ML phylogenetic inference and GARLI

Derrick Zwickl

University of Arizona

(and University of Kansas)

Workshop on Molecular Evolution 2015

Outline

- Heuristics and tree searches
- ML phylogeny inference and GARLI
- Using GARLI in practice
- Computer exercises

Heuristics

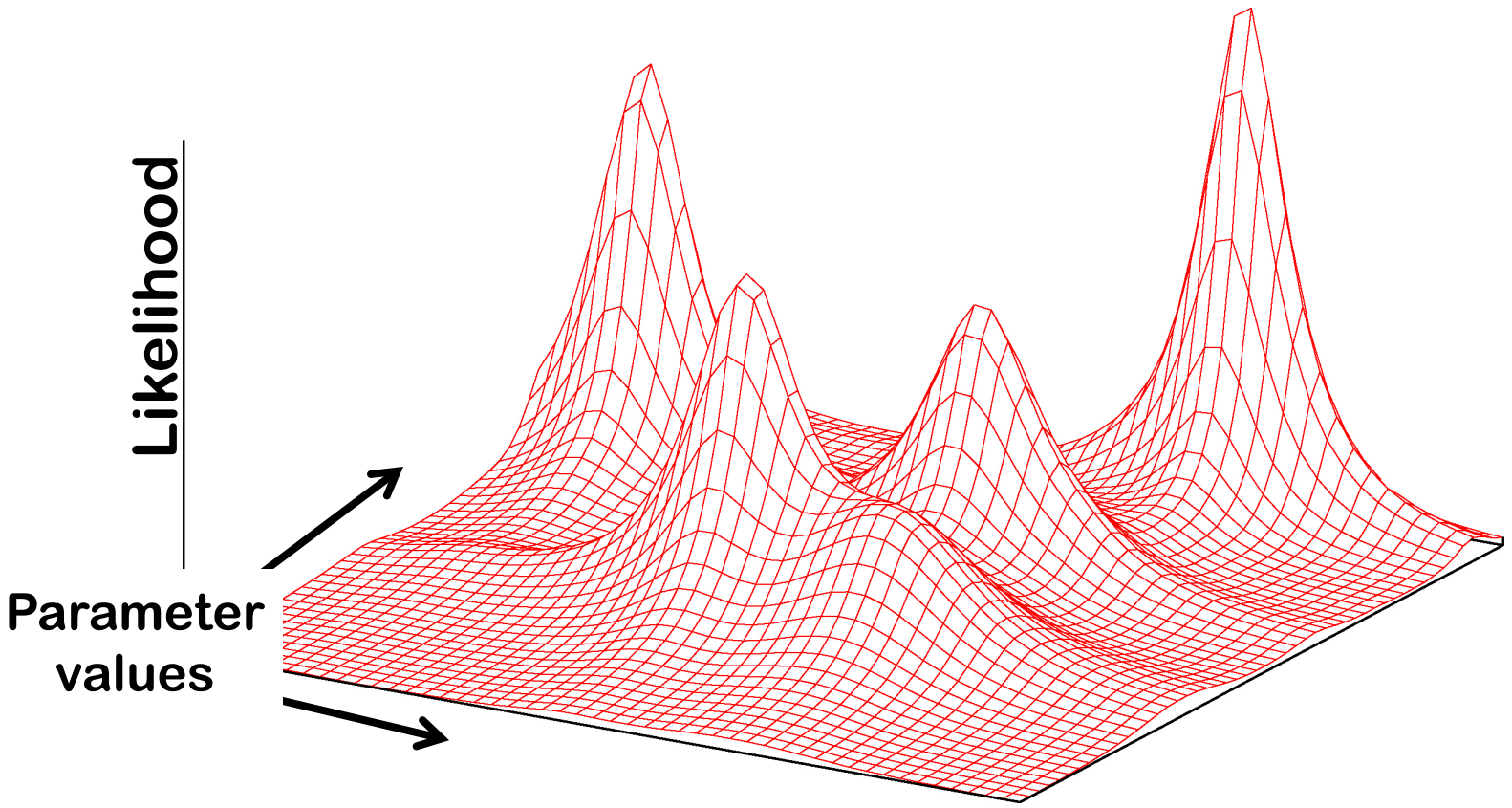
Aim to optimize a function or solve some problem

Finding a good solution is not guaranteed

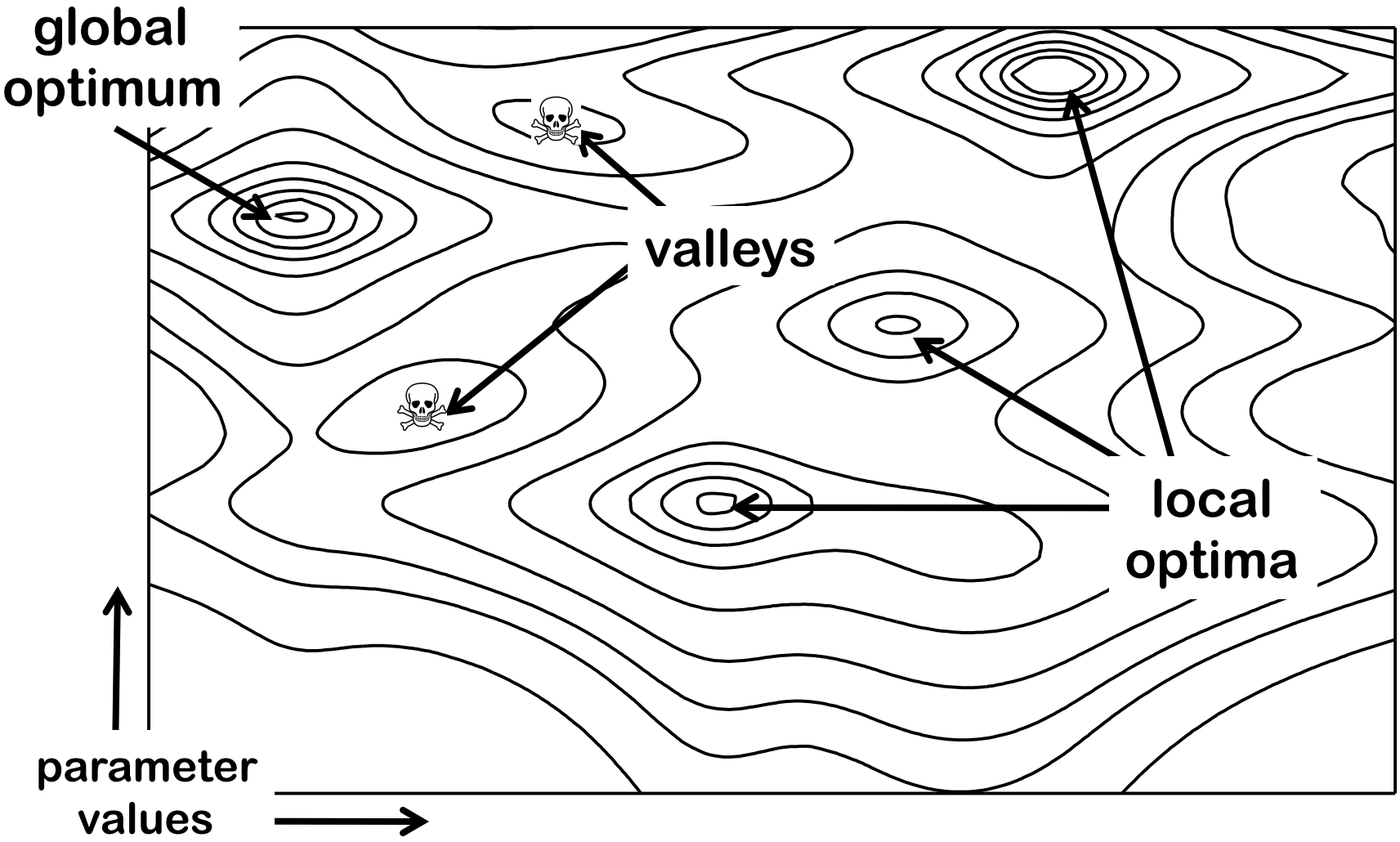
Deterministic and/or stochastic

Many types (greedy hill climbing, GA, simulated annealing, etc.)

A likelihood surface



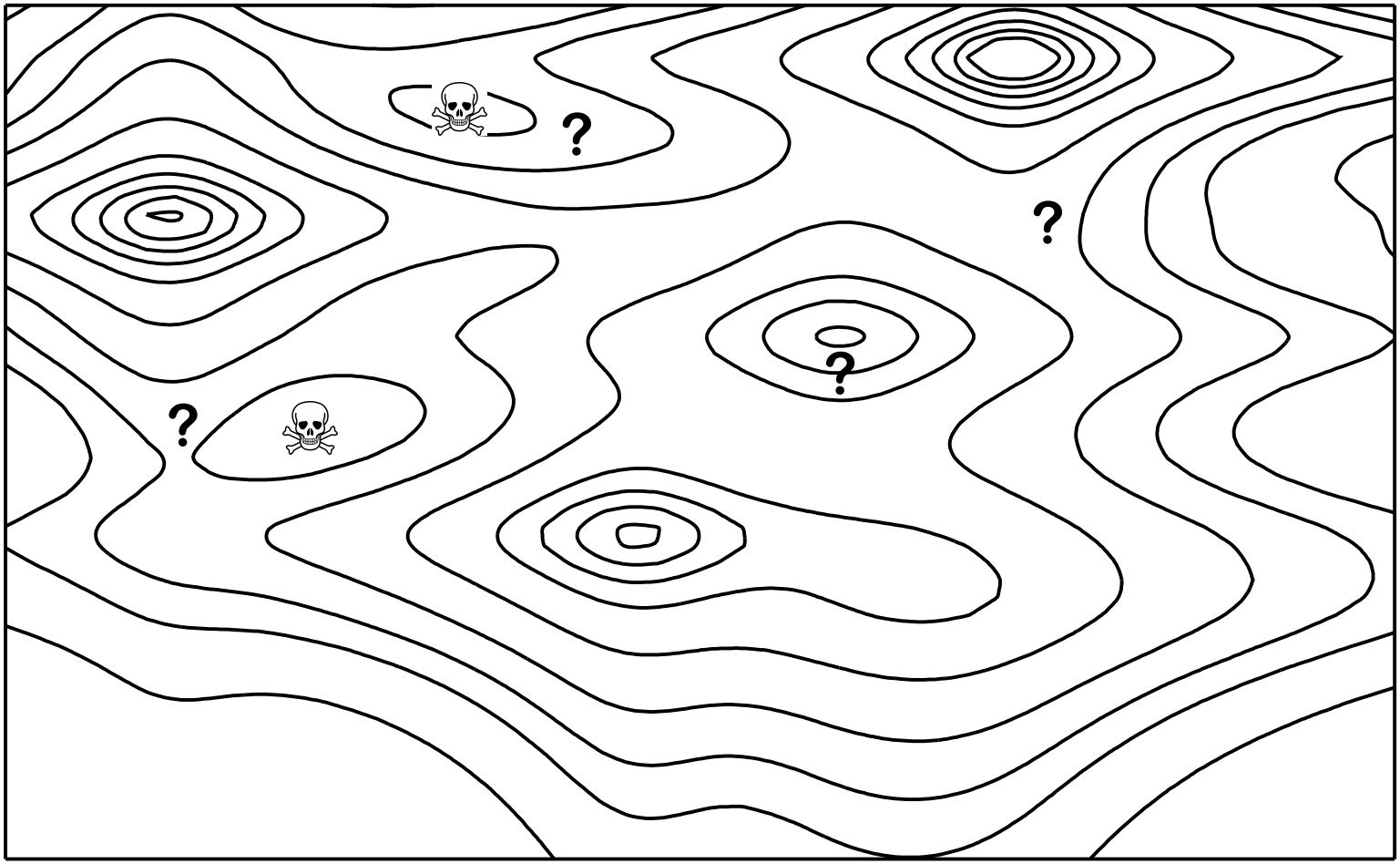
A likelihood surface (from above)



Heuristic features

Where does it start?

Heuristics: starting point

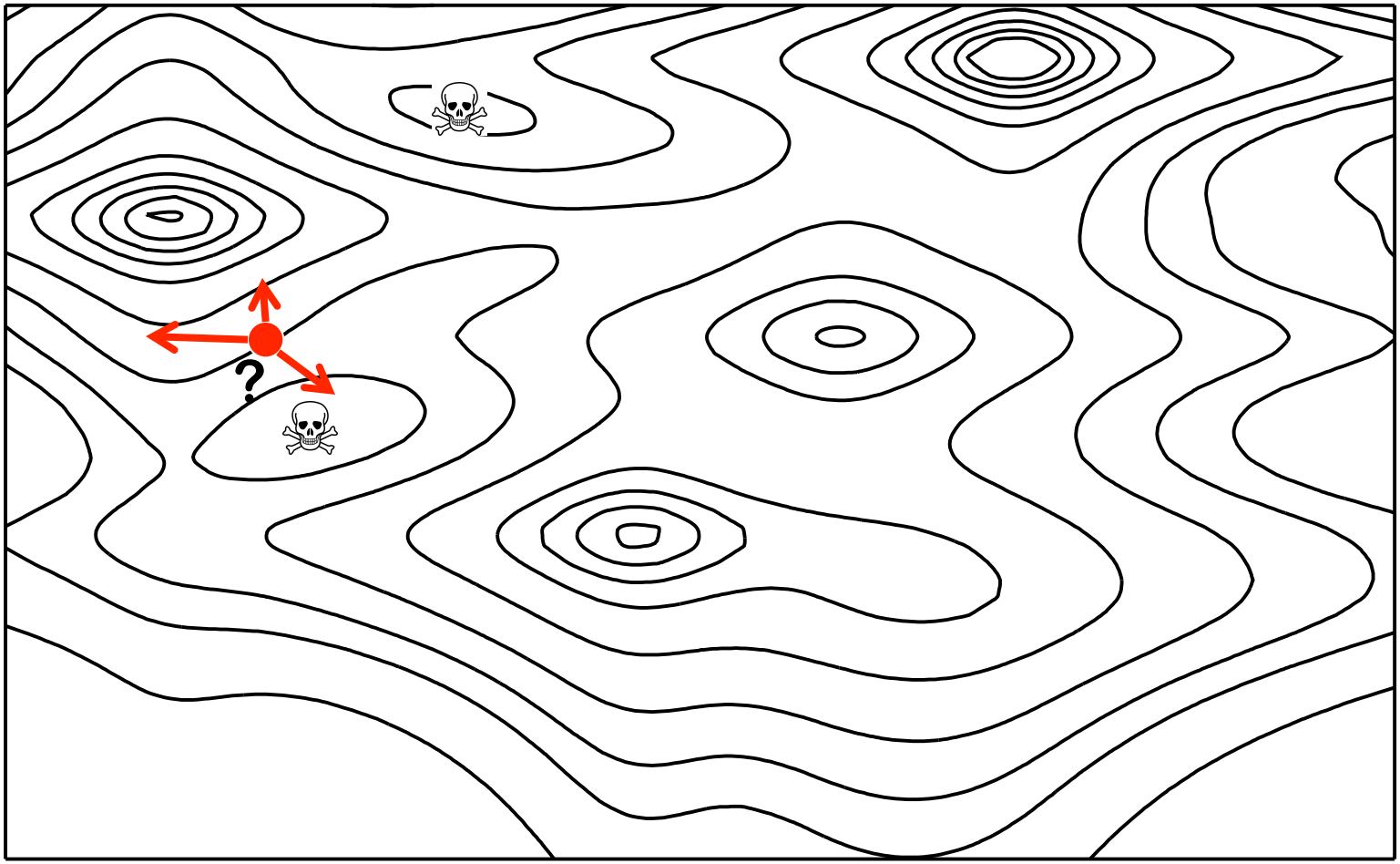


Heuristic features

Where does it start?

How are new values proposed?

Heuristics: proposing new values



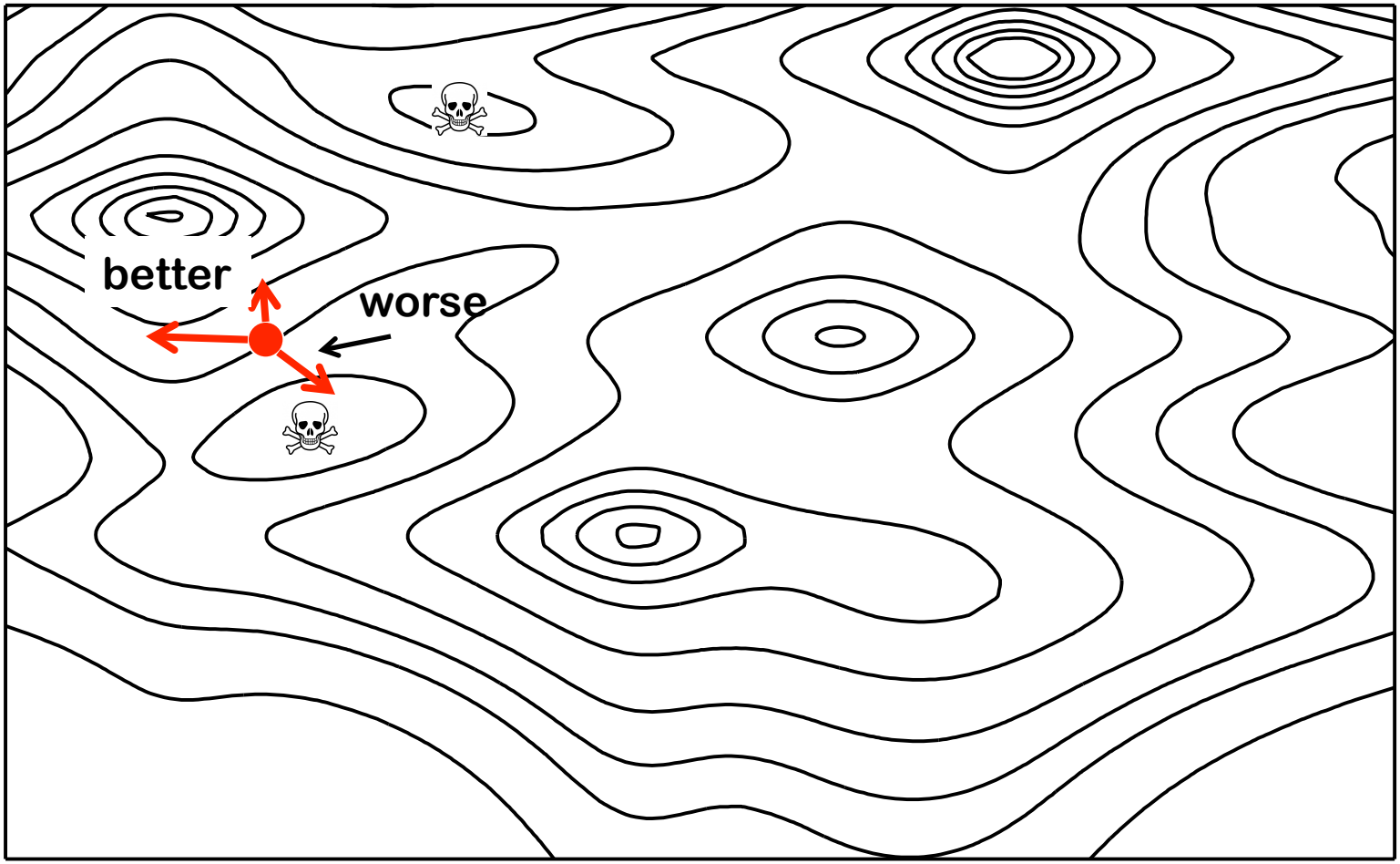
Heuristic features

Where does it start?

How are new values proposed?

How does it move between proposed values?

Heuristics: choosing a new value



Heuristics

Few restrictions on how a heuristic can work

Best choice likely problem-specific

ML phylogenetic heuristics

Goal: find tree with highest likelihood

Difficulties:

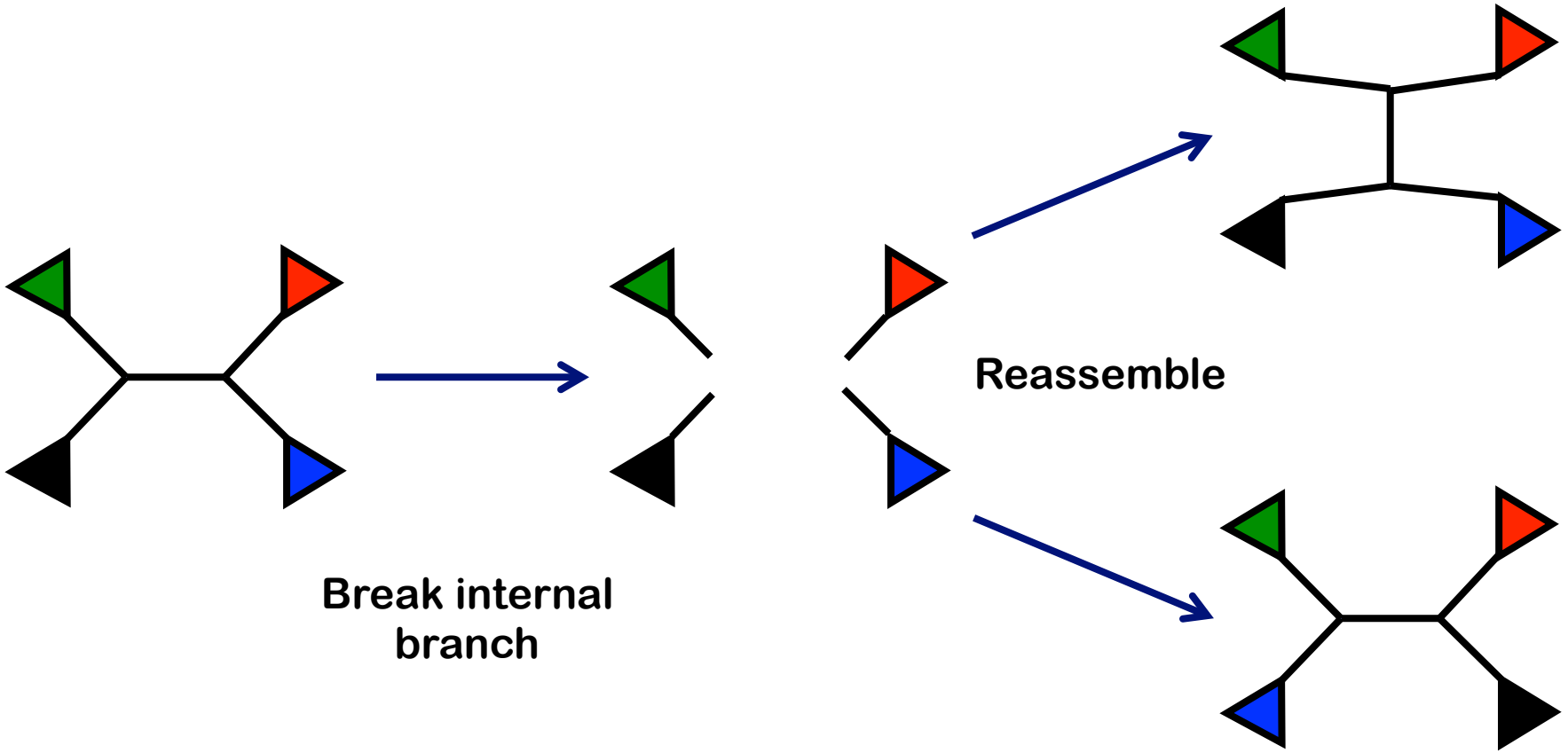
- **Enormous number of trees to consider**
- **Significant computation needed to score each tree (parameter optimization)**
- **Branch-length parameters aren't equivalent on different trees**
- **Optimal parameter values are strongly correlated**

Phylogenetic searches

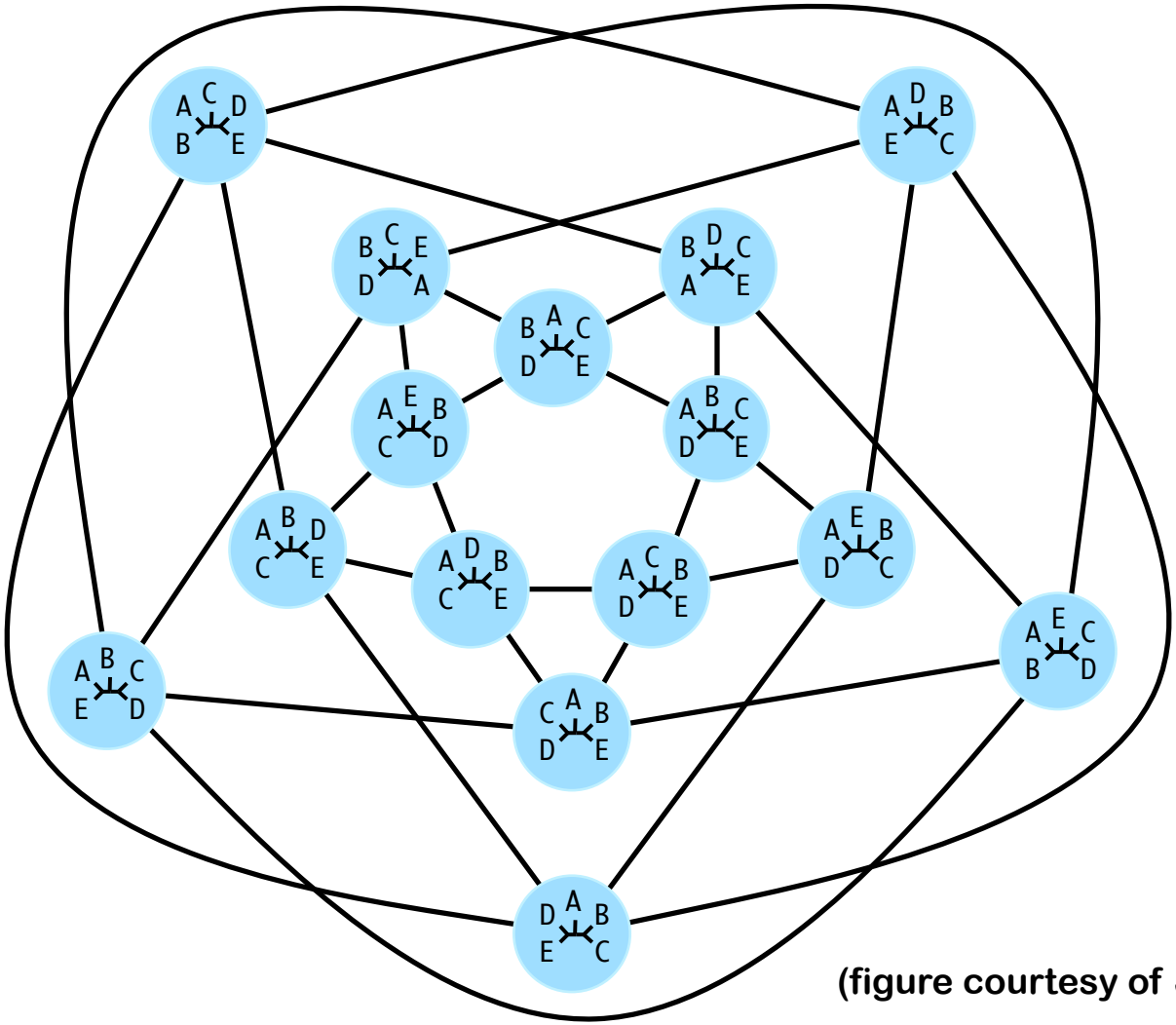
Think about moving through an abstract "treespace"

Nearby points in this treespace are connected by NNI (nearest neighbor interchange) branch swaps

Moving through treespace: NNI branch swaps

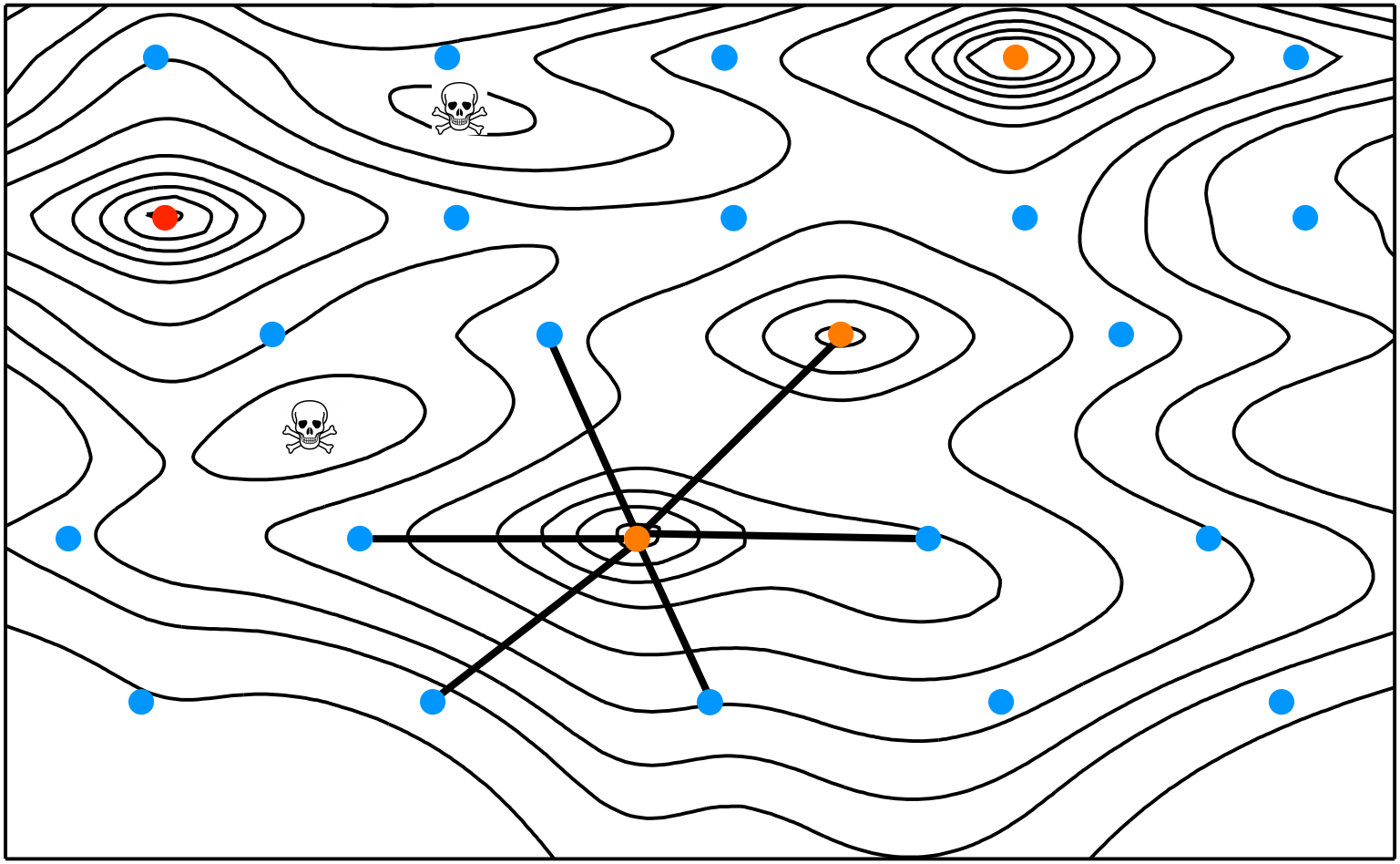


Schoenberg graph – edges connect NNI neighbors

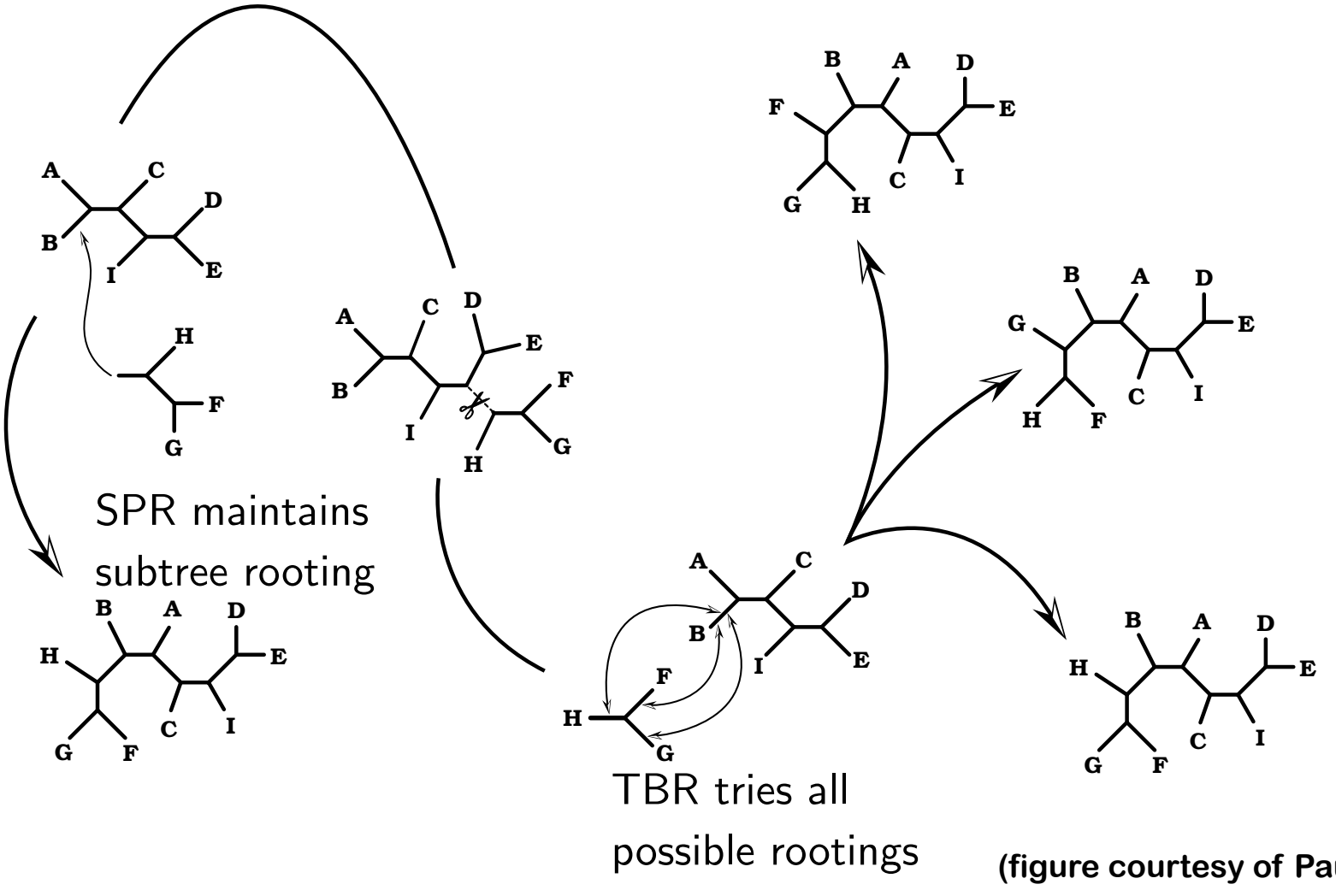


(figure courtesy of Joe Felsenstein)

NNI Treespace

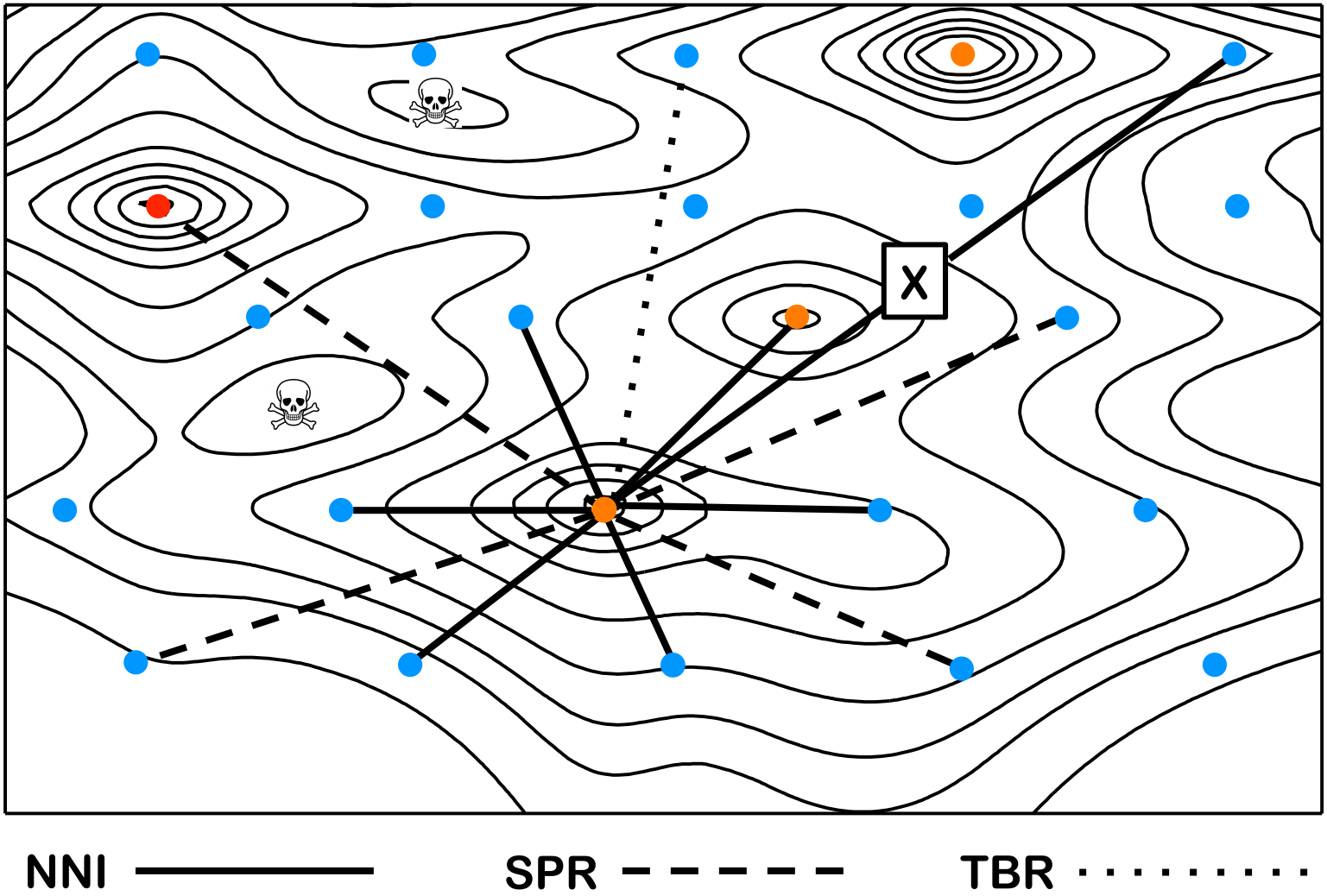


Subtree Pruning Regrafting (SPR) and Tree Bisection Reconnection (TBR)



(figure courtesy of Paul Lewis)

SPR/TBR moves in NNI treespace



GARLI

Genetic Algorithm for Rapid Likelihood Inference

Descendent of GAML (Lewis, 1998)

Development goal: make ML phylogenetic searches feasible for large datasets

GARLI

Stochastic, genetic algorithm-like approach instead of deterministic hill climbing

Gradually optimizes tree topology, branch lengths and model parameters

Accurate ML tree inference on large datasets (hundreds of sequences) in hours

The Genetic Algorithm

Computational analogue of evolution by natural selection

A few simple requirements:

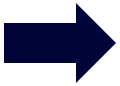
- **Measure of fitness**
- **Method of selection**
- **Mutation operators**
- **Recombination operators**

GA terminology

- Individual
  (topology+model parameter values+branch length values)
- Population
- Fitness (log-likelihood)
- Selection function (fitness proportional)
- Generation

One generation

Create initial population of individuals



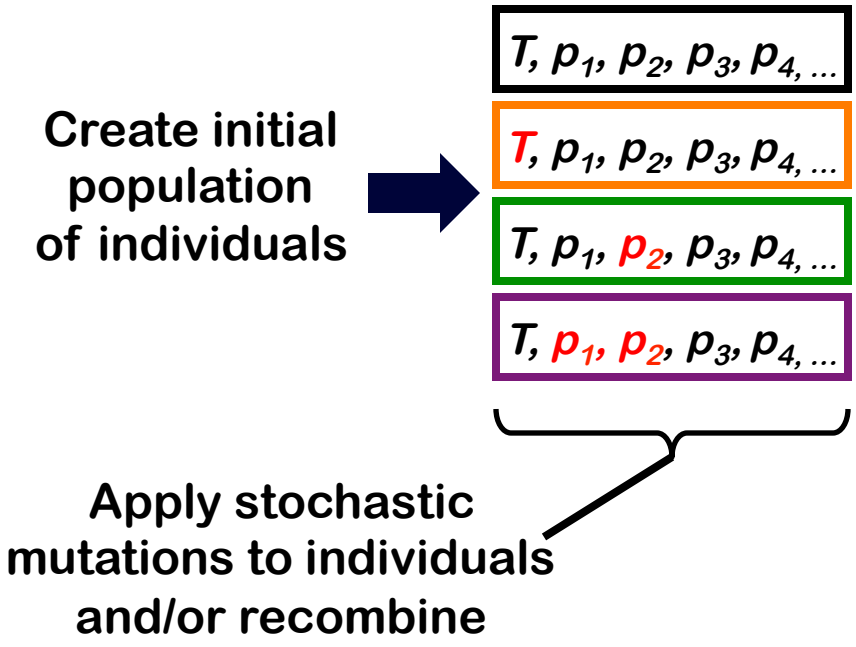
$T, p_1, p_2, p_3, p_4, \ldots$

$T, p_1, p_2, p_3, p_4, \ldots$

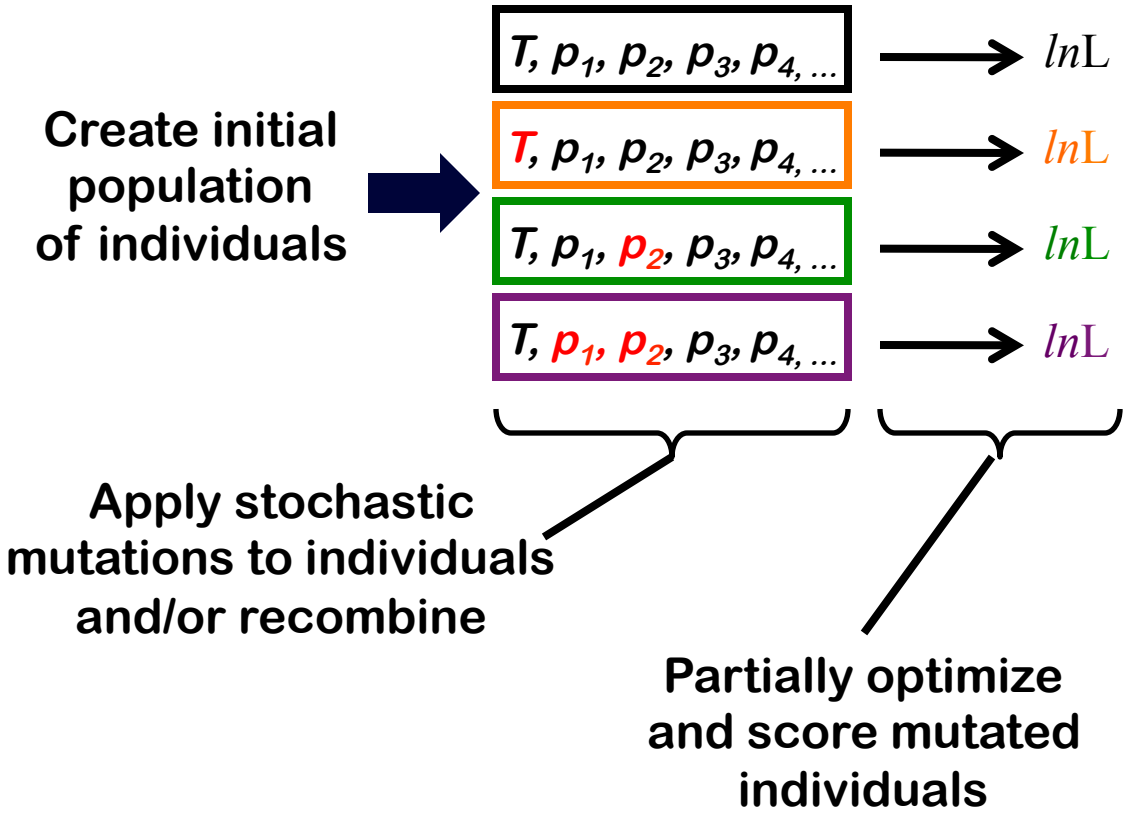
$T, p_1, p_2, p_3, p_4, \ldots$

$T, p_1, p_2, p_3, p_4, \ldots$

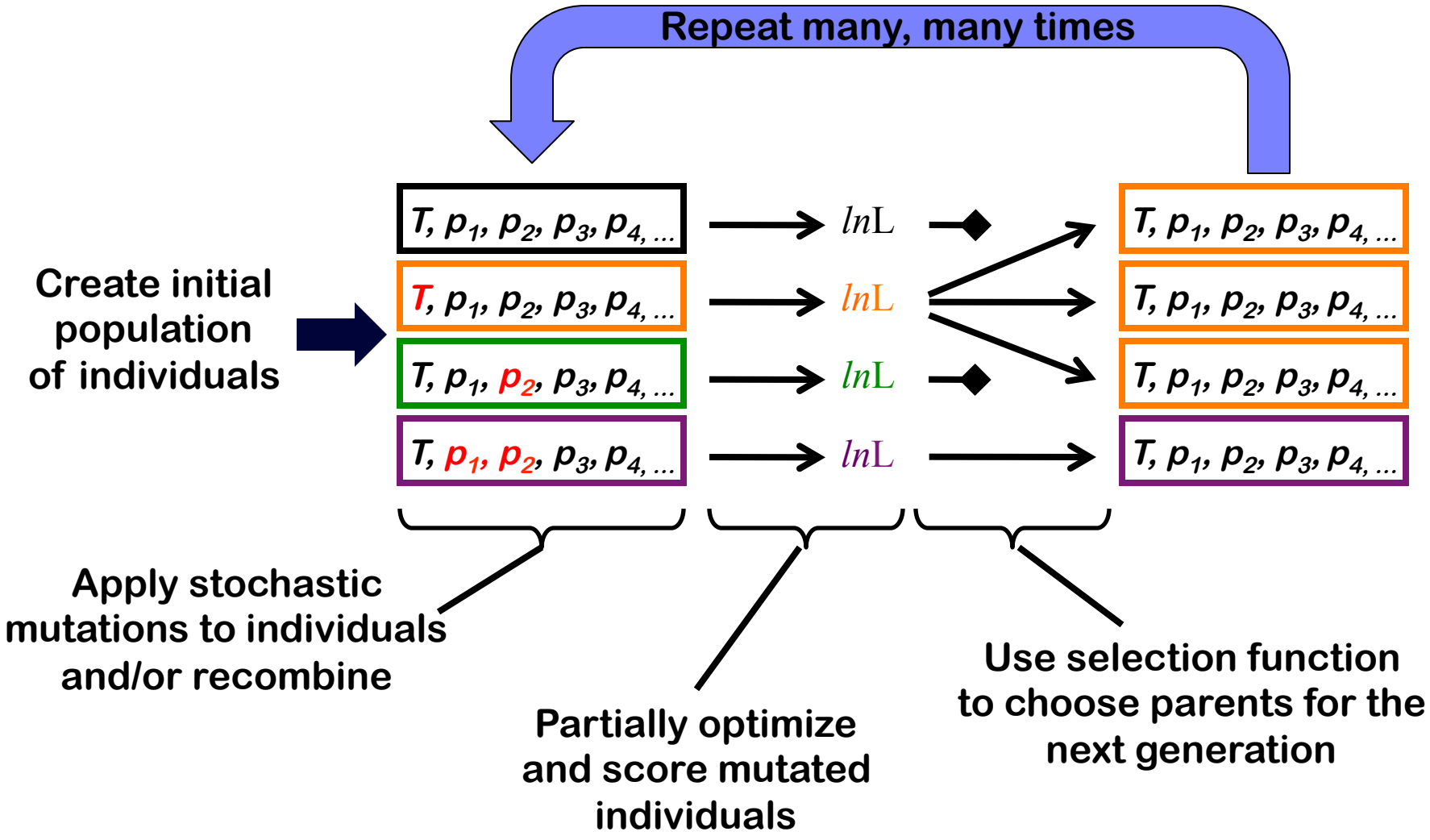
One generation



One generation



One generation



Maximized likelihood: pros

By definition, our goal is to find the topology with the highest maximized likelihood

If we calculate it for every tree we examine, we'll always know which tree is the best

Maximized likelihood: cons

Fully optimizing a single branch length can require significant computation

When one parameter changes, optimal values of all others also change

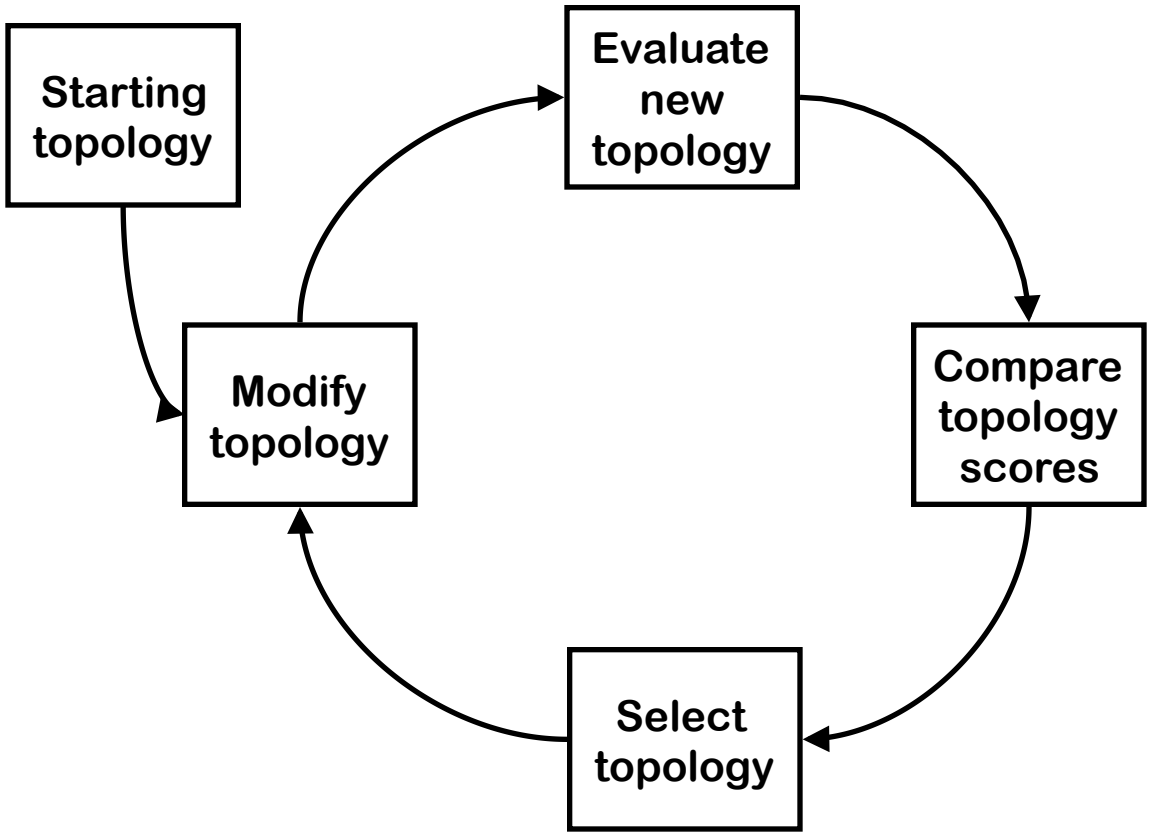
Maximized likelihood: cons

As optimization is applied (and applied, and applied, …), maximized likelihood only approached asymptotically

This is the majority of the computation required in ML inference, and it grows quickly with the number of sequences

Heuristic runtimes

Inference time = # of topologies to evaluate X time to evaluate each



Both are strongly a function of the # of sequences when calculating maximized likelihood

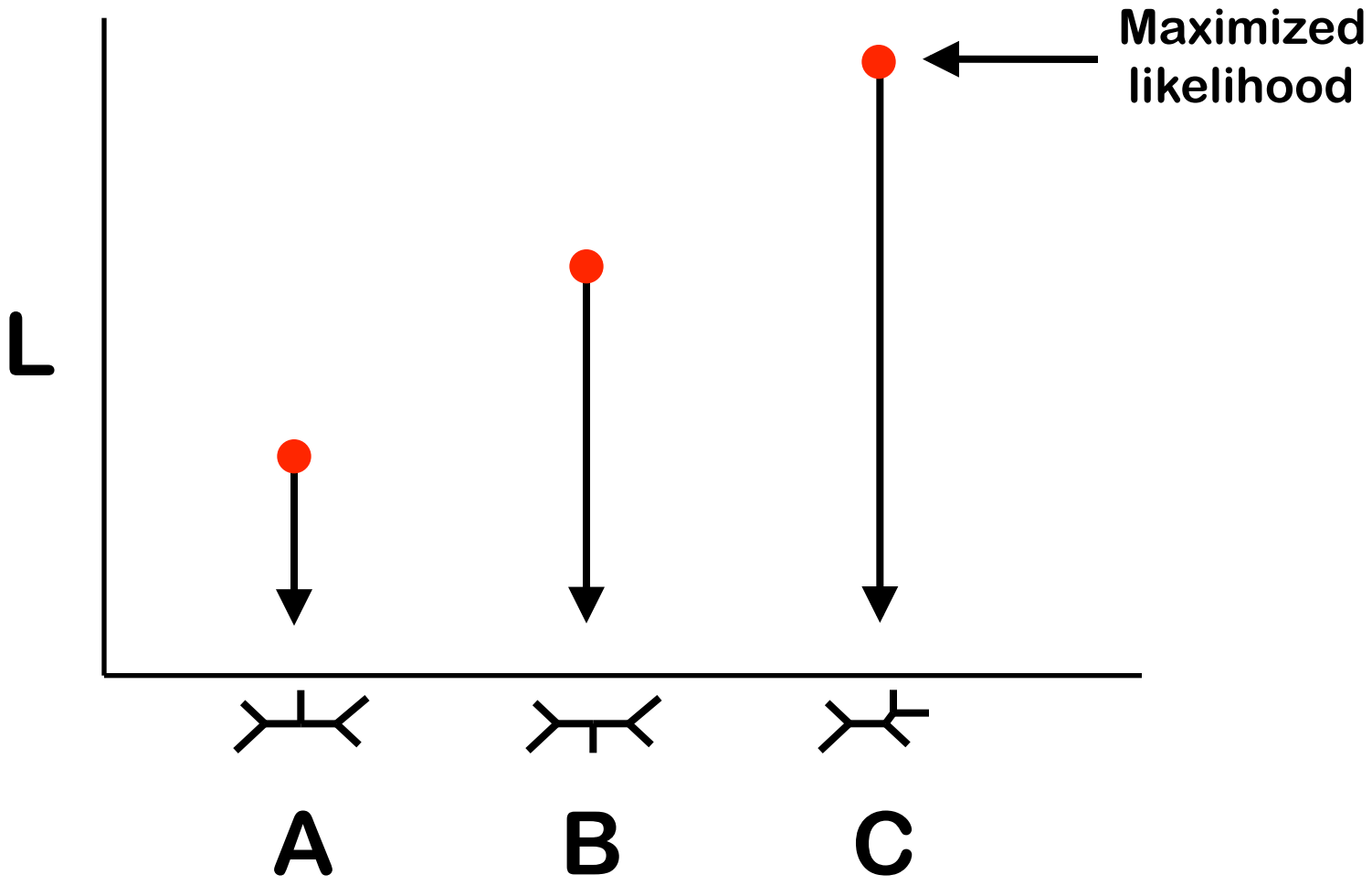
Avoiding the maximized likelihood

We want to accurately judge the merits of topologies, *as if* we had the maximized likelihood

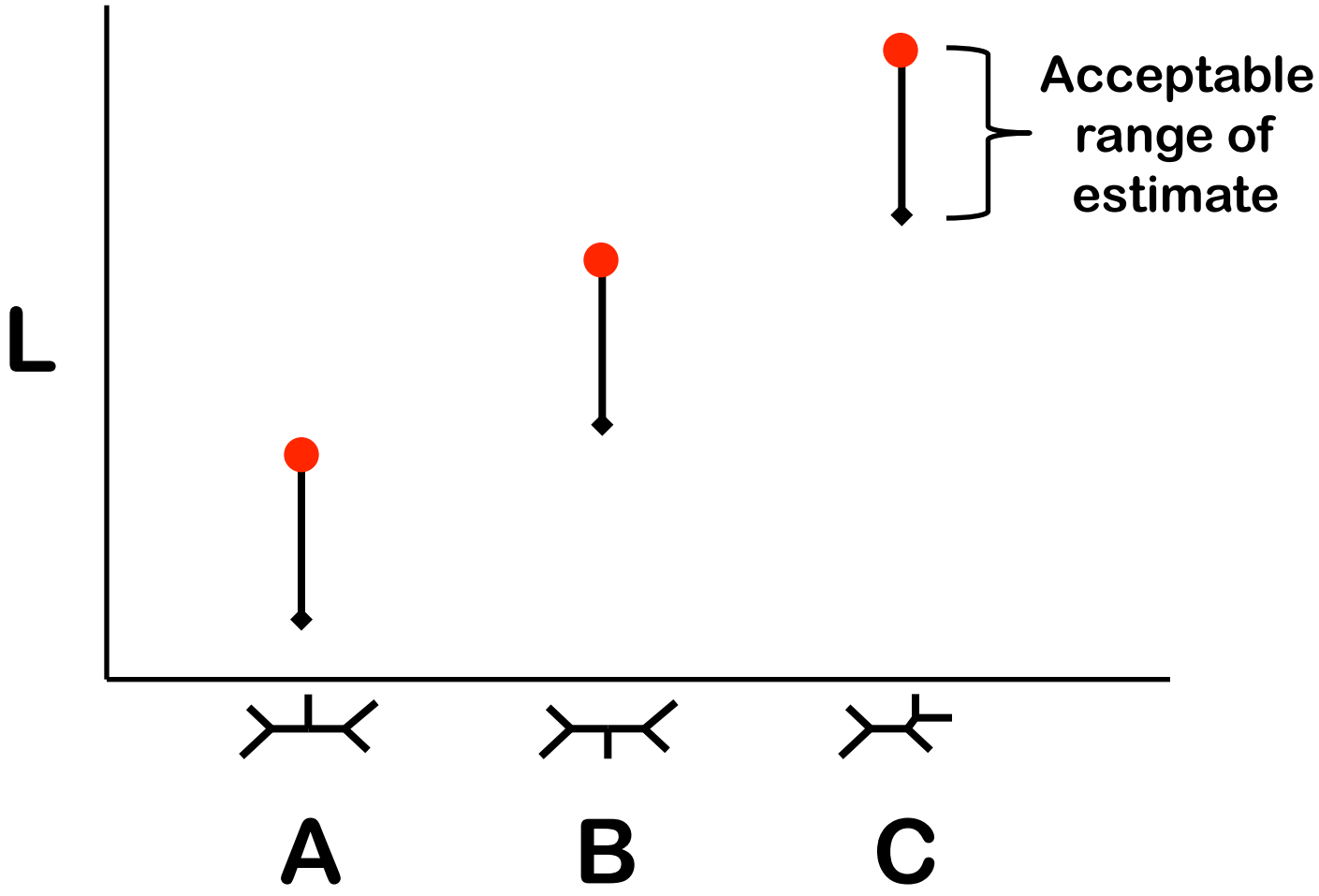
… *but without actually calculating it*

We'll explore the idea of an approximate likelihood score for topologies

How accurate does a tree likelihood estimate need to be?

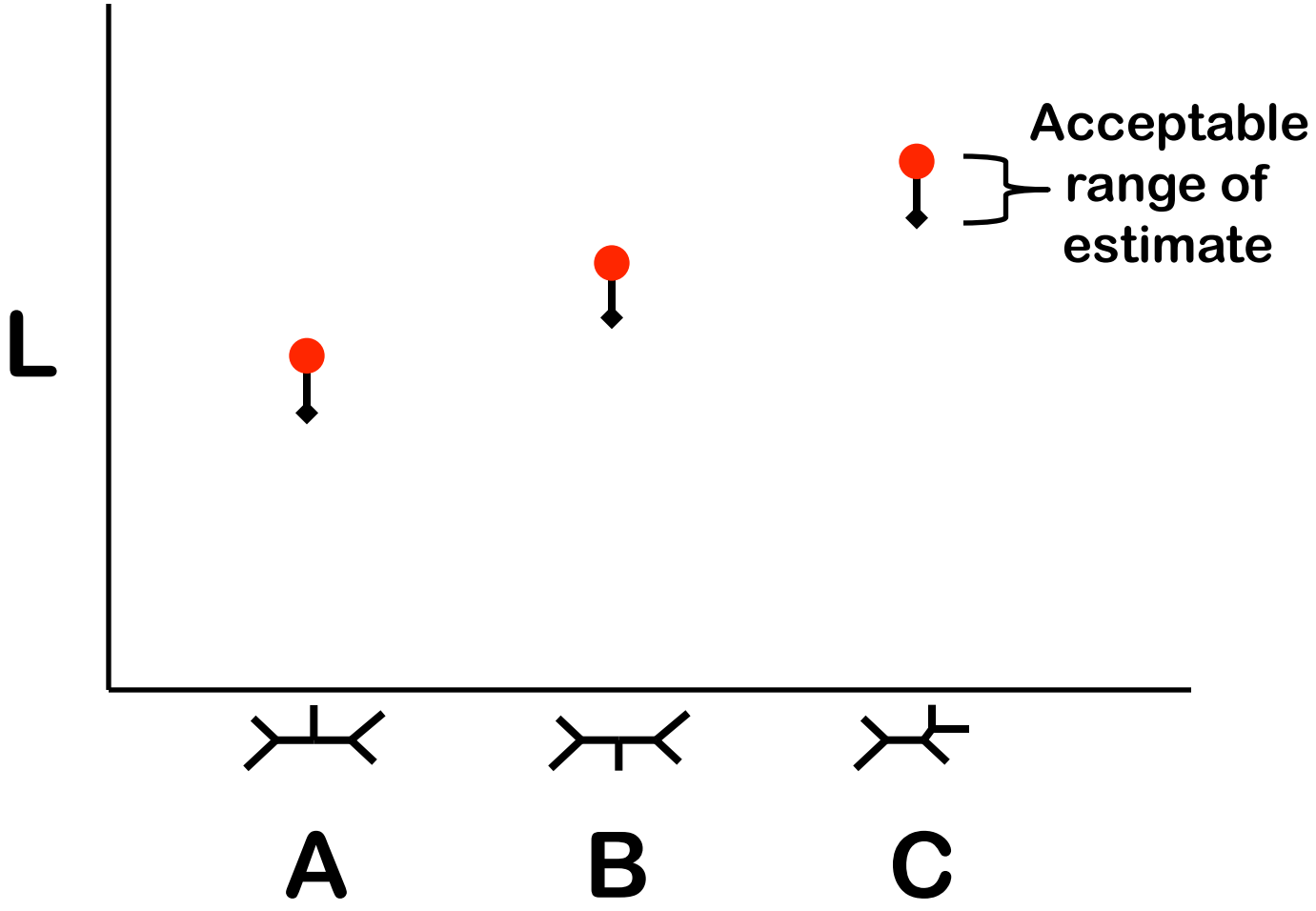


How accurate does a tree likelihood estimate need to be?



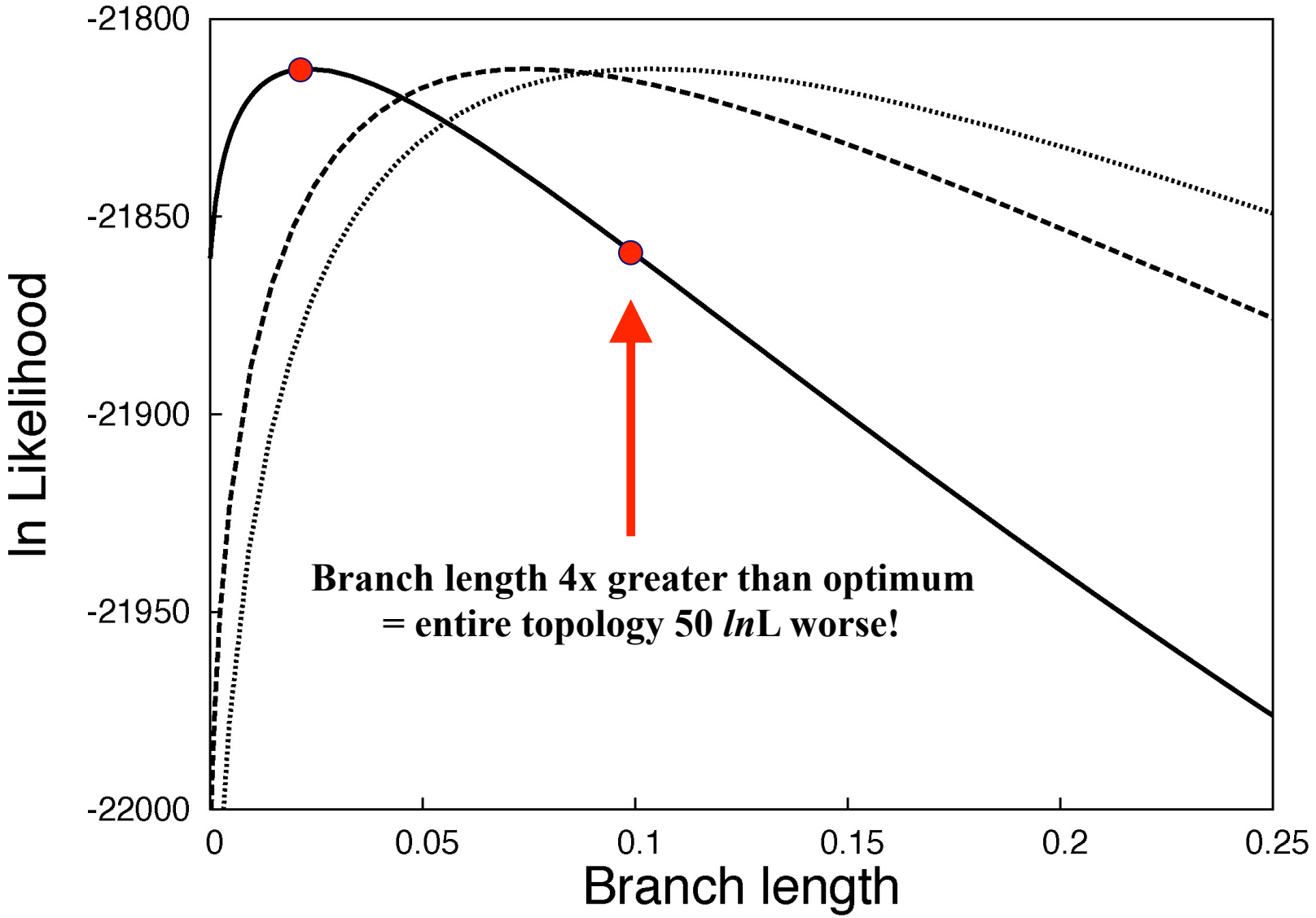
How accurate does a tree likelihood estimate need to be?

(when tree scores are more similar)



How important are branch-length values?

(three example branches in a specific 64-taxon tree)



Branch length importance

If even one branch length is far from optimal, the estimated likelihood will not be useful

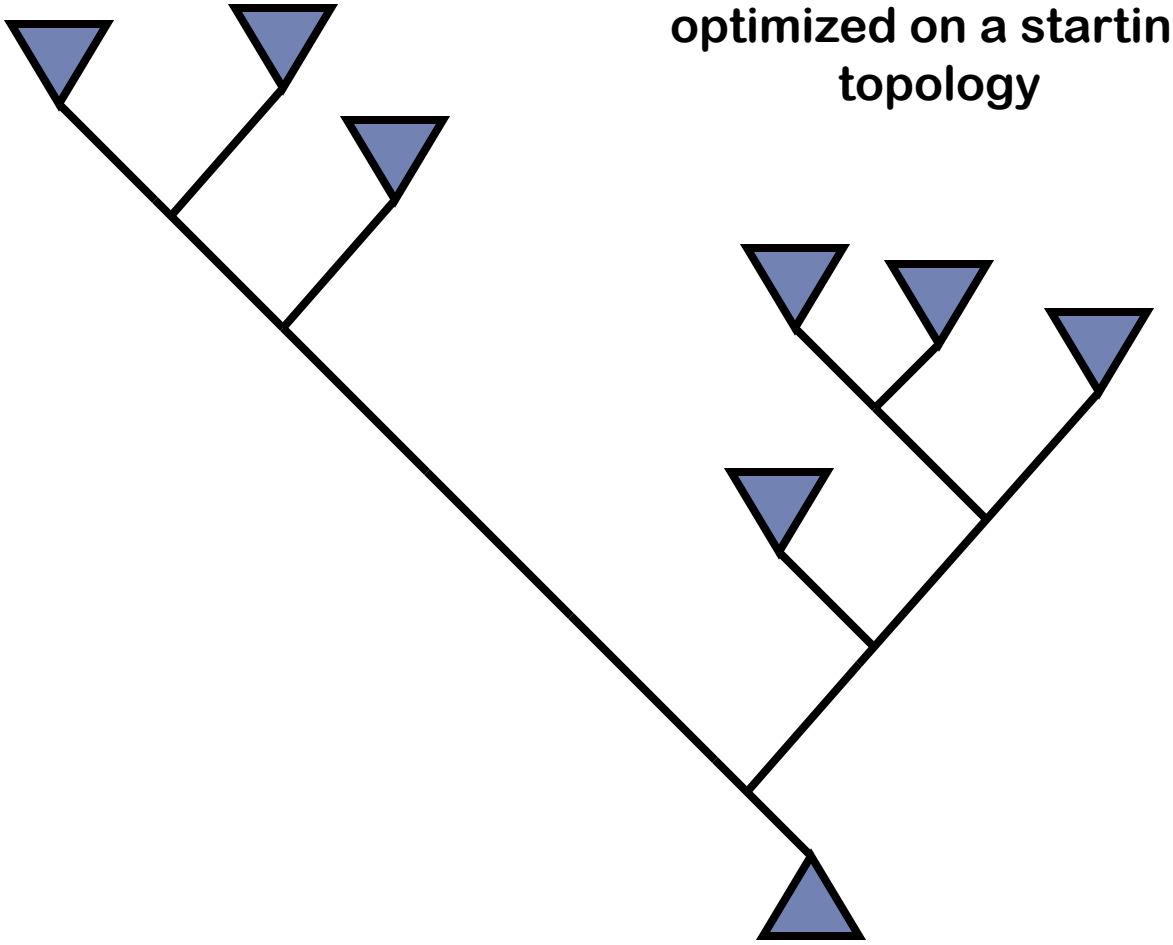
How can we get around optimizing every branch length on every tree?

Using topological similarity

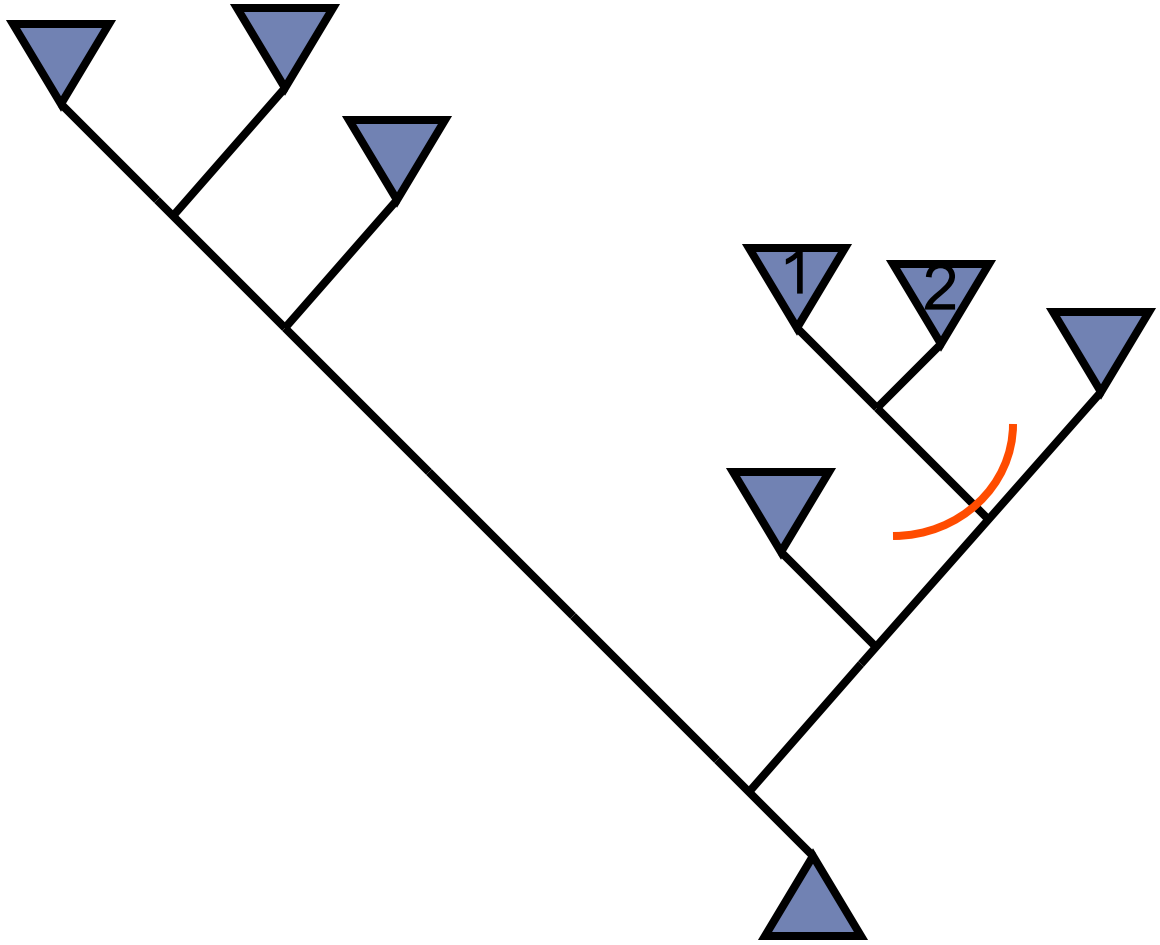
Successive trees are created by slightly modifying an existing tree

We can capitalize on this when dealing with branch-length parameters

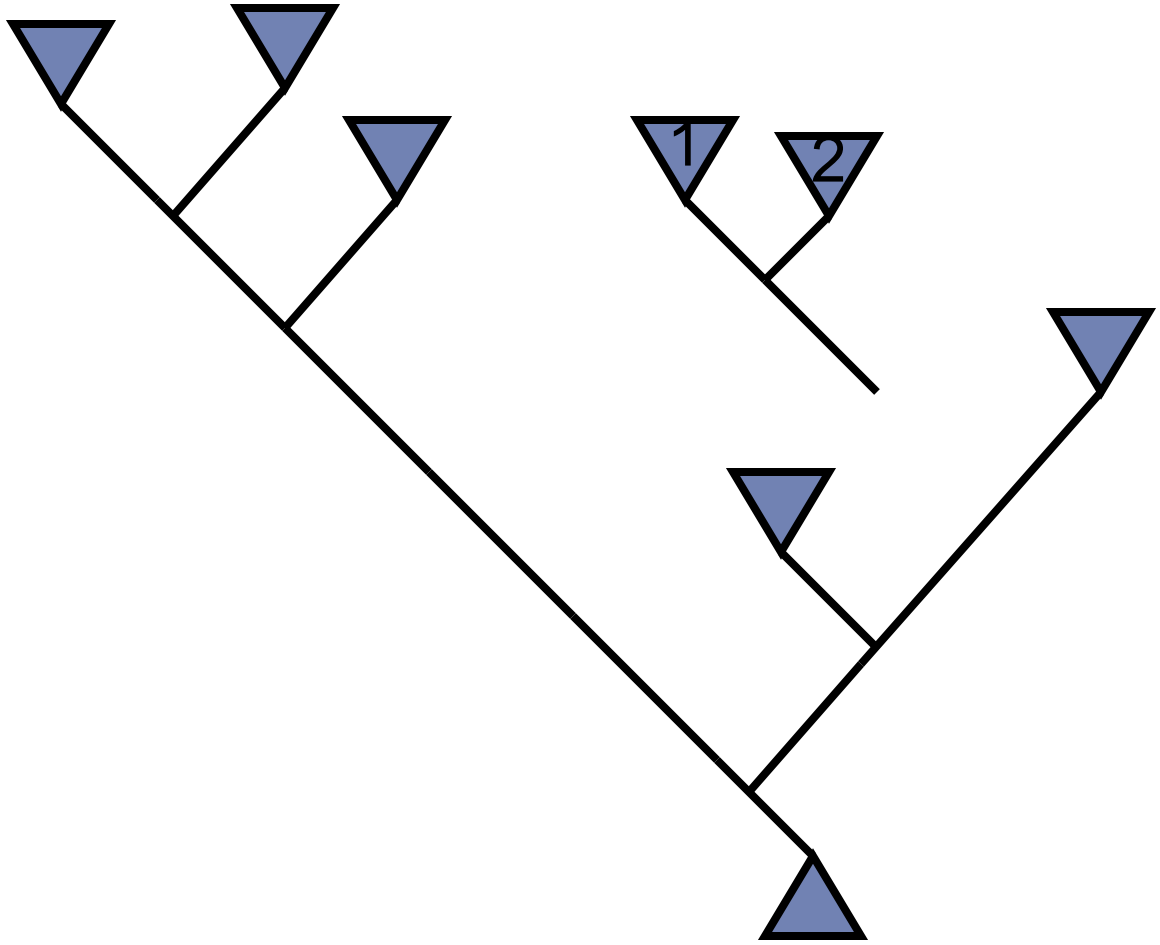
Searching with approximate likelihoods



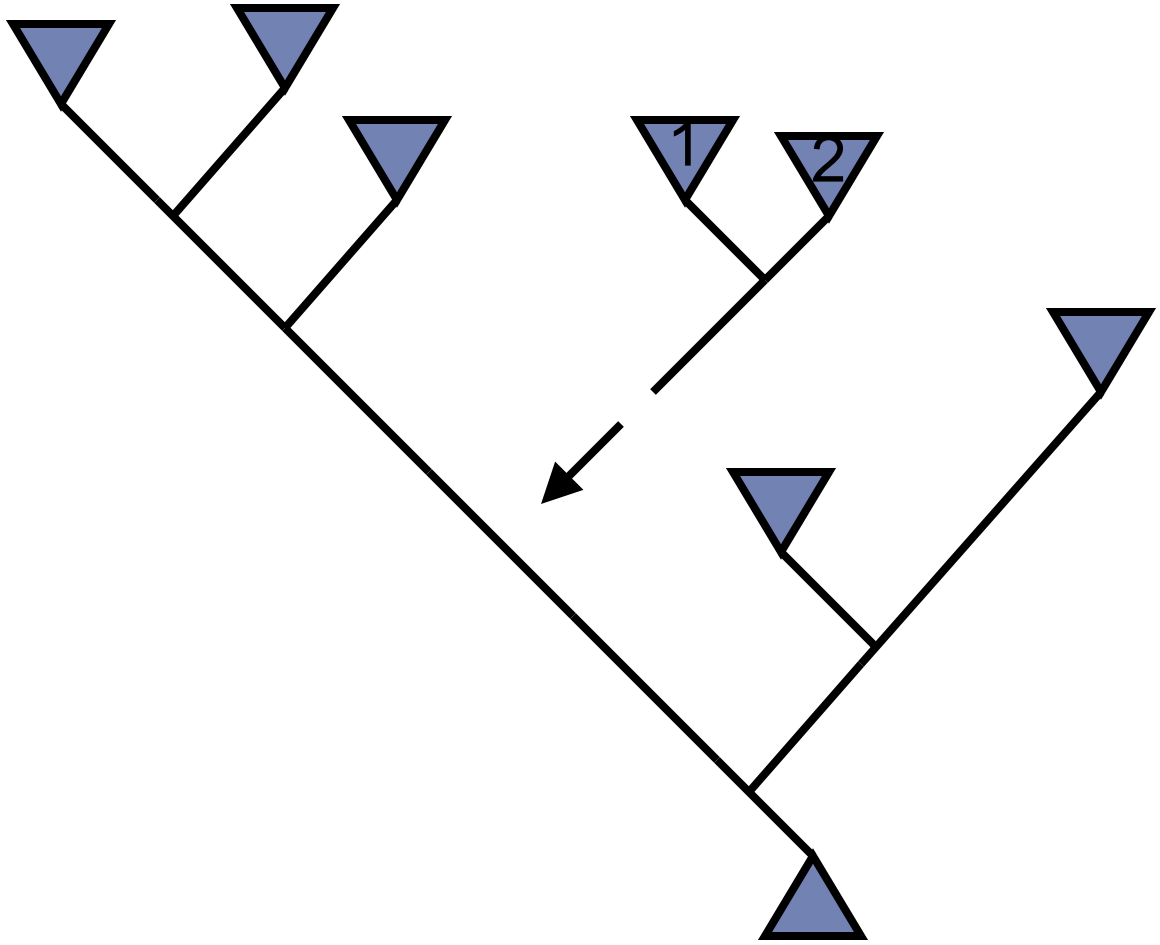
Altering the tree: subtree pruning-regrafting (SPR)



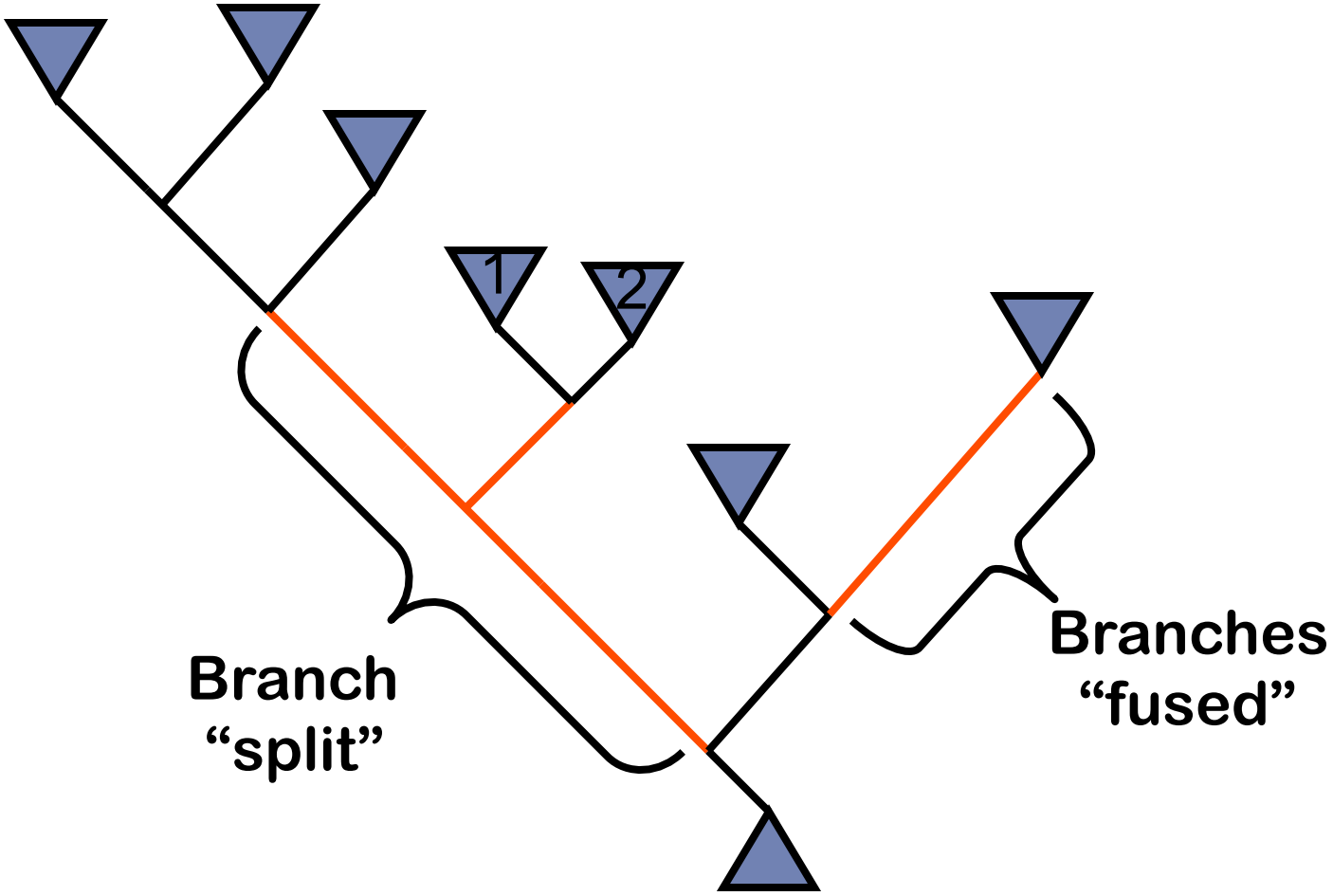
Altering the tree: subtree pruning-regrafting (SPR)



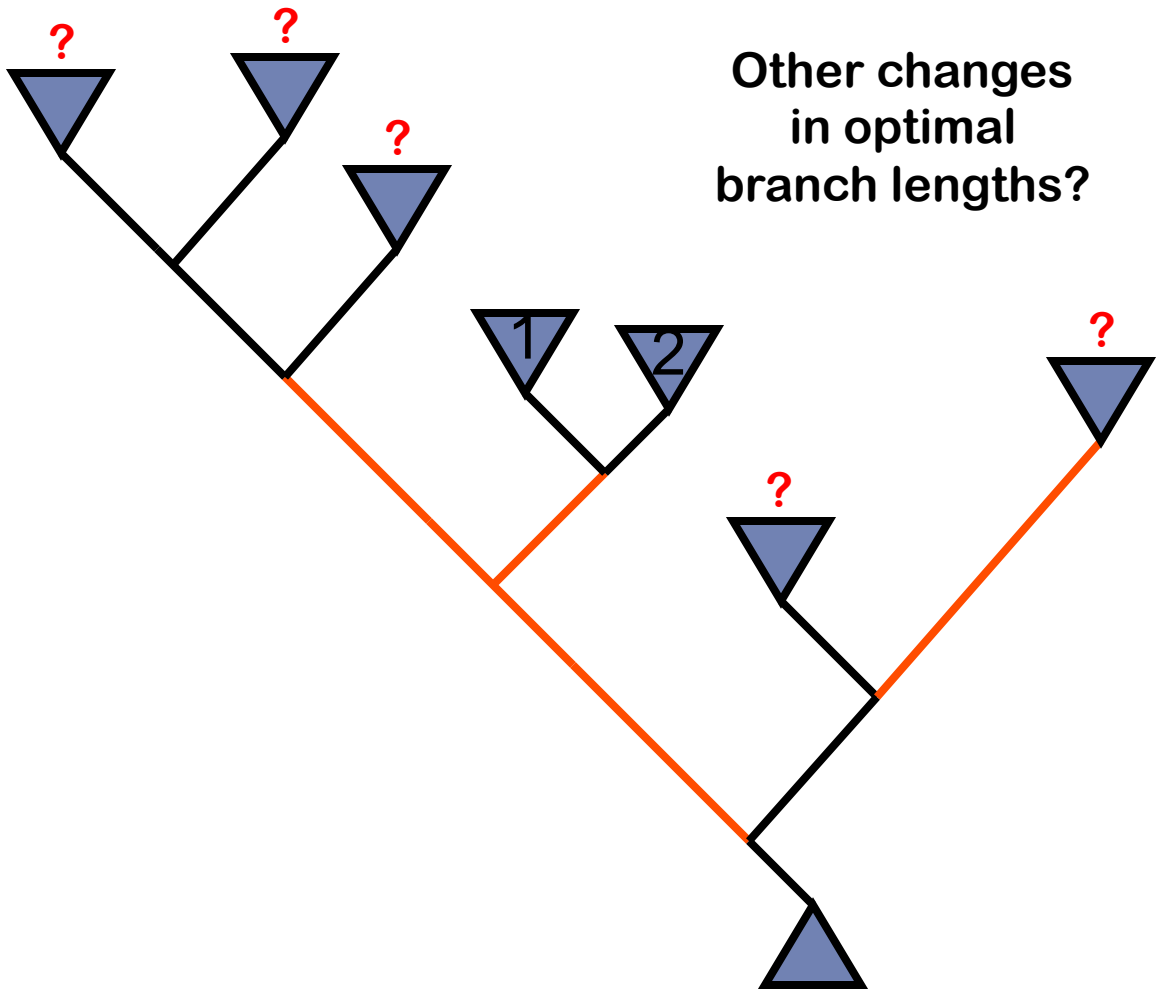
Altering the tree: subtree pruning-regrafting (SPR)



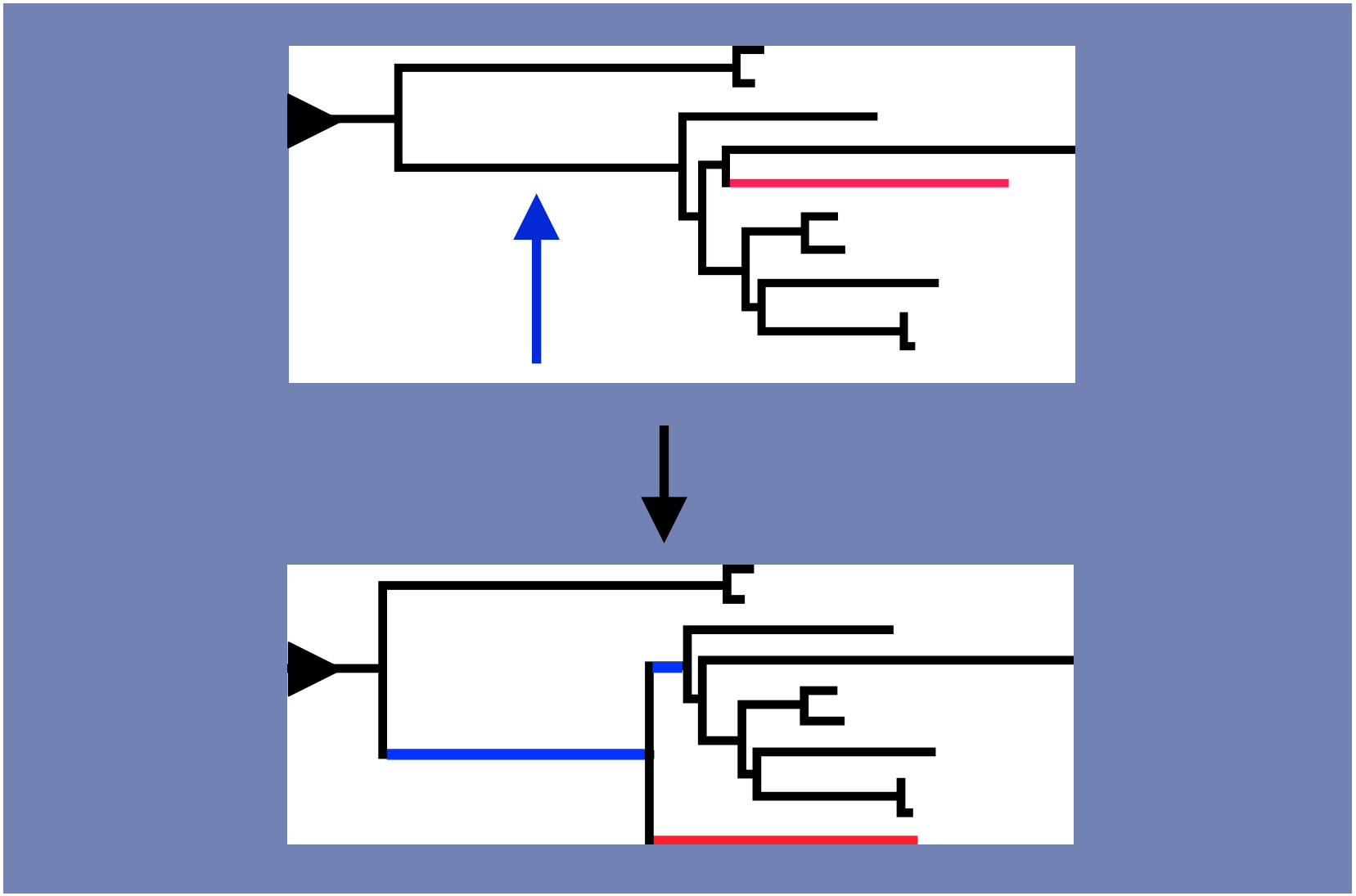
Scoring and optimizing the new topology

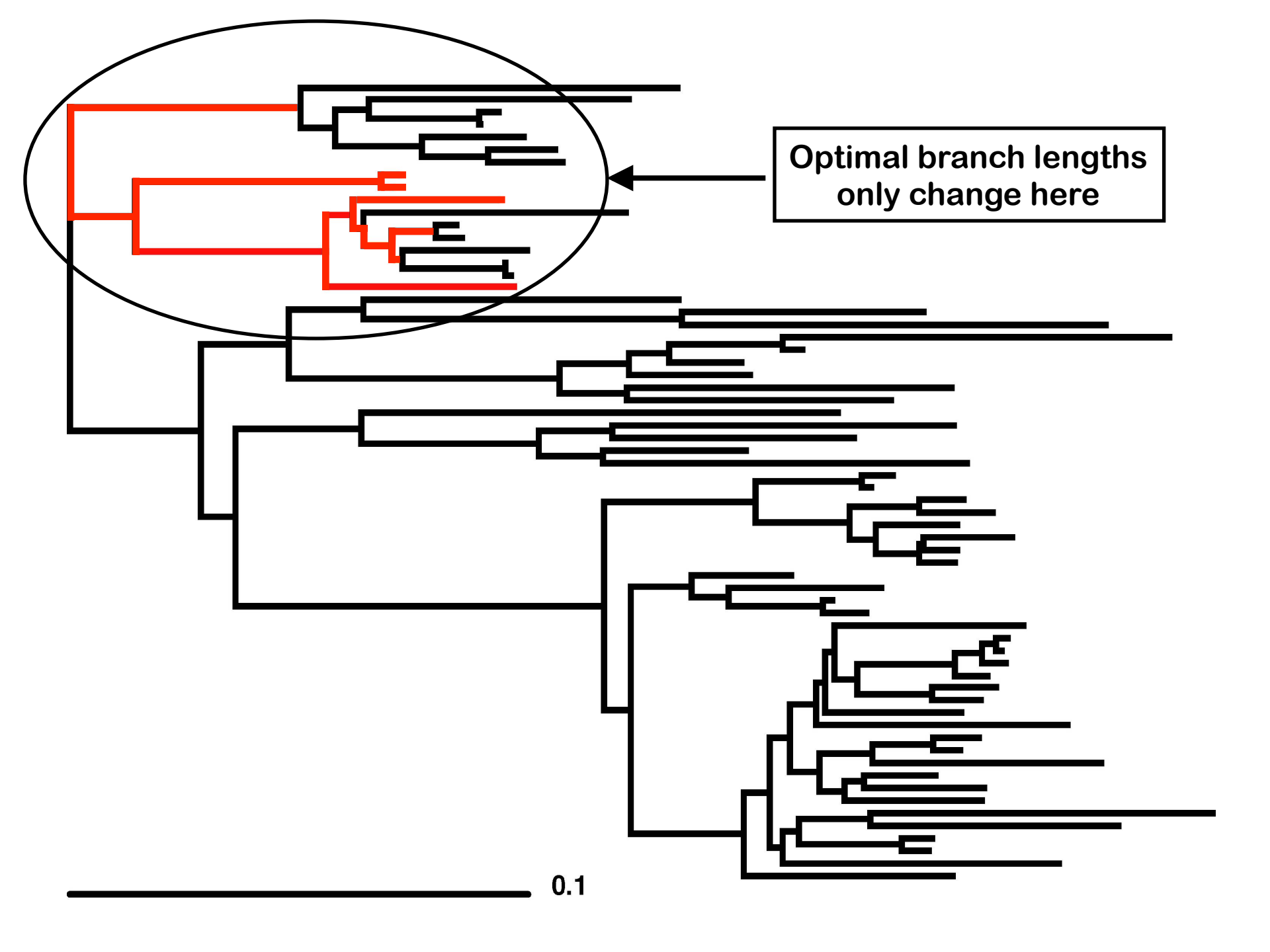


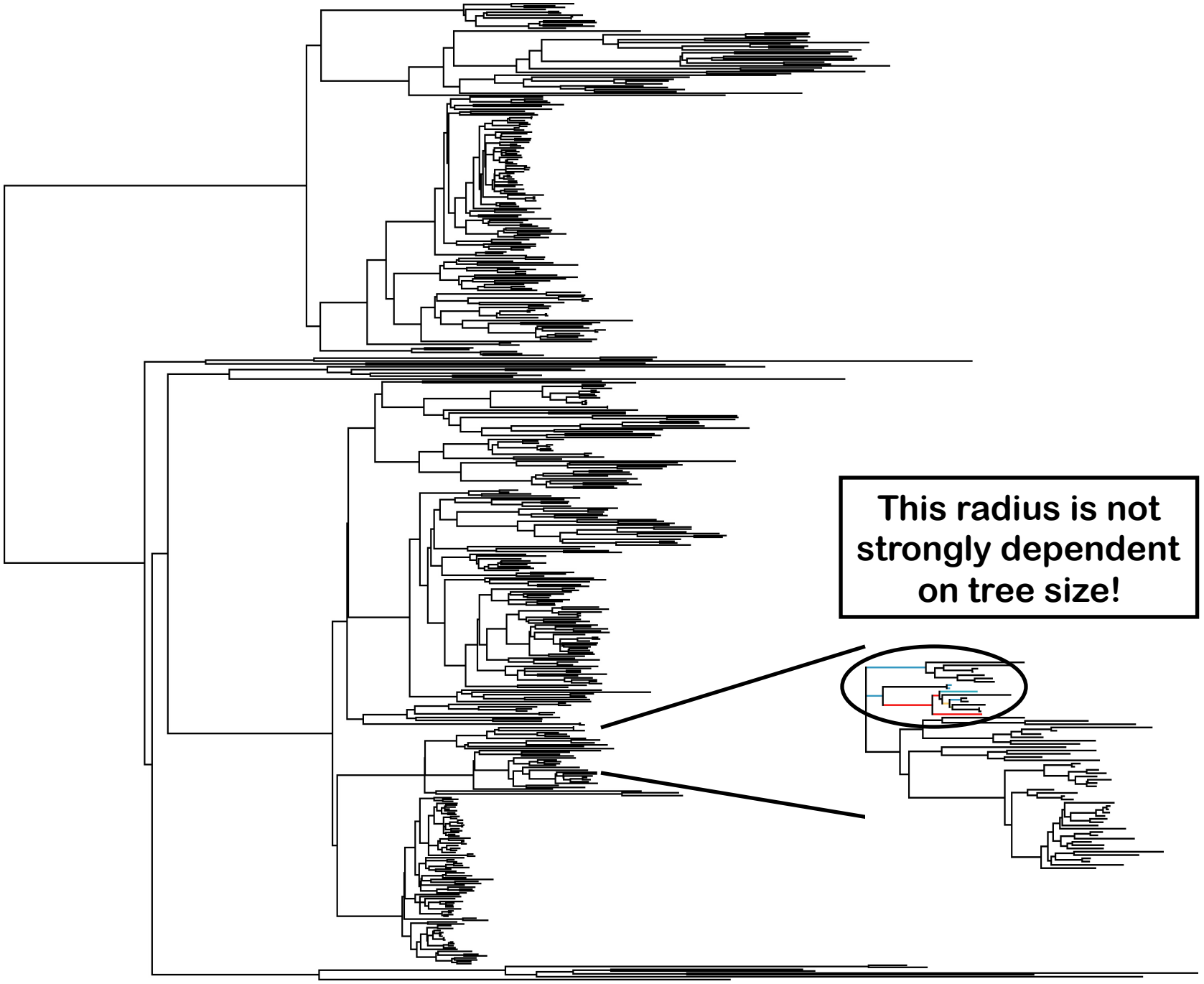
Scoring and optimizing the new topology



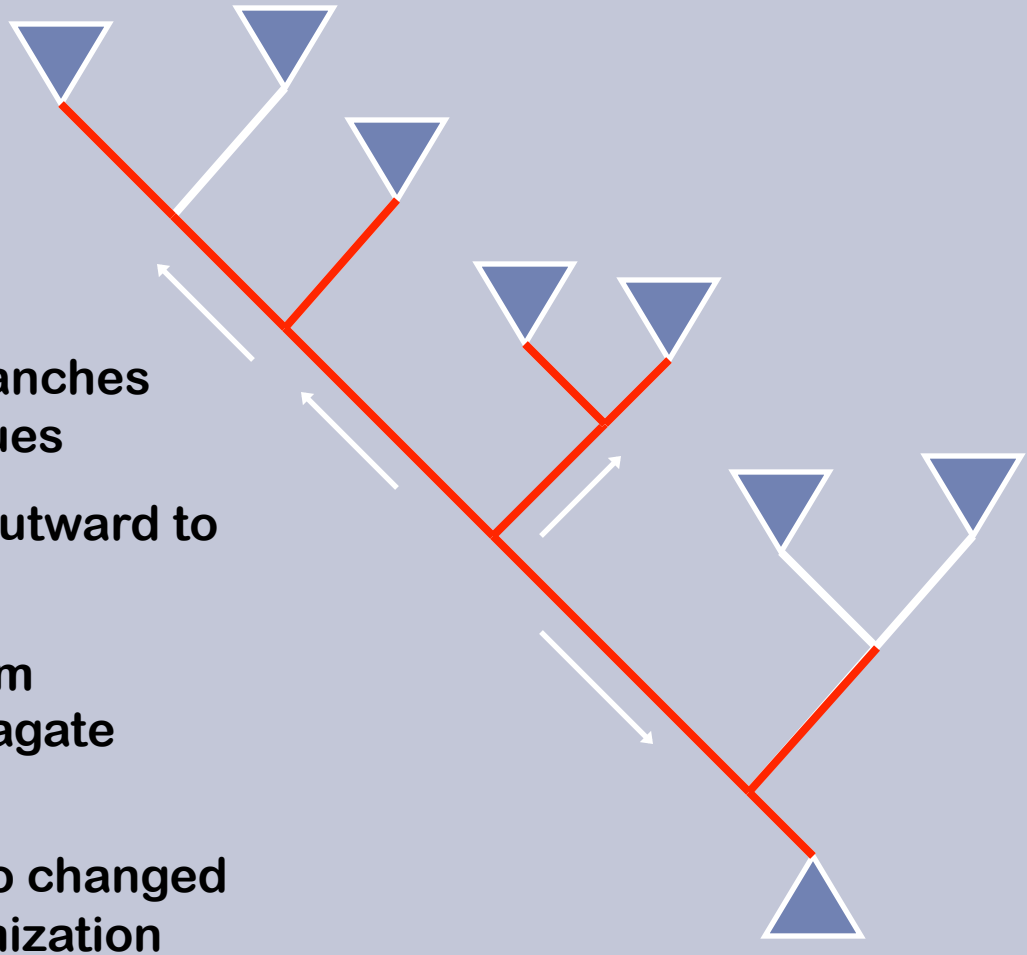
Where do optimal branch lengths change?




Optimal branch lengths
only change here
0.1


This radius is not strongly dependent on tree size!

GARLI's post-swap optimization

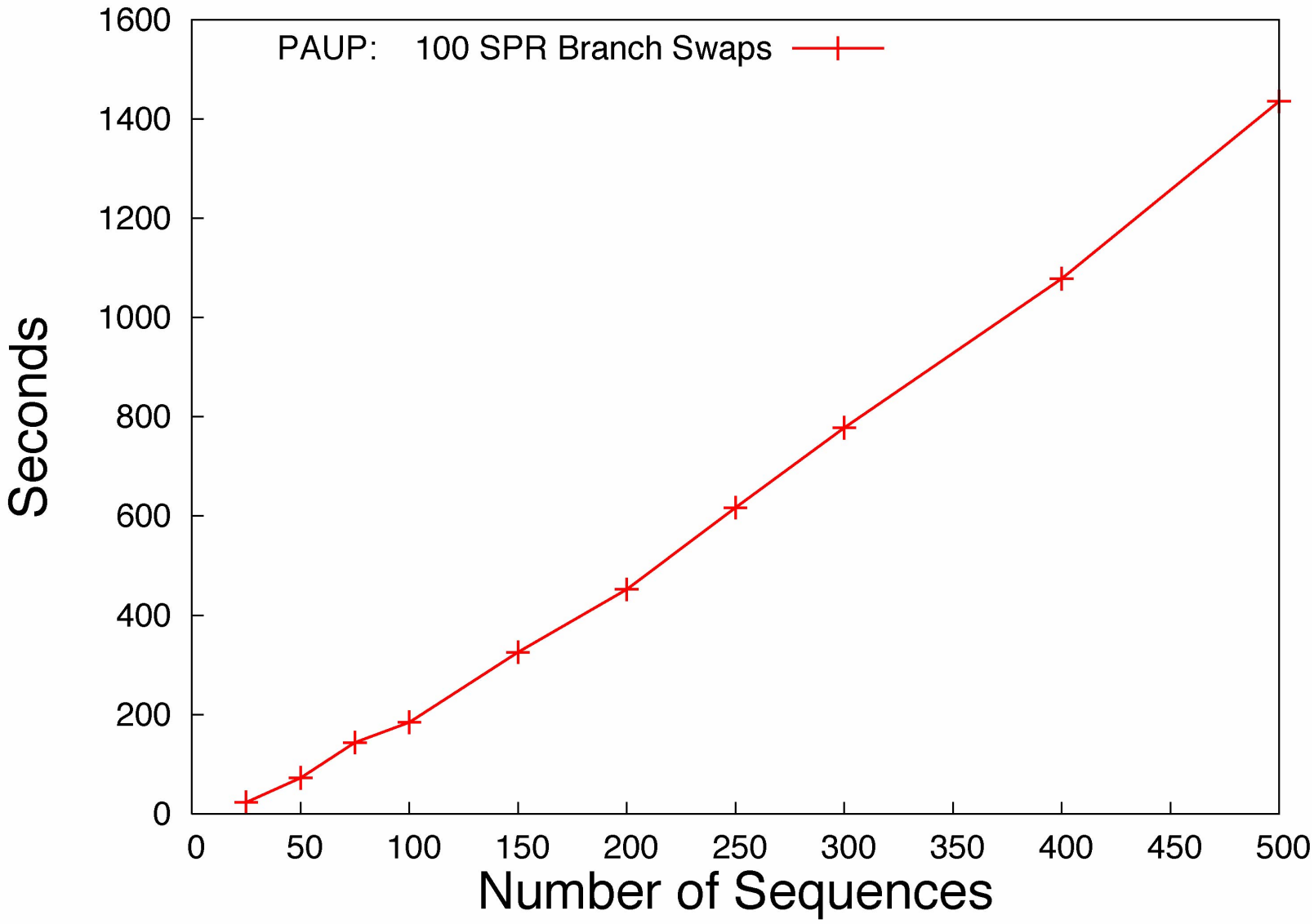


Optimization rules:

1. Optimize the 3 proximal branches until near their optimal values
2. "Propagate" optimization outward to other branches
3. If a branch length is far from optimum, continue to propagate outward
4. After propagation, return to changed branches for another optimization pass

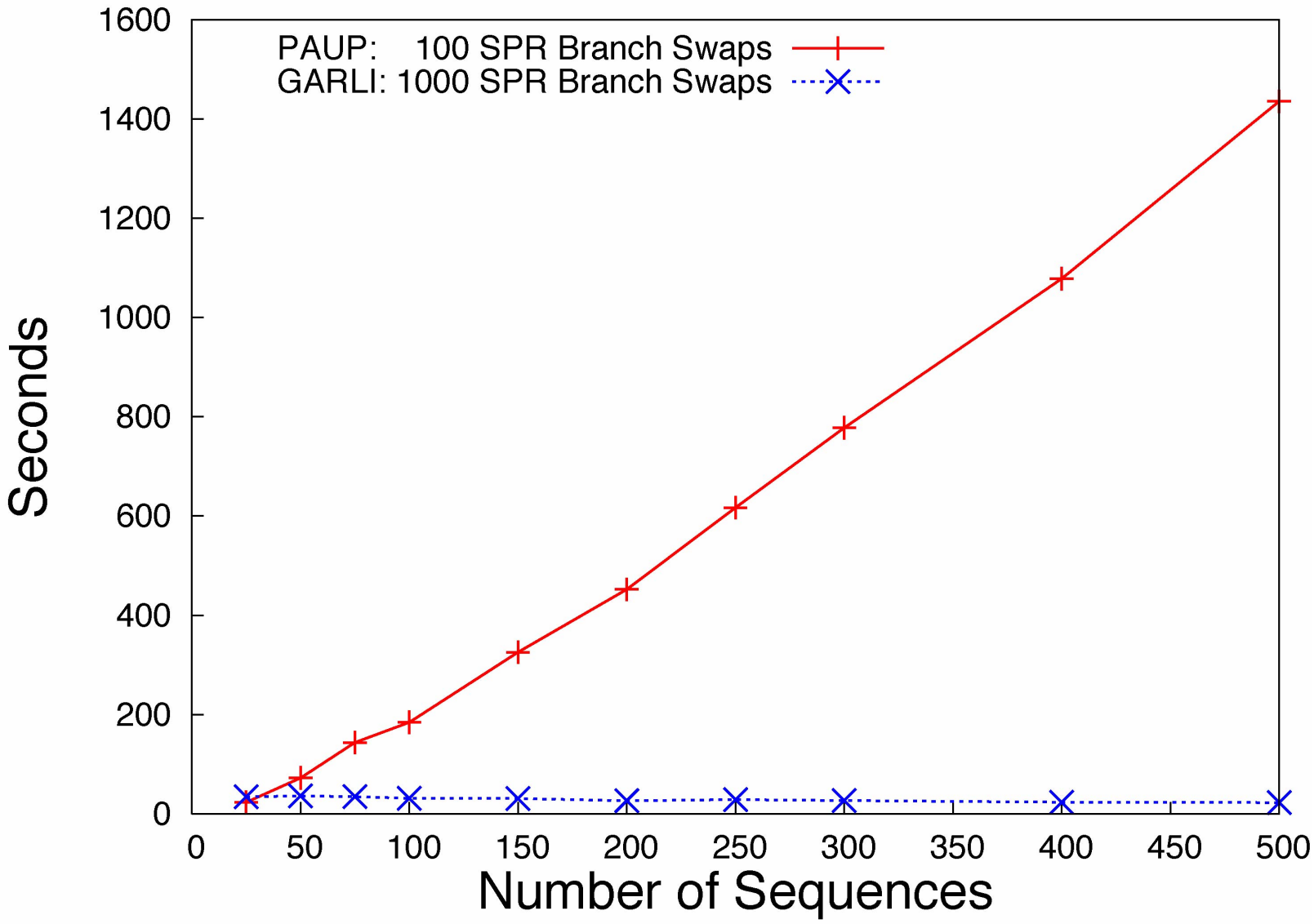
Topology evaluation times

(normalized with respect to # of site patterns)



Topology evaluation times

(normalized with respect to # of site patterns)



Conclusions

GARLI's localized method makes branch-length optimization largely independent of the number of sequences

Several other fast ML heuristics also owe much of their speed to localized optimization (PHYML, RAxML)

Using GARLI in practice

Performance comparisons (brief)

Allowed models

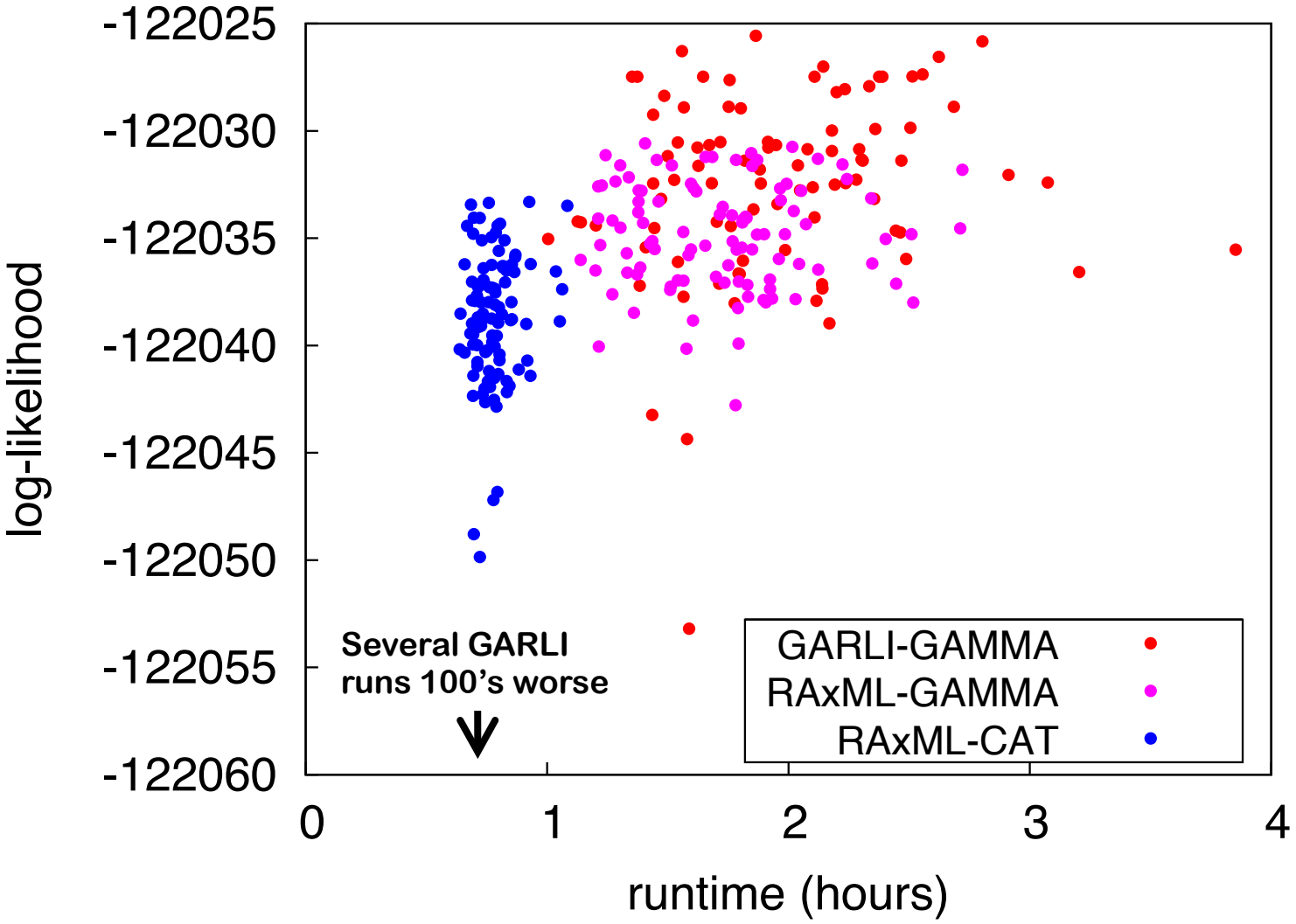
Search strategies

Performance comparisons against other software

More subjective than one would like:

- **What constitutes comparable analyses?**
- **What criteria should be used to compare methods?**
- **Models and likelihood values often not exactly comparable**
- **Most software can be "tuned" to perform better on any particular dataset**
- **Simulated datasets are far too easy to analyze**

Performance comparison: 228 taxon x 4811 nucleotide dataset



ML tree inference software

Some of the most used (alphabetically): GARLI, PAUP*, PHYML, RAxML

For small datasets (< 50 taxa), all of the ML tree inference programs perform well

For large datasets (hundreds of sequences):

- PAUP* is very rigorous, but slowest
- RAxML is generally the fastest
- GARLI often has a slight edge over RAxML in optimality (although often more variability)

RAxML is very efficient for huge datasets (1000+ sequences)

ML tree inference software

NOTE: There can be substantial differences in which program performs best depending on the specific dataset!

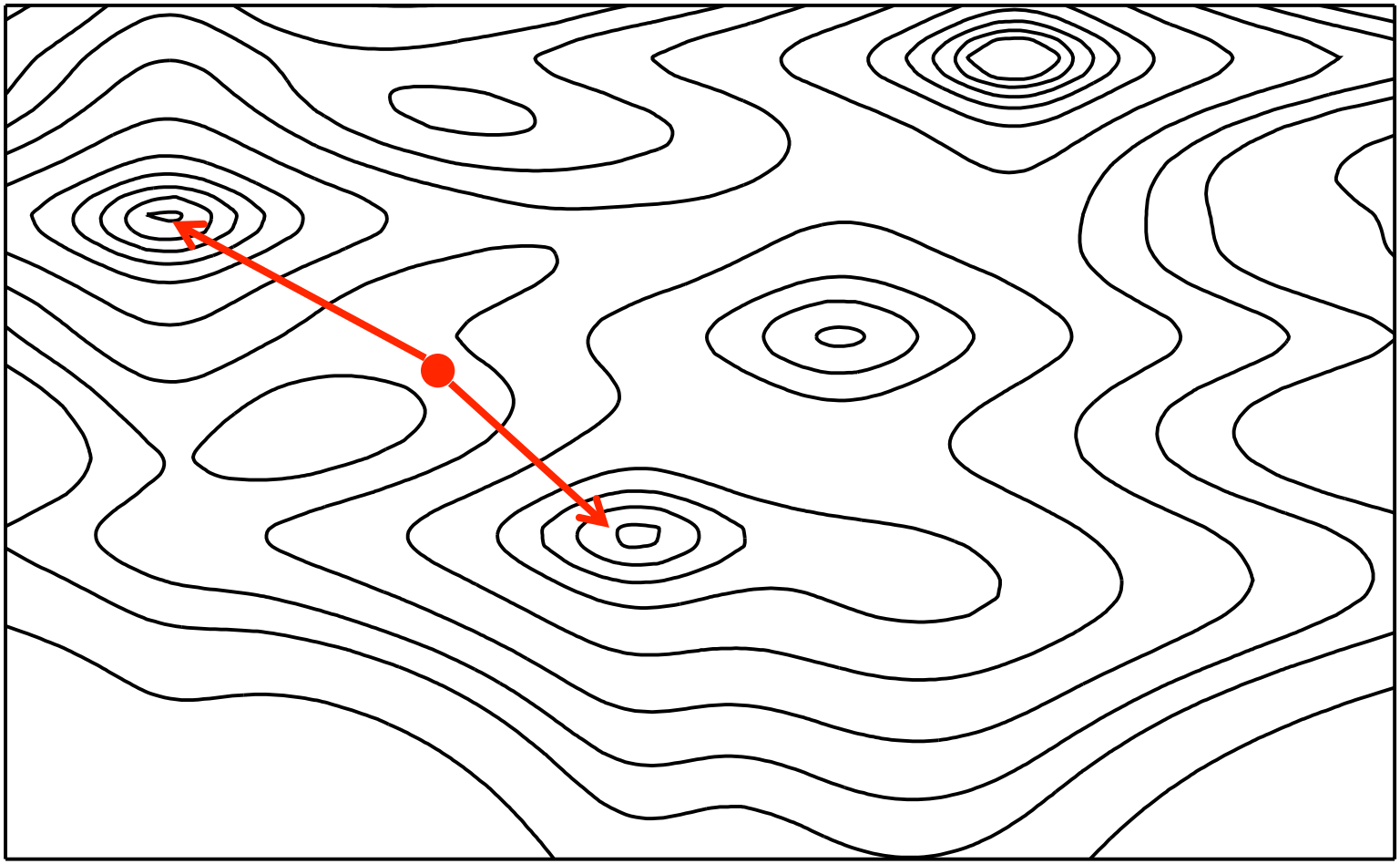
Search strategies in GARLI

Multiple search replicates must ALWAYS be done

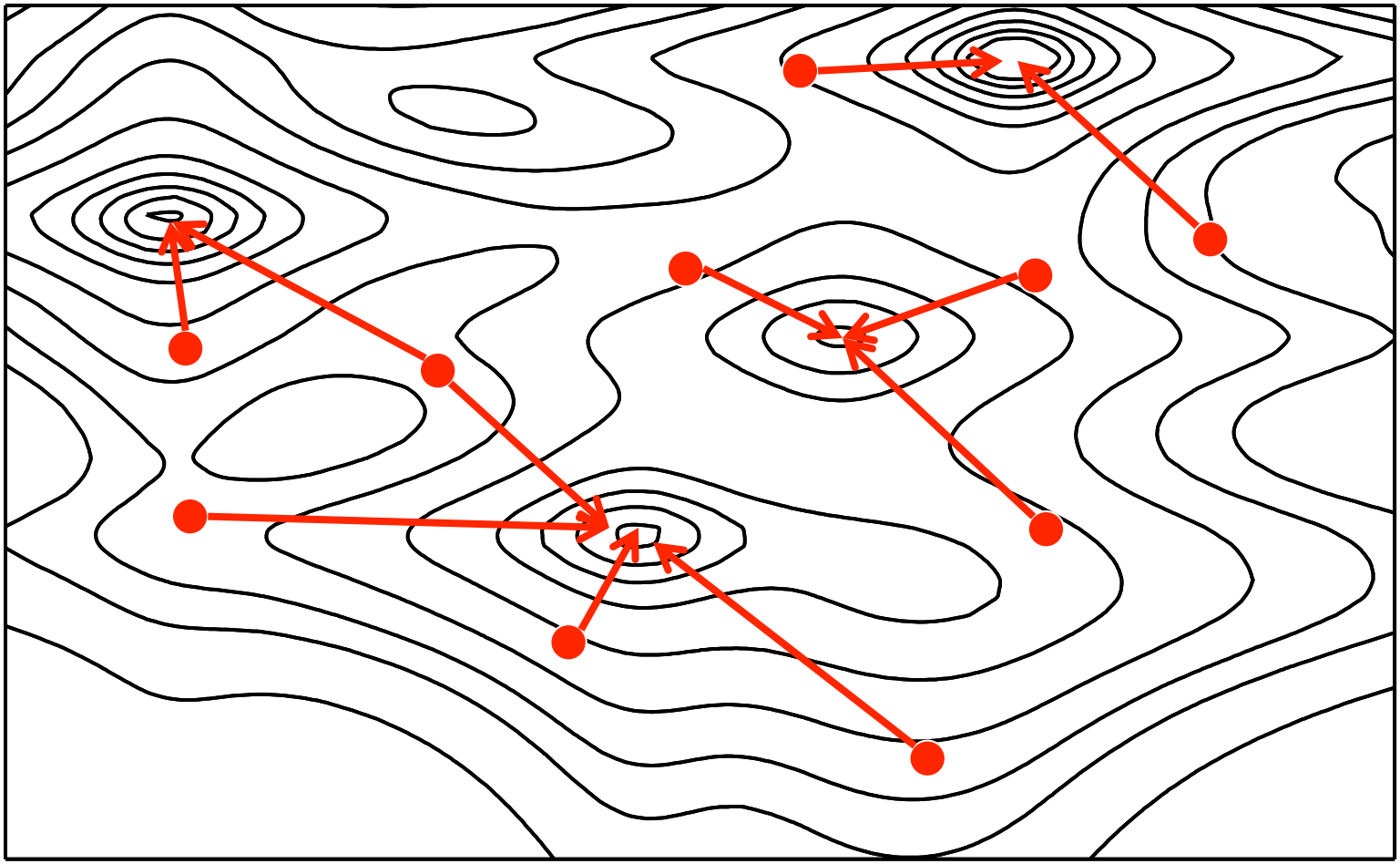
If variable results across search replicates seen:

- **Make changes to improve the search**
- **and/or do more search replicates**

Search repeatability and multiple replicates



Search repeatability and multiple replicates



Search difficulty

On average:

- More sequences = worse
- More characters (signal) = better

parsimony informative sites better indicator of signal than total # of sites

Tuning search intensity

Tradeoff between search intensity and runtimes

Not always a direct relationship between search intensity and solution optimality

Given a certain amount of time, how can we best use it?

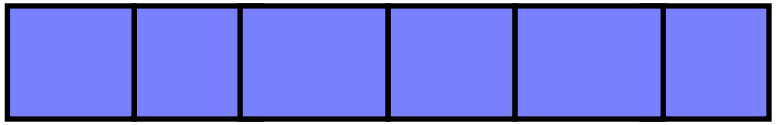
Balancing search intensity and runtimes

H hours

Per run, more likely to find global optimum

3 thorough searches

May be more likely to find global optimum within H hours



6 fast searches

Practical search recommendations

Search repeatability is an indicator of how analyses are going (much like convergence of independent MCMC runs)

Saturating the search space (lots of searches) may be better than very long searches

On some large datasets, unlikely to find the same tree twice

How else can I speed up/improve searches?

Eliminate identical sequences!

Constrained tree searches won't help (in GARLI)

Starting tree

- **Providing a decent (potentially unresolved) starting tree can help on large datasets**

What about bootstrapping?

GARLI can run multiple search replicates per bootstrap reweighting, with the best scoring tree saved

More intense searches add up quickly when bootstrapping

Find fastest settings that give consistent results on full data, use those for bootstrap searches

Evolutionary models

GARLI is geared toward model flexibility and rigorous parameter estimation

Model types

- **Any GTR submodel for nucleotides**
- **Various common amino acid models**
- **Simple codon models**
- **Non-sequence data (**$\mathrm{M}k$ **and** $\mathrm{M}k_v$**)**
- **Partitioned models**
- **Indel models (unreleased)**

How/when to partition

PartitionFinder may prove to be a great approach to partitioned model selection

Smaller subsets increase sampling error, lead to parameter estimation difficulties and model breakdown

Over-partitioning may have serious consequences in ML inference, less in Bayesian

Non-bifurcating trees

GARLI returns trees with polytomies when branches have an optimal length of zero, but some programs do not

This can become very important in low divergence phylogenomic studies

Assorted GARLI features

Single data file may be analyzed at the nucleotide, amino acid and codon levels without making changes to it

Multithreaded version for multiple CPU cores

MPI version simplifies bootstrapping on clusters

Full checkpointing

Topological constraints (positive, negative, backbone)

Other assorted GARLI features

Specification and fixation of model parameter values

Site-likelihood output for all models including partitioned, for input into CONSEL, etc.

Ancestral state reconstruction for all models

Eventually: Beagle GPU version

Computer exercises