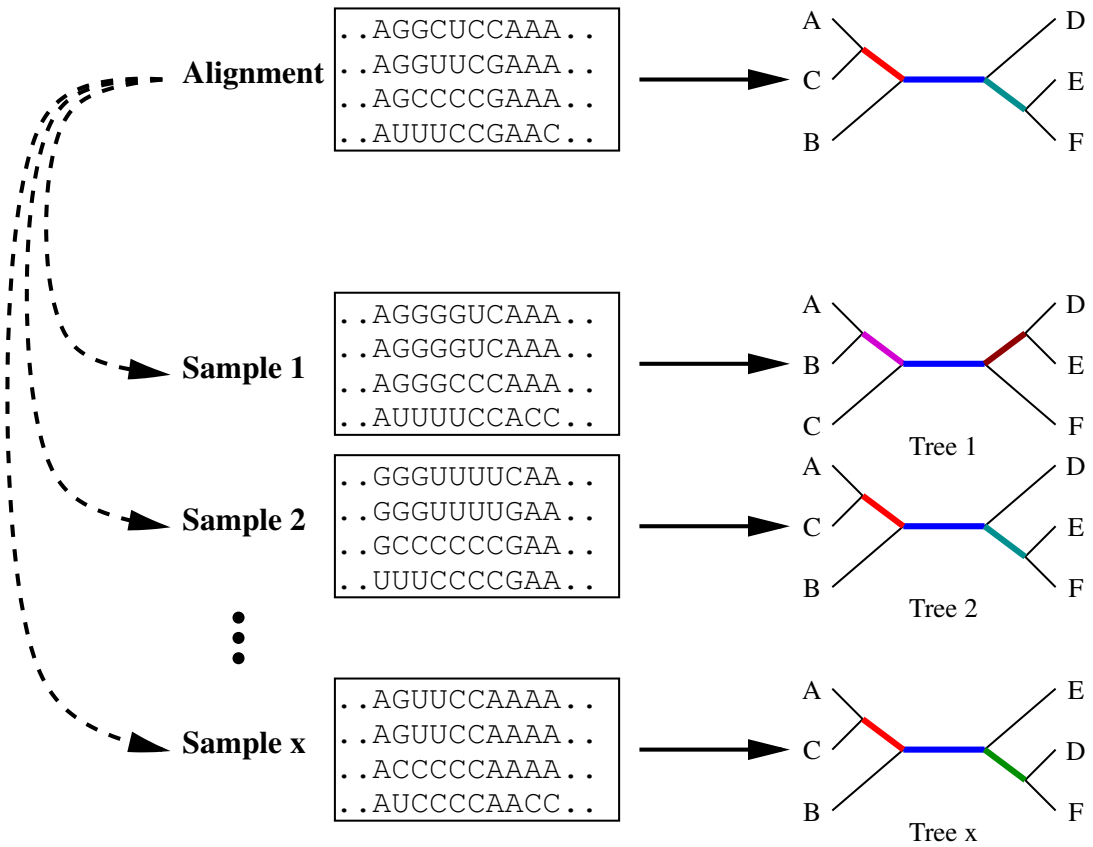
Estimating Confidence: The Bootstrap



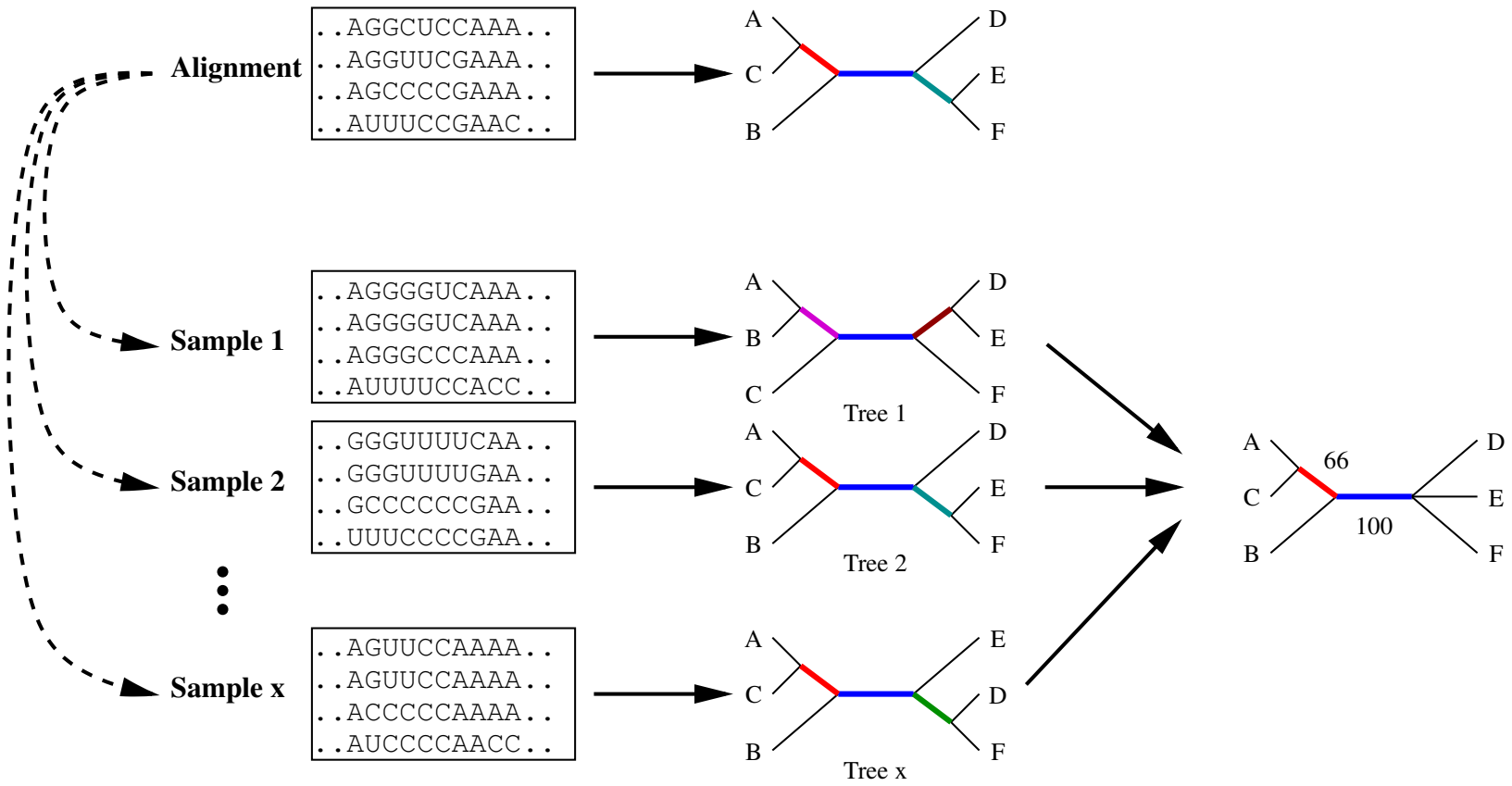
Estimating Confidence: The Bootstrap



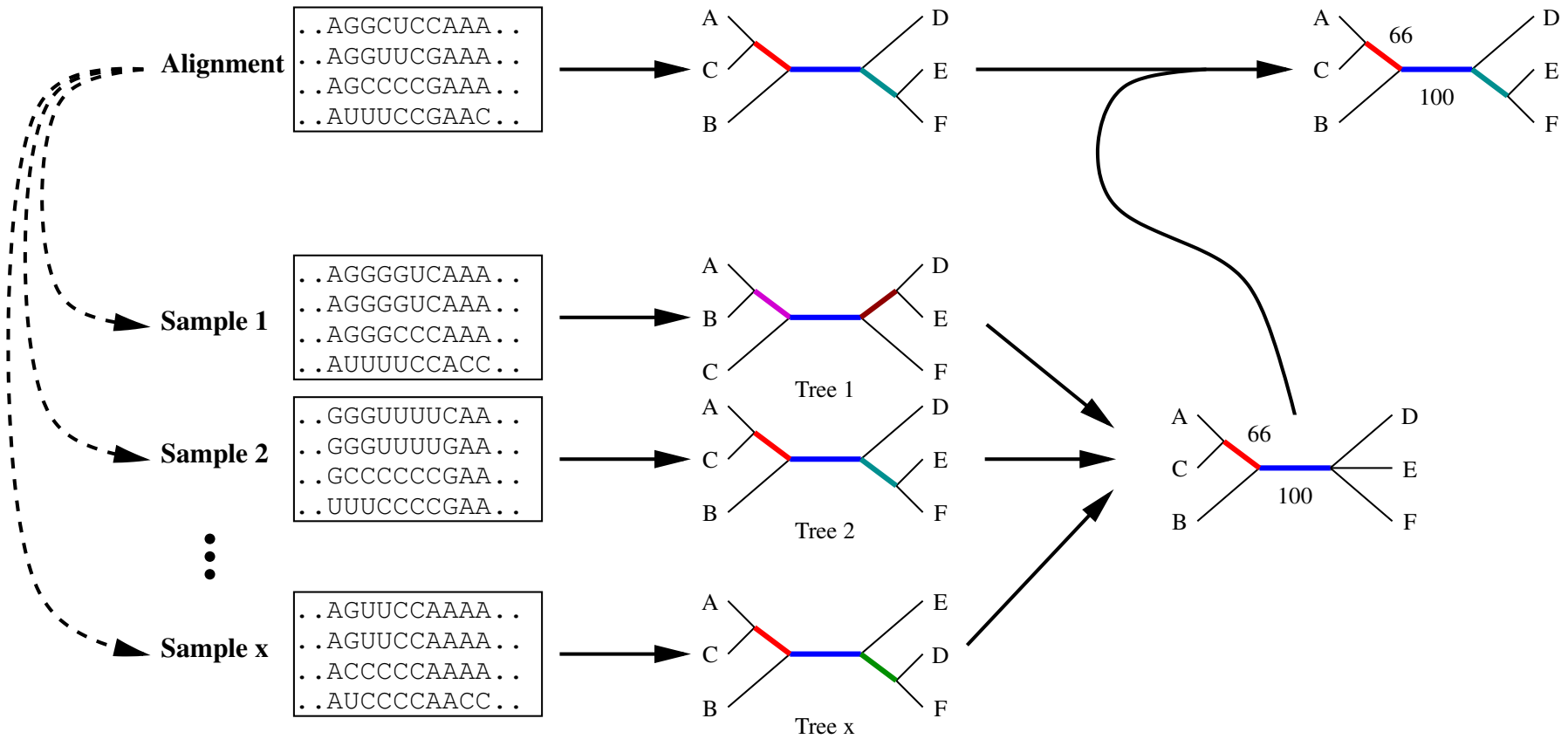
Estimating Confidence: The Bootstrap



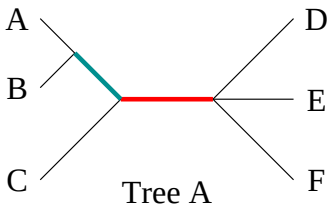
Estimating Confidence: The Bootstrap



Estimating Confidence: The Bootstrap



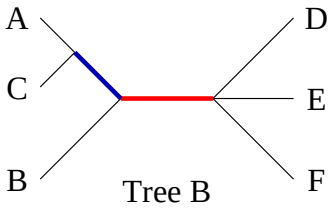
Summarizing Trees: Consensus Methods



Tree A

AB|CDEF

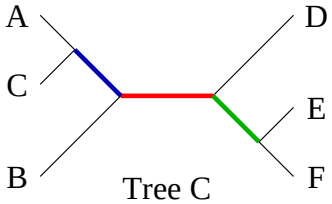
ABC|DEF



Tree B

AC|BDEF

ABC|DEF



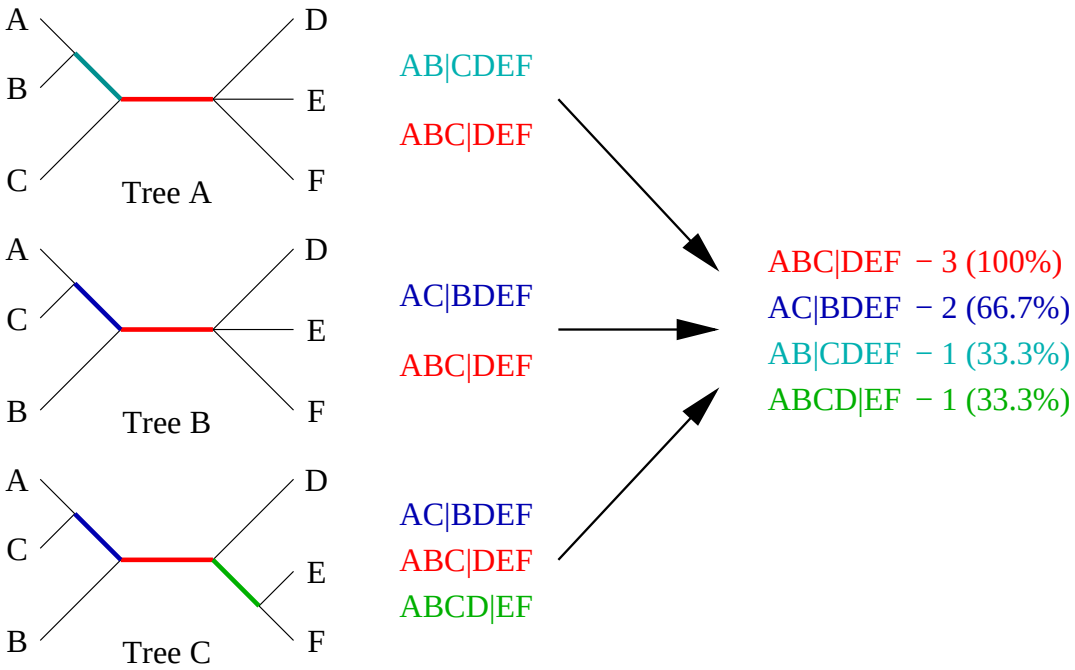
Tree C

AC|BDEF

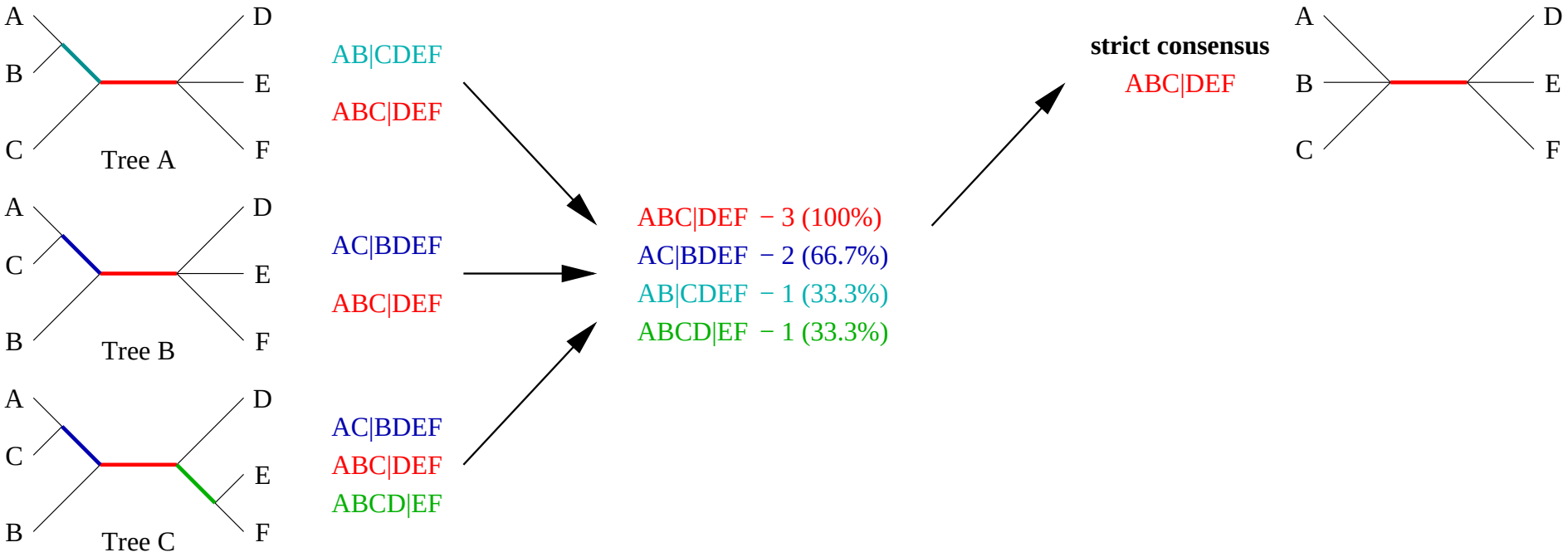
ABC|DEF

ABCD|EF

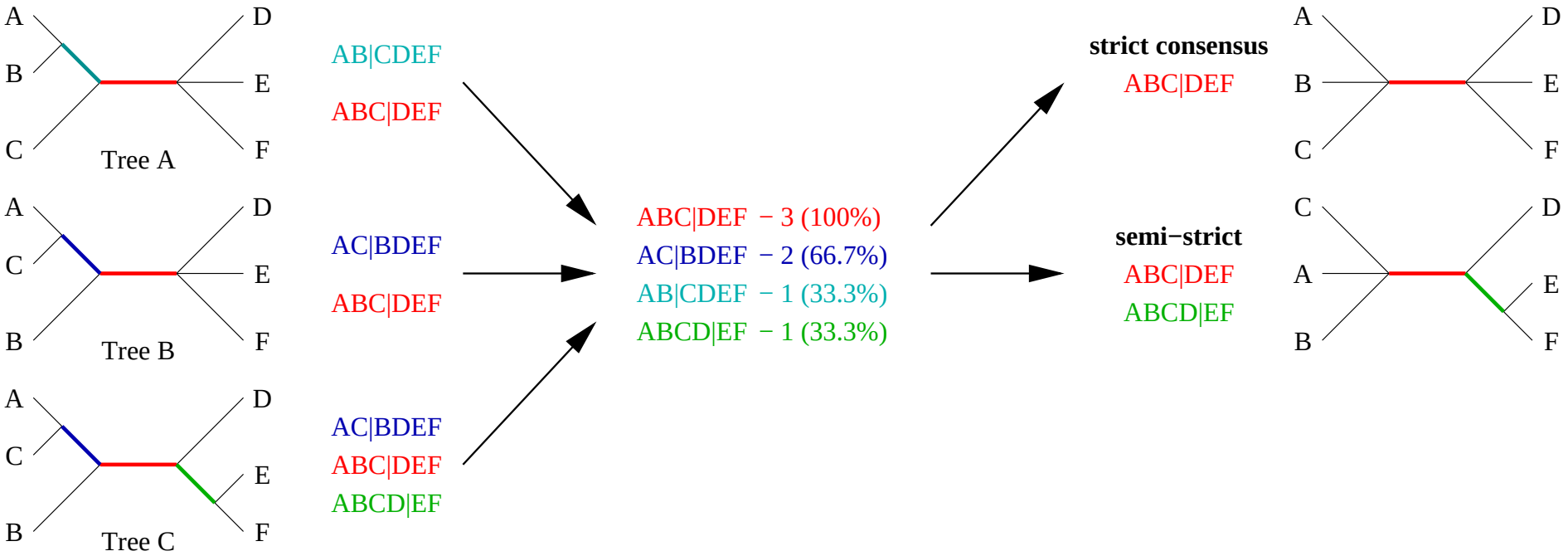
Summarizing Trees: Consensus Methods



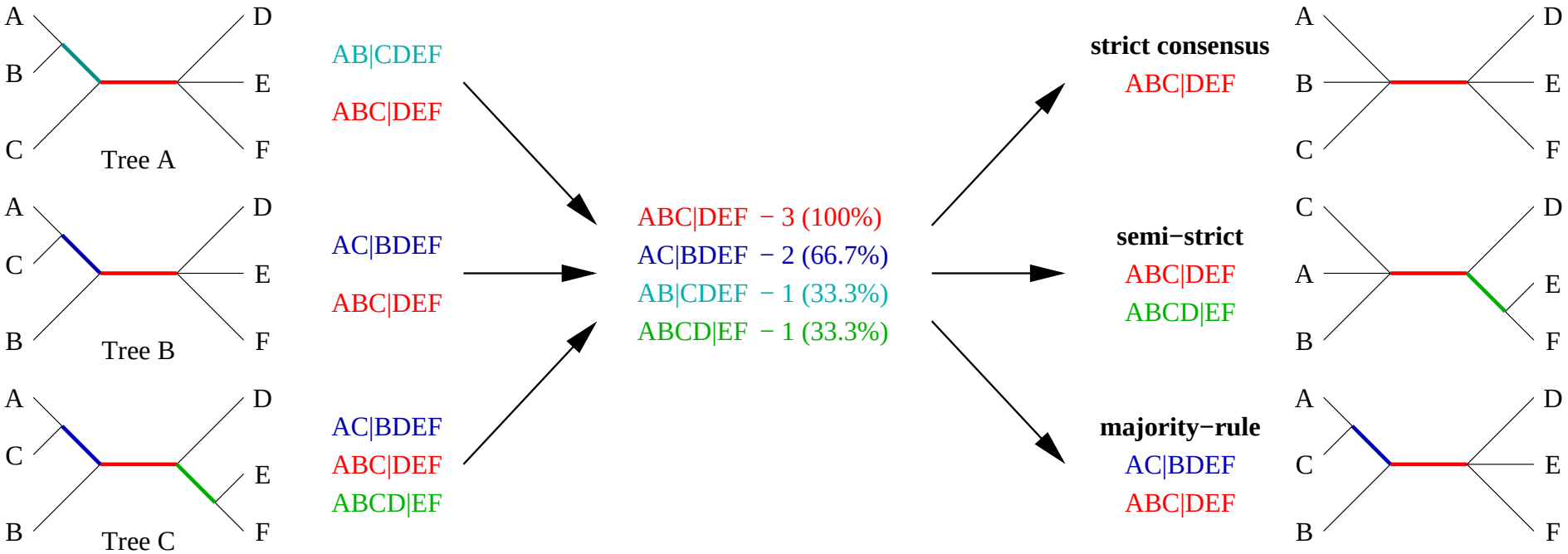
Summarizing Trees: Consensus Methods



Summarizing Trees: Consensus Methods



Summarizing Trees: Consensus Methods



Overview over Likelihood-based Analyses

- Comparing hypothesis with Likelihood-Ratio-Test (=LRT)
  - different models of evolution (ModelTest)
  - testing molecular clock assumption and root position (TREE-PUZZLE)
- Parameter estimation (TREE-PUZZLE, PAUP, ModelTest, . . .)
- Testing for phylogenetic content (TREE-PUZZLE)
- Comparing/testing different tree topologies with Kishino-Hasegawa test, Shimodaira-Hasegawa test (TREE-PUZZLE), SOWH-test, ELW
- Constructing confidence sets on posterior likelihoods (MrBayes)

Posterior Probabilities and Empirical Bayes

- Problem: How different are likelihoods? Just from the value of likelihoods one often cannot tell whether they are significantly different.

Posterior Probabilities and Empirical Bayes

- Problem: How different are likelihoods? Just from the value of likelihoods one often cannot tell whether they are significantly different.
- Normalization: Posterior probabilities are computed:

$$p_i = \frac{L_1}{\sum_n L_n}$$

- Usage:
  - Which sites along an alignment support a tree most?
  - Are there sites/partitions not supporting a tree?
  - Which model of evolution (e.g. dependent, independent) is supported by which site/partition? (PAML)
  - Is a site fast/medium/slowly evolving? (PAML, TREE-PUZZLE)
  - Constructing confidence sets on posterior tree likelihoods (MrBayes)

LRT – Likelihood Ratio Test (1)

The Likelihood function offers a natural way of comparing nested evolutionary hypothesis using the Likelihood Ratio (LR) statistics:

$$\Delta = 2(\ln L_1 - \ln L_0)$$

- L_1 maximum likelihood under the more parameter-rich, complex model (alternative hypothesis, H_A)
- L_0 maximum likelihood under the less parameter-rich simple model (Null-hypothesis, H_0)

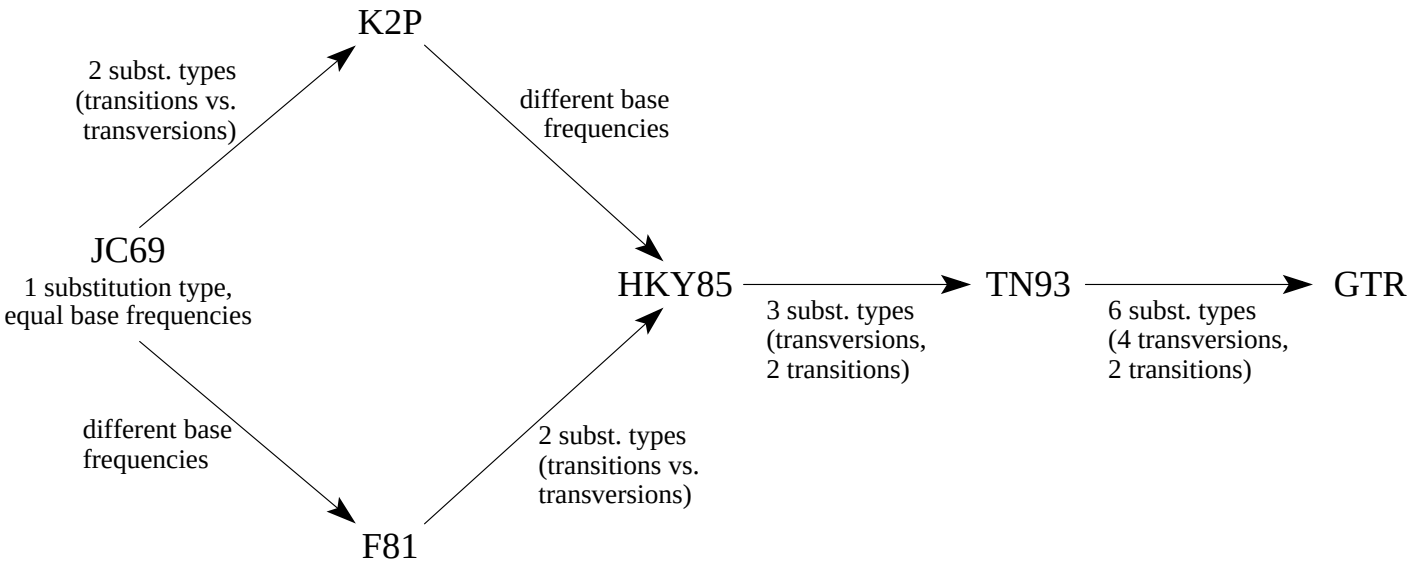
If the models are nested, i.e., H_0 is a special case of H_A and the Null-hypothesis (H0) is correct, Δ is asymptotically χ^2-distributed with the number of degrees of freedom equal to the difference in number of free parameters between the two models.

LRT – Likelihood Ratio Test (2)

- If the LRT is significant (i.e., $p < 0.05$ or $p < 0.01$): the use of the additional parameters in the alternative model H_A increases the likelihood significantly.
- If Δ is close to zero, that is, $p > 0.05$: the alternative hypothesis H_A does not fit the data significantly better than H_0, that means using the additional parameters of H_A does not explain the data better.
- Only nested models can be tested:
  One model (H_0, Null-model, constraint model) is nested in another model (H_A, alternative, unconstraint model) if the model H_0 can be produced by restricting parameters in model H_A.

LRT – Typical cases of nested models

- Different levels of evolutionary models:



- *rate-homogeneous models* (H_0) are nested in *rate-heterogeneous models* (H_A)
- A tree assuming *molecular clock* (H_0) are nested its *non-clock* version (H_A)

Exercises:

the exercises can be found at

`http://www.cibiv.at/~hschmidt/VEME/ML`