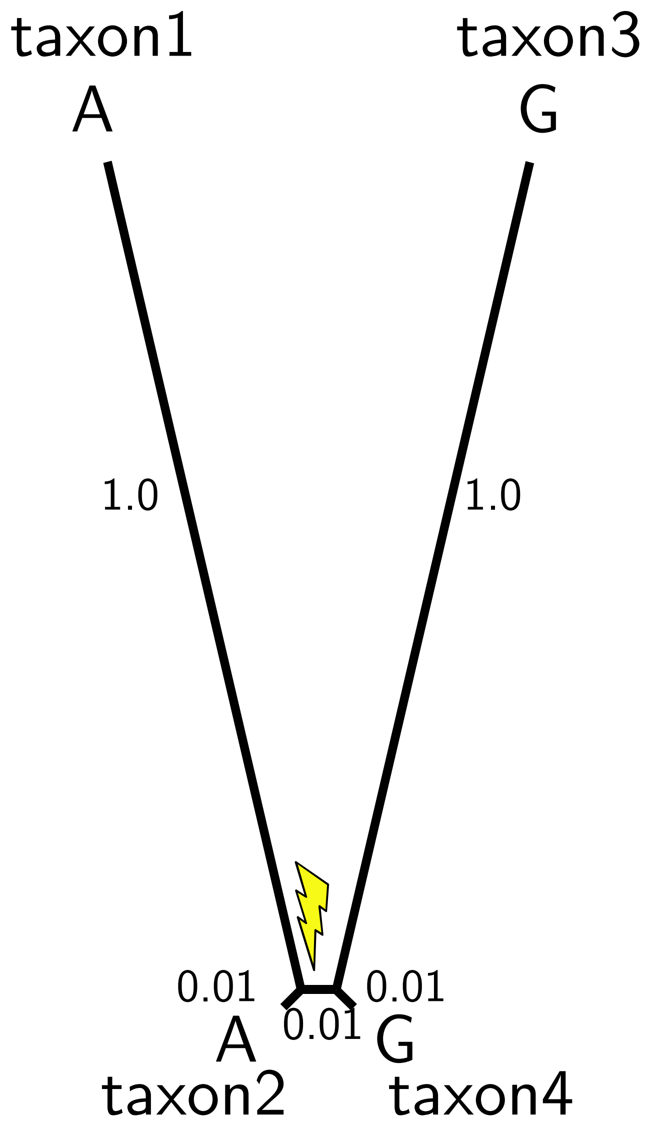
Thanks to Paul Lewis and Joe Felsenstein for the use of slides

Review

- Hennigian logic reconstructs the tree if we know **polarity** of characters and there is **no homoplasy**
- UPGMA infers a tree from a distance matrix:
  - groups based on **similarity**
  - fails to give the correct tree if rates of character evolution vary much
- Modern distance-based approaches:
  - find trees and branch lengths: patristic distances $\approx$ distances from character data.
  - do **not** use all of the information in the data.
- Parsimony:
  - prefer the tree that **requires** the fewest character state changes. Minimize the number of times you invoke homoplasy to explain the data.
  - can work well if if homoplasy is not rare
  - fails if homoplasy very common **or is concentrated on certain parts of the tree**

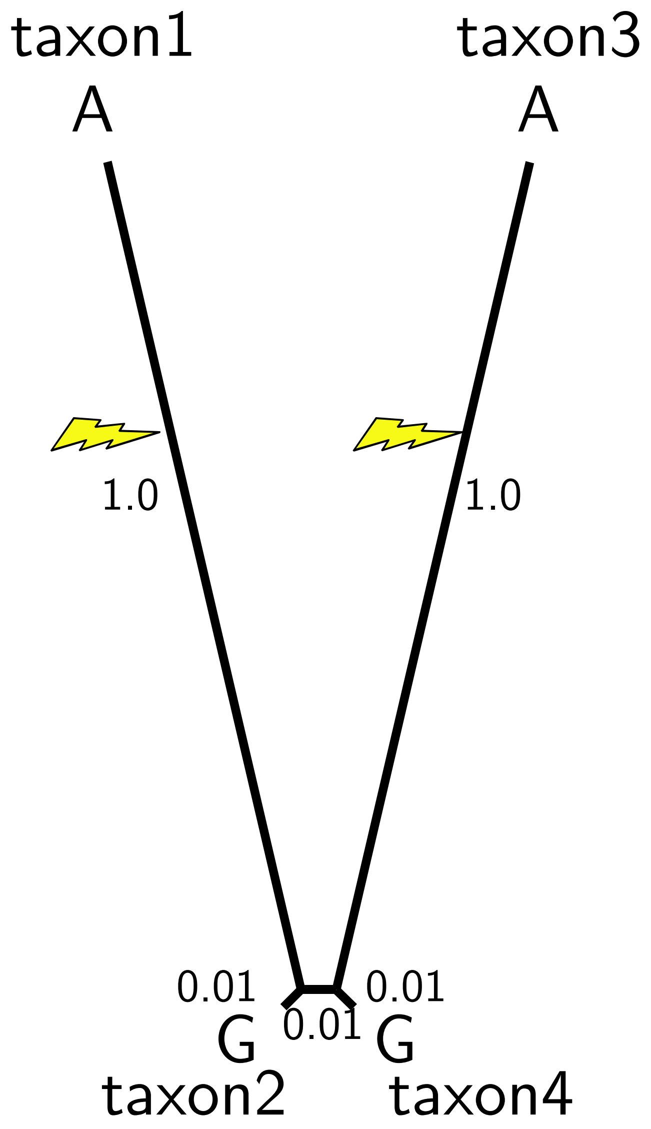
Long branch attraction



Felsenstein, J. 1978. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology* **27**: 401-410.

The probability of a parsimony informative site due to inheritance is very low, (roughly 0.0003).

Long branch attraction



Felsenstein, J. 1978. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Zoology* **27**: 401-410.

The probability of a parsimony informative site due to inheritance is very low, (roughly 0.0003).

The probability of a misleading parsimony informative site due to parallelism is much higher (roughly 0.008).

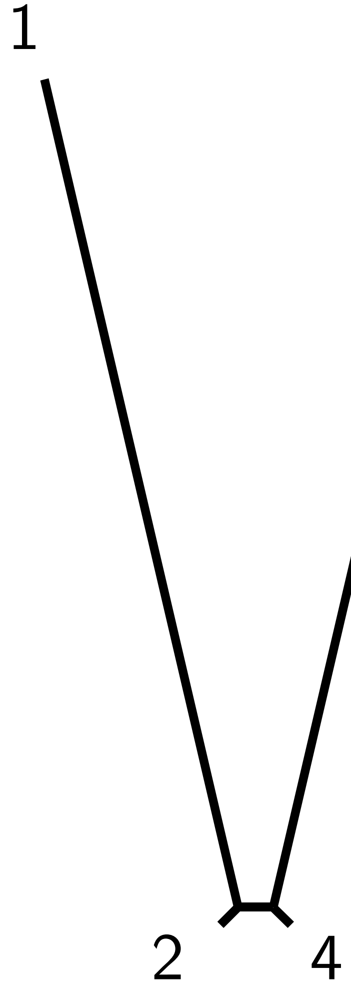
Long branch attraction data

Under such a tree misleading characters are more common that characters that favor the true tree.

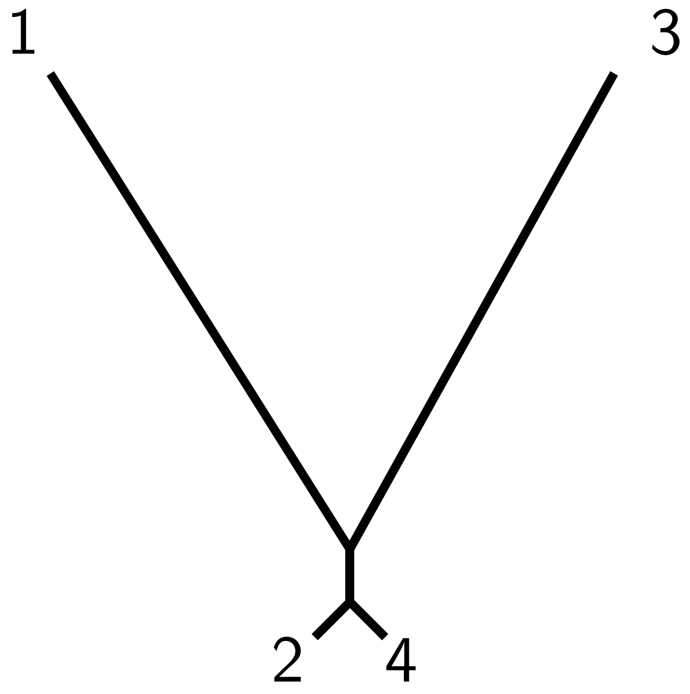
	Rare					Common			
taxon1	A	A	C	C		A	A	C	C
taxon2	A	A	C	C		G	C	T	G
taxon3	G	C	T	G		A	A	C	C
taxon4	G	C	T	G		G	C	T	G

Long branch attraction

Parsimony is almost guaranteed to get this tree wrong.



True



Inferred

Likelihood

X is the data.

T is the tree.

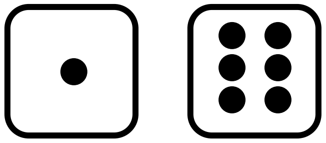
ν is a vector of branch lengths.

$\Pr(X|T,\nu)$ is the *likelihood*; this is sometimes denoted $L(T,\nu)$.

Maximum likelihood: find the T and ν that gives the highest likelihood.

Combining probabilities

- Multiply probabilities if the component events must happen simultaneously (i.e. whereever you would naturally use the word AND when describing the problem)

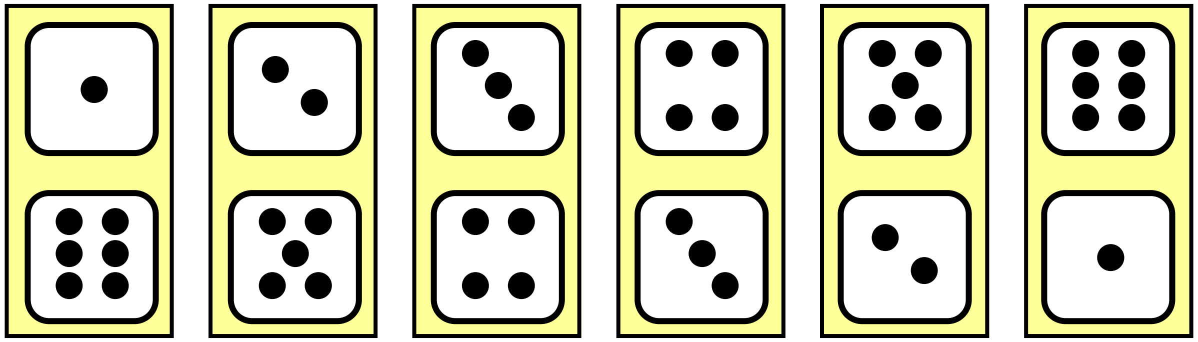


$(1/6) \times (1/6) = 1/36$

What is the probability of rolling two dice and having the first show 1 dot AND the second show 6 dots?

Combining probabilities

- Add probabilities if the component events are mutually exclusive (i.e. whereever you would naturally use the word OR)



(1/36) + (1/36) + (1/36) + (1/36) + (1/36) + (1/36) = 1/6

What is the probability of rolling 7 using two dice? This is the same as asking "What is the probability of rolling (1 and 6) OR (2 and 5) OR (3 and 4) OR (4 and 3) OR (5 and 2) OR (6 and 1)?"

Likelihood of a single sequence

First 32 nucleotides of the $\psi\eta$-globin gene of gorilla:

GAAGTCCTTGAGAAATAAACTGCACACACTGG

$$\begin{aligned} L &= \pi_G\pi_A\pi_A\pi_G\pi_T\pi_C\pi_C\pi_T\pi_T\pi_G\pi_A\pi_G\pi_A\pi_A\pi_A\pi_T\pi_A\pi_A\pi_A\pi_C\pi_T\pi_G\pi_C\pi_A\pi_C\pi_A\pi_C\pi_A\pi_C\pi_T\pi_G\pi_G \\ &= \pi_A^{12}\pi_C^{7}\pi_G^{7}\pi_T^{6} \end{aligned}$$

$$\ln L = 12\ln(\pi_A) + 7\ln(\pi_C) + 7\ln(\pi_G) + 6\ln(\pi_T)$$

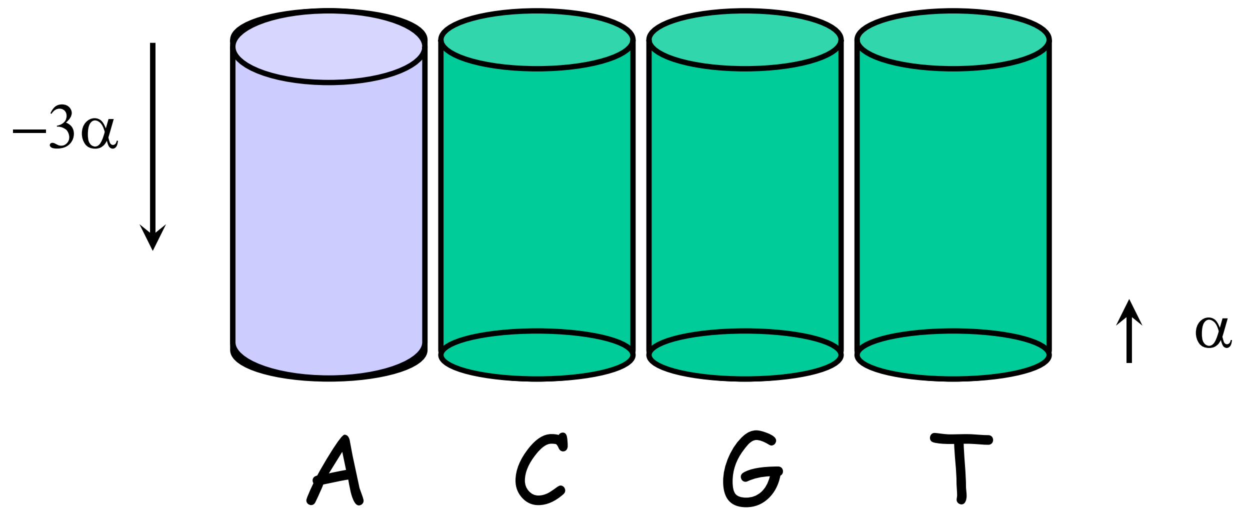
We can already see by eye-balling this that the F81 model (which allows unequal base frequencies) will fit better than the JC69 model (which assumes equal base frequencies) because there are about twice as many As as there are Cs, Gs and Ts.

Likelihoods on the simplest possible tree

$$\texttt{GA}\longrightarrow\texttt{GG}$$

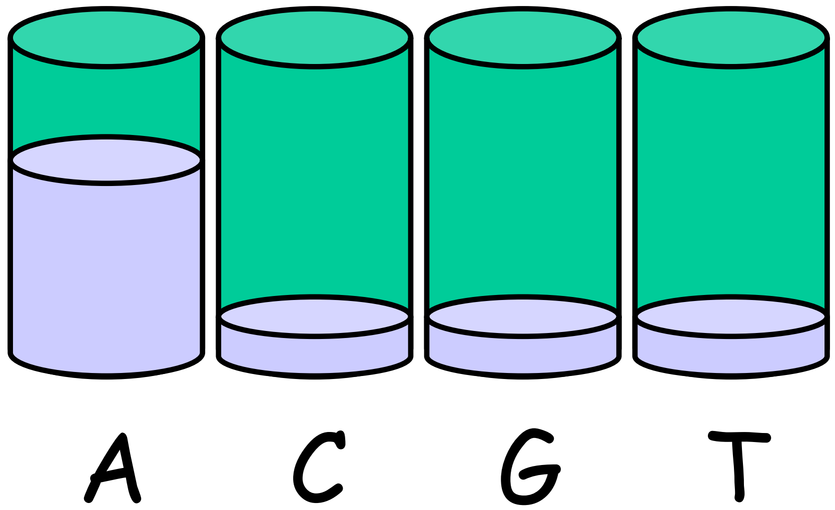
$$\begin{aligned}
L &= L_1 L_2 \\
&= \Pr(G)\,\Pr(G \rightarrow G)\,\Pr(A)\,\Pr(A \rightarrow G) \\
&= \Pr(G)\,\Pr(G \rightarrow G|\nu)\,\Pr(A)\,\Pr(A \rightarrow G|\nu)
\end{aligned}$$

Water analogy (time 0)



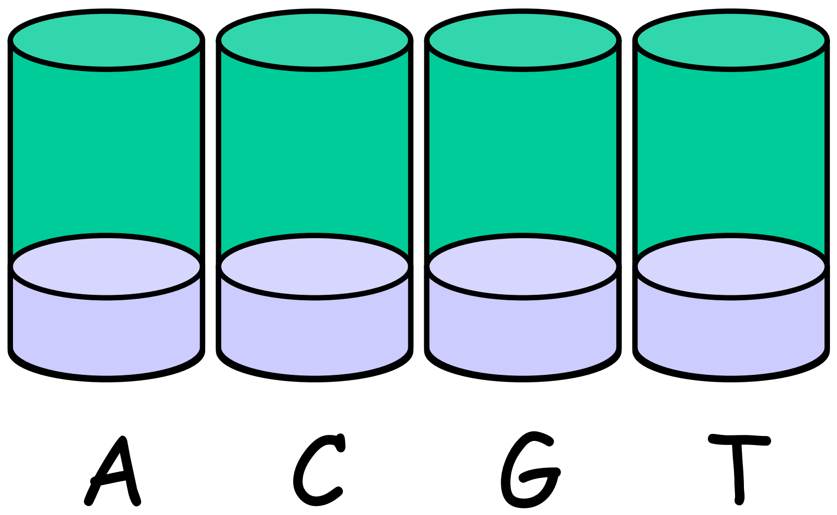
- Start with container A completely full and others empty
- Imagine that all containers are connected by tubes that allow same rate of flow between any two
- Initially, A will be losing water at 3 times the rate that C (or G or T) gains water

Water analogy (after some time)



A’s level is not dropping as fast now because it is now also *receiving* water from C, G and T

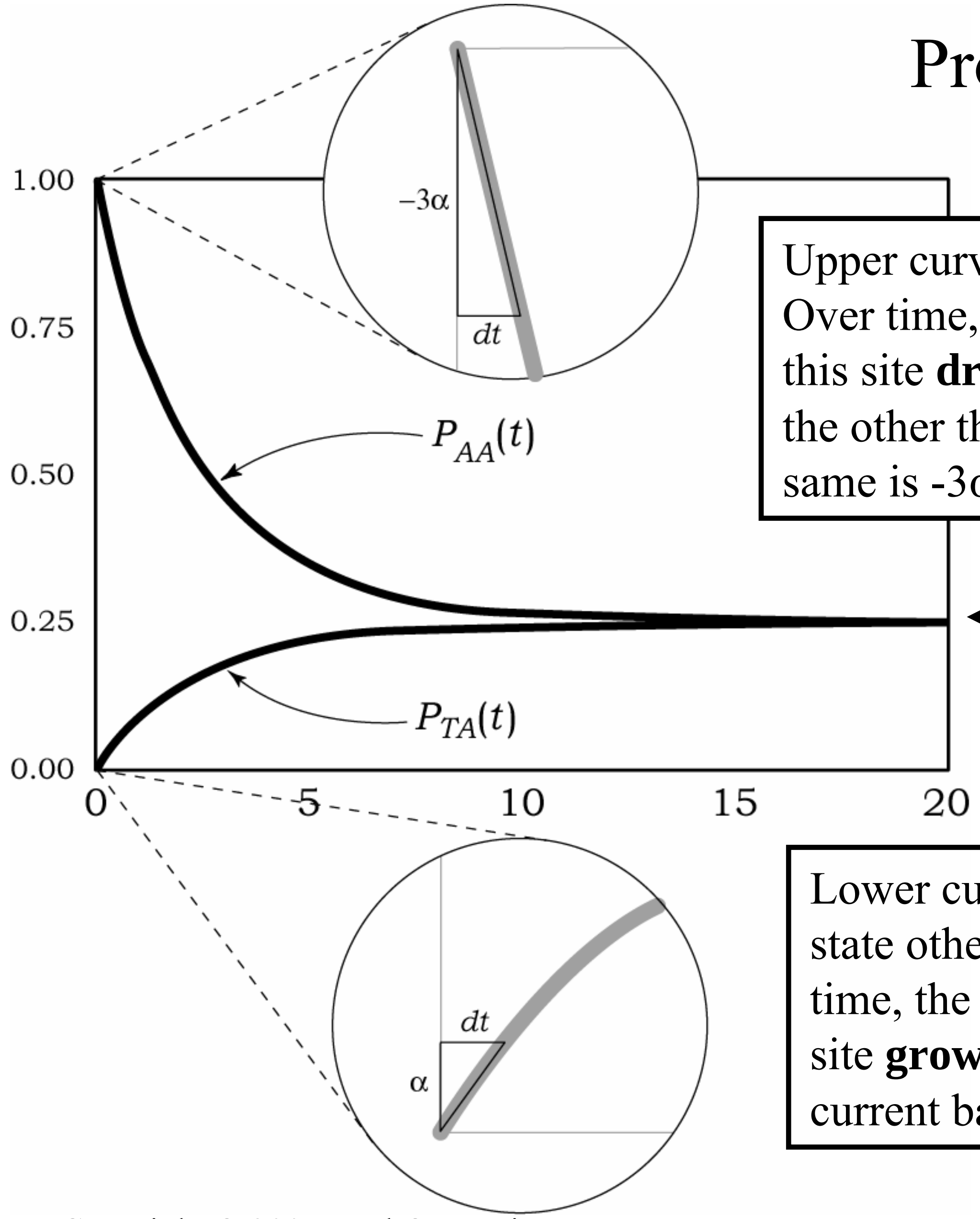
Water analogy (after a very long time)



Eventually, all containers are one fourth full and there is zero *net* volume change – **stationarity** (equilibrium) has been achieved

(Thanks to Kent Holsinger for this analogy)

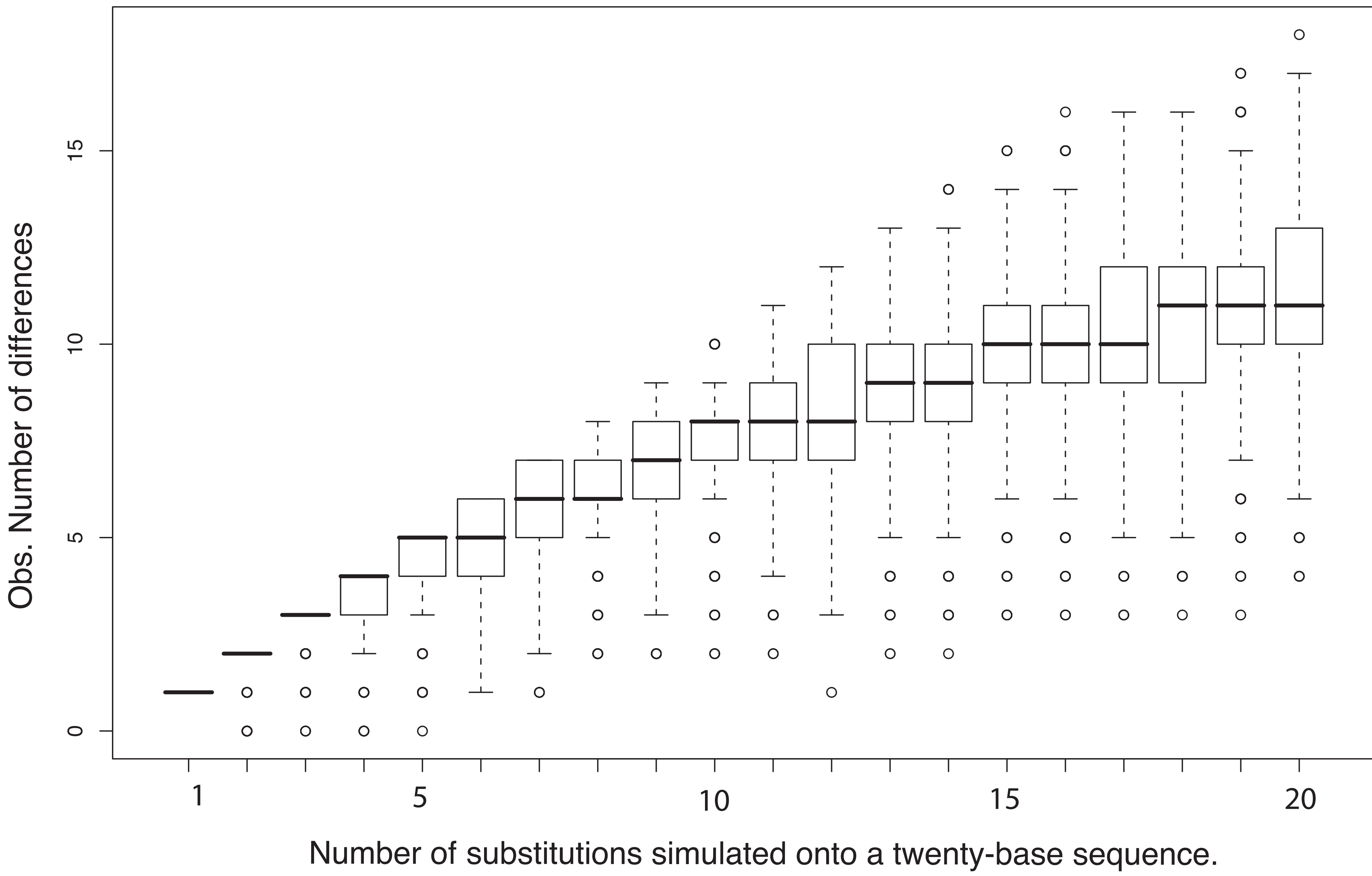
Probability of "A present" as a function of time



Upper curve assumes we started with A at time 0. Over time, the probability of still seeing an A at this site **drops** because rate of changing to one of the other three bases is 3α (so rate of staying the same is -3α).

The equilibrium relative frequency of A is 0.25

Lower curve assumes we started with some state other than A (T is used here). Over time, the probability of seeing an A at this site **grows** because the rate at which the current base will change into an A is α.


Obs. Number of differences
0
5
10
15
1
5
10
15
20
Number of substitutions simulated onto a twenty-base sequence.

Jukes-Cantor model

$$\begin{aligned}\Pr(G \rightarrow G|\nu) &= \frac{1}{4} + \frac{3}{4}\, e^{\frac{-4\nu}{3}} \\ \Pr(A \rightarrow G|\nu) &= \frac{1}{4} - \frac{1}{4}\, e^{\frac{-4\nu}{3}}\end{aligned}$$

Likelihoods on the simplest possible tree

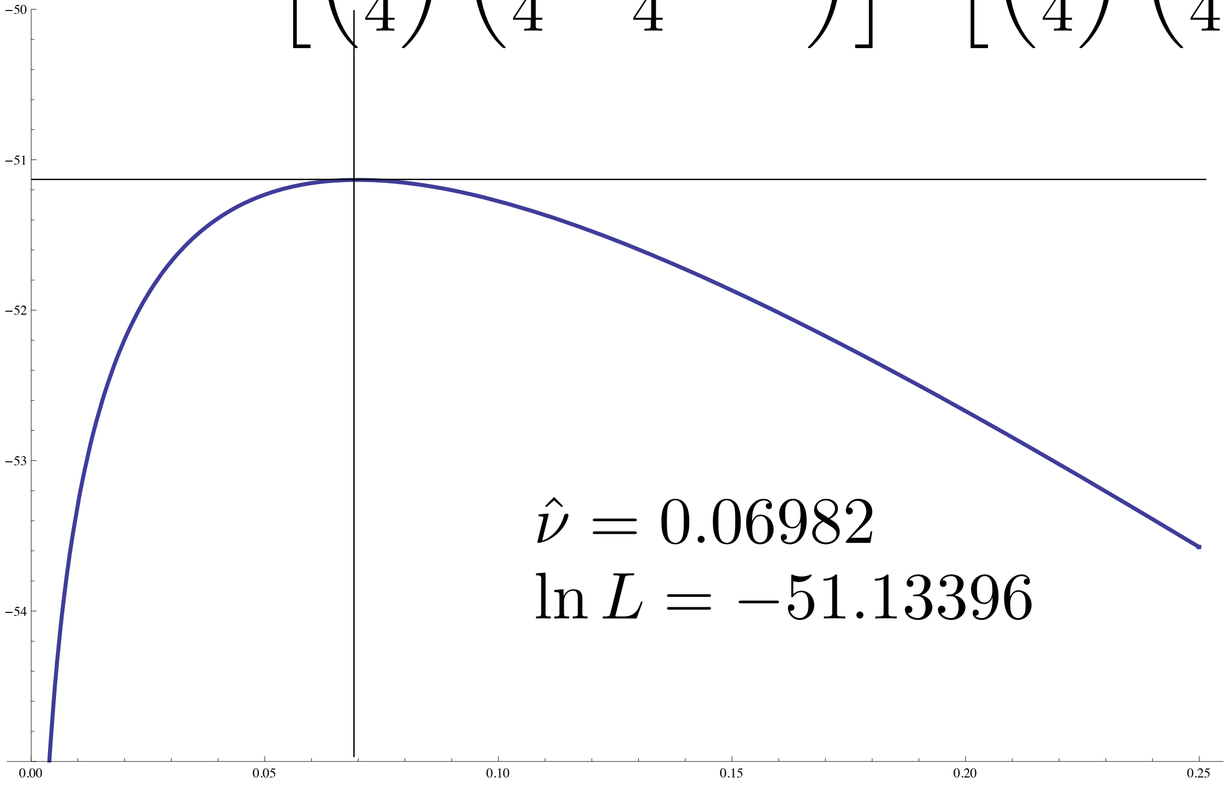
$$\texttt{GA} \longrightarrow \texttt{GG}$$

$$\begin{aligned}
L &= L_1 L_2 \\
&= \Pr(G)\,\Pr(G \rightarrow G)\,\Pr(A)\,\Pr(A \rightarrow G) \\
&= \Pr(G)\,\Pr(G \rightarrow G|\nu)\,\Pr(A)\,\Pr(A \rightarrow G|\nu) \\
&= \left(\frac{1}{4}\right)\left(\frac{1}{4} + \frac{3}{4}\,e^{\frac{-4\nu}{3}}\right)\left(\frac{1}{4}\right)\left(\frac{1}{4} - \frac{1}{4}\,e^{\frac{-4\nu}{3}}\right)
\end{aligned}$$

The first 30 nucleotides of the $\psi\eta$-globin gene

```
gorilla    GAAGTCCTTGAGAAATAAACTGCACACTGG
orangutan  GGACTCCTTGAGAAATAAACTGCACACTGG
```

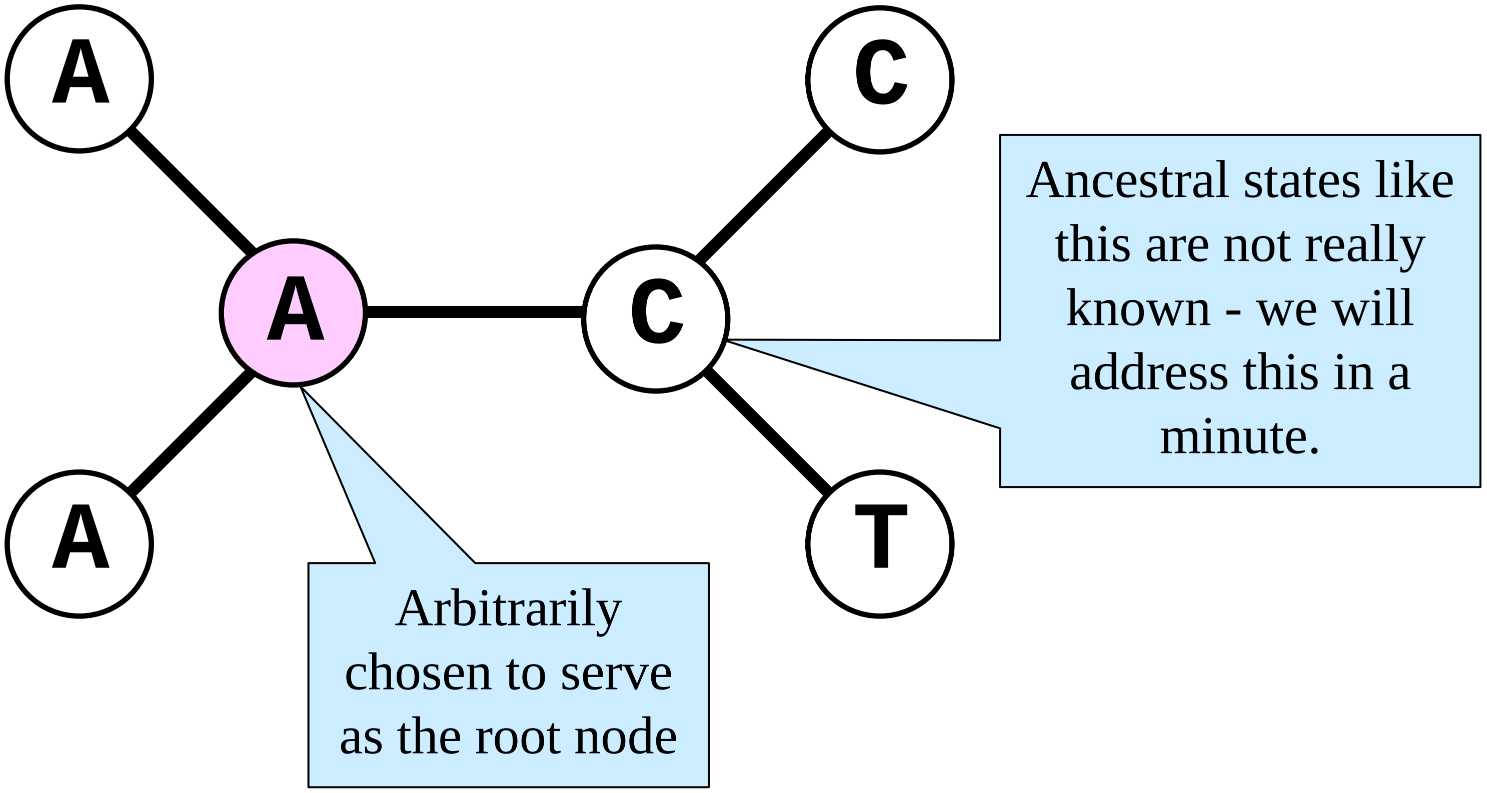
$$L = \left[\left(\frac{1}{4}\right)\left(\frac{1}{4}+\frac{3}{4}\,e^{\frac{-4\nu}{3}}\right)\right]^{28}\left[\left(\frac{1}{4}\right)\left(\frac{1}{4}-\frac{1}{4}\,e^{\frac{-4\nu}{3}}\right)\right]^{2}$$



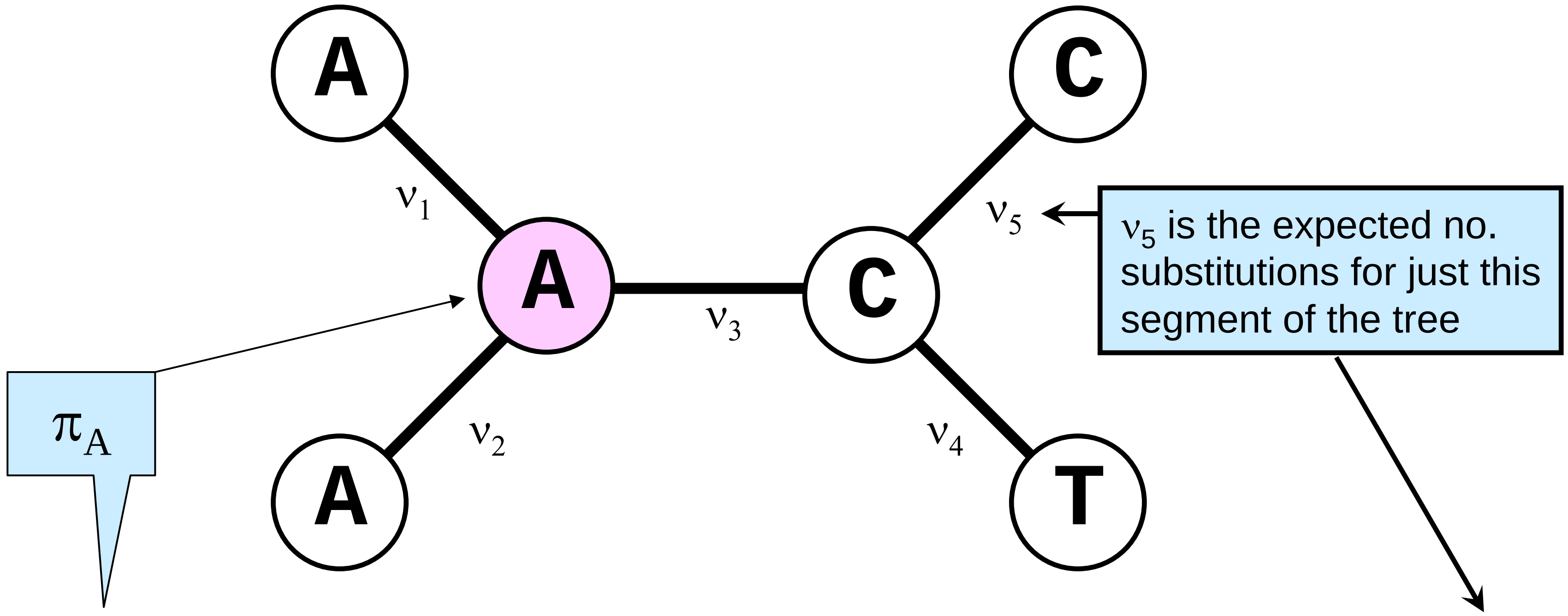
$$\hat{\nu} = 0.06982$$
$$\ln L = -51.13396$$

Likelihood of a tree

(data for only one site shown)


A
C
A
C
A
T
Ancestral states like this are not really known - we will address this in a minute.
Arbitrarily chosen to serve as the root node

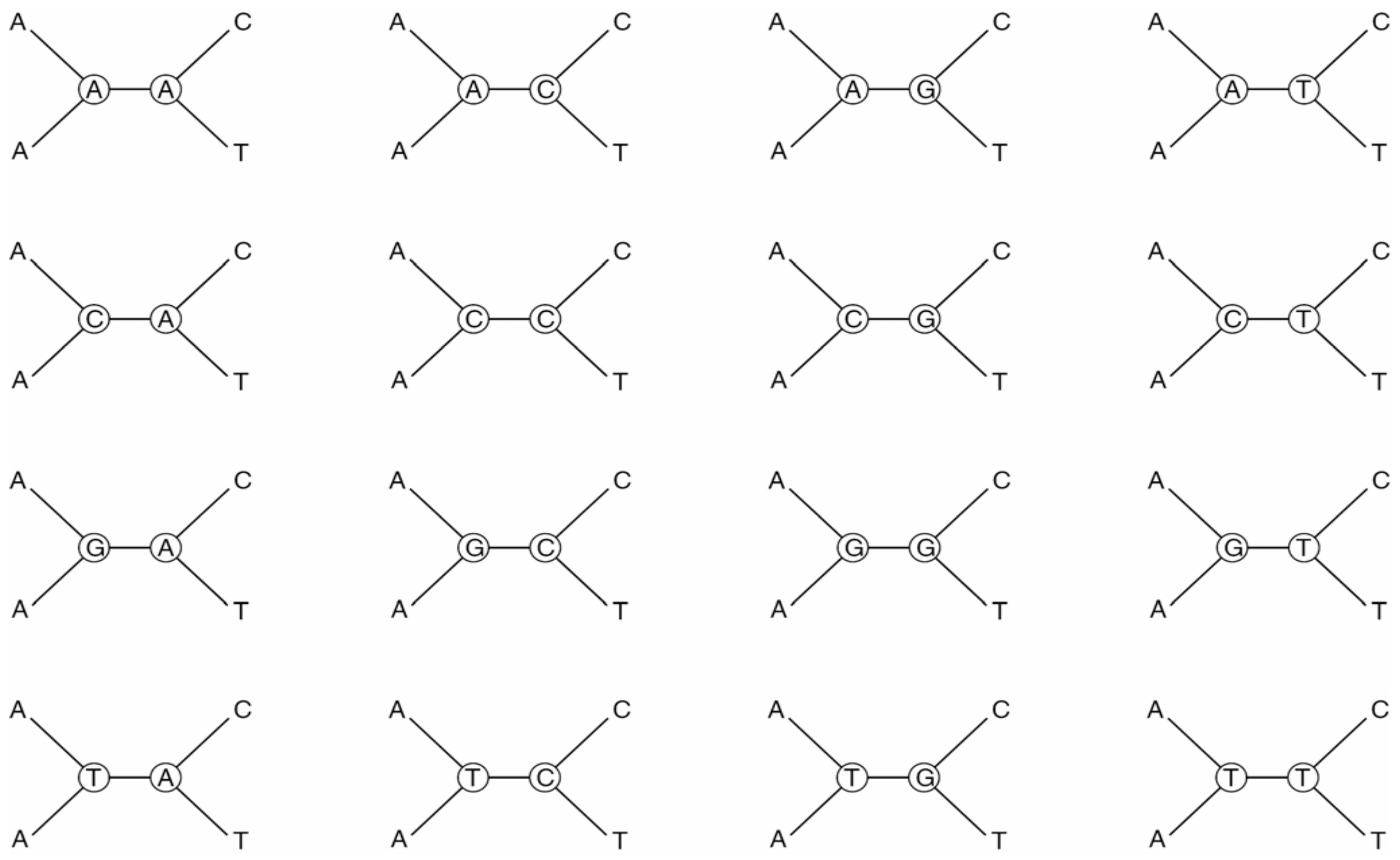
Likelihood for site k


A
A
A
C
C
T
ν_1
ν_2
ν_3
ν_4
ν_5
π_A
ν_5 is the expected no. substitutions for just this segment of the tree

$$L_k = \tfrac{1}{4}\left[\tfrac{1}{4}+\tfrac{3}{4}e^{-4\nu_1/3}\right]\left[\tfrac{1}{4}+\tfrac{3}{4}e^{-4\nu_2/3}\right]\left[\tfrac{1}{4}-\tfrac{1}{4}e^{-4\nu_3/3}\right]\left[\tfrac{1}{4}-\tfrac{1}{4}e^{-4\nu_4/3}\right]\left[\tfrac{1}{4}+\tfrac{3}{4}e^{-4\nu_5/3}\right]$$

$P_{AA}(\nu_1)$ $P_{AA}(\nu_2)$ $P_{AC}(\nu_3)$ $P_{CT}(\nu_4)$ $P_{CC}(\nu_5)$

Brute force approach would be to calculate L_k for all 16 combinations of ancestral states and sum



Likelihood and Bayesian procedures

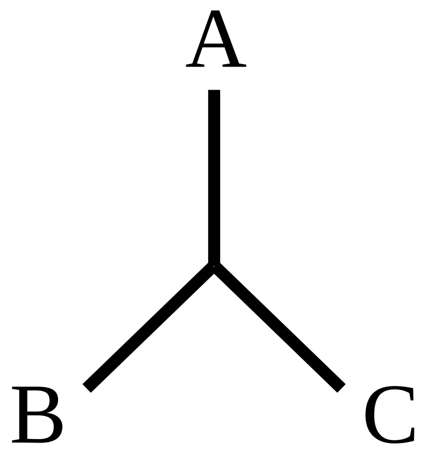
1. very computationally intensive,
2. Use all of the information in the data,
3. Let us estimate the forces of character evolution while estimating trees,
4. Uses models to detect concerted patterns of homoplasy (this is how likelihood based procedures avoid long-branch attraction).

Tree Searching

Parsimony and ML give us ways to deciding whether one tree is fits our data better than another tree, but . . .

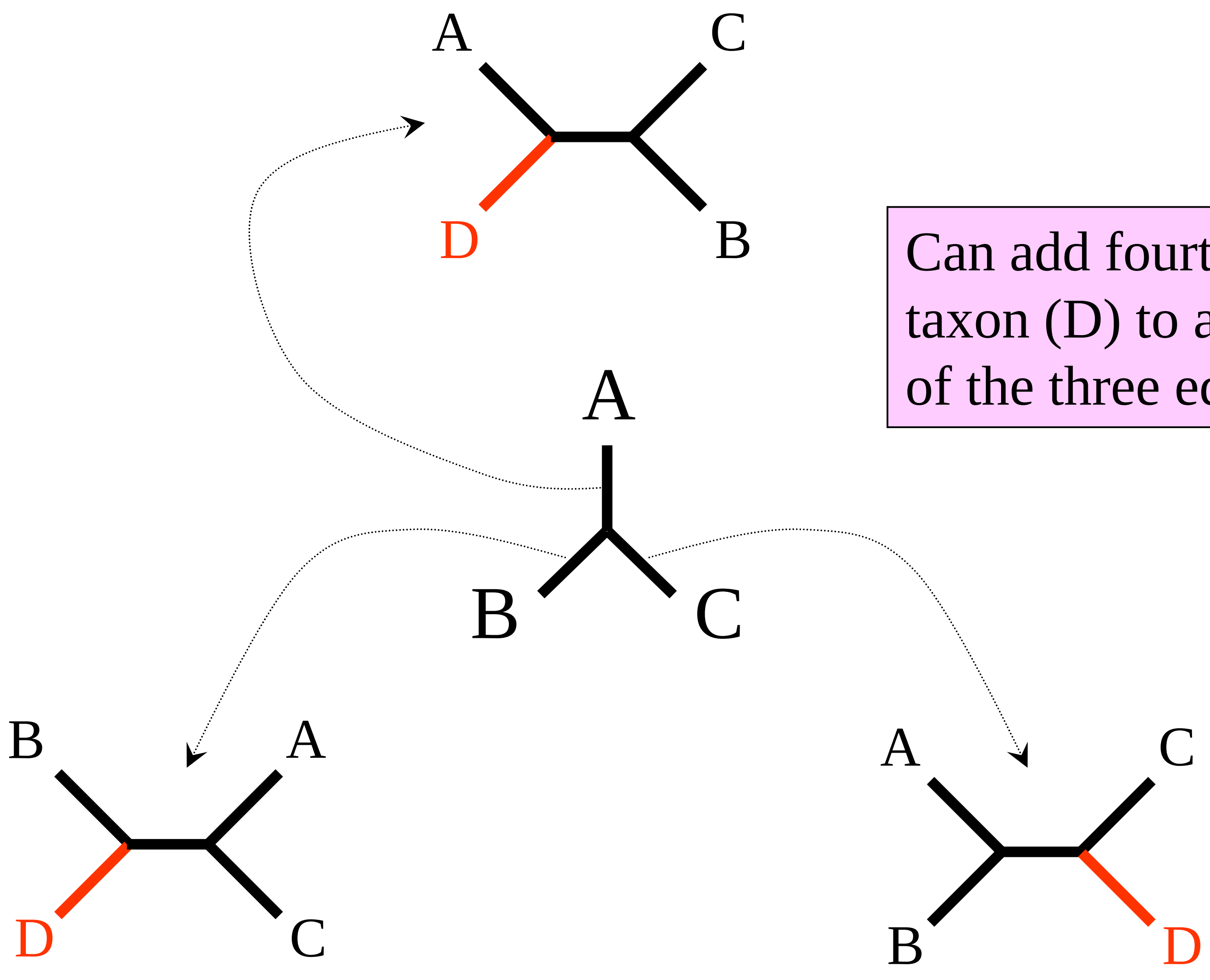
How do we find the best tree?
(or one that is good enough)

Exhaustive Enumeration



With the first three taxa, create the trivial unrooted tree

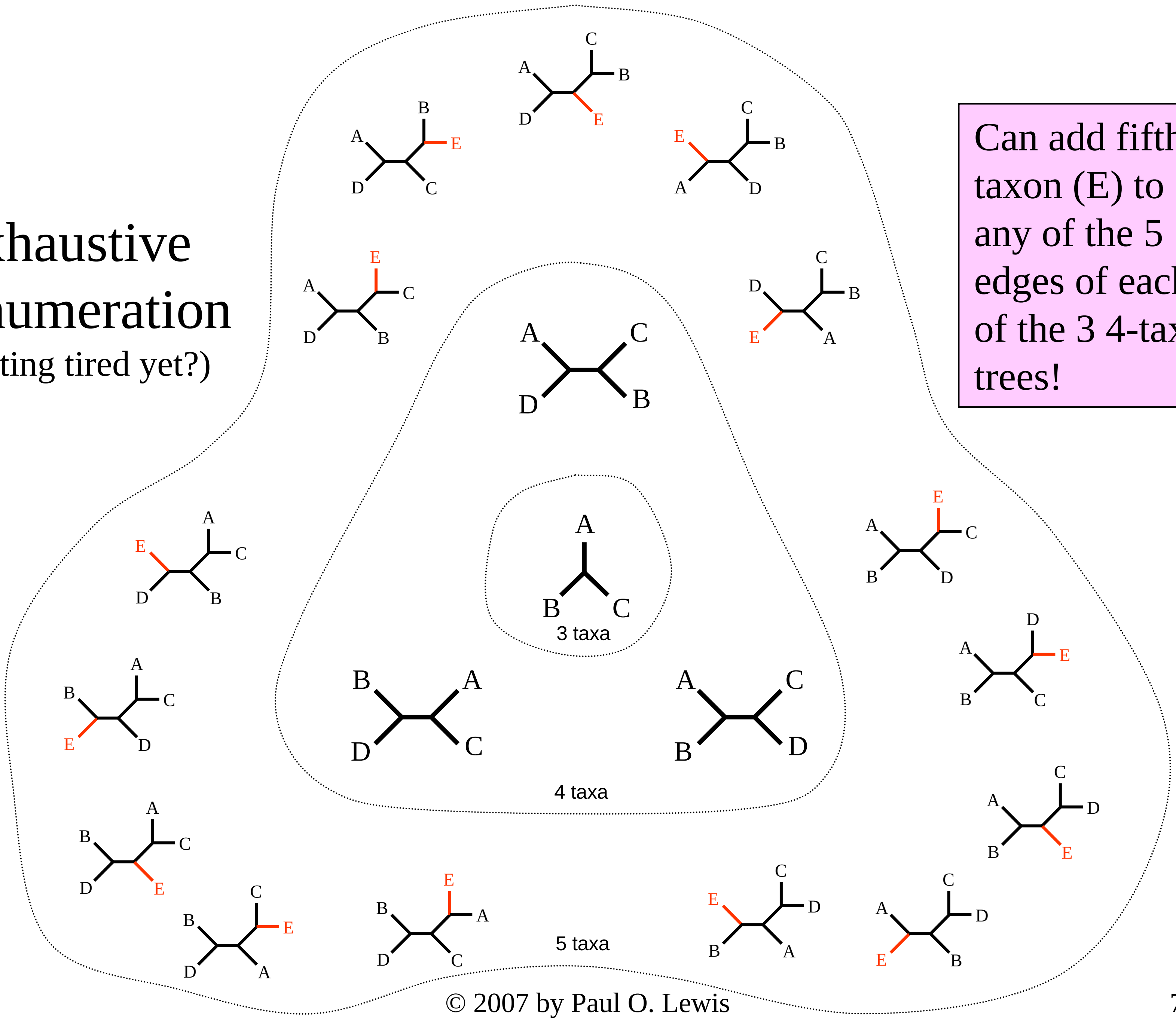
Exhaustive Enumeration...



Can add fourth taxon (D) to any of the three edges

Exhaustive Enumeration

(getting tired yet?)



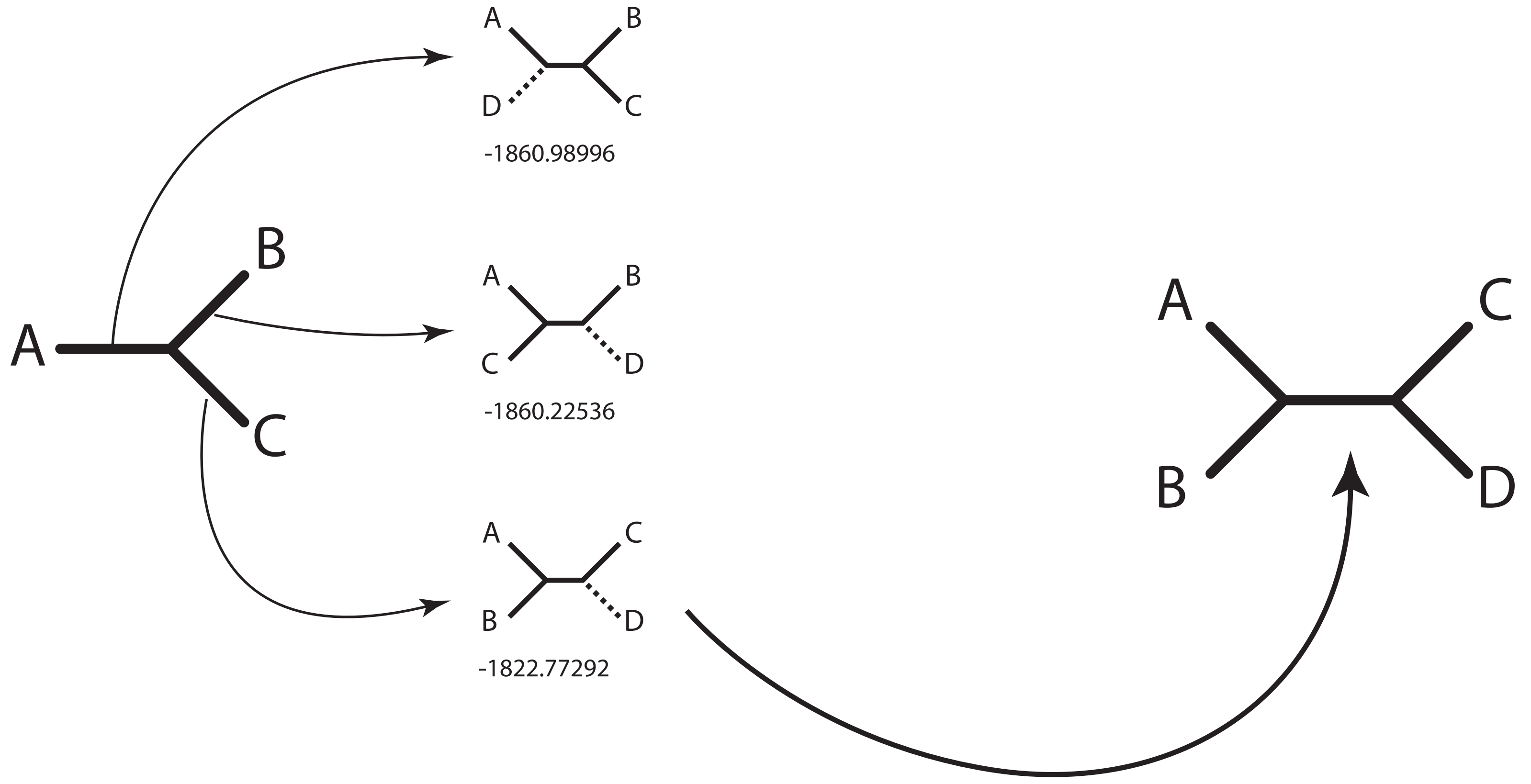
Can add fifth taxon (E) to any of the 5 edges of each of the 3 4-taxon trees!

Tips	Number of unrooted (binary) trees	
4	3	
5	15	
6	105	
7	945	
8	10,395	
9	135,135	
10	2,027,025	
11	34,459,425	
12	654,729,075	
13	13,749,310,575	
14	316,234,143,225	
15	7,905,853,580,625	
16	213,458,046,676,875	
17	6,190,283,353,629,375	
18	191,898,783,962,510,625	
19	6,332,659,870,762,850,625	
20	22,164,309,5476,699,771,875	
21	8,200,794,532,637,891,559,375	
22	319,830,986,772,877,770,815,625	
23	13,113,070,457,687,988,603,440,625	> 21 moles of trees
24	563,862,029,680,583,509,947,946,875	

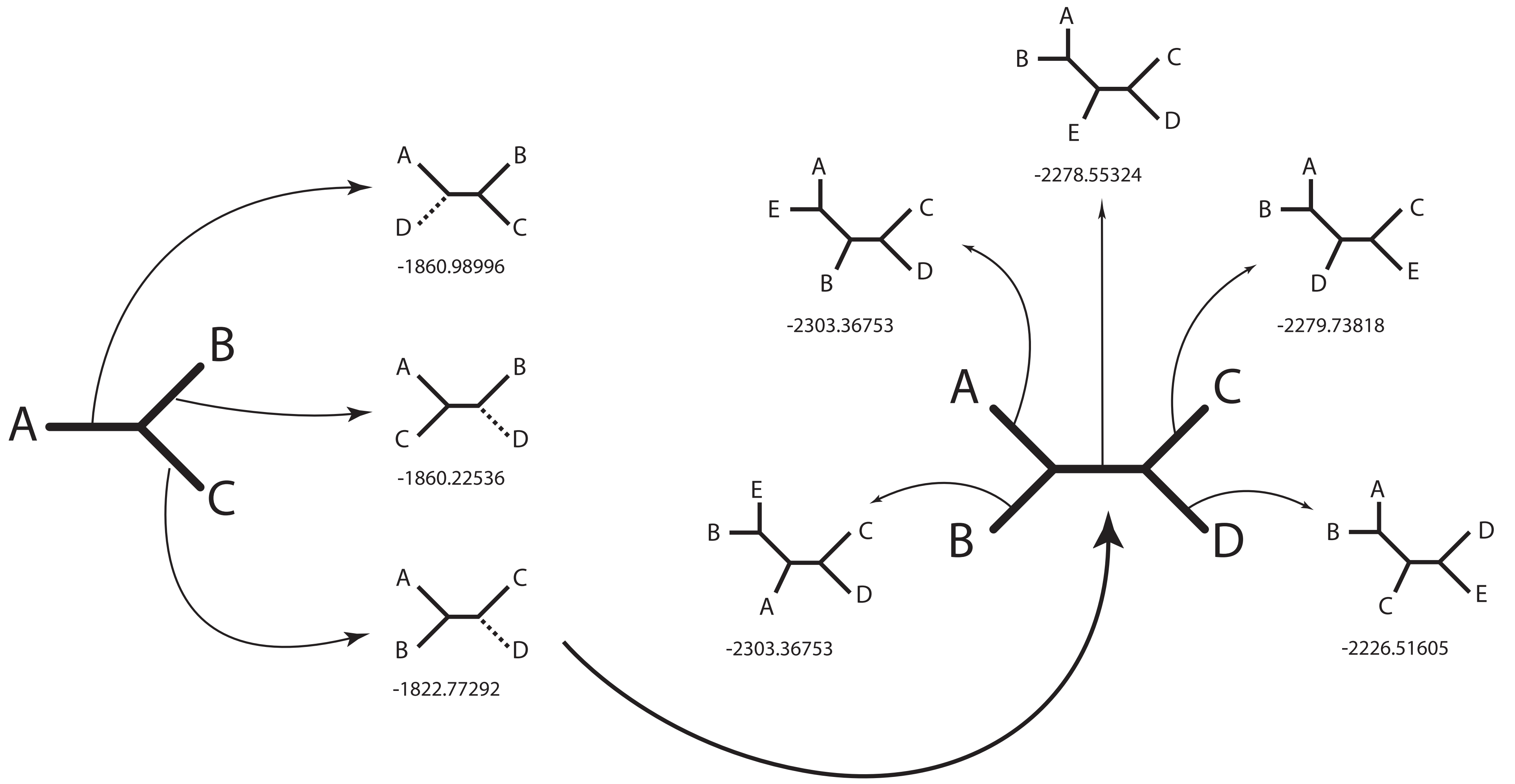
For N taxa:

$$\begin{aligned}
\#\text{ unrooted, binary trees} &= \prod_{i=3}^{N-1}(2i-3) \\
&= \prod_{i=4}^{N}(2i-5) \\
\#\text{ rooted, binary trees} &= \prod_{i=3}^{N}(2i-3) \\
&= (2N-3)(\#\text{ unrooted, binary trees})
\end{aligned}$$

Stepwise addition



Stepwise addition



Is stepwise addition guaranteed to find the best tree?

	1	2	3	4	5
taxonA	A	A	A	A	A
taxonB	A	C	C	A	C
taxonC	C	C	C	T	T
taxonD	C	A	A	C	T
taxonE	A	A	A	T	C

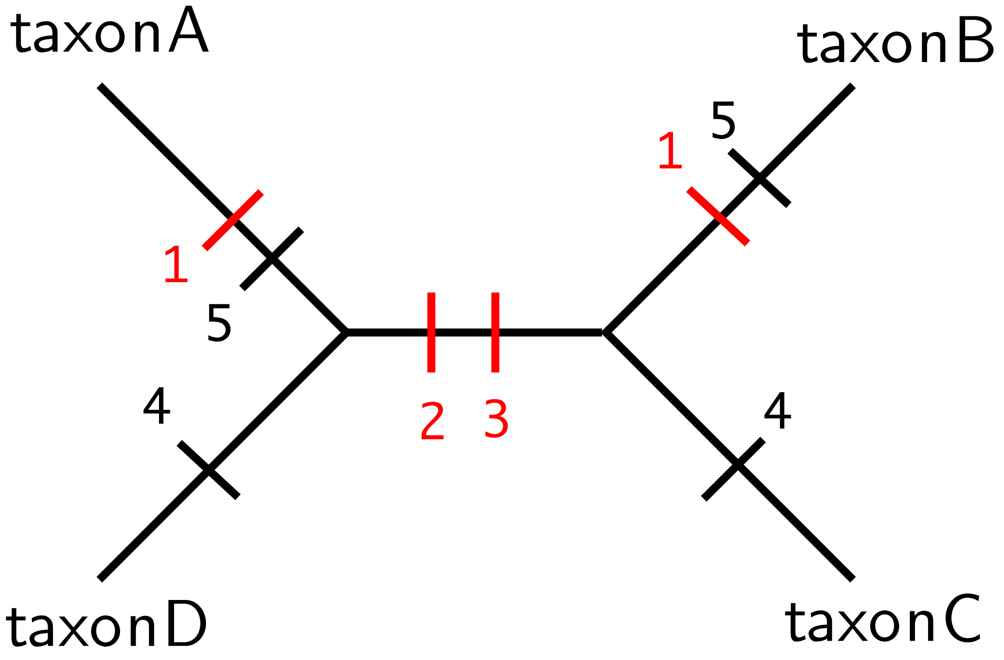
First step of stepwise addition

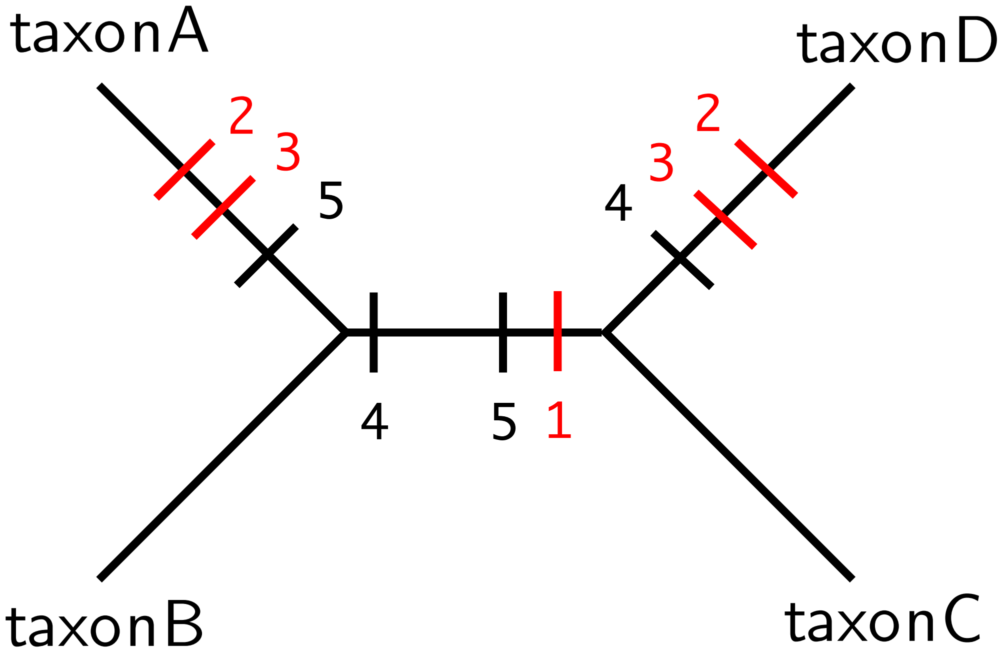
	1	2	3	4	5
taxonA	A	A	A	A	A
taxonB	A	C	C	A	C
taxonC	C	C	C	T	T
taxonD	C	A	A	C	T

First step of stepwise addition

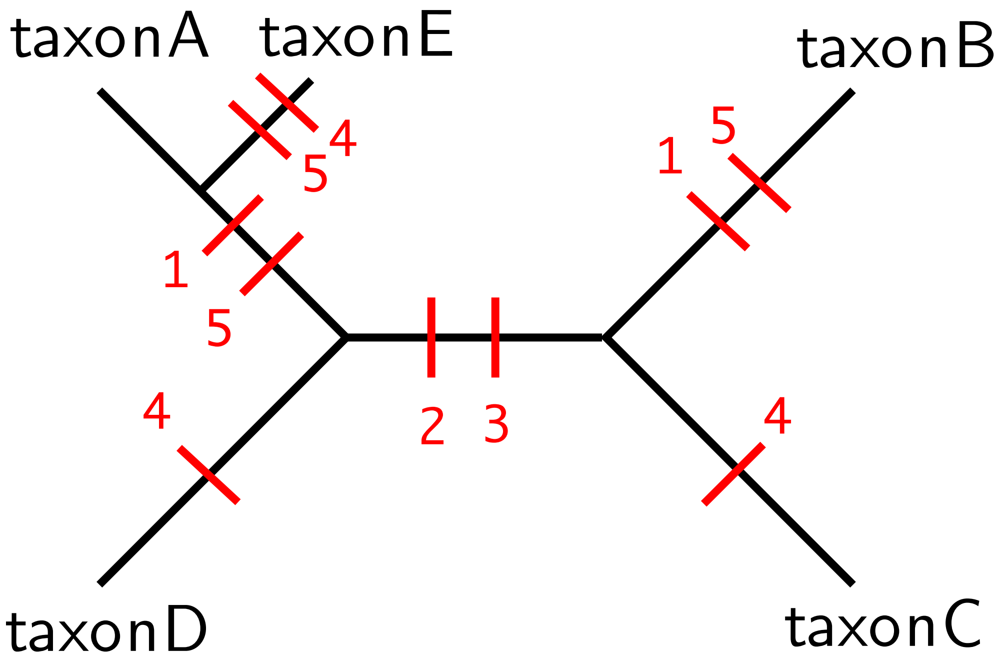
	1	2	3	4	5	
taxonA	A	A	A	A	A	
taxonB	A	C	C	A	C	
taxonC	C	C	C	T	T	
taxonD	C	A	A	C	T	
tree (A, B, (C, D))	1	2	2	2	2	9
tree (A, C, (B, D))	2	2	2	2	2	10
tree (A, D, (B, C))	2	1	1	2	2	8

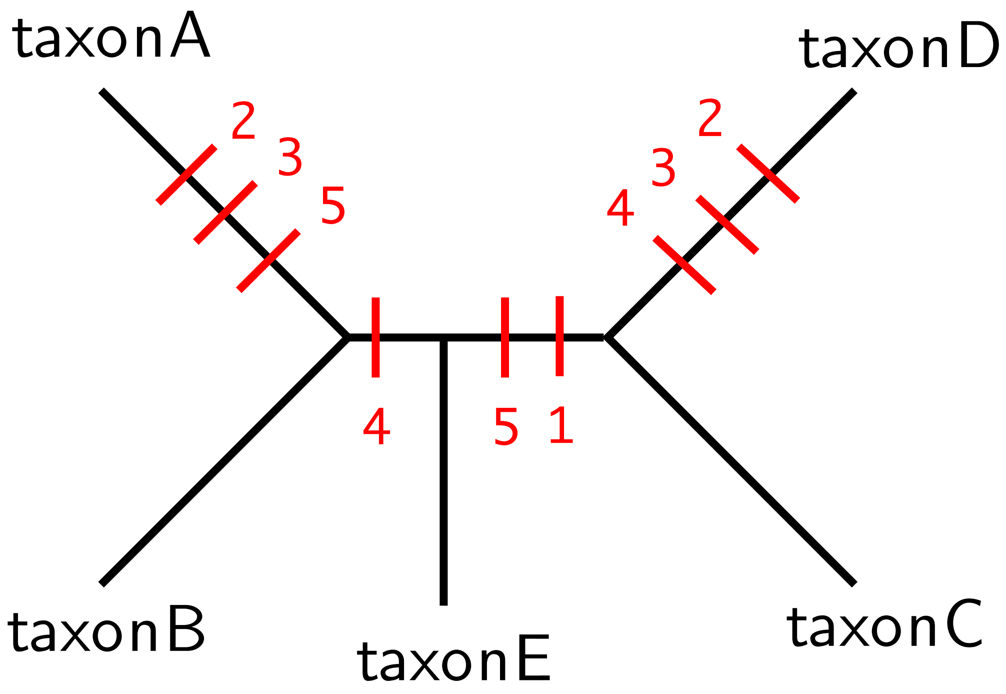
	1	2	3	4	5
taxonA	A	A	A	A	A
taxonB	A	C	C	A	C
taxonC	C	C	C	T	T
taxonD	C	A	A	C	T





	1	2	3	4	5
taxonA	A	A	A	A	A
taxonB	A	C	C	A	C
taxonC	C	C	C	T	T
taxonD	C	A	A	C	T
taxonE	A	A	A	T	C





Comparison of two five taxon trees

	1	2	3	4	5	
taxonA	A	A	A	A	A	
taxonB	A	C	C	A	C	
taxonC	C	C	C	T	T	
taxonD	C	A	A	C	T	
taxonE	A	A	A	T	C	
tree ((A, B), E, (C, D))	1	2	2	2	2	9
tree ((A,E), D, (B, C))	2	1	1	3	3	10

Stepwise addition

- heuristic – not guaranteed to find the best tree
- Number of trees scored for N taxa :

$$\begin{aligned}\text{\# trees scored} &= \sum_{i=3}^{N-1} (2i-3) \\ &= (N-1)(N-3)\end{aligned}$$

Thus, stepwise addition is $O(N^2)$. For N=10:

$$63 = 3 + 5 + 7 + 9 + 11 + 13 + 15$$

Trying to improve a tree

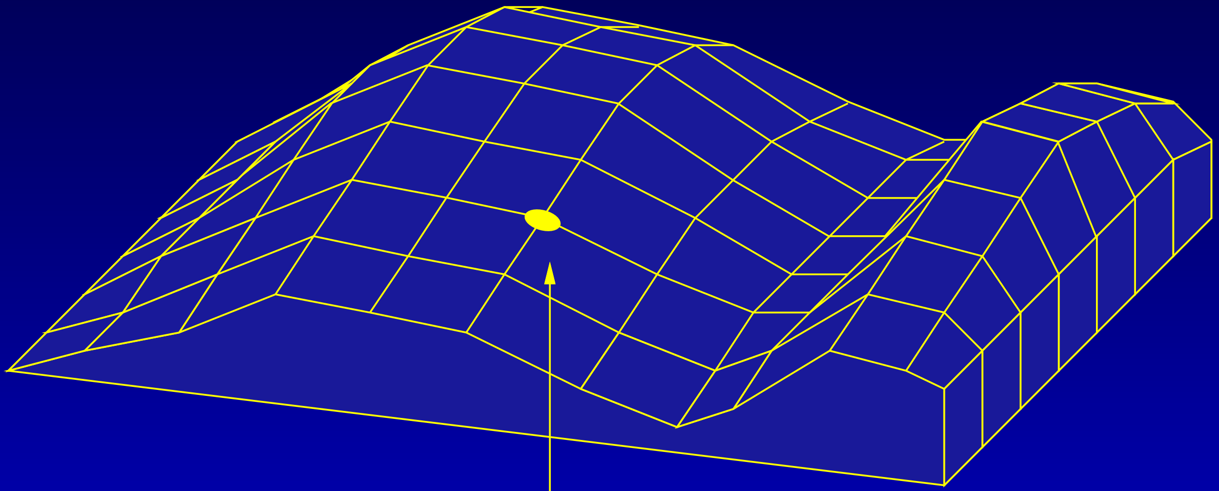
Heuristic hill-climbing searches can work quite well:

1. Start with a tree
2. Score the tree
3. Consider a new tree within the neighborhood of the current tree:
   (a) Score the new tree.
   (b) If the new tree has a better tree, use it as the “current tree”
   (c) Stop if there are no other trees within the neighborhood to consider.

These are **not** guaranteed to find even one of the optimal trees.

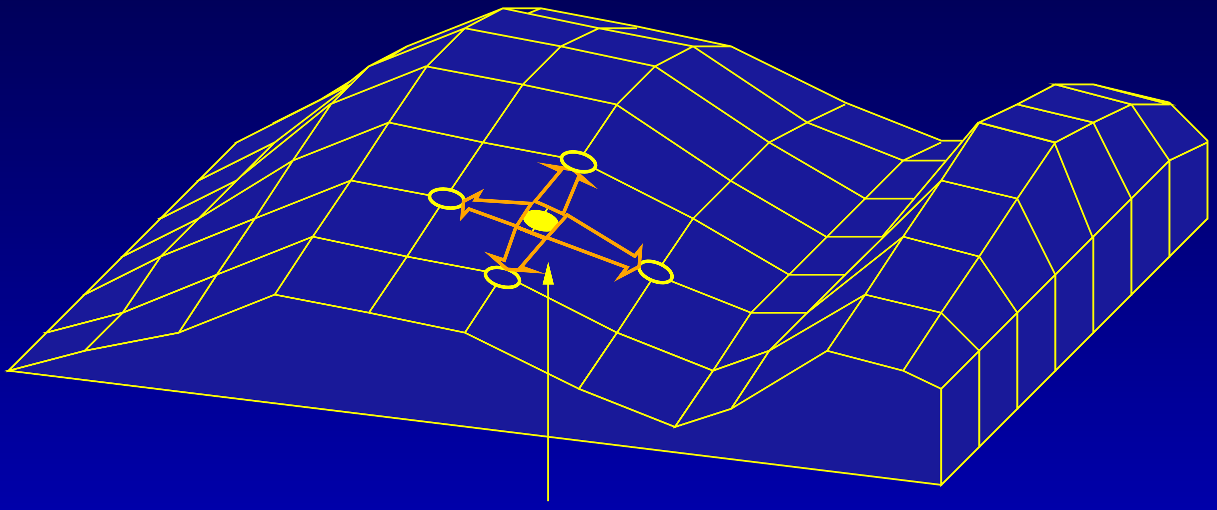
The most common way to explore the neighborhood of a tree is to swap the branches of the tree to construct similar trees.

Greedy search for a maximum



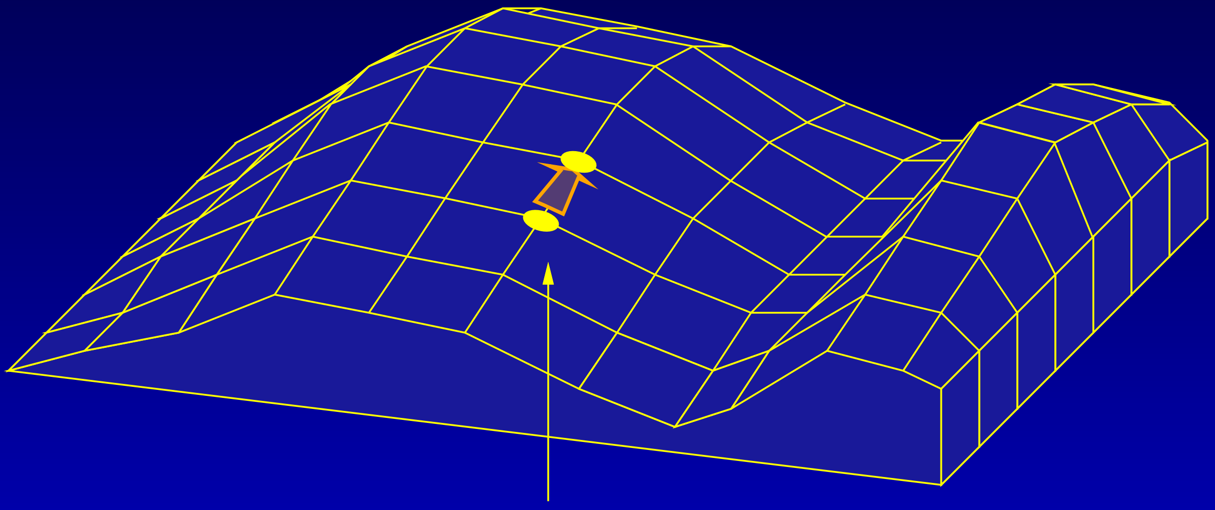
If start here

Greedy search for a maximum



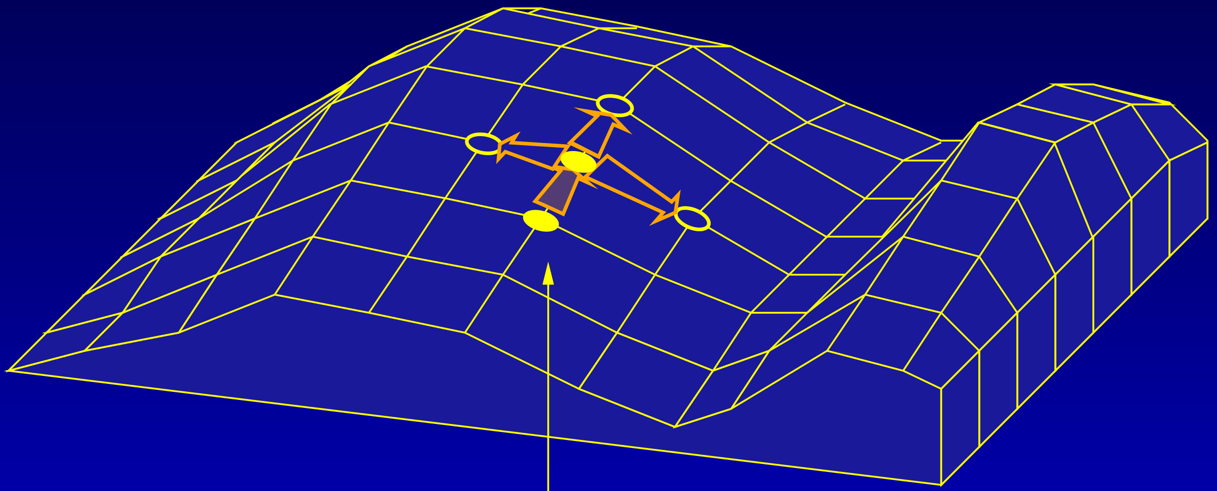
If start here

Greedy search for a maximum

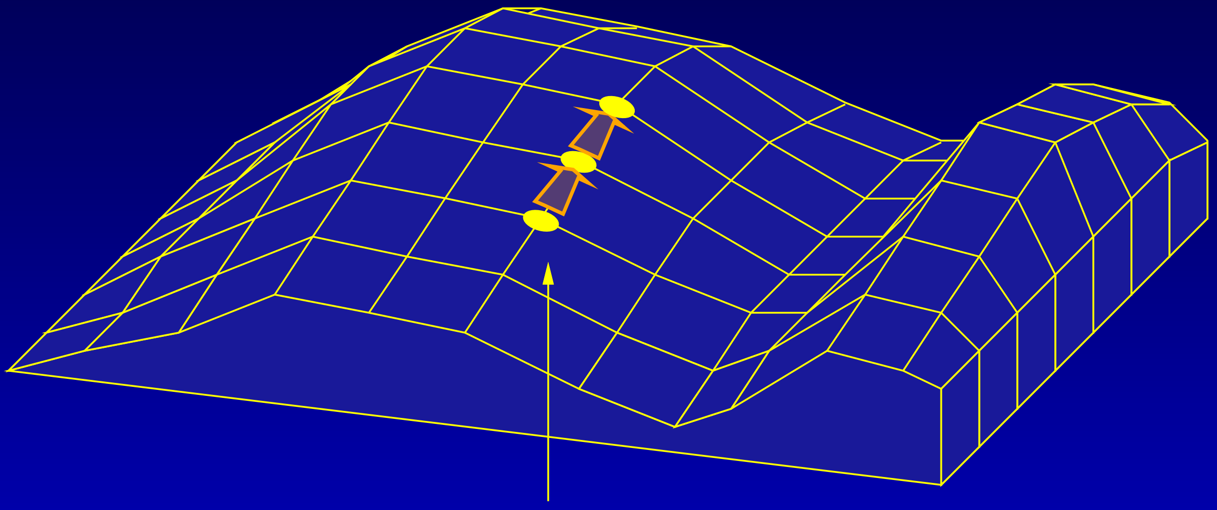


If start here

Greedy search for a maximum

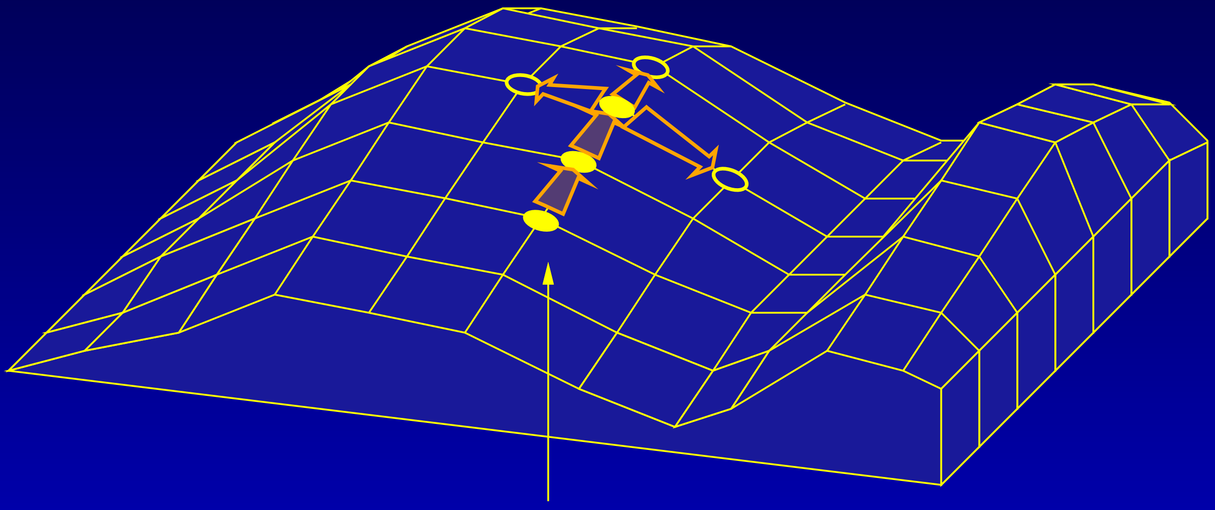


Greedy search for a maximum



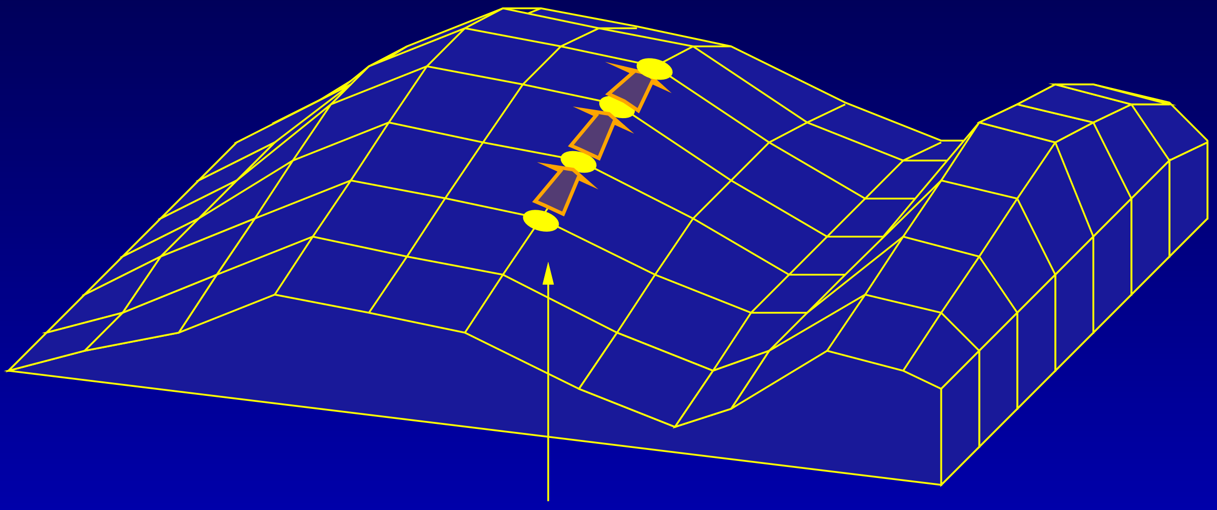
If start here

Greedy search for a maximum



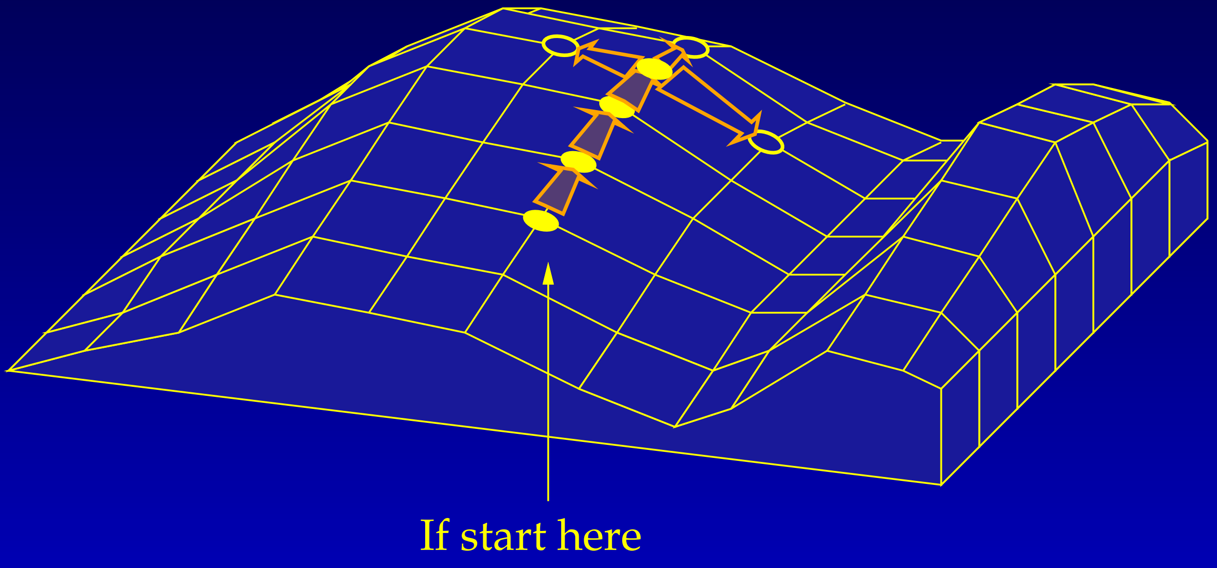
If start here

Greedy search for a maximum

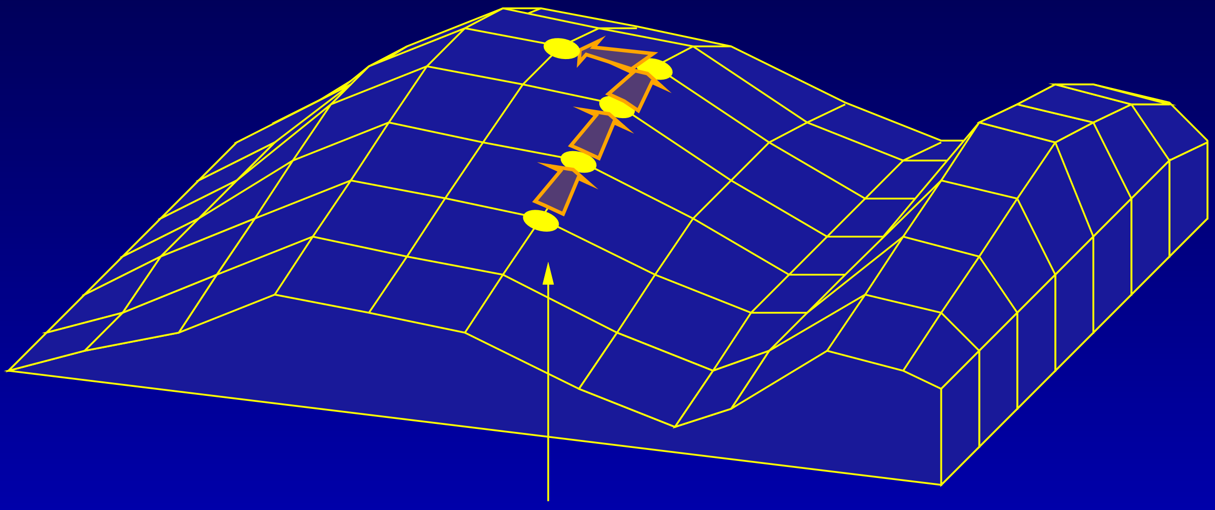


If start here

Greedy search for a maximum

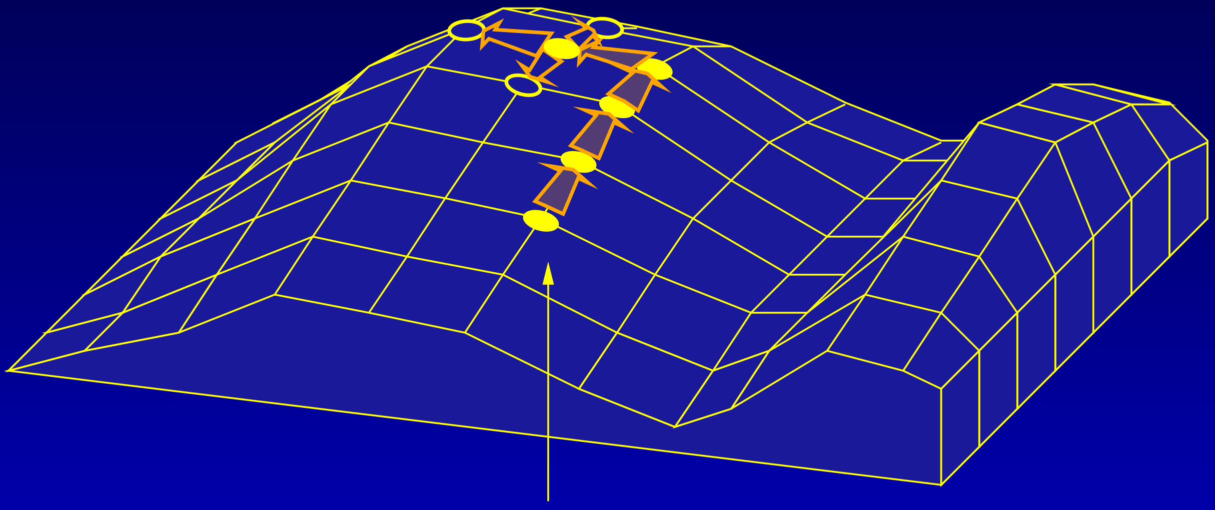


Greedy search for a maximum



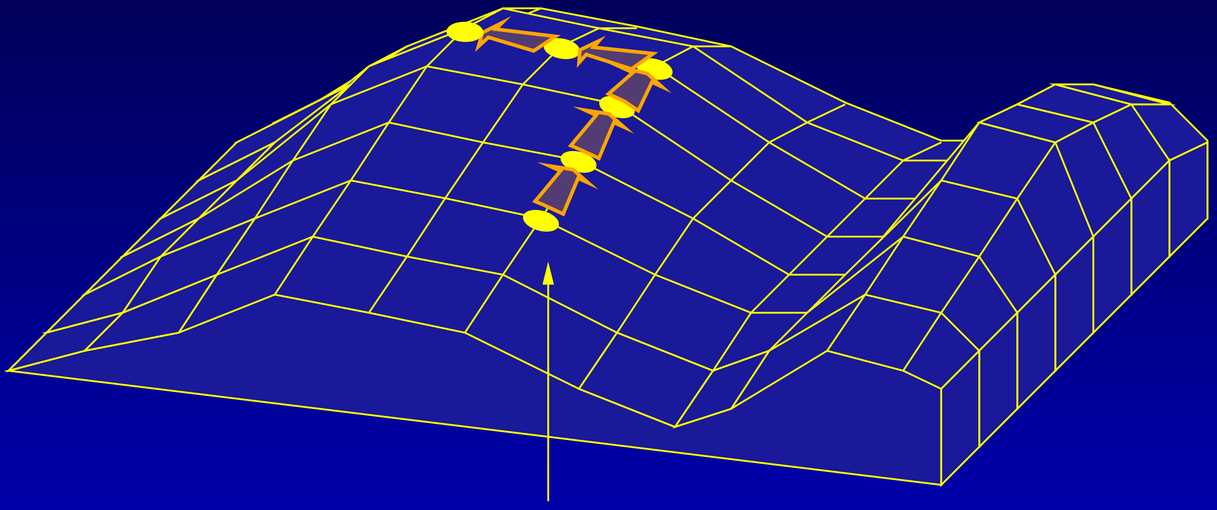
If start here

Greedy search for a maximum

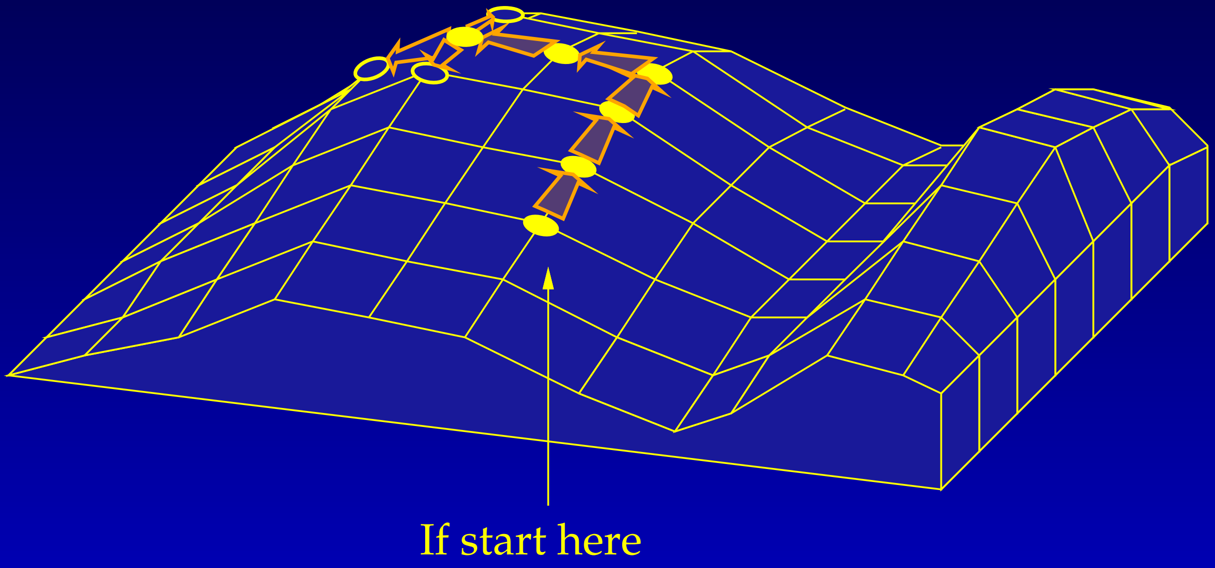


If start here

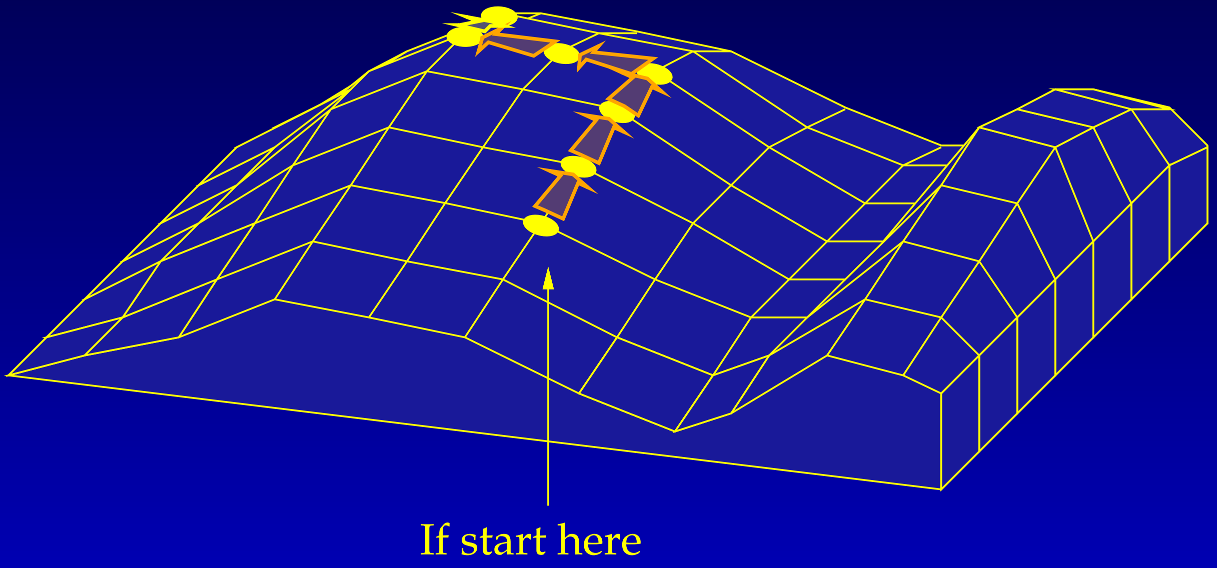
Greedy search for a maximum



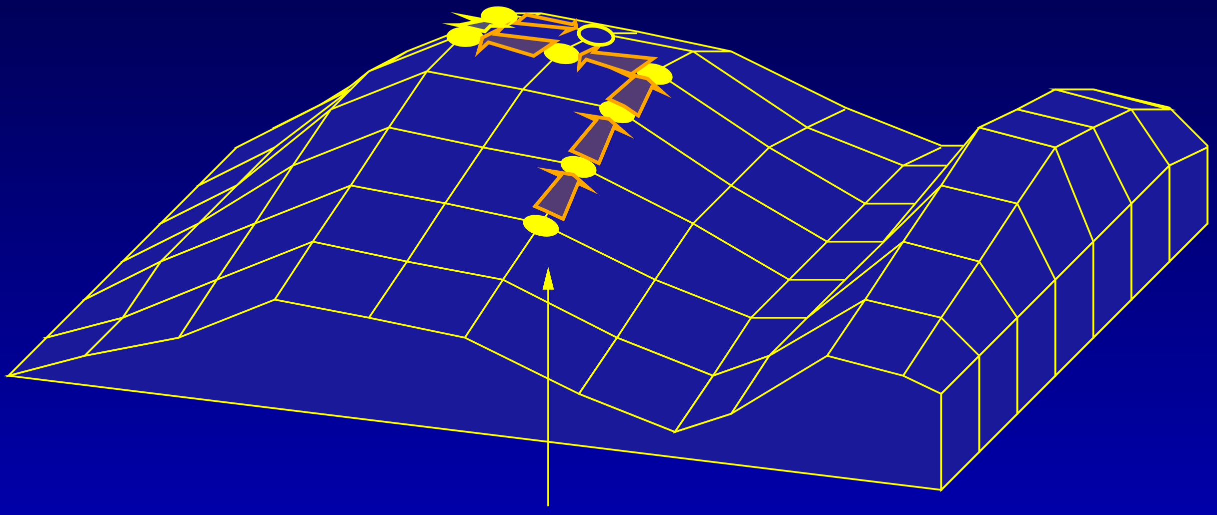
Greedy search for a maximum



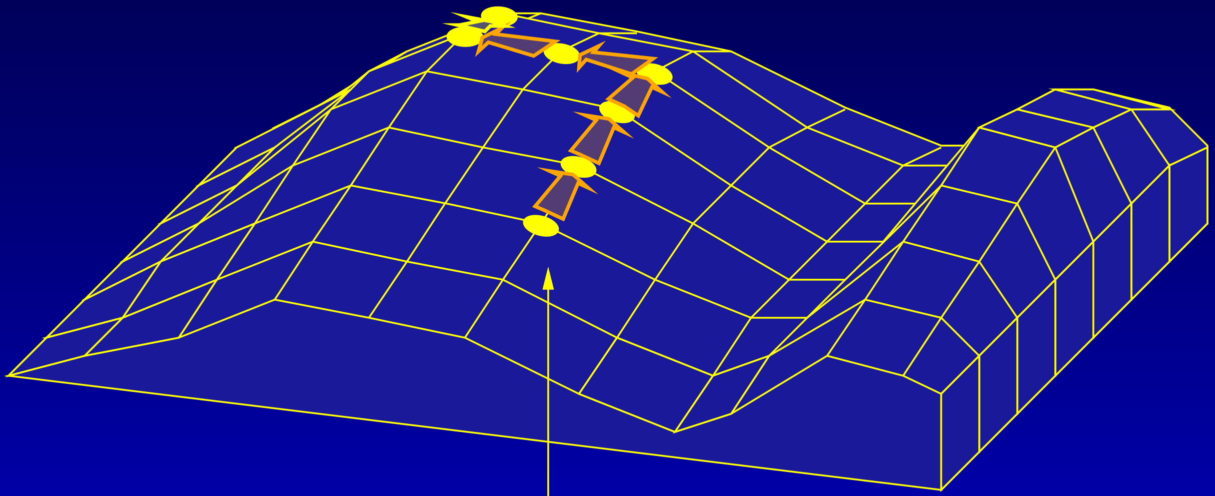
Greedy search for a maximum



Greedy search for a maximum

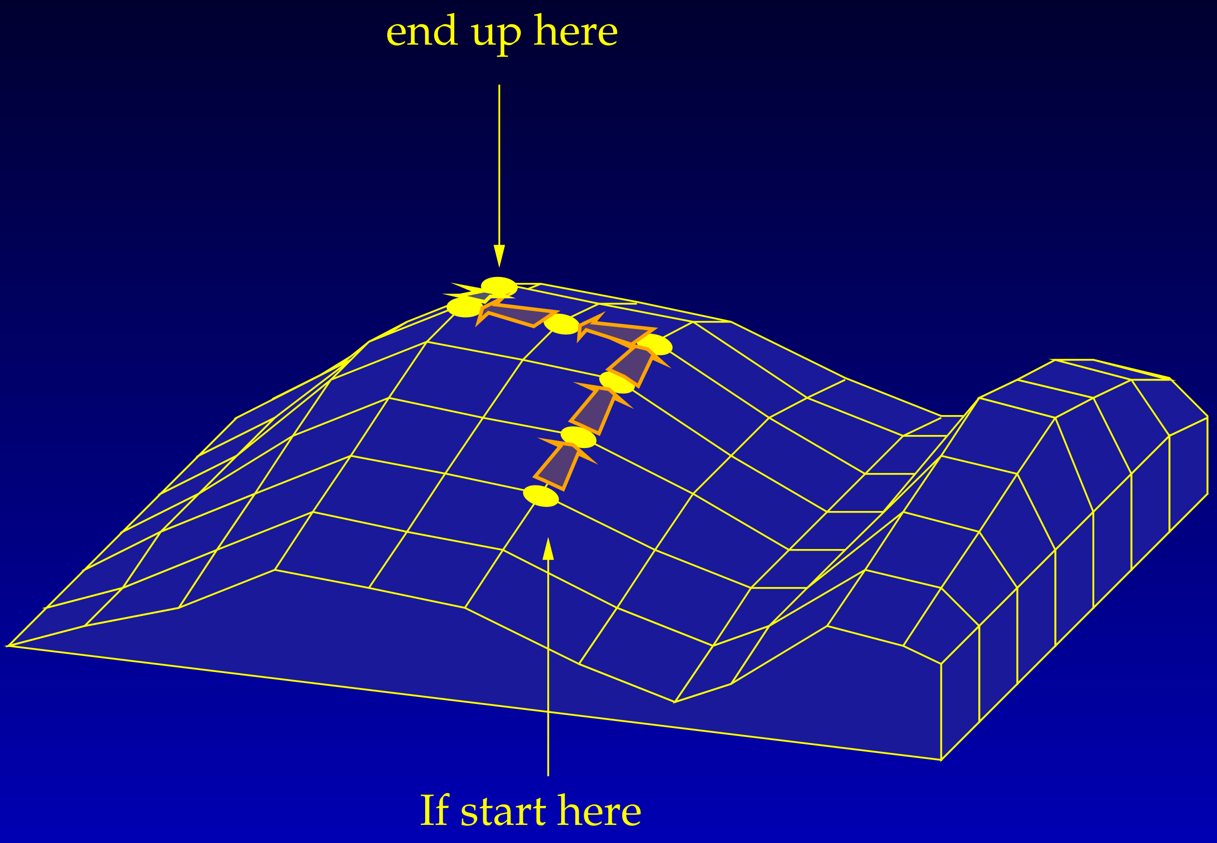


Greedy search for a maximum

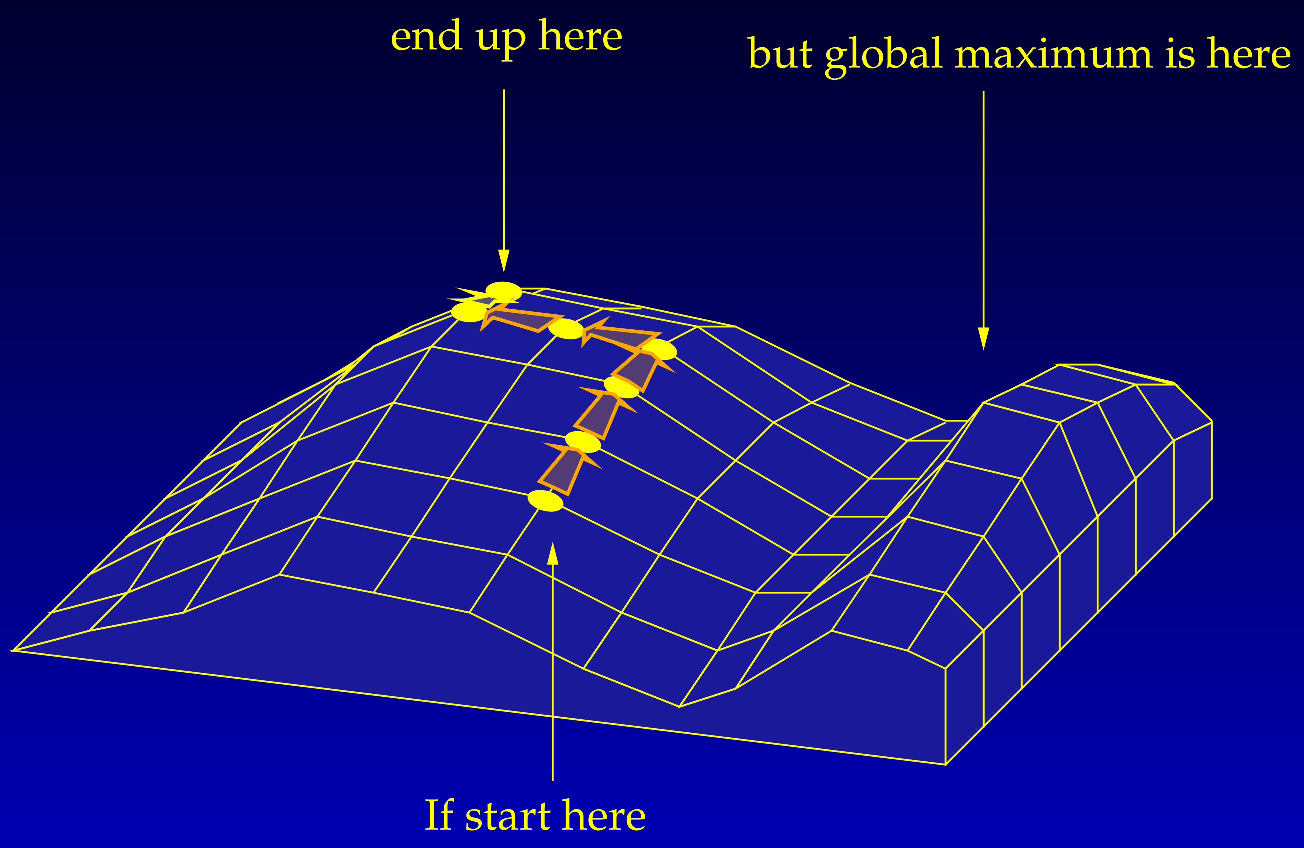


If start here

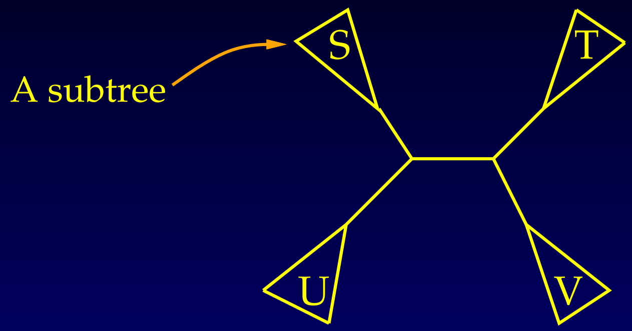
Greedy search for a maximum



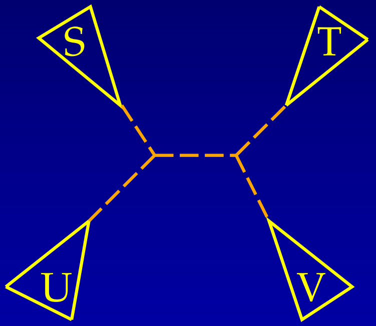
Greedy search for a maximum



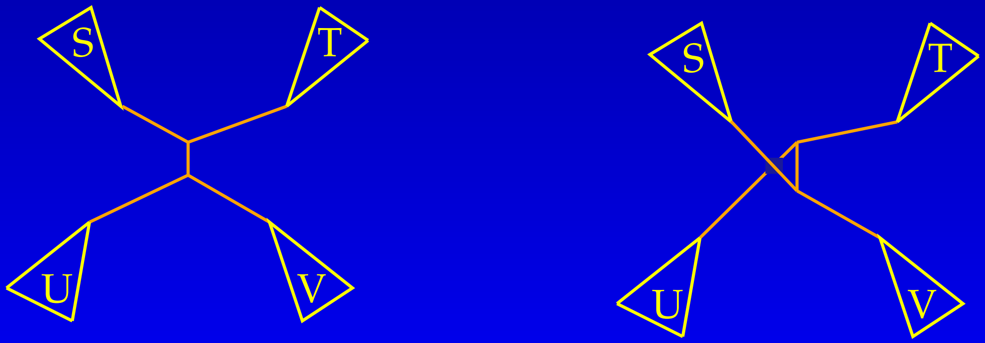
Nearest-neighbor rearrangements



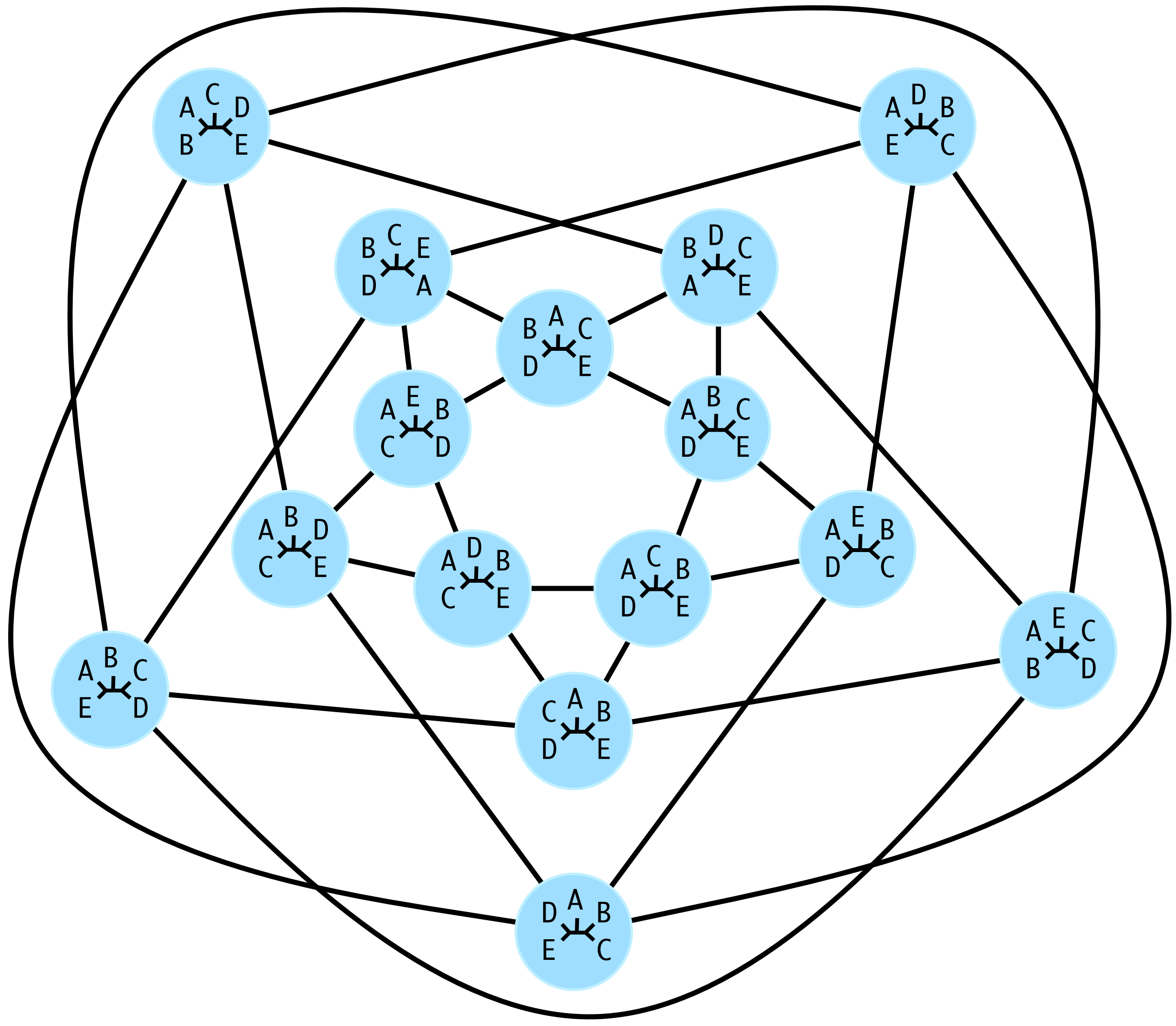
is rearranged by dissolving the connections to an interior branch



and reforming them in one of the two possible alternative ways:



Schoenberg graph – edges connect NNI neighbors



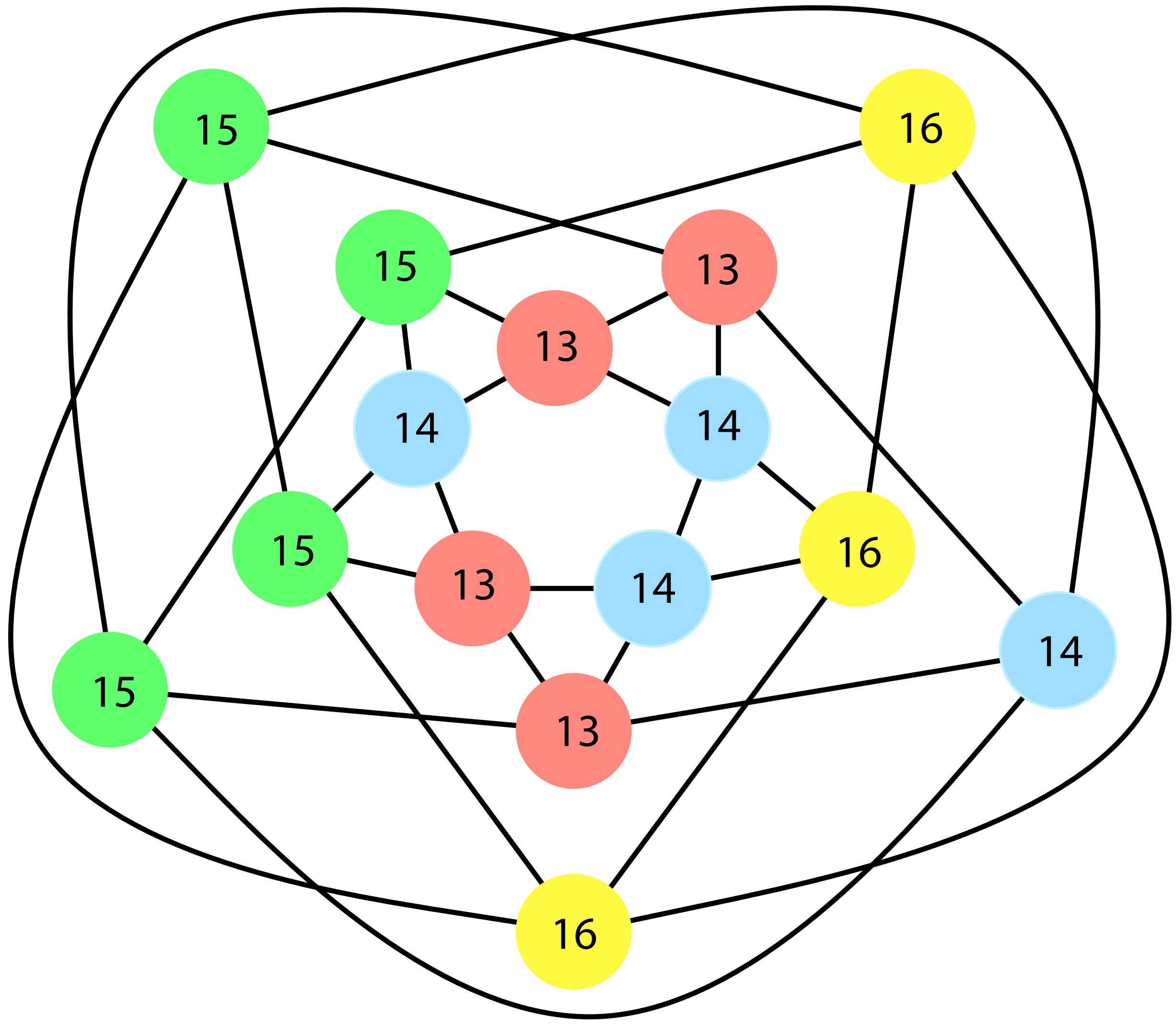
Tree "Islands" possible

An $Op - L$ tree island (*sensu* *Maddison, 1991*): A set of trees with score $\leq L$ that are connected to each other by Op operations such that you can get from any tree in the set to any other tree by repeated Op changes and all intermediate trees along the path are also members of the set.

The following Schoenberg graph shows the scores of the 15 trees on the following dataset (contrived data by POL):

```
A ACGCAGGT
B ATGGTGAT
C GCTCACGG
D ACTGTCGT
E GTTCTGAG
```

Schoenberg graph with parsimony scores



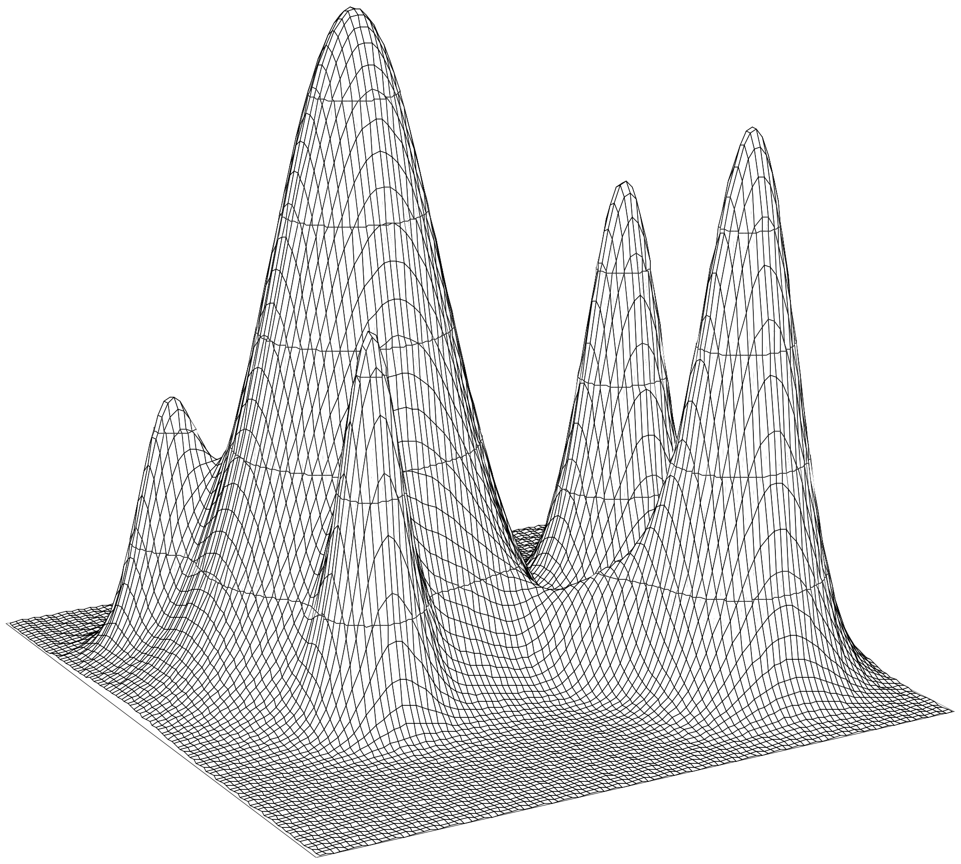
Tree Islands implications

1. Islands can be larger than 1 tree – we must consider ties if we want to find all trees that optimize the score.

2. Swapping to completion on all optimal trees found in a search is **not** guaranteed to succeed.

3. The delimitation of an island depends on tree changing operation used.

Heuristics explore “Tree Space”

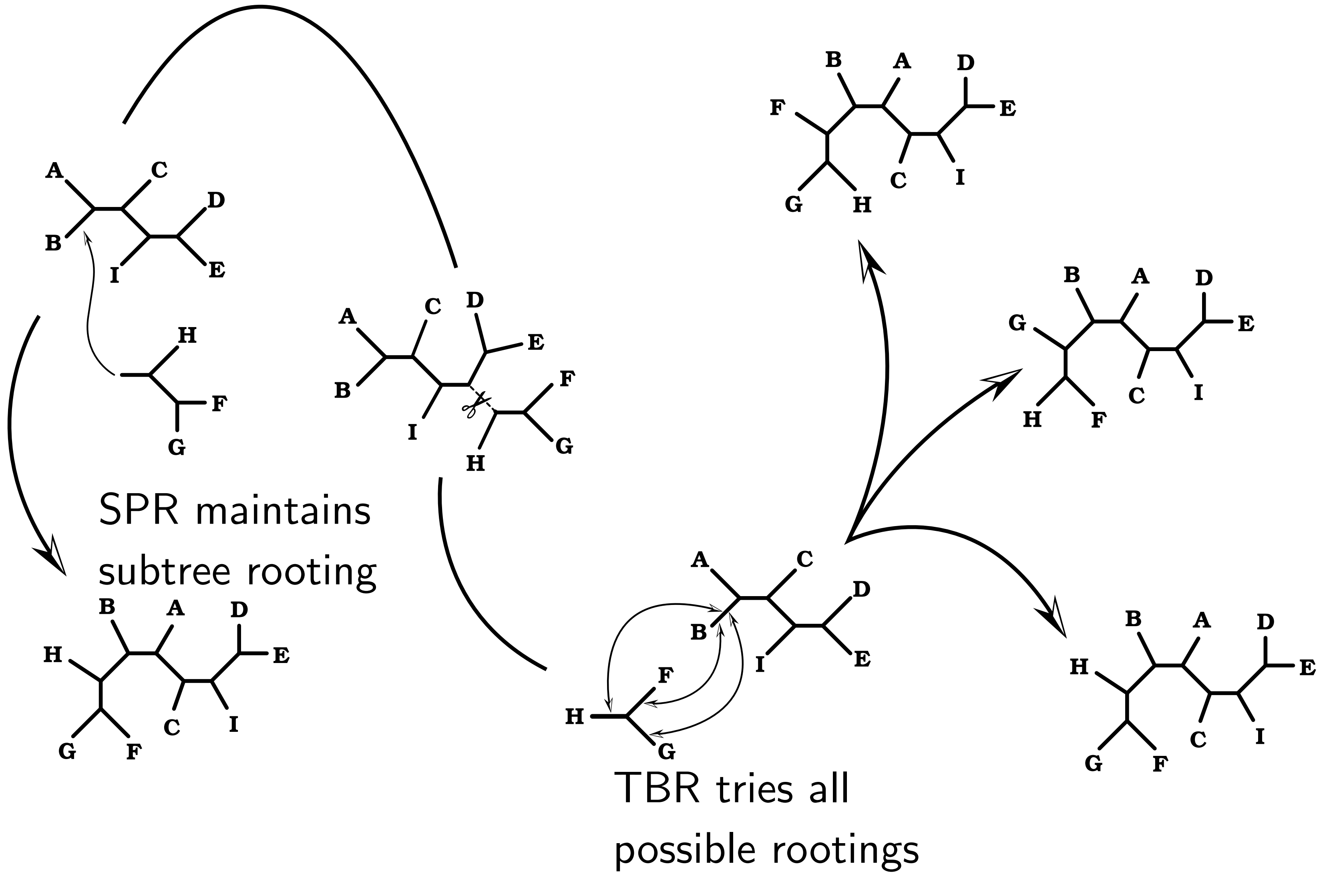


Most commonly used methods are “hill-climbers.”

Multiple optima found by repeating searches from different origins.

Severity of the problem of multiple optima depends on step size.

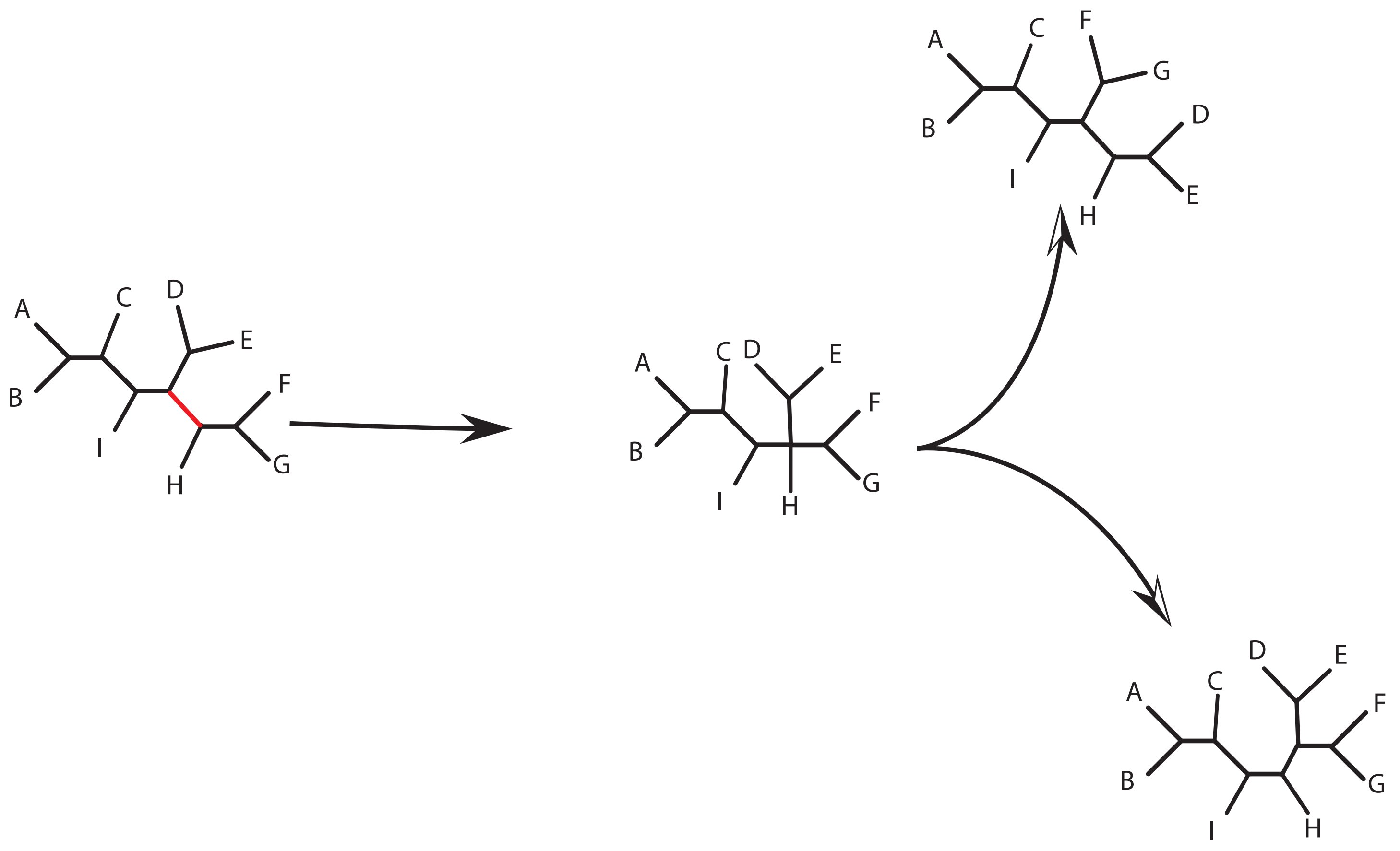
Subtree Pruning Regrafting (SPR) and Tree Bisection Reconnection (TBR)



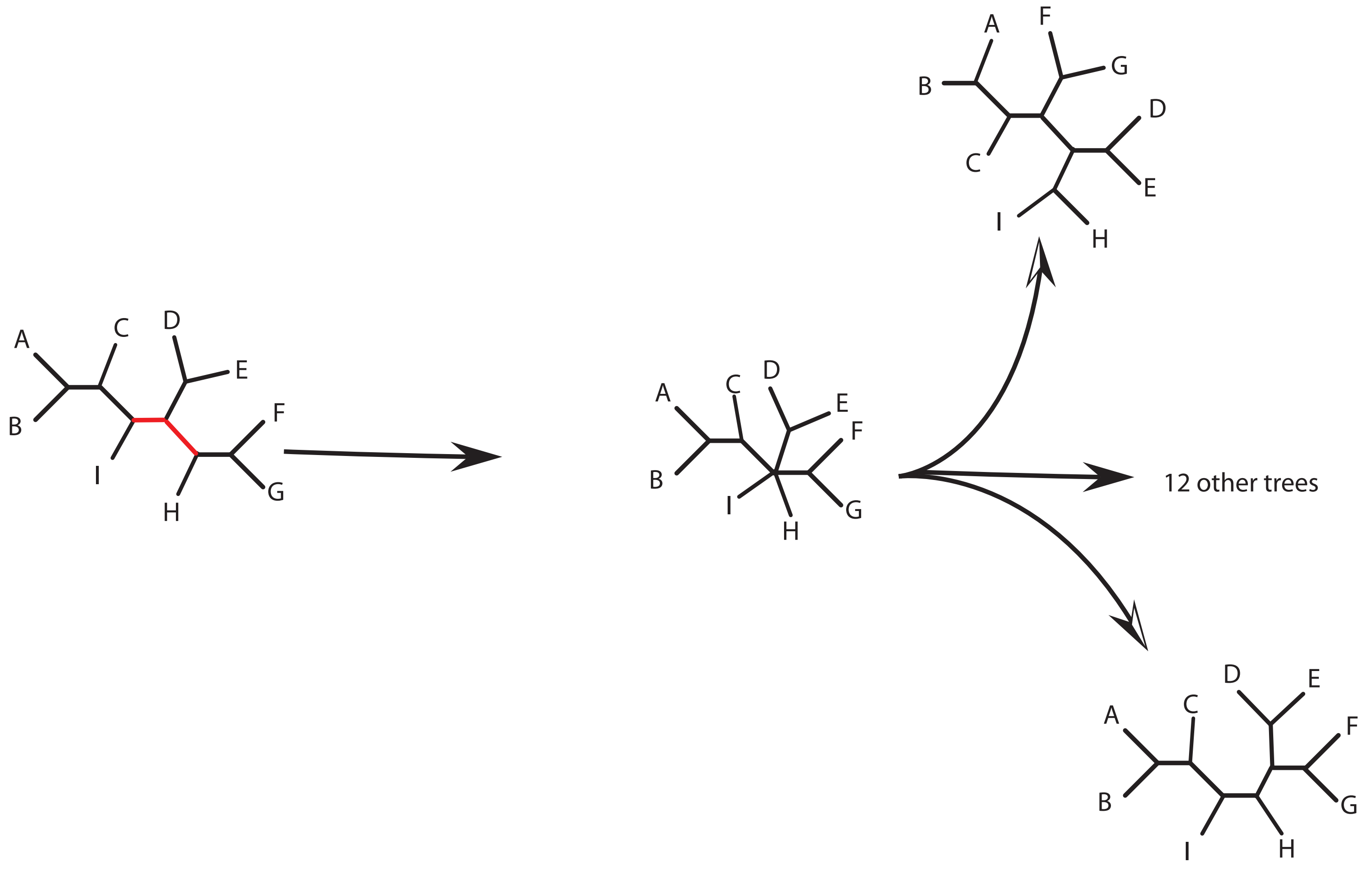
SPR maintains subtree rooting

TBR tries all possible rootings

1-Edge-contract Refine



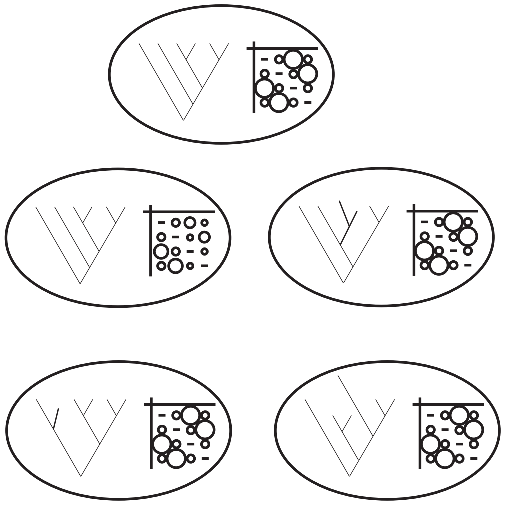
2-Edge-contract Refine

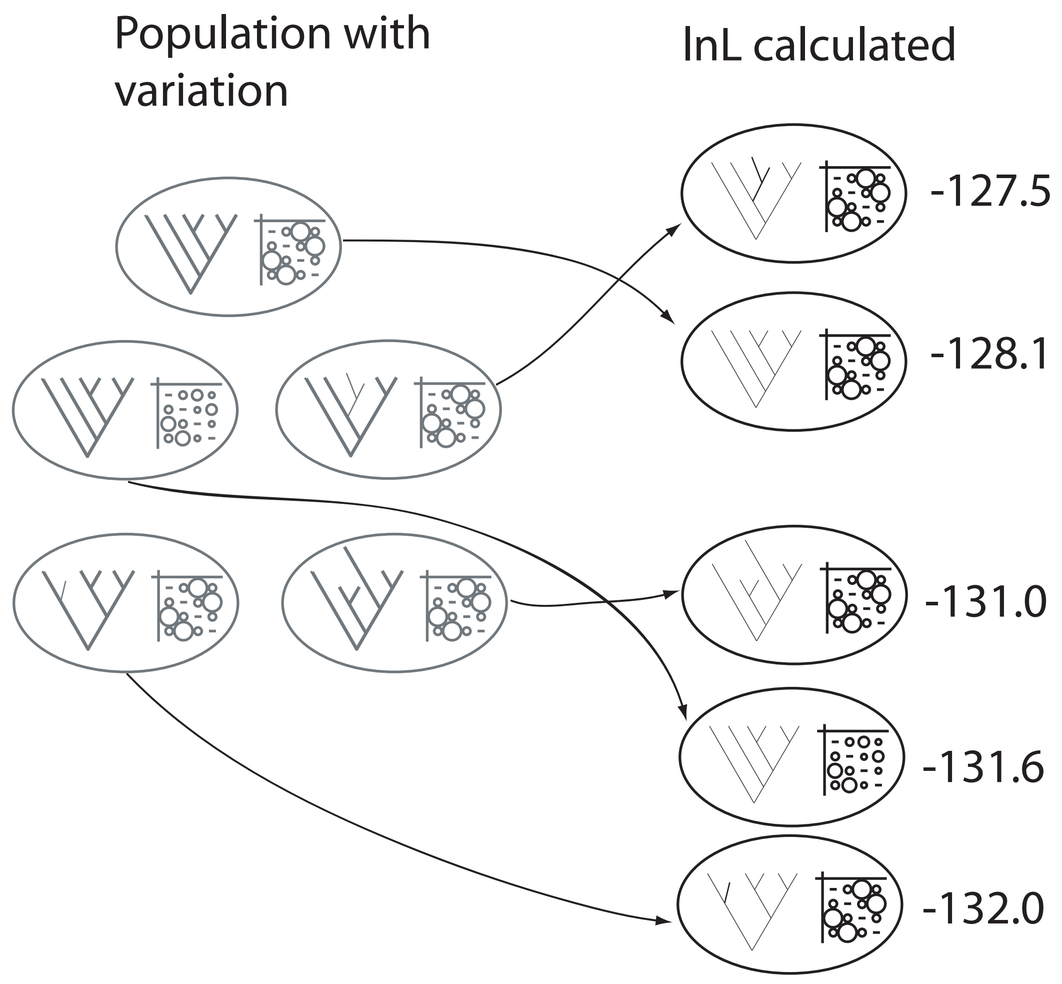


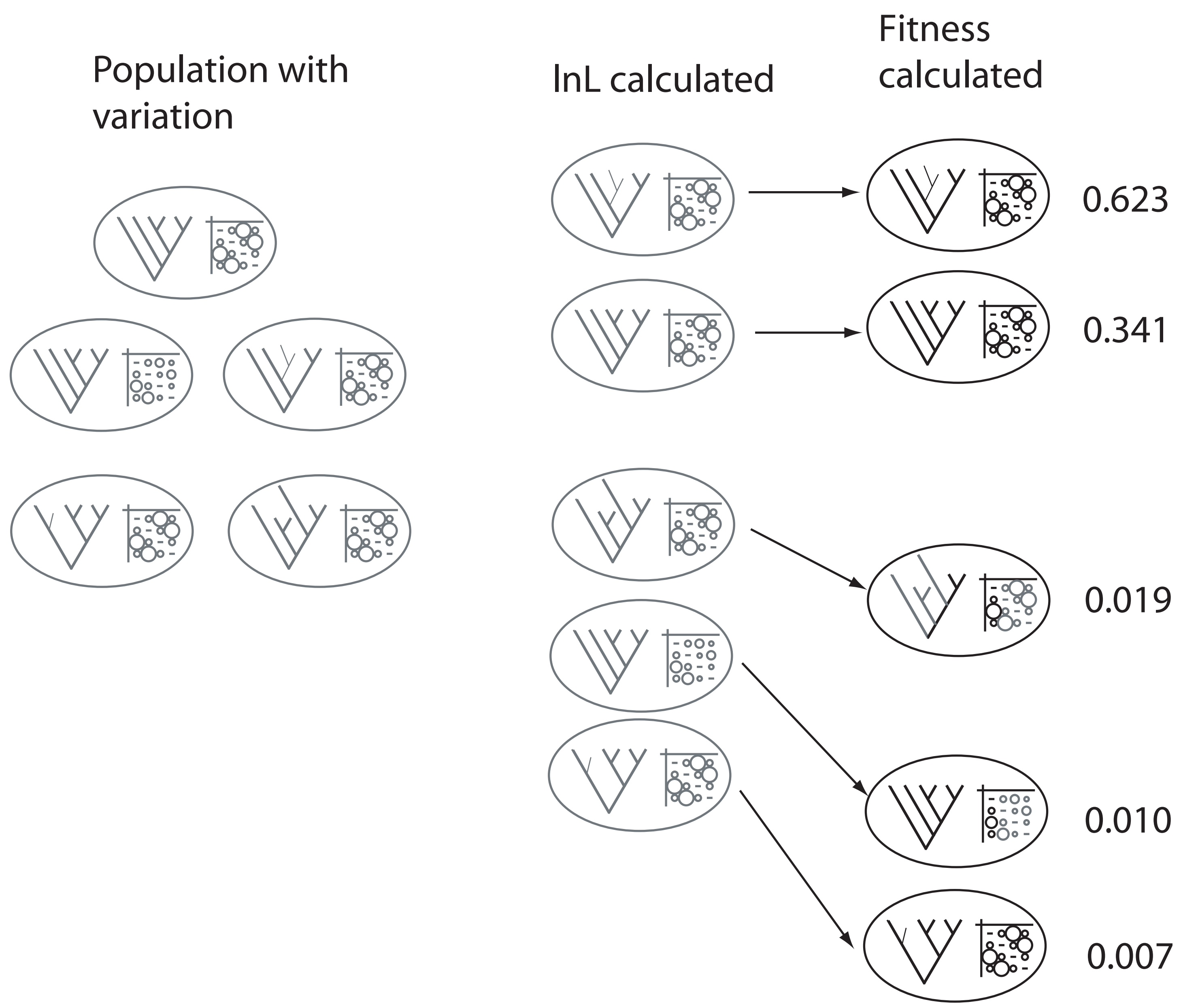
Many other heuristic strategies proposed

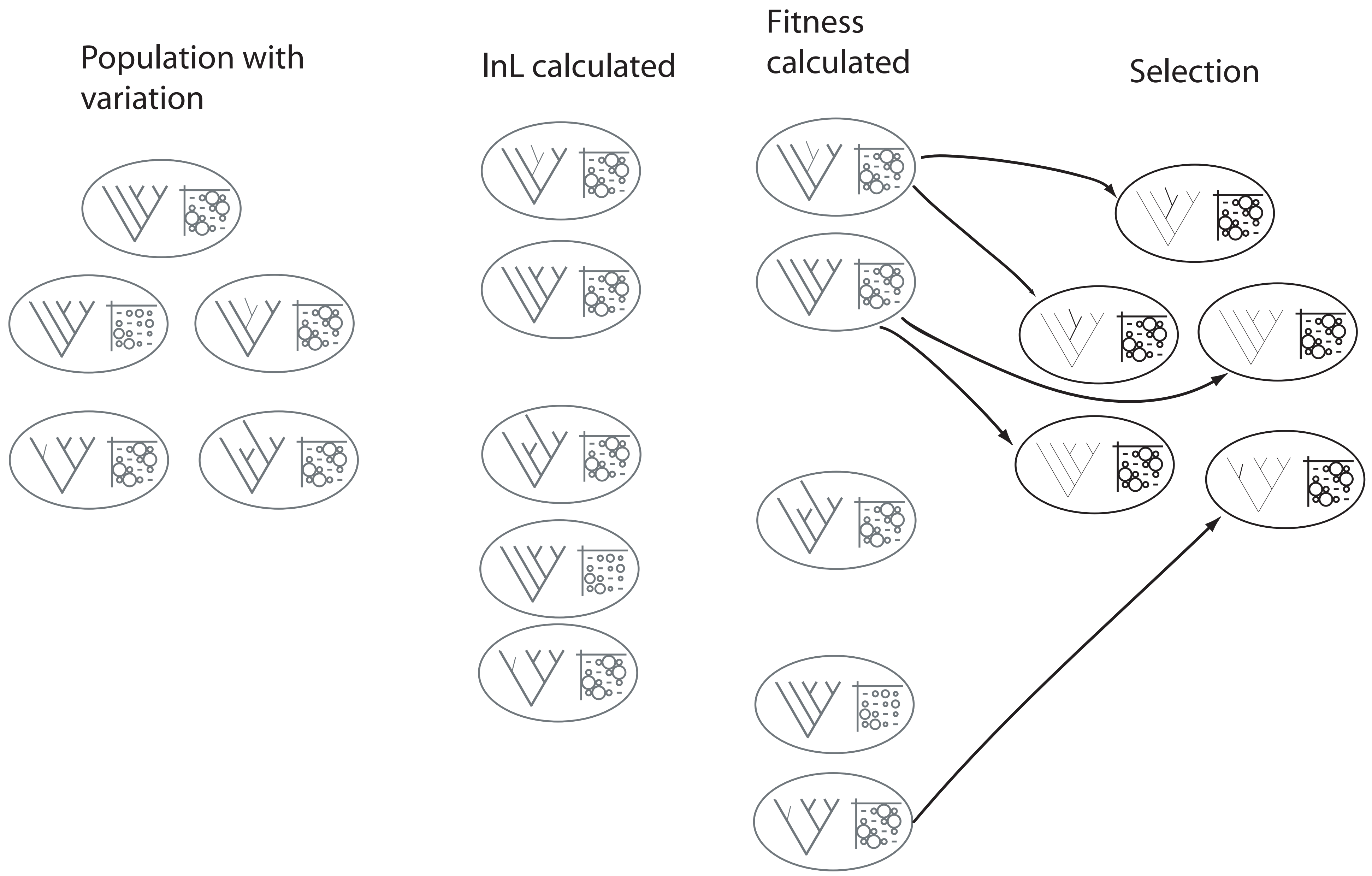
- Swapping need not include *all* neighbors (RAxML, `reconlimit` in PAUP*)
- "lazy" scoring of swaps (RAxML)
- Ignoring (at some stage) interactions between different branch swaps (PHYML)
- Stochastic searches
  - Genetic algorithms (GAML, MetaPIGA, GARLI)
  - Simulated annealing
- Divide and conquer methods (the sectortial searching of Goloboff, 1999; Rec-I-DCM3 Roshan 2004)
- Data perturbation methods (*e.g.* Kevin Nixon's "ratchet")

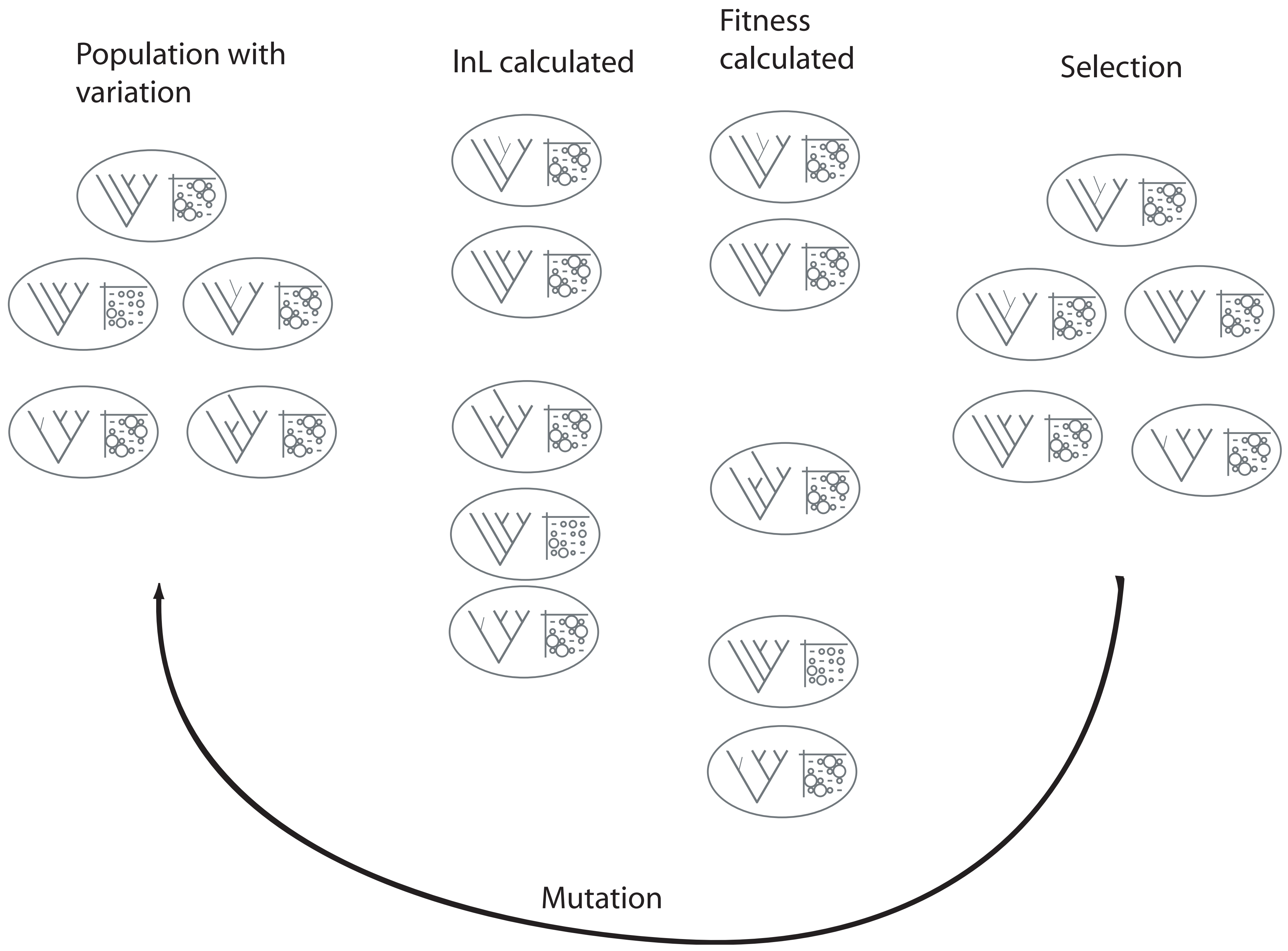
Population with variation




Population with variation
lnL calculated
-127.5
-128.1
-131.0
-131.6
-132.0


Population with variation
lnL calculated
Fitness calculated
0.623
0.341
0.019
0.010
0.007


Population with variation
lnL calculated
Fitness calculated
Selection


Population with variation
lnL calculated
Fitness calculated
Selection
Mutation

Divide-and-Conquer Methods

The basic outline of a phylogenetic Divide-and-Conquer approach is:

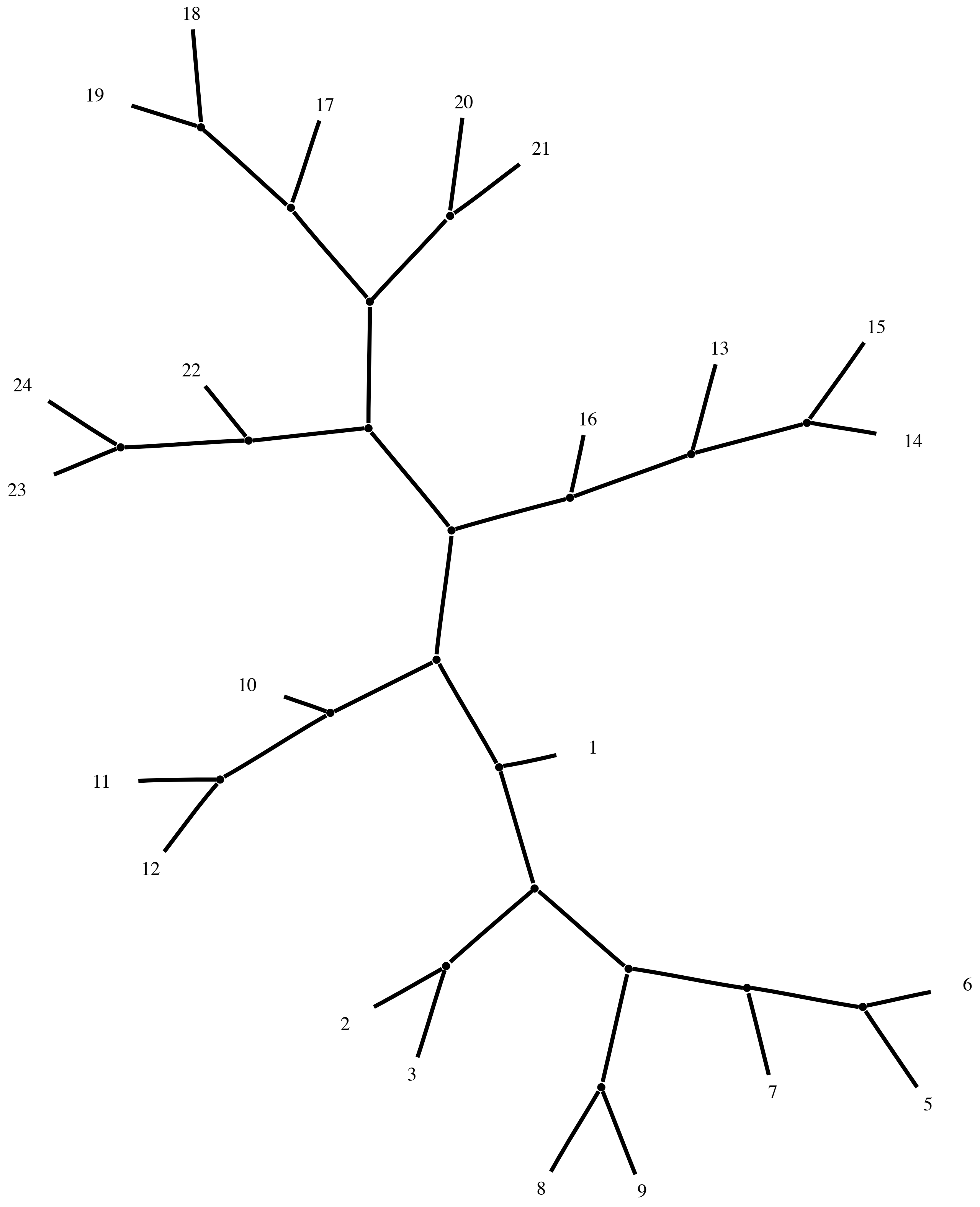
1. **Decompose** a starting tree into subsets of the taxa.
2. **Improve** the tree for each of the subsets of taxa.
3. **Merge** the resulting trees into a tree for the full set of taxa.
4. **Refine** the full tree (it will often have polytomies).
5. **Improve** the full tree using a simple (and fast) heuristic.

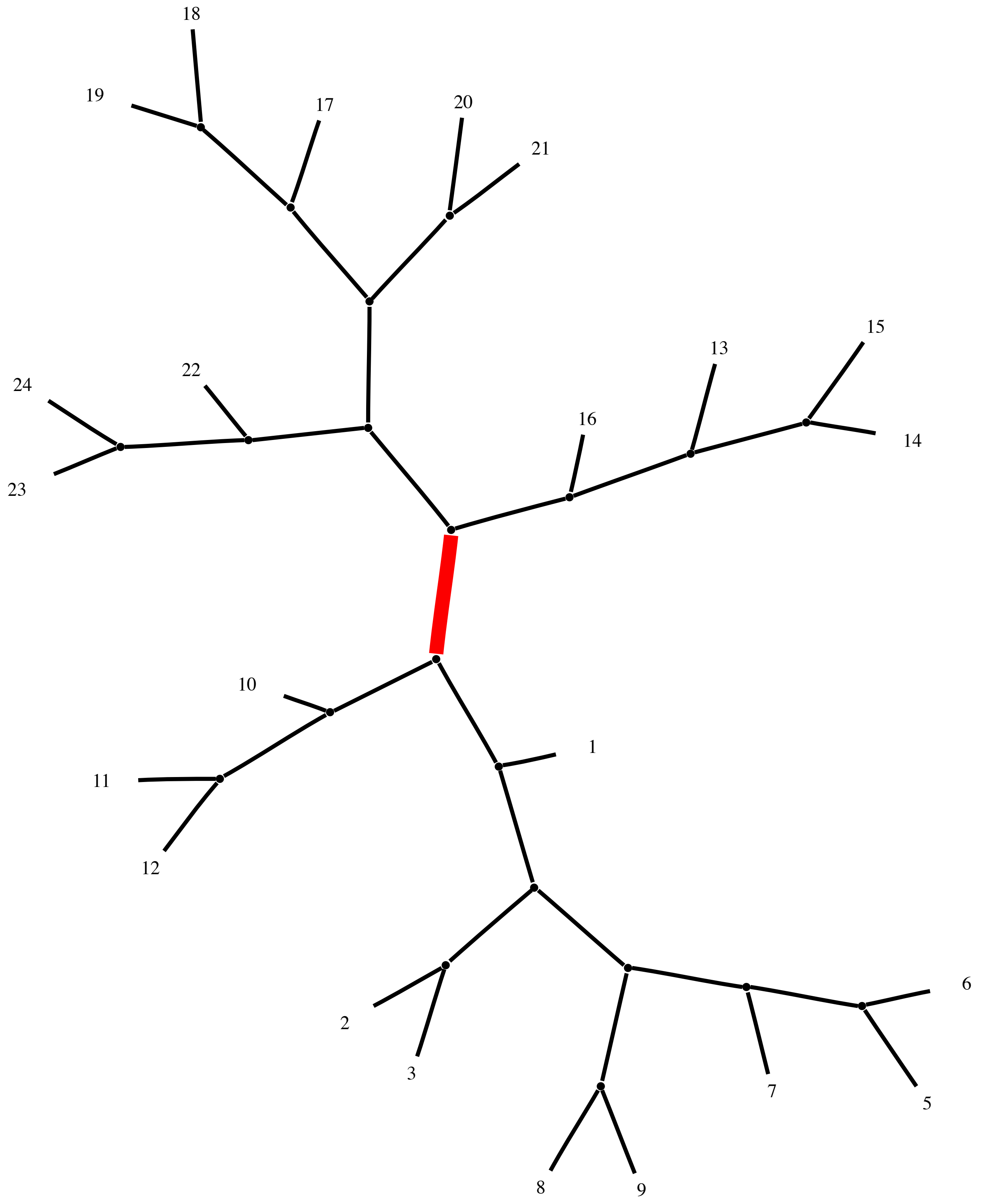
Examples include Rec-I-DCM3 by Roshan *et al.*(2004). See Goloboff and Pol (*Systematic Biology*, 2007) for a contrasting viewpoint about the relative efficiency of Rec-I-DCM3 compared to heuristics implemented in TNT.

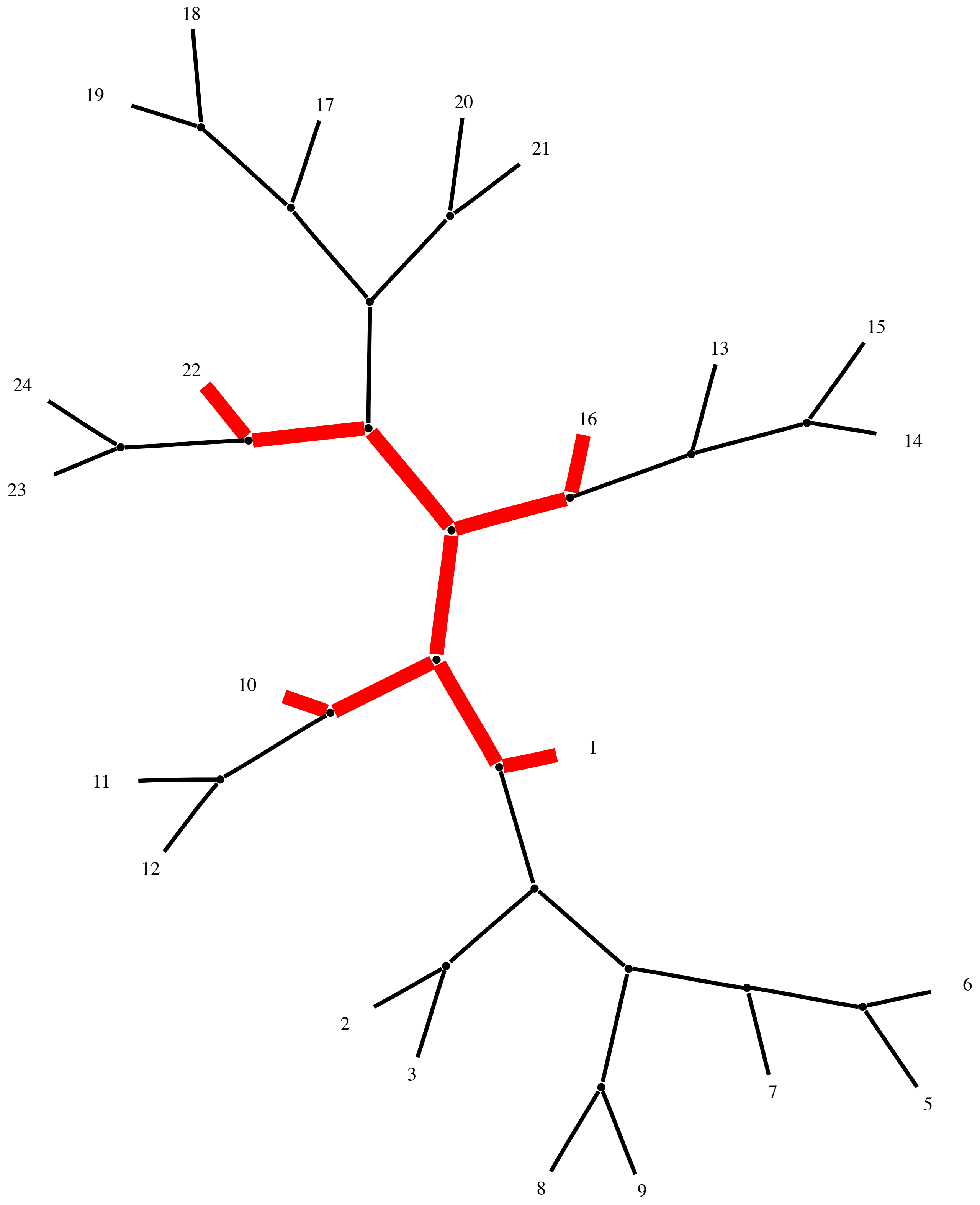
Step 1: Leaf set decomposition

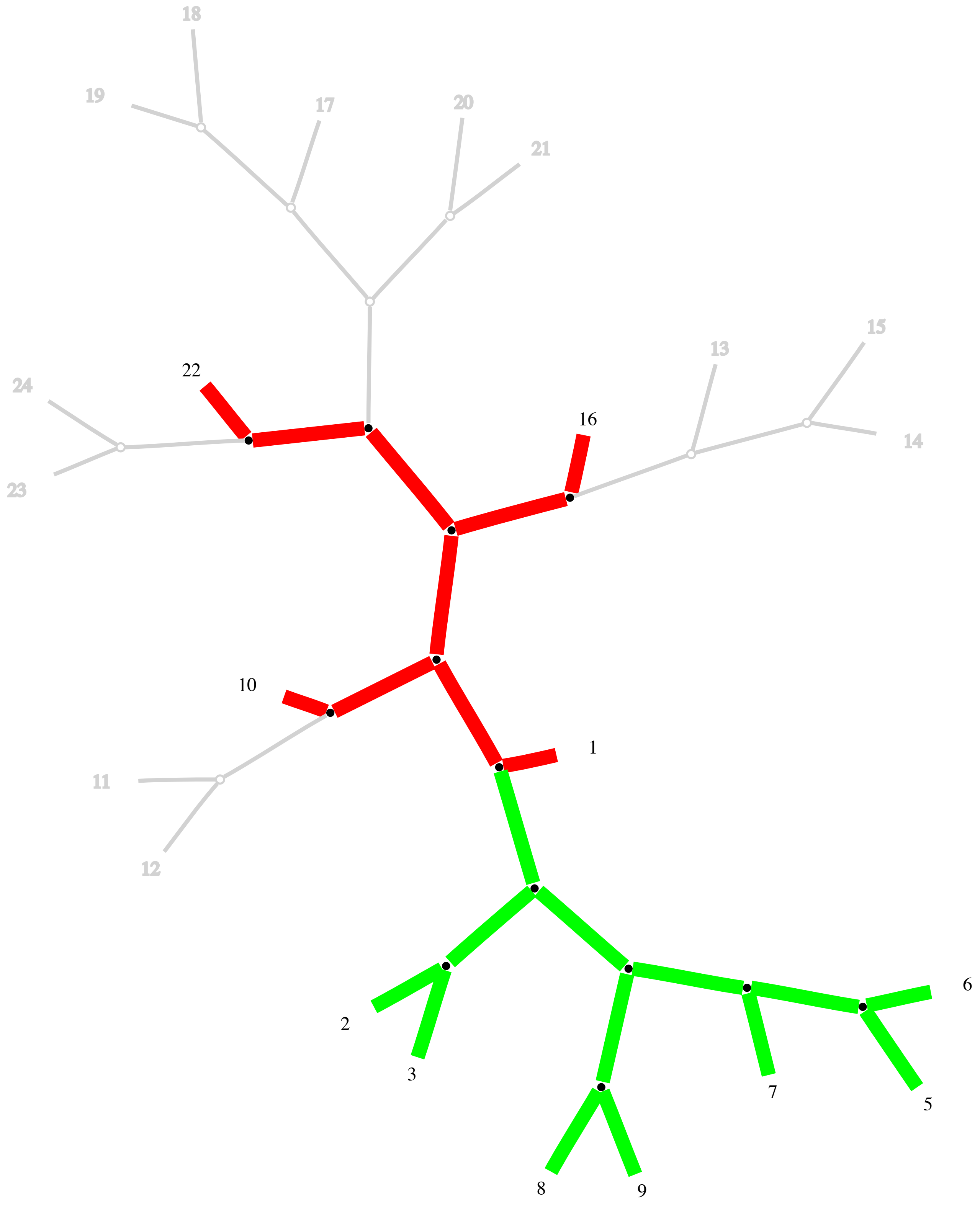
In Rec-I-DCM3 Roshan *et al.* (2004):

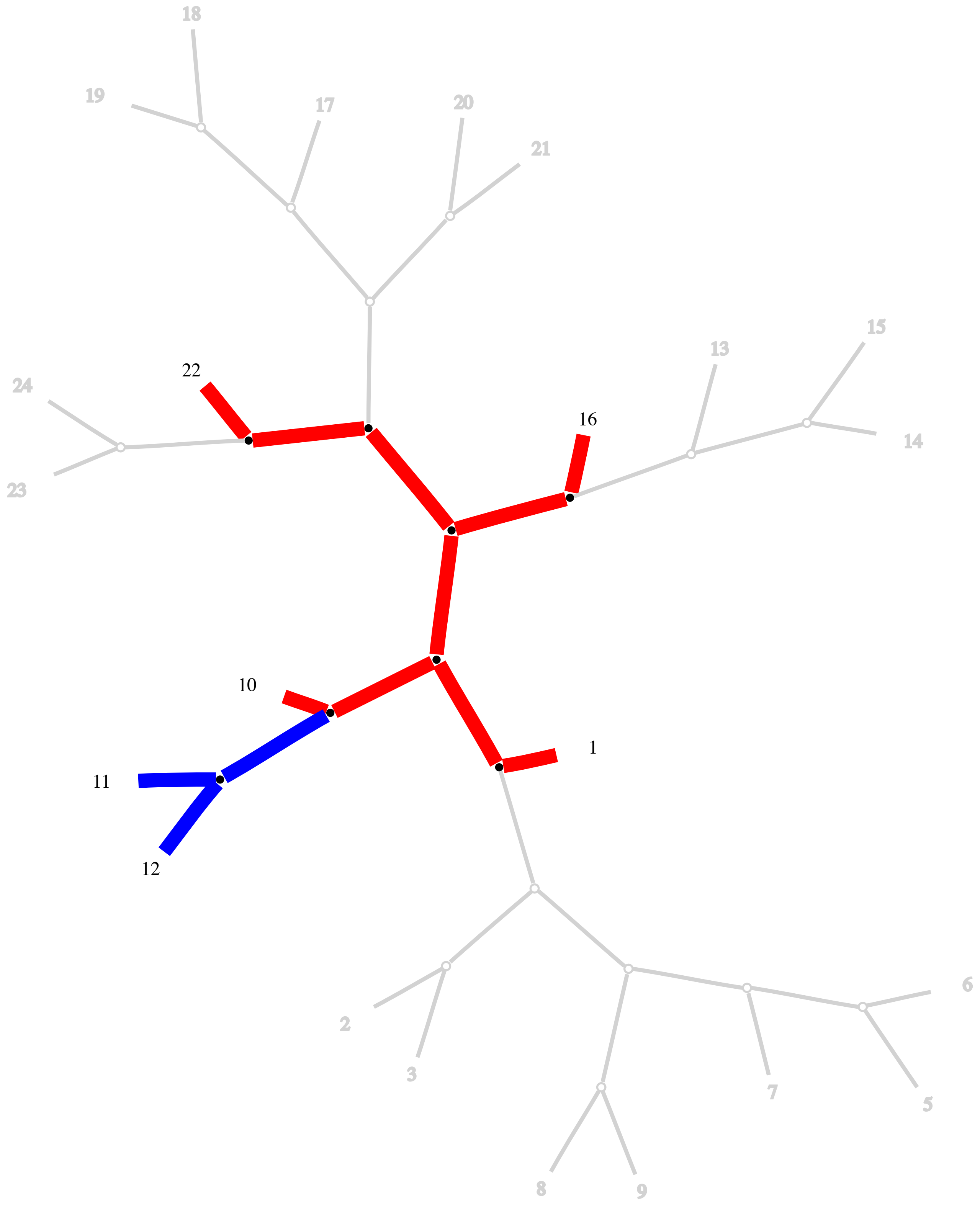
- A tree is divided ("decomposed") into 4 trees around a central edge. The edge is chosen such that it comes as close as possible to dividing the taxa into 2 equally-sized groups.
- The short quartet (taxa closest to this edge in each of the 4 directions) is selected.
- 4 sub-problems are produced. Each contains 1 subtree connected to the central edge and all leaves that are a part of the short quartet.

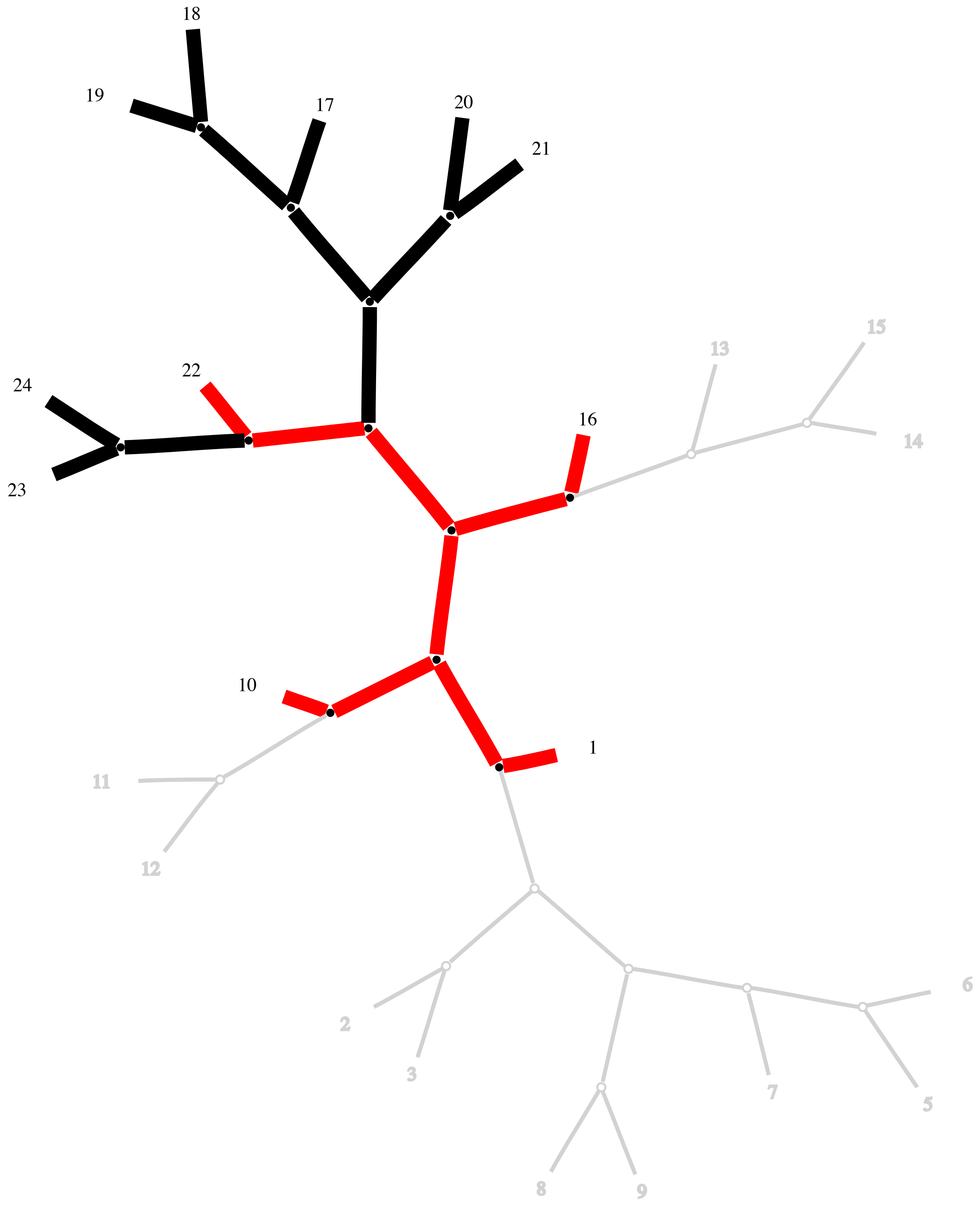

18
19
17
20
21
15
13
22
24
16
14
23
10
1
11
12
6
2
3
7
5
8
9

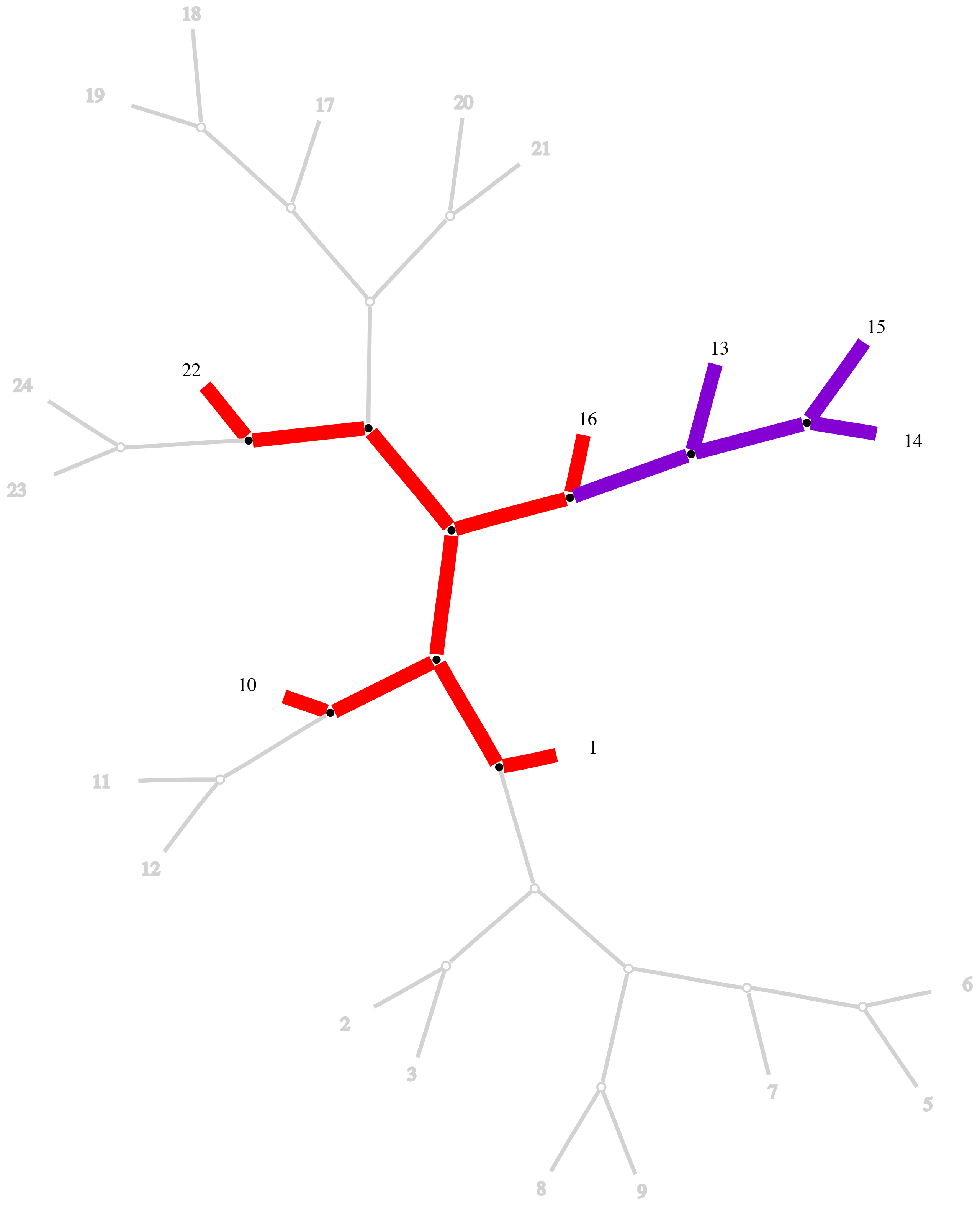

18
19
17
20
21
15
13
22
24
16
14
23
10
1
11
12
6
2
7
5
3
8
9


18
19
17
20
21
22
24
23
16
13
15
14
10
1
11
12
2
3
6
7
5
8
9


18
19
17
20
21
22
24
23
16
13
15
14
10
1
11
12
2
3
6
7
5
8
9


18
19
17
20
21
13
15
22
24
16
14
23
10
1
11
12
6
2
3
7
5
8
9


18
19
17
20
21
22
24
23
16
13
15
14
10
1
11
12
2
3
6
7
5
8
9


18
19
17
20
21
15
13
22
24
16
14
23
10
1
11
12
6
2
3
7
5
8
9

Step 2: Tree improvement

Simply a tree search on a smaller tree

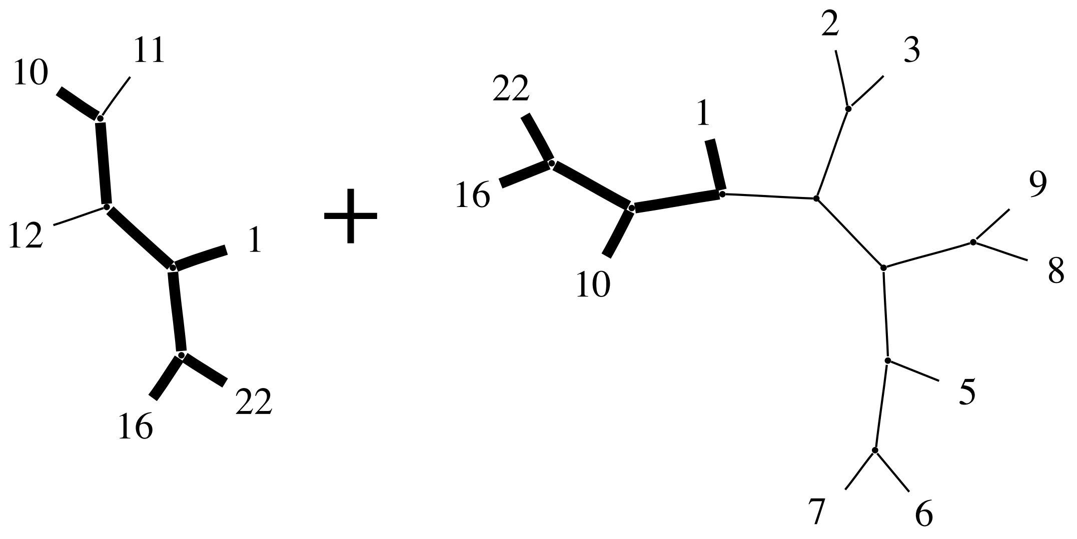
DCM is a “meta-method” that can be used with almost any type of large-scale tree inference.

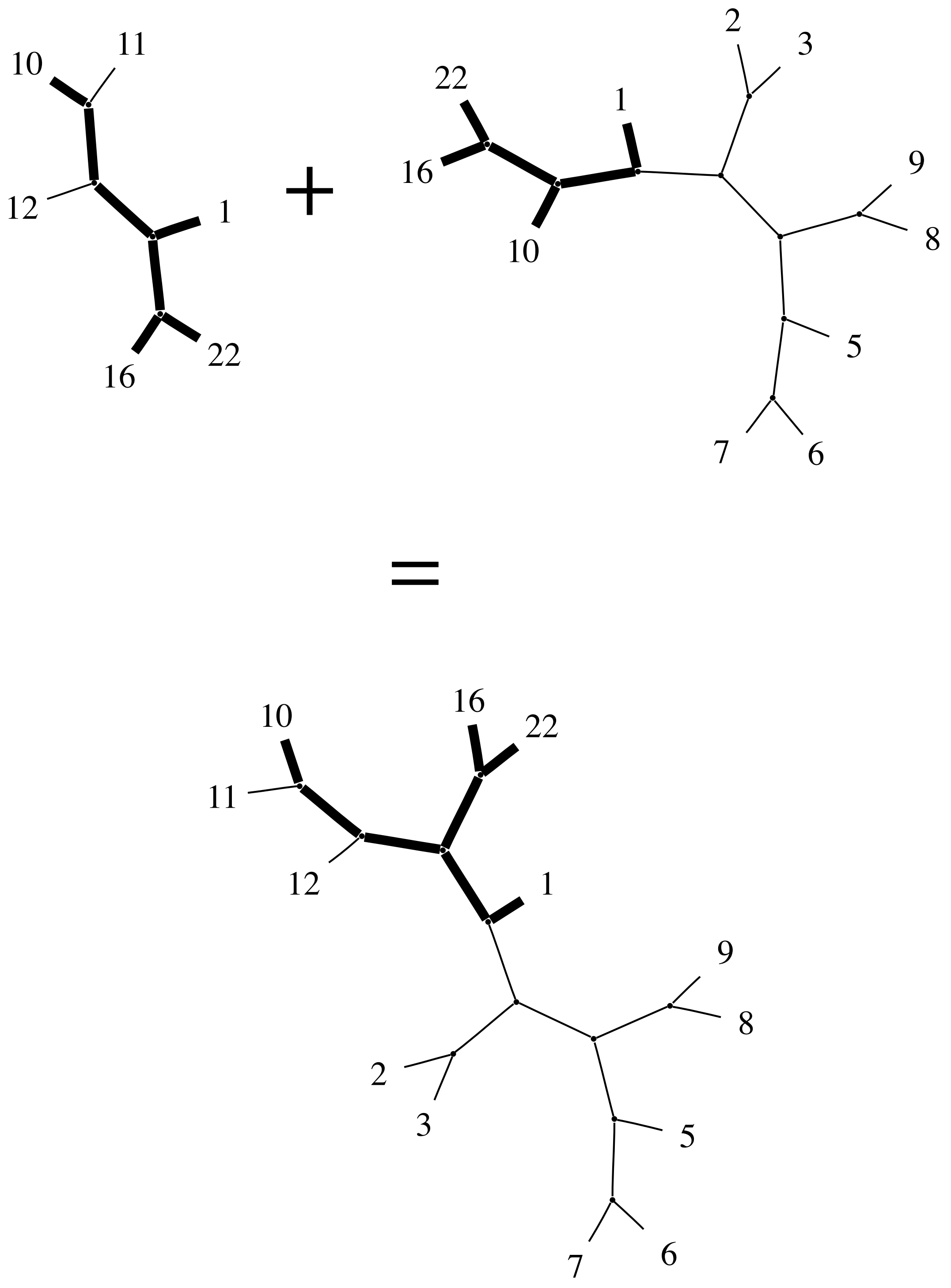
Step 3: Tree Merge (Supertree analysis)

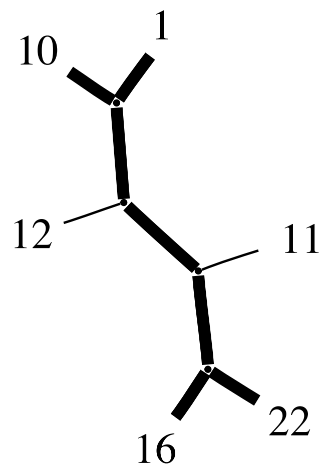
The step of “glueing” the trees for subproblems together is a supertree analysis.

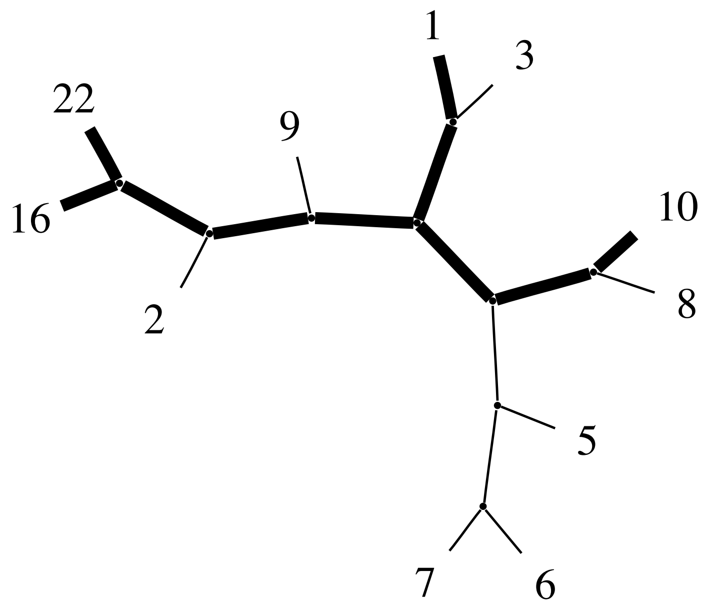
If there is no conflict between the input trees, the problem is trivial.

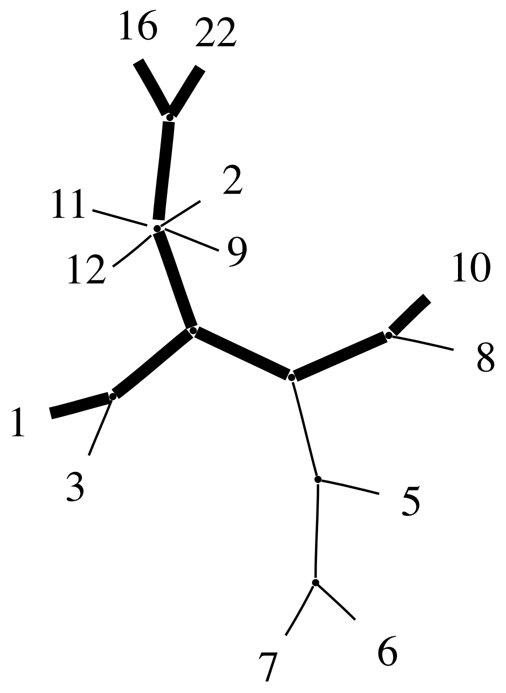
Roshan et recommend using a Strict Consensus Merger - collapse the minimal number of edges required to make 2 trees display the same tree (for the leaves that they have in common).


10
11
12
1
16
22
+
22
1
16
10
2
3
9
8
5
7
6


10
11
12
1
16
22
+
22
16
10
1
2
3
9
8
5
7
6
=
10
11
12
16
22
1
9
8
2
3
5
7
6


10
1
12
11
16
22


1
3
22
9
16
10
2
8
5
7
6

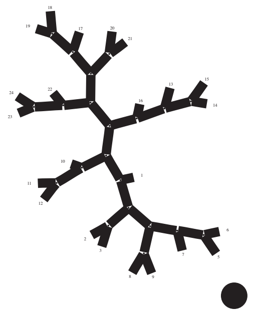

16
22
2
11
9
12
10
8
1
3
5
7
6

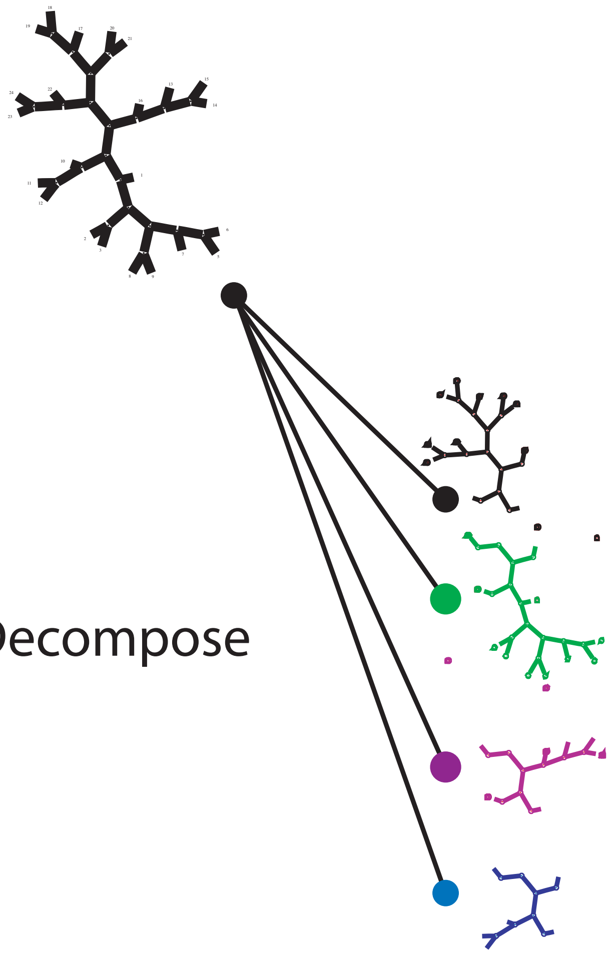
Step 4: Tree Refine

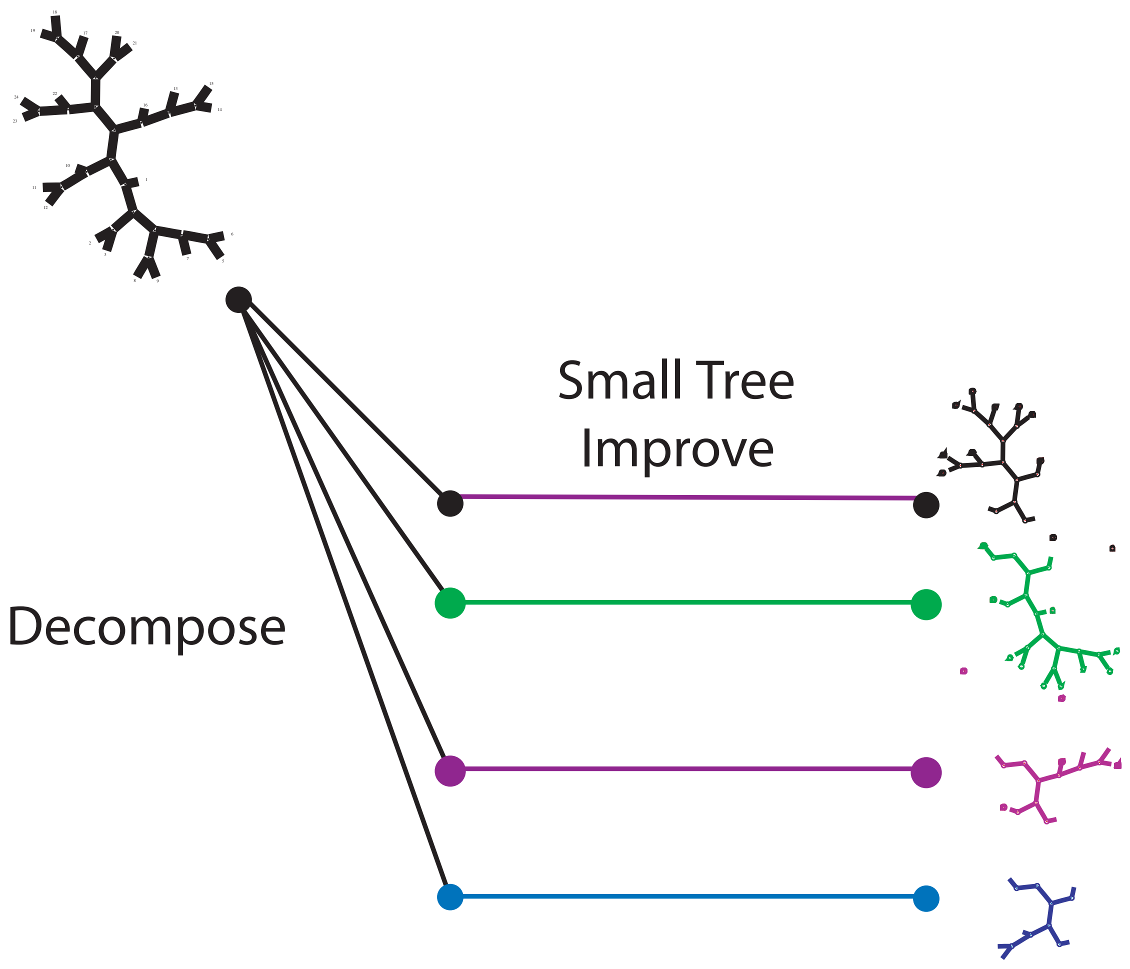
Optional step - some tree searching methods require binary trees

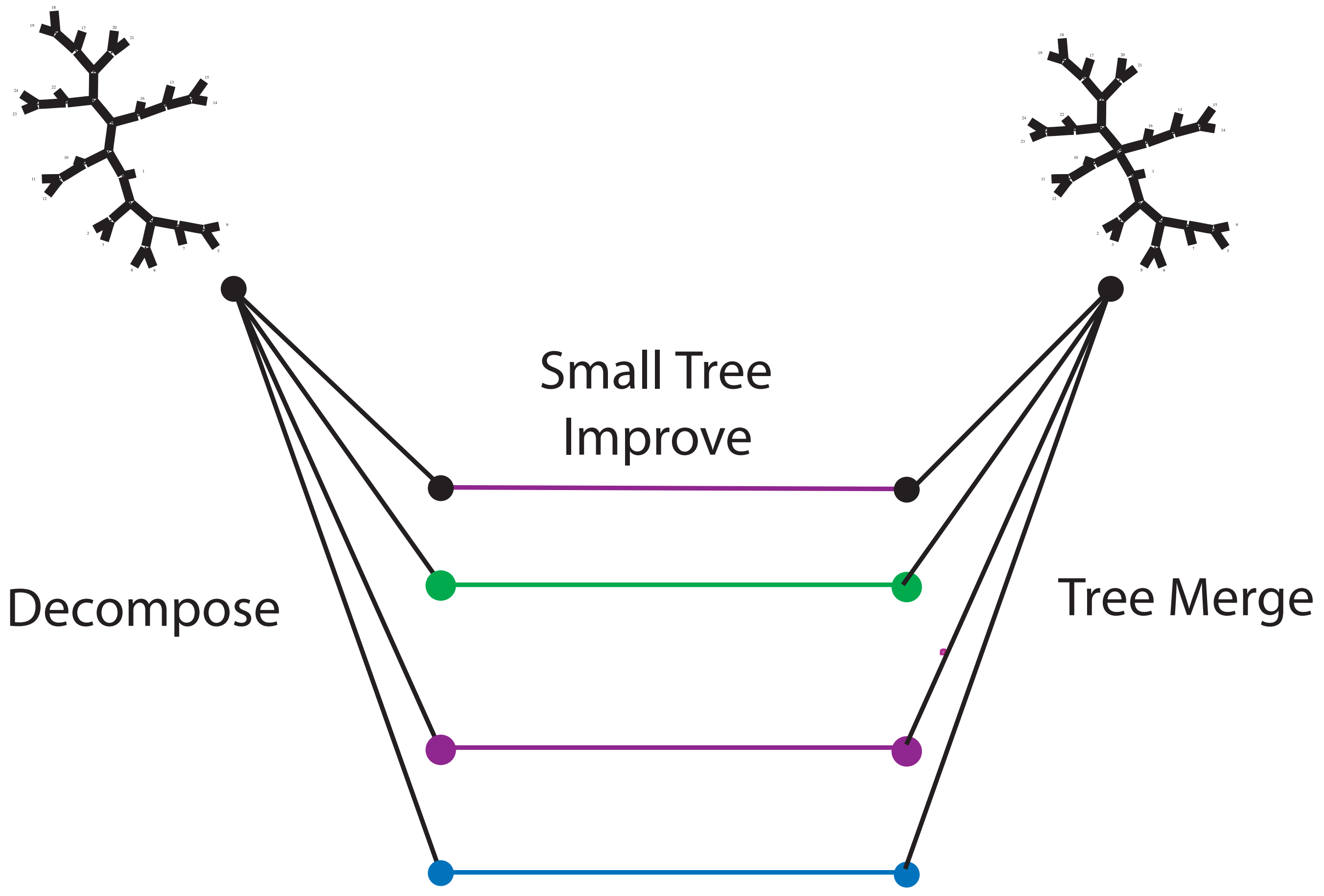
Step 5: Tree Improve

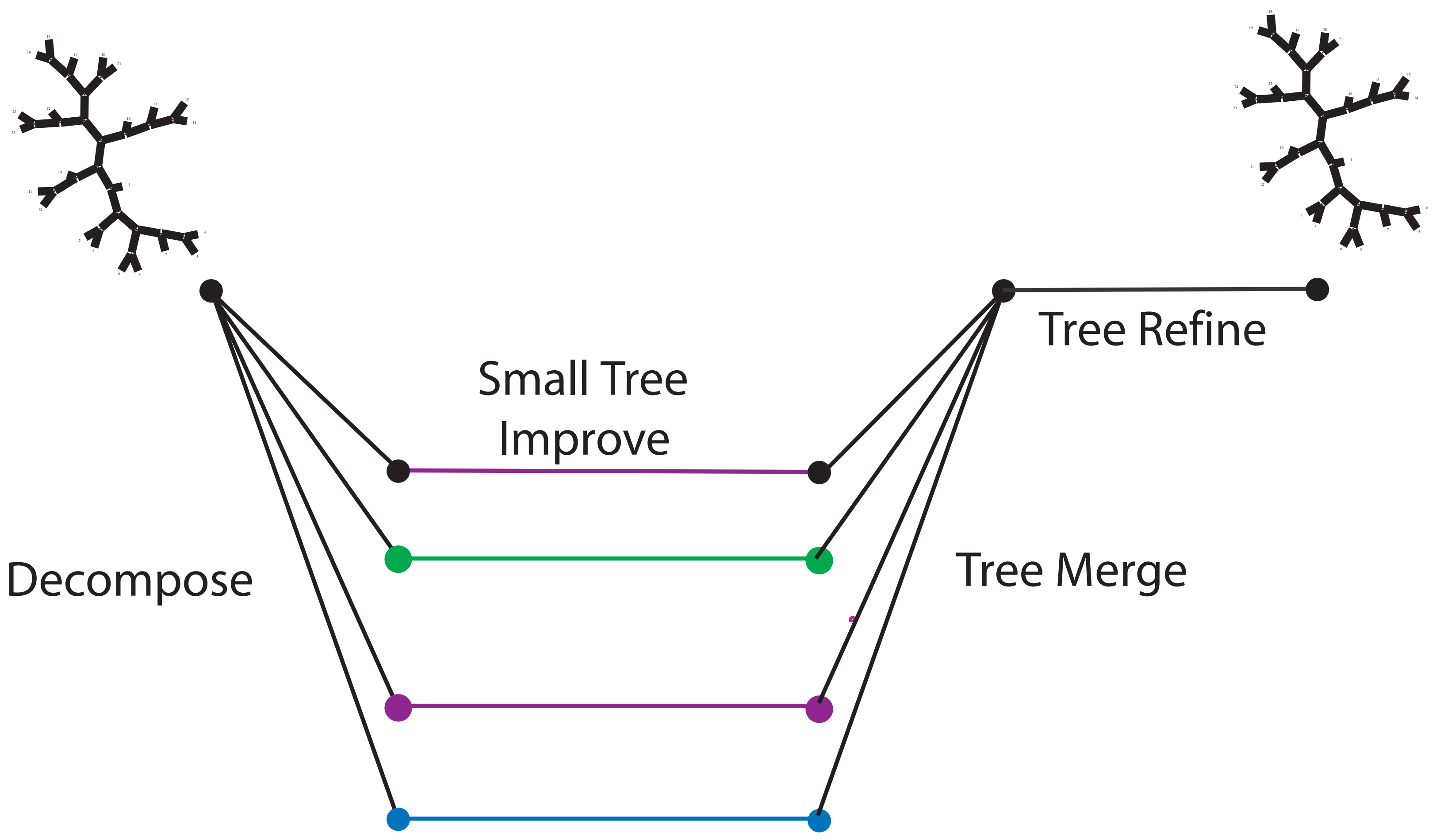
Another “base method” tree search (but with a large set of taxa, so the seach often has to be less thorough)

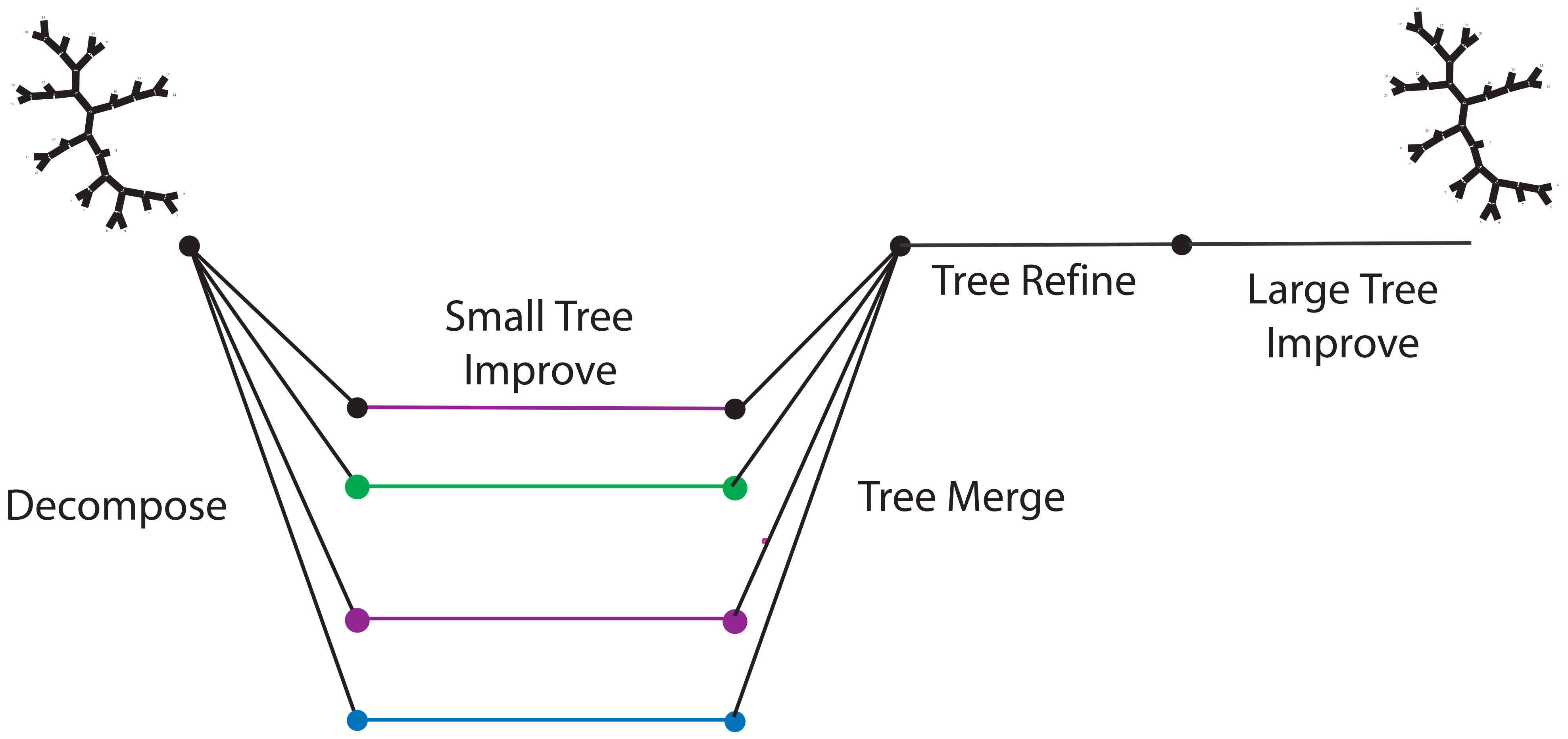

18
19
17
20
21
24
22
15
13
16
14
23
10
1
11
12
6
2
3
7
5
8
9


Decompose


Small Tree
Improve
Decompose


Small Tree
Improve
Decompose
Tree Merge


Small Tree
Improve
Tree Refine
Decompose
Tree Merge


Small Tree
Improve
Tree Refine
Large Tree
Improve
Decompose
Tree Merge

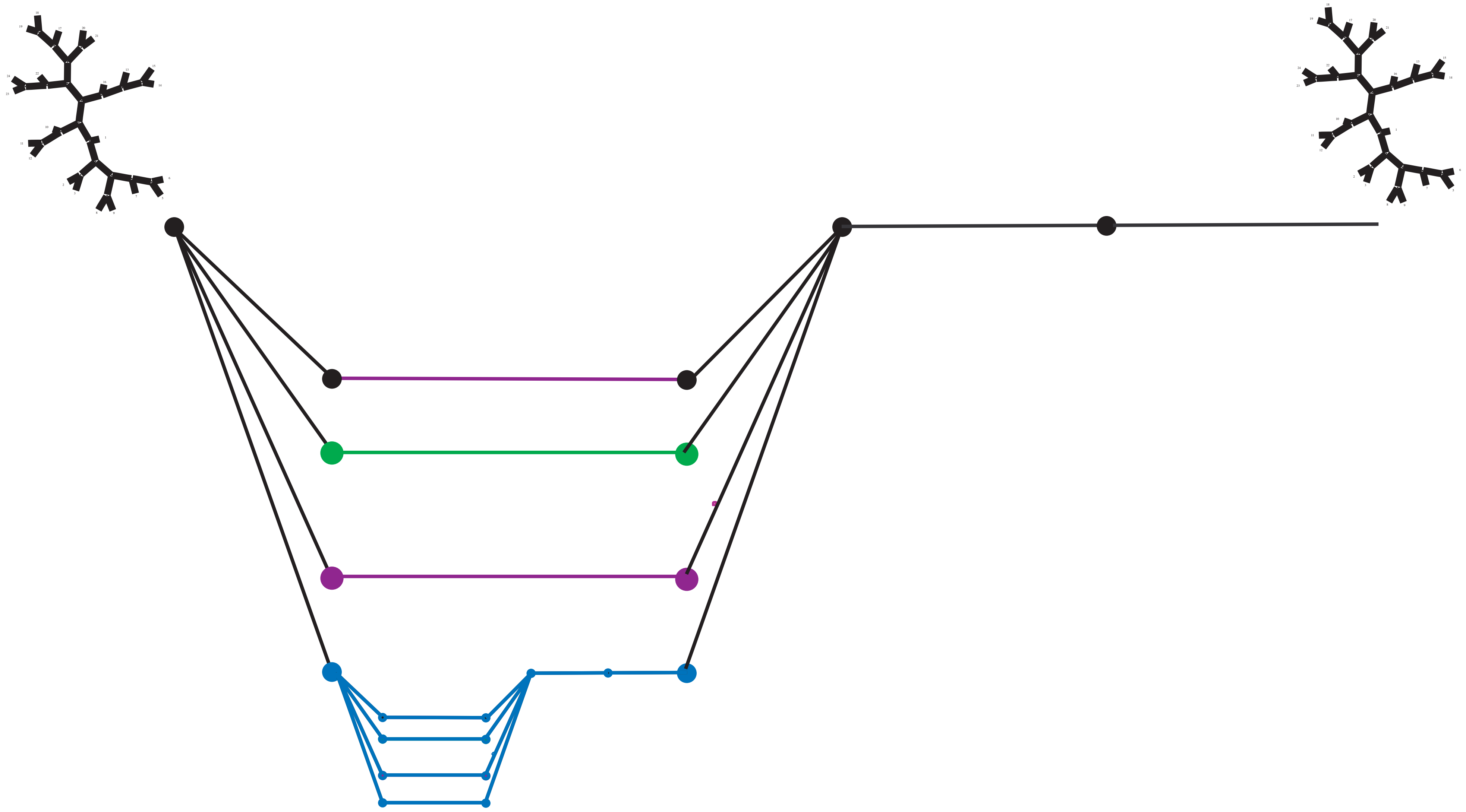
Recursion

A *recursive* algorithm is one that calls (invokes) itself.

A definition of the function to compute the factorial is the classic example:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

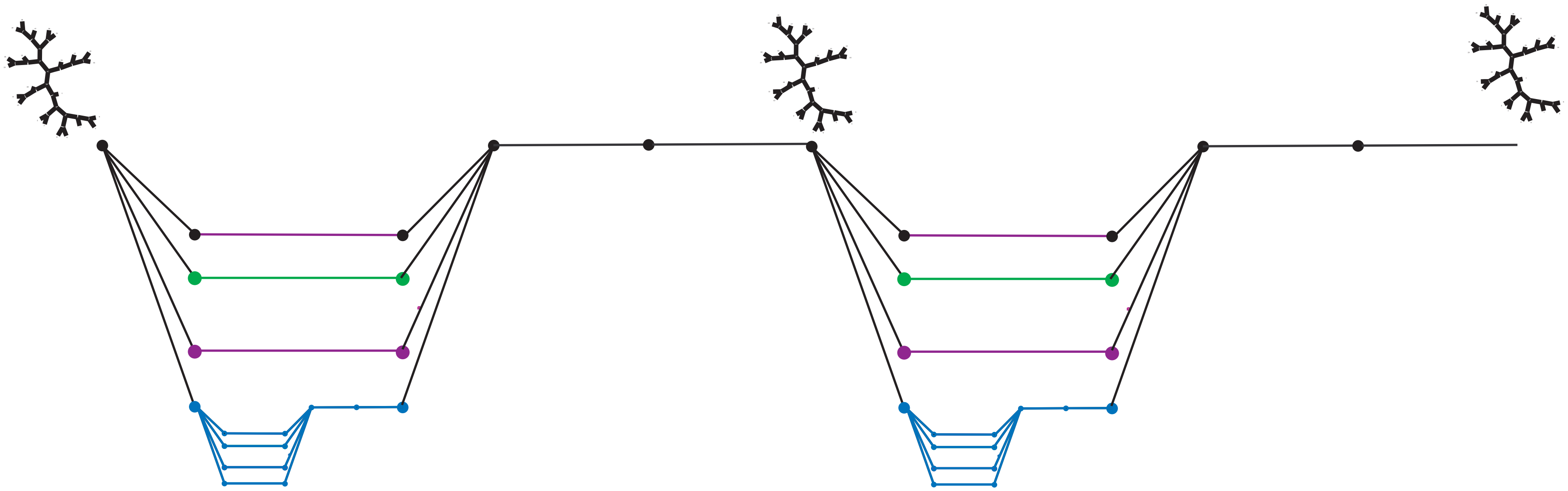
Recursion is often used when it is easy to perform a few tasks, but then you are faced with the same problem you originally faced, but on a smaller scale.
Recursive DCM3 arises from the recognition that, when we break our full set of taxa into subsets some of them may *still* be too large for thorough searching. We can use another level of DCM to break them down into smaller problems.



Iteration

Because the decompositions are sensitive to the starting tree, we may do a better job decomposing the tree into closely related subtrees if we have a better estimate of the tree.

So we can simply repeat the whole recursive DCM process



References

Maddison, D. (1991). The discovery and importance of multiple islands of most-parsimonious trees. *Systematic Zoology*, 40(3):315–328.