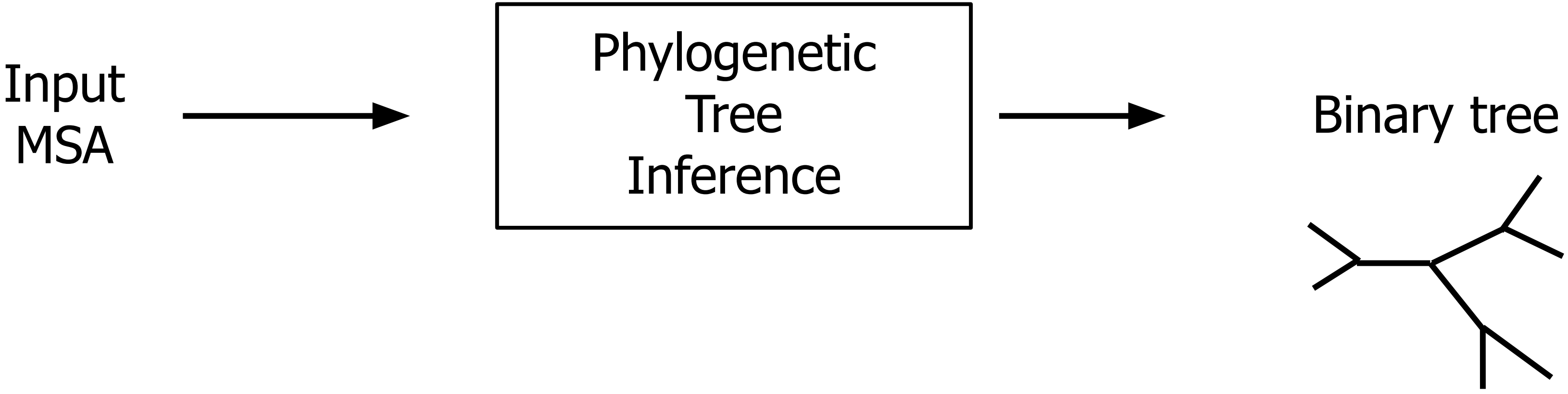
Adaptive RAxML-NG: Accelerating Phylogenetic Inference under Maximum Likelihood using dataset difficulty

Anastasis Togkousidis, Oleksiy M Kozlov, Julia Haag,
Dimitri Höhler, Alexandros Stamatakis

HITS

Introduction

Introduction

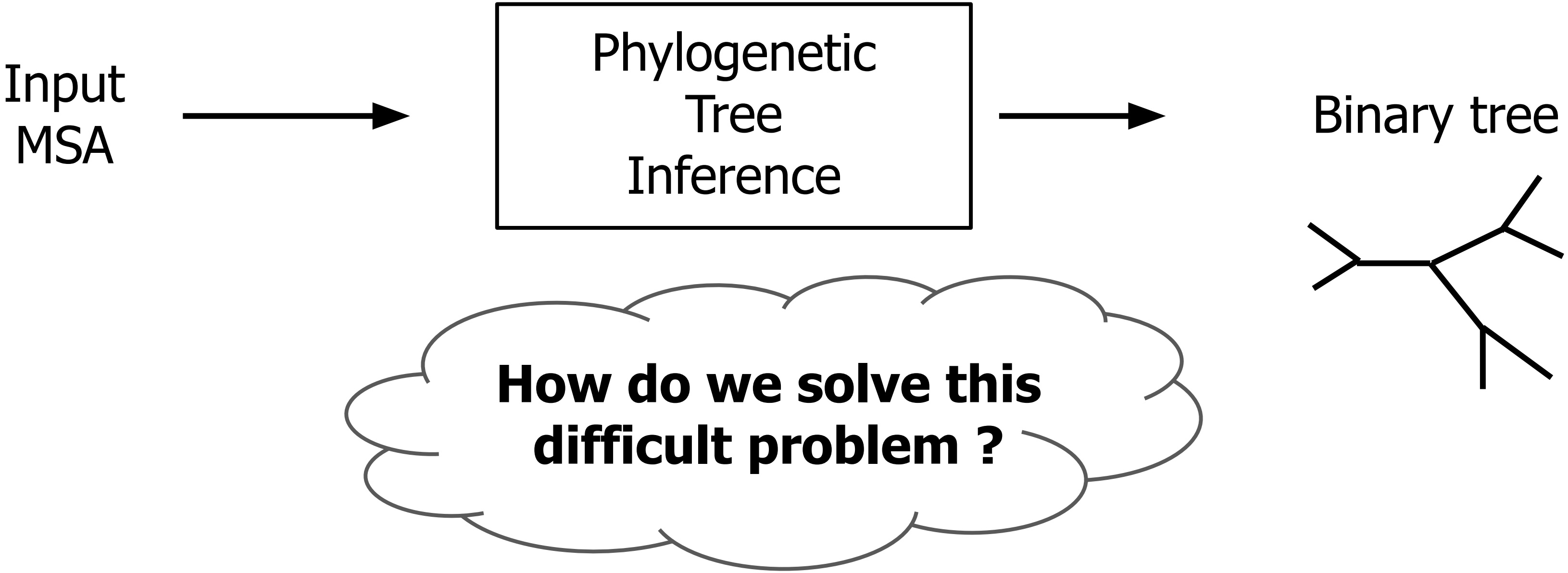


Input MSA

Phylogenetic Tree Inference

Binary tree

Introduction



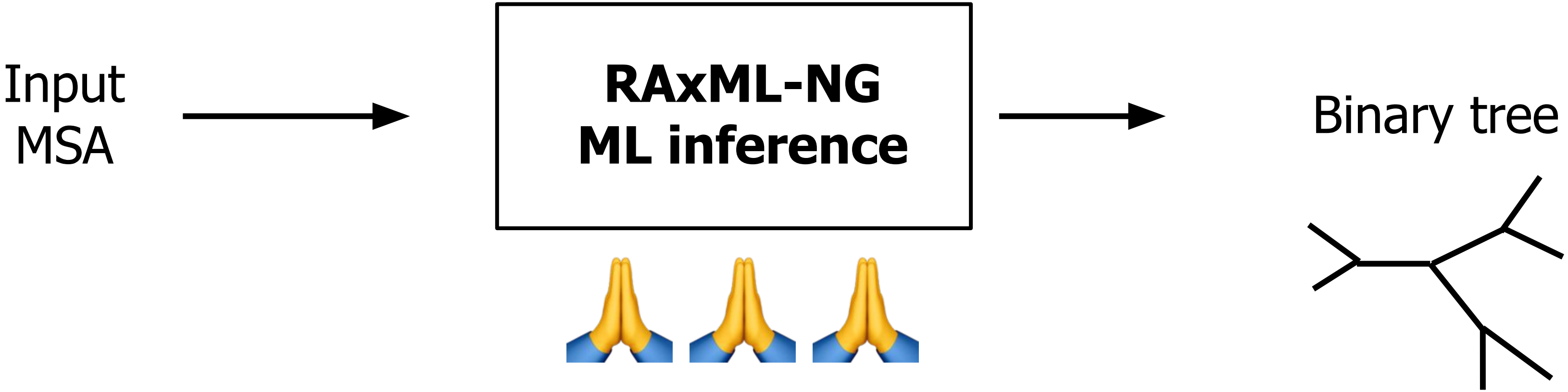
Input
MSA

Phylogenetic
Tree
Inference

Binary tree

How do we solve this difficult problem ?

Introduction



Input
MSA

RAxML-NG
ML inference

🙏🙏🙏

Binary tree

Introduction

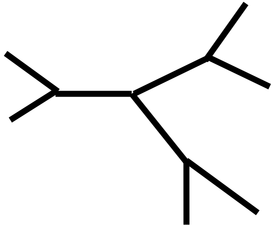
Input
MSA

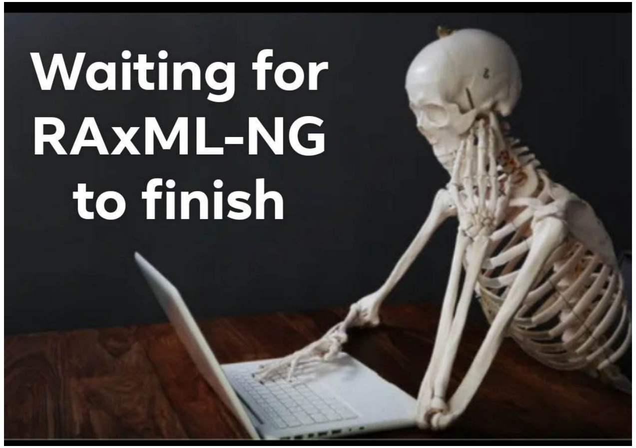
→

RAxML-NG
ML inference

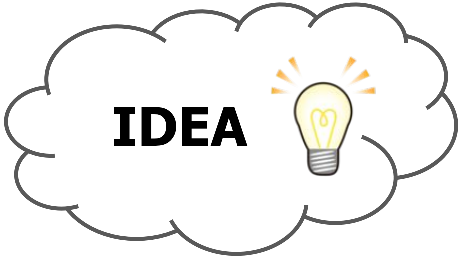
→

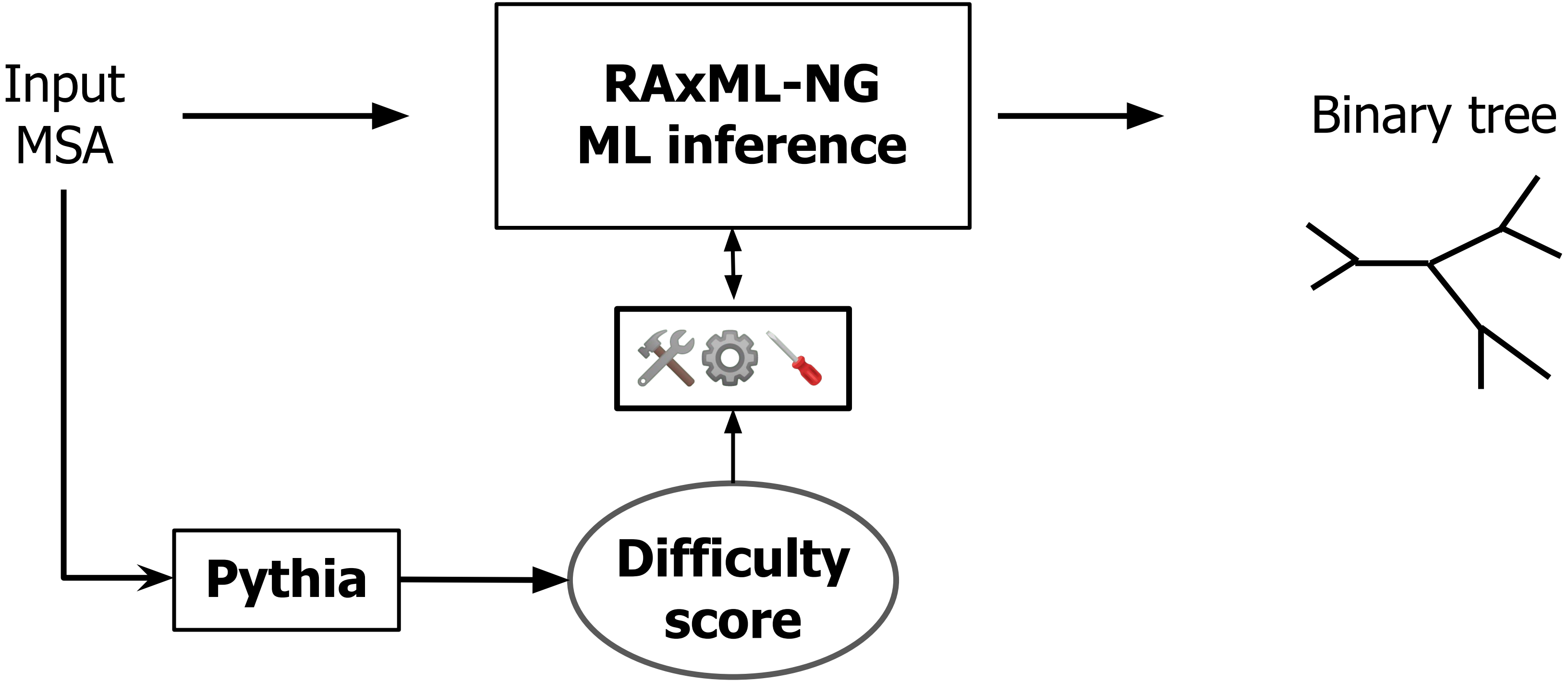
Binary tree





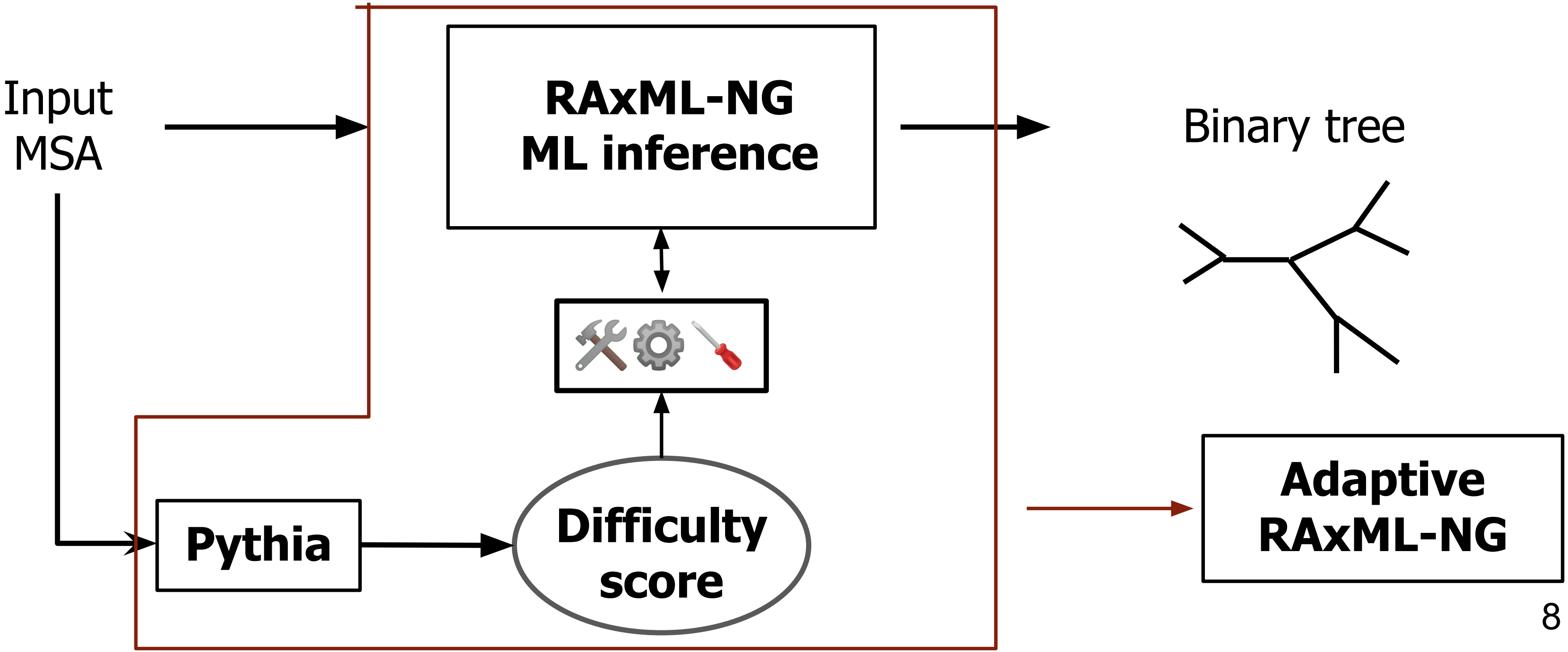
Introduction





Introduction

IDEA



Input
MSA

RAxML-NG
ML inference

Binary tree

Pythia

Difficulty
score

Adaptive
RAxML-NG

Background

Tree inference

1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

Tree inference

1.

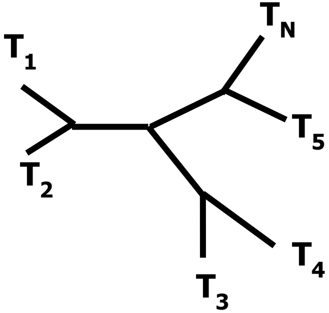
Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

→

2.

LH = -100



Initial tree
(Random, MP)

Tree inference

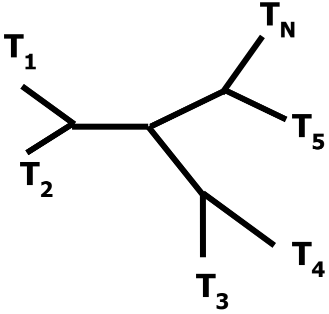
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

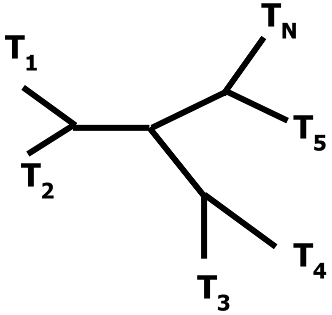
LH = -100



Initial tree
(Random, MP)

3.

LH = -100



Intermediate Trees
(Topological moves)

Tree inference

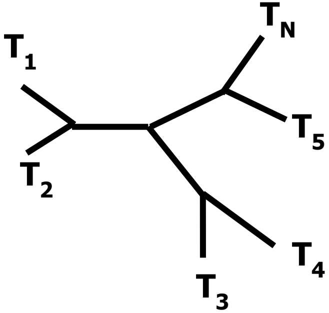
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

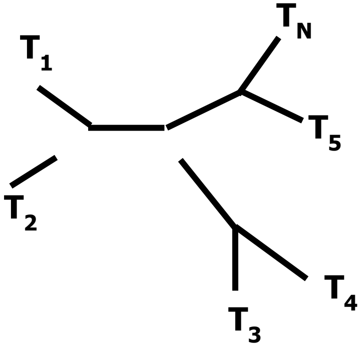
LH = -100



Initial tree
(Random, MP)

3.

LH = -100



Intermediate Trees
(Topological moves)

Tree inference

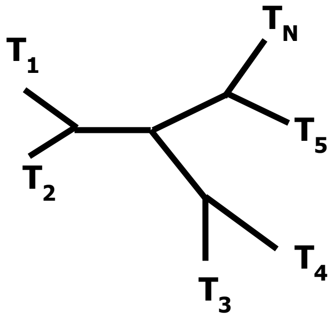
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

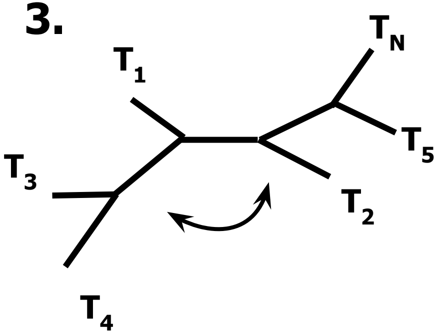
LH = -100



Initial tree
(Random, MP)

3.

LH = -60



Intermediate Trees
(Topological moves)

Tree inference

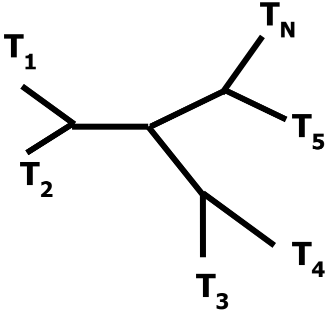
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

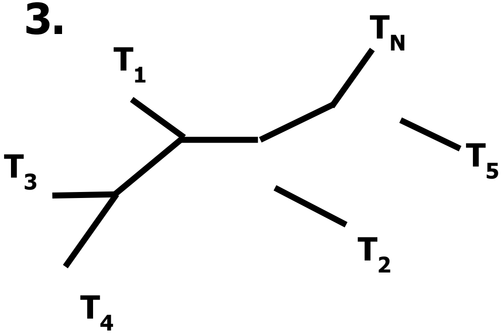
LH = -100



Initial tree
(Random, MP)

3.

LH = -60



Intermediate Trees
(Topological moves)

Tree inference

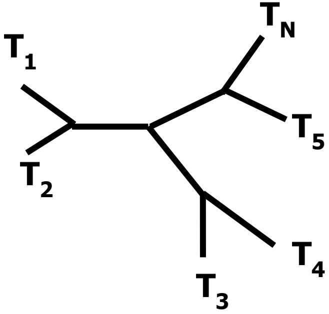
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

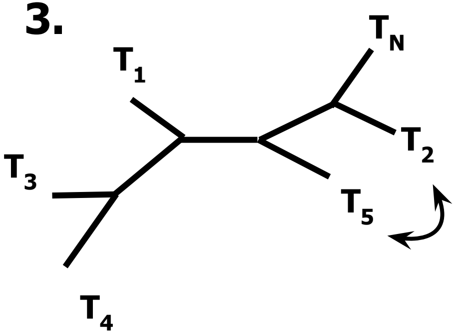
LH = -100



Initial tree
(Random, MP)

3.

LH = -40



Intermediate Trees
(Topological moves)

Tree inference

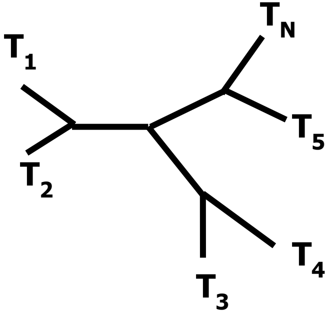
1.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA

2.

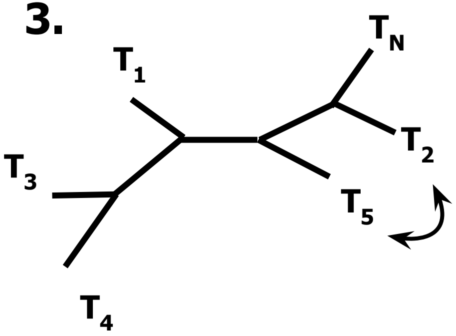
LH = -100



Initial tree
(Random, MP)

3.

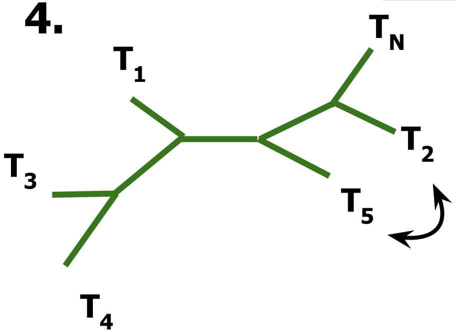
LH = -40



Intermediate Trees
(Topological moves)

4.

LH = -35



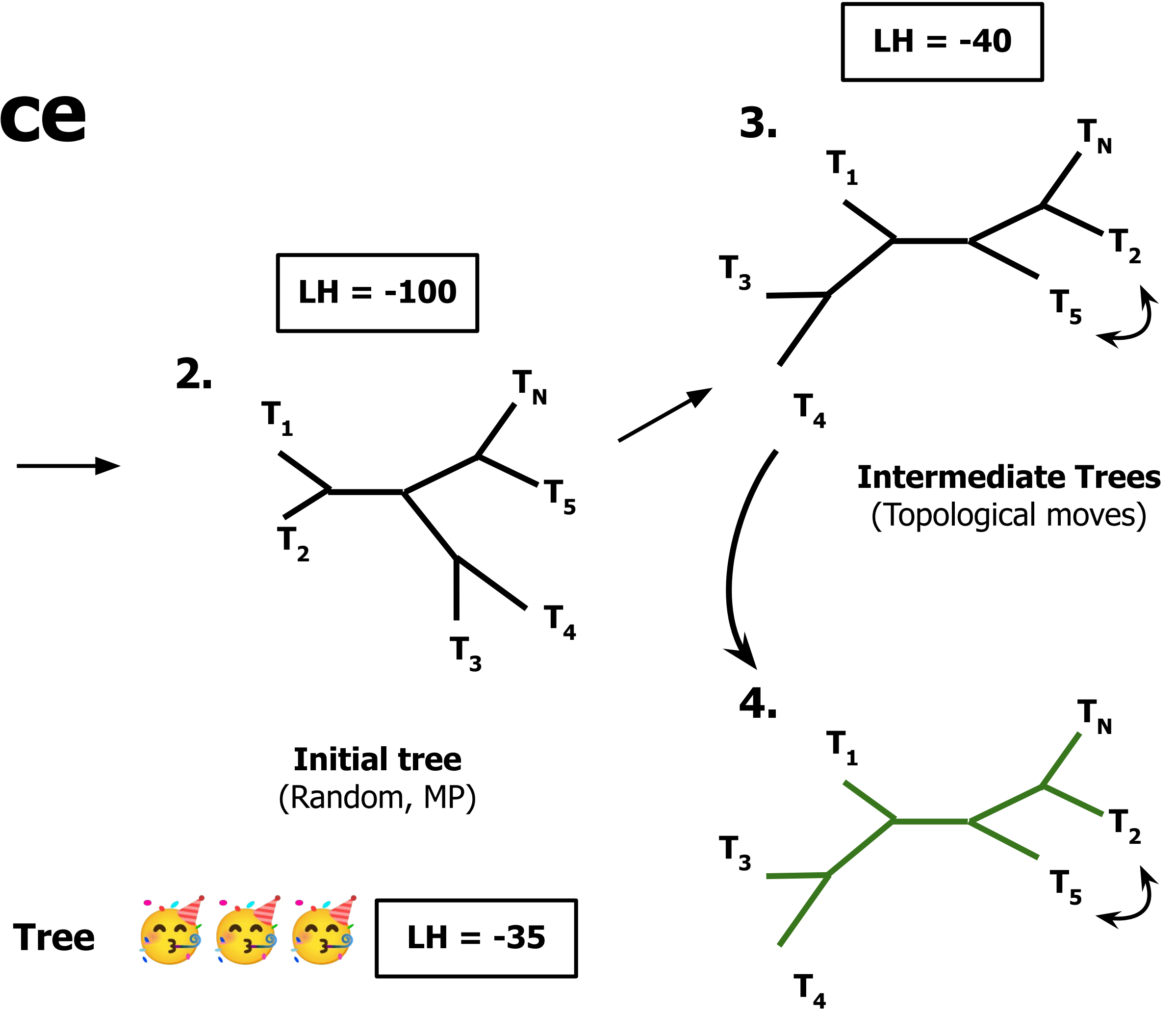
Branch-Length Optimization

Tree inference

1\.

Taxon1: AACAGTAC--AA
Taxon2: ATCATTACC-AA
Taxon3: AAGAGTACC-AA
.
.
.
TaxonN: AGCAGTAC--AA

MSA



Initial tree
(Random, MP)

Intermediate Trees
(Topological moves)

Final Tree 🥳🥳🥳 **LH = -35**

The concept of difficulty

Easy datasets:

- For some datasets, independent ML tree searches starting from different trees, converge to a **single - or topologically similar - tree(s)**
- We say that these datasets exhibit a “**clear phylogenetic signal**”

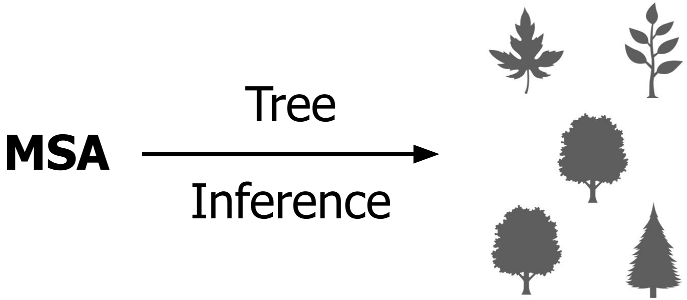
The concept of difficulty

Easy datasets:

MSA $\xrightarrow[\text{Inference}]{\text{Tree}}$

The concept of difficulty

Easy datasets:



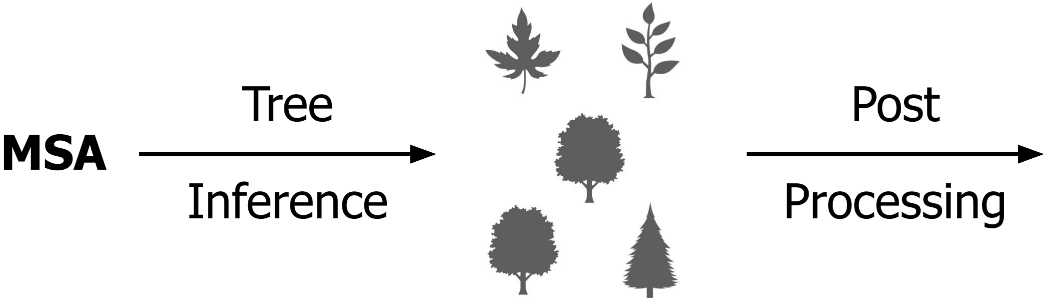
MSA

Tree

Inference

The concept of difficulty

Easy datasets:



MSA

Tree

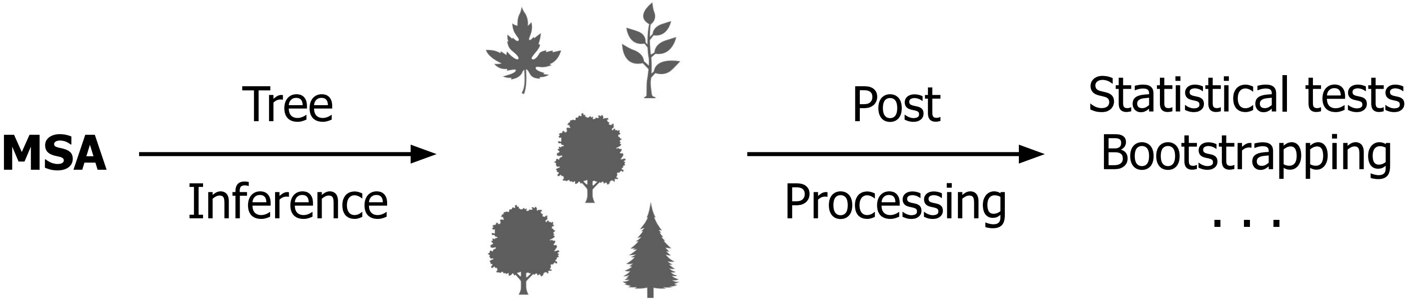
Inference

Post

Processing

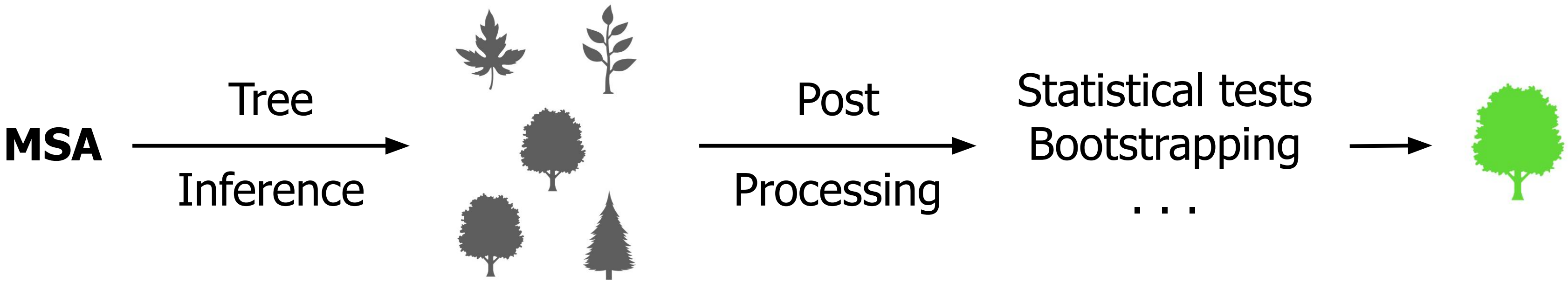
The concept of difficulty

Easy datasets:



The concept of difficulty

Easy datasets:



MSA

Tree Inference

Post Processing

Statistical tests
Bootstrapping
. . .

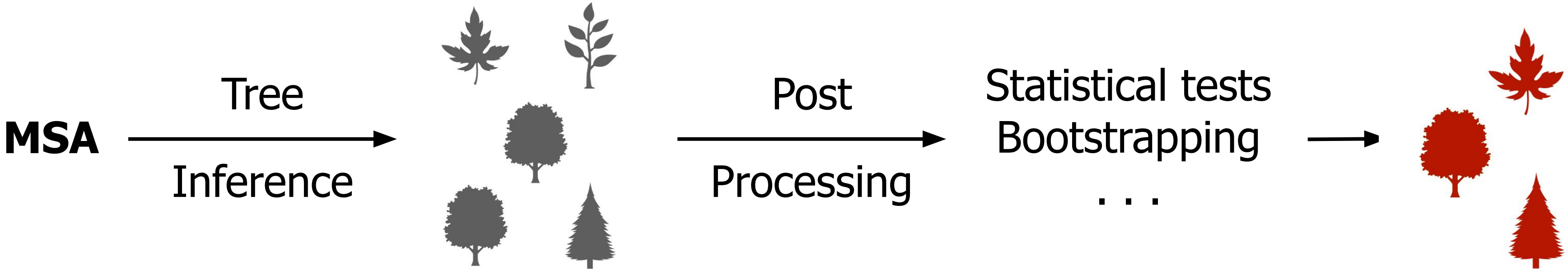
The concept of difficulty

Difficult datasets:

- Other datasets yield **topologically highly distinct**, yet **equally likely trees** (equal score)
- These trees are also **statistically indistinguishable** based on IQ-TREE 2 significance tests

The concept of difficulty

Difficult datasets:



MSA

Tree

Inference

Post

Processing

Statistical tests

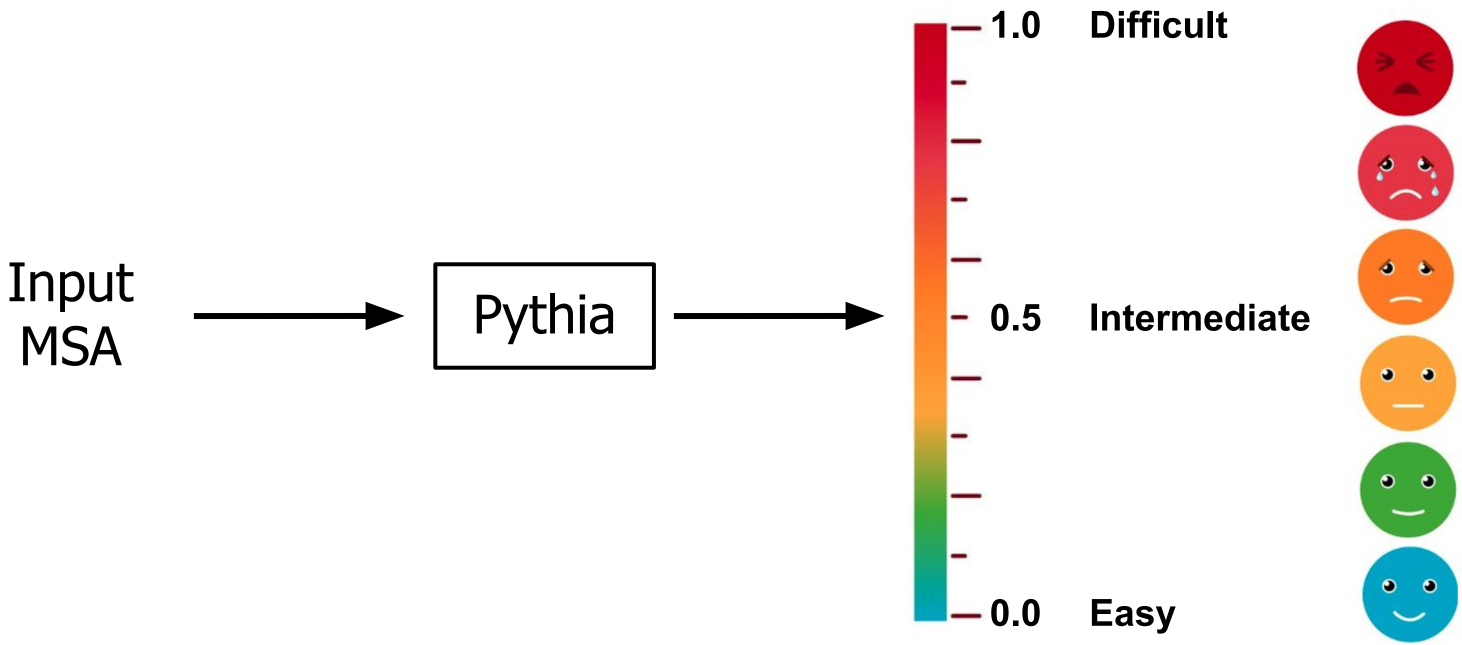
Bootstrapping

. . .

Comments

- These two examples demonstrate two extreme-case dataset types
- There is a whole spectrum of **in-between dataset** cases (intermediate datasets)
- This diverse **behavior** is essentially **quantified** by **pythia**

Difficulty prediction (Haag *et al.*)



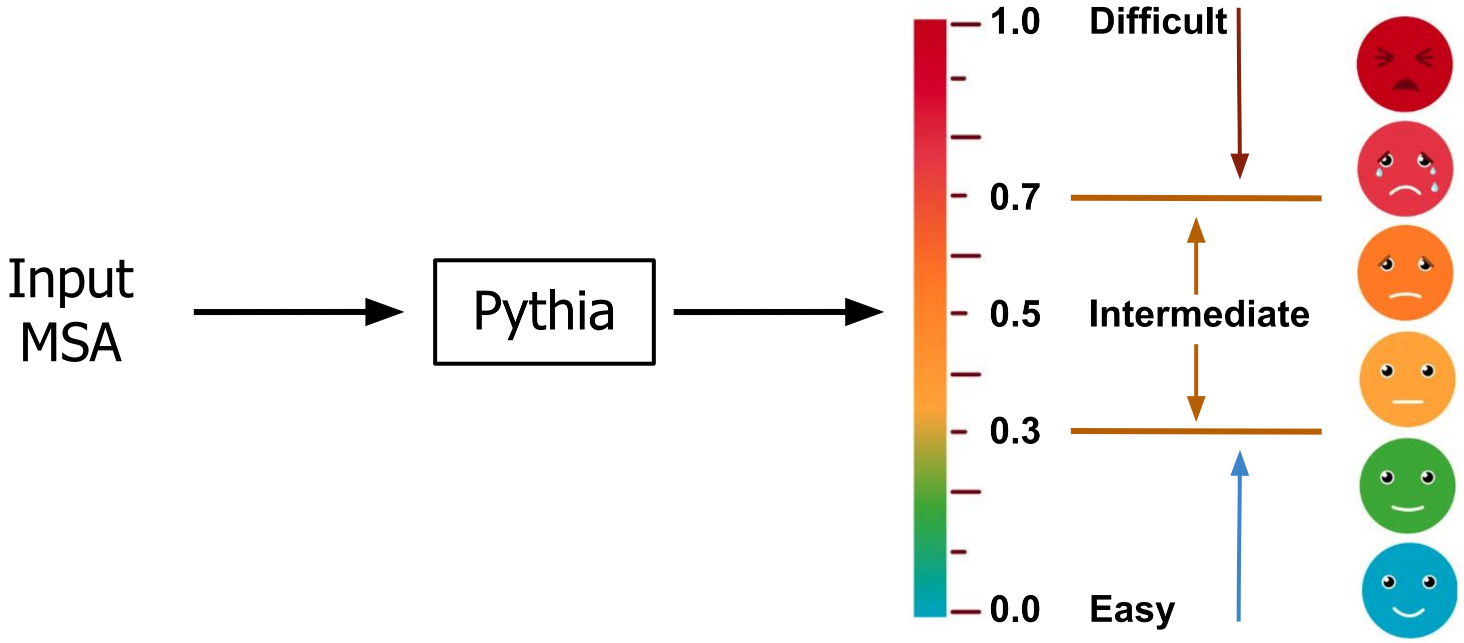
Input MSA → Pythia →

1.0 Difficult

0.5 Intermediate

0.0 Easy

Difficulty prediction (Haag *et al.*)



Input MSA

Pythia

1.0 Difficult

0.7

0.5 Intermediate

0.3

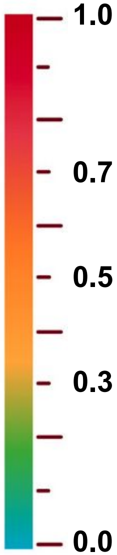
0.0 Easy

Höhler *et al.* (preprint)

- Compared RAxML-NG, IQ-TREE 2 and FastTree 2 on **datasets** with **varying difficulty**.

Höhler *et al.* (preprint)

- Compared RAxML-NG, IQ-TREE 2 and FastTree 2 on **datasets** with **varying difficulty**.



On **difficult** MSAs, all tools perform **similarly** but **only** in terms of **likelihood** score

---

On **intermediate** MSAs, **RAxML-NG** and **IQ-TREE 2** infer significantly **better trees**

On **easy** MSAs, all tools perform **similarly** in terms of **likelihood** score and **topological accuracy**

Adaptive RAxML-NG

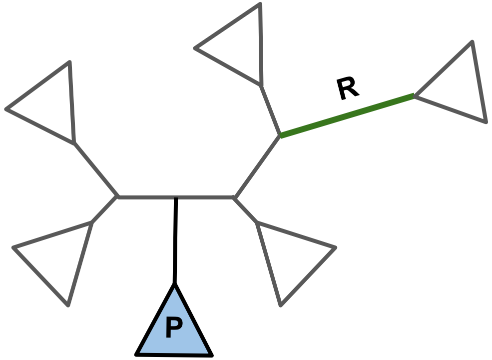
Standard RAxML-NG

- Initiates **10 random** + **10 MP** starting trees
- Applies a tree search **heuristic** to each of them, using exclusively **SPR** moves

Standard RAxML-NG

- Initiates **10 random** + **10 MP** starting trees
- Applies a tree search **heuristic** to each of them, using exclusively **SPR** moves

Subtree Prune and Regraft (SPR) move

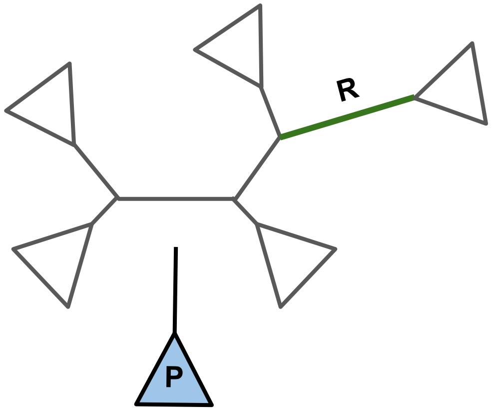


Initial tree topology

Standard RAxML-NG

- Initiates **10 random** + **10 MP** starting trees
- Applies a tree search **heuristic** to each of them, using exclusively **SPR** moves

Subtree Prune and Regraft (SPR) move

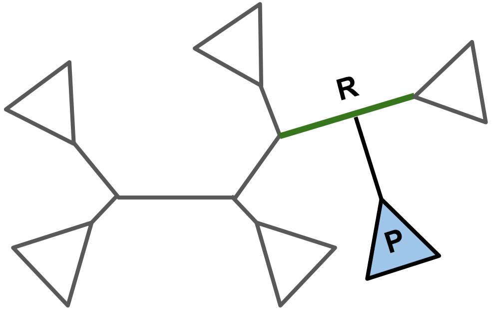


Prune

Standard RAxML-NG

- Initiates **10 random** + **10 MP** starting trees
- Applies a tree search **heuristic** to each of them, using exclusively **SPR** moves

Subtree Prune and Regraft (SPR) move

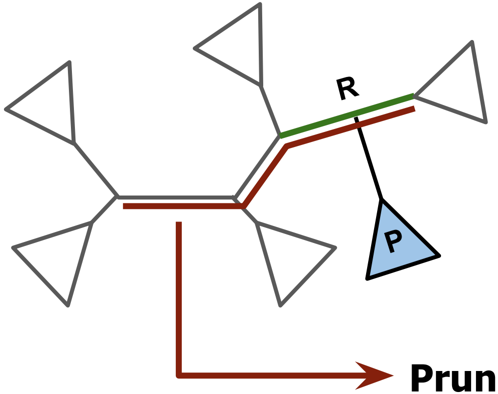


Regraft

Standard RAxML-NG

- Initiates **10 random** + **10 MP** starting trees
- Applies a tree search **heuristic** to each of them, using exclusively **SPR** moves

Subtree Prune and Regraft (SPR) move



Regraft

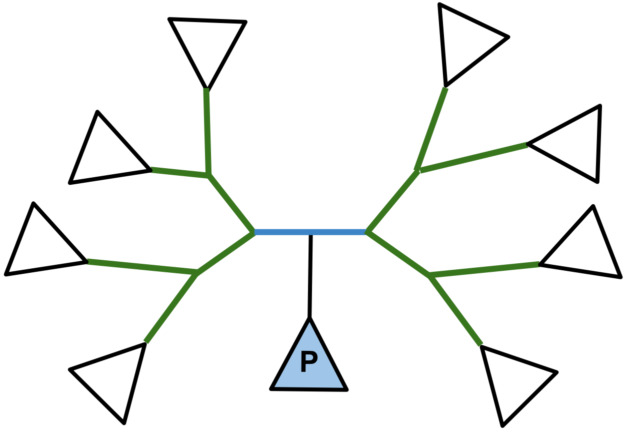
Prune-Regraft distance: 2

SPR round

- Sequence of SPR moves
- Hill-climbing - greedy heuristic
- Two types of SPRs round in **adaptive** RAxML-NG: **Fast** and **Slow** SPR round
- SPR radius: parameter of SPR rounds, denotes the **maximum distance** between the **pruning** and **regrafting** edge

SPR radius

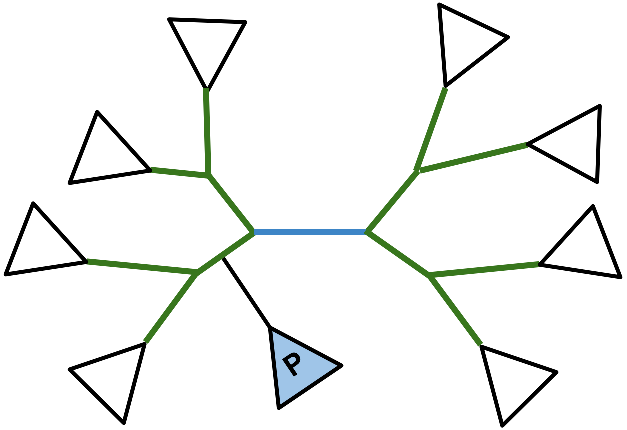
- Example with SPR **radius = 2**



- In an SPR round, RAxML-NG tests **all** possible **regrafting** (green) **edges** up to a certain distance (**radius**)
- It accepts the topology with the highest likelihood score

SPR radius

- Example with SPR **radius = 2**

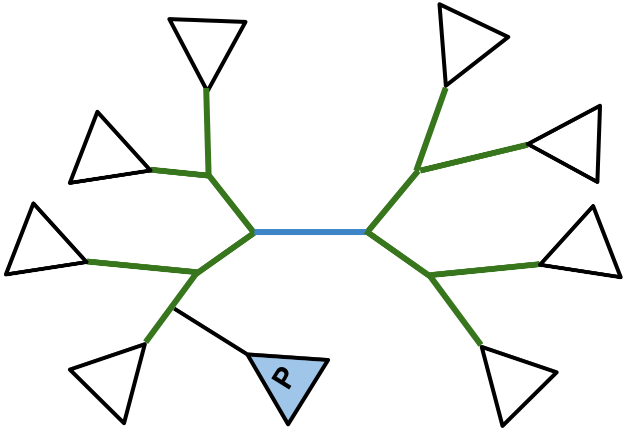


━ **Pruning edge**
━ **Regrafting edge**

- In an SPR round, RAxML-NG tests **all** possible **regrafting** (green) **edges** up to a certain distance (**radius**)
- It accepts the topology with the highest likelihood score

SPR radius

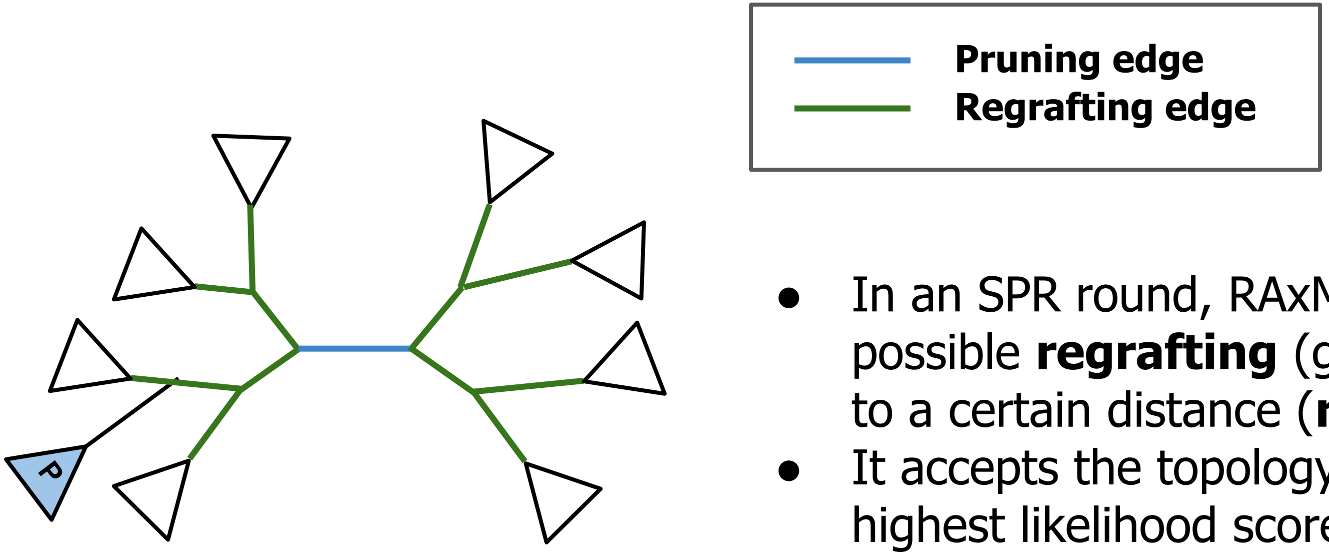
- Example with SPR **radius = 2**



- In an SPR round, RAxML-NG tests **all** possible **regrafting** (green) **edges** up to a certain distance (**radius**)
- It accepts the topology with the highest likelihood score

SPR radius

- Example with SPR **radius = 2**

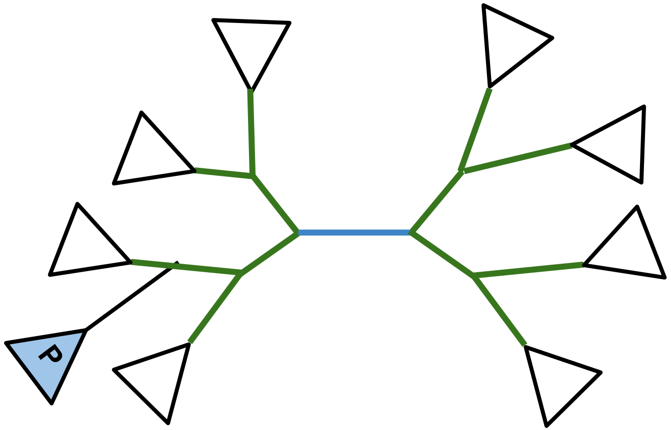


- In an SPR round, RAxML-NG tests **all** possible **regrafting** (green) **edges** up to a certain distance (**radius**)
- It accepts the topology with the highest likelihood score

SPR radius

- Example with SPR **radius = 2**

🟦 **Pruning edge**
🟩 **Regrafting edge**

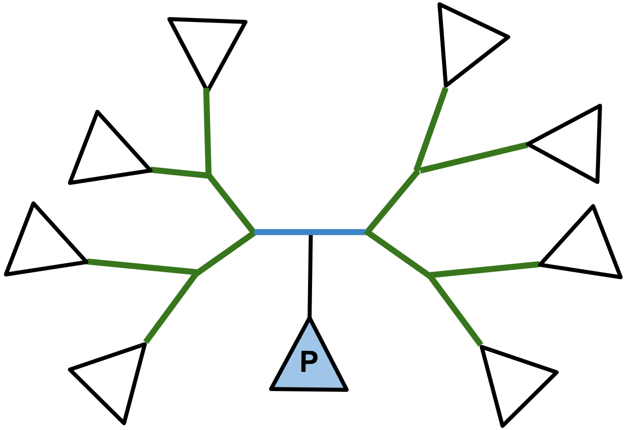


- In an SPR round, RAxML-NG tests **all** possible **regrafting** (green) **edges** up to a certain distance (**radius**)
- It accepts the topology with the highest likelihood score

And so on ….

Fast SPR round

- In **Fast SPR** round, RAxML-NG computes the likelihood using the **branch lengths** of the **initial tree** topology

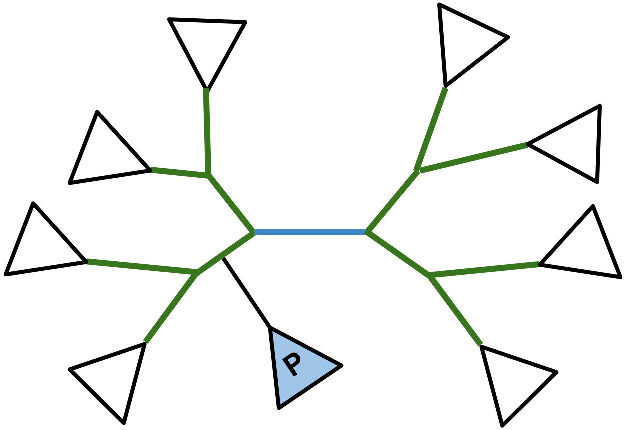


LH_0

List of LH scores:
- LH_0

Fast SPR round

- In **Fast SPR** round, RAxML-NG computes the likelihood using the **branch lengths** of the **initial tree** topology



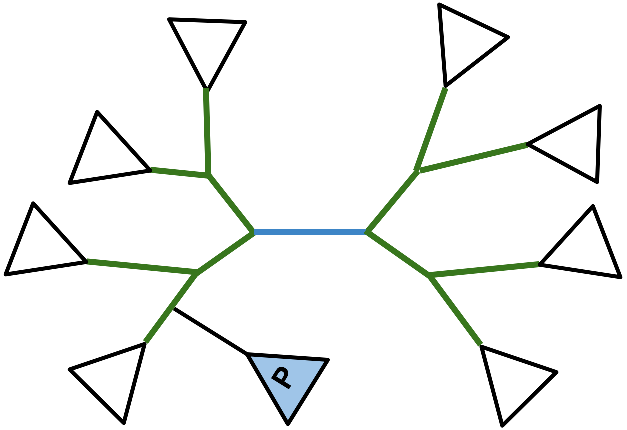
LH_1

List of LH scores:

- LH_0
- LH_1

Fast SPR round

- In **Fast SPR** round, RAxML-NG computes the likelihood using the **branch lengths** of the **initial tree** topology



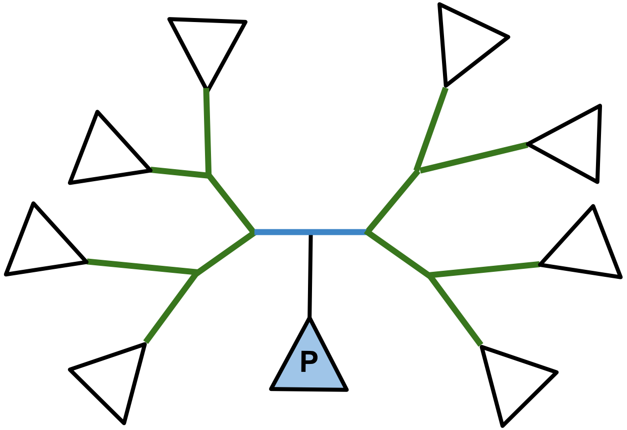
LH_2

List of LH scores:
- LH_0
- LH_1
- LH_2

And so on ….

Slow SPR round

- In **Slow SPR** round, RAxML-NG computes the likelihood by **optimizing** the **three branch lengths** around the **insertion node**



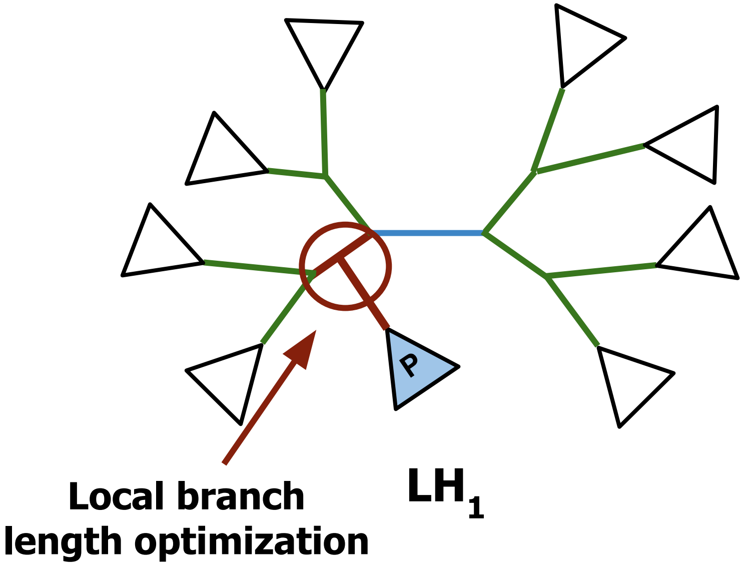
LH_0

List of LH scores:

- LH_0

Slow SPR round

- In **Slow SPR** round, RAxML-NG computes the likelihood using the **branch lengths** of the **initial tree** topology



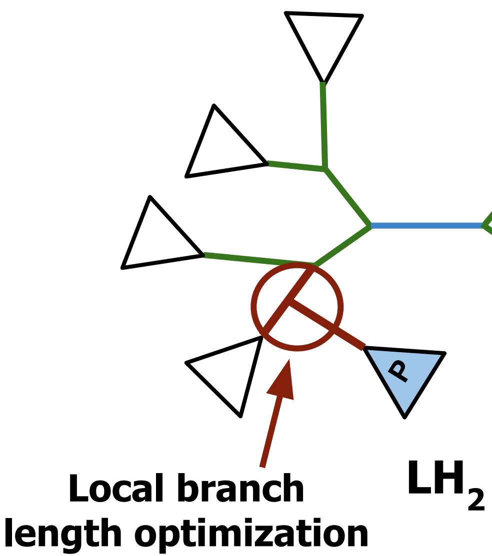
Local branch length optimization

LH_1

List of LH scores:
- LH_0
- LH_1

Slow SPR round

- In **Slow SPR** round, RAxML-NG computes the likelihood using the **branch lengths** of the **initial tree** topology



List of LH scores:
- LH_0
- LH_1
- LH_2

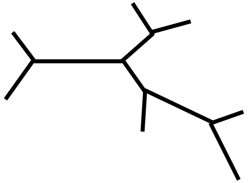
And so on ….

Slow SPR round

- Also, in **Slow SPR** round, RAxML-NG stores the **top-20** best-scoring **topologies** on a list

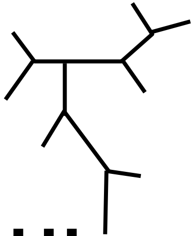
Top-20 topologies:

1.



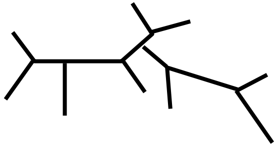
LH_1

2.



LH_2

3.

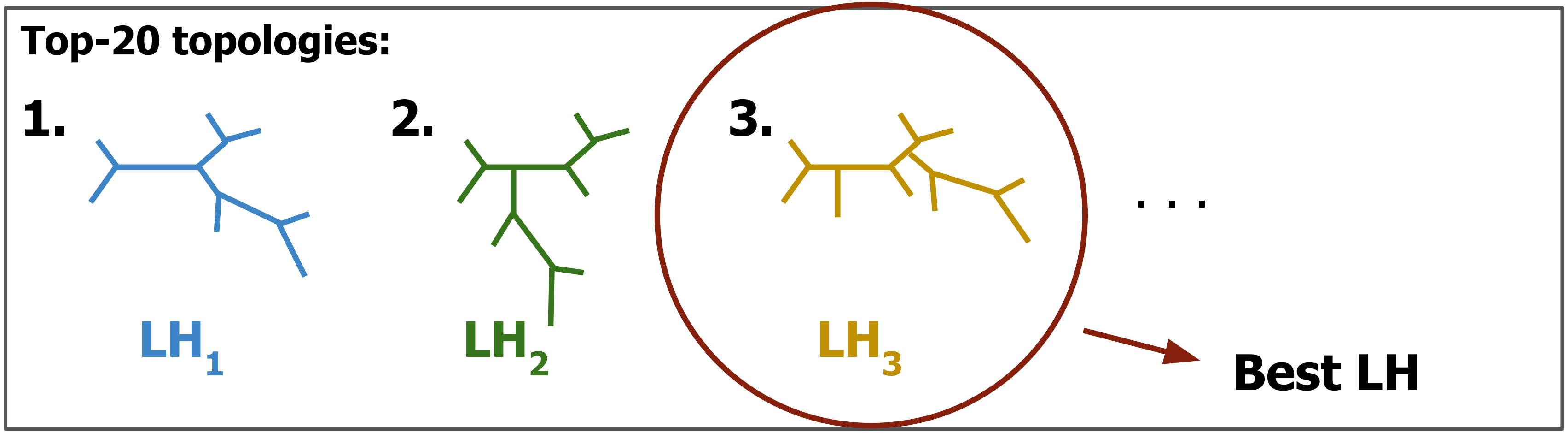


LH_3

. . .

Slow SPR round

- Also, in **Slow SPR** round, RAxML-NG stores the **top-20** best-scoring **topologies** on a list
- At the end of the round, **branch lengths** are **optimized** in all top-20 topologies to check whether any of them gives a **higher LH score**



Top-20 topologies:

1. LH_1

2. LH_2

3. LH_3

. . .

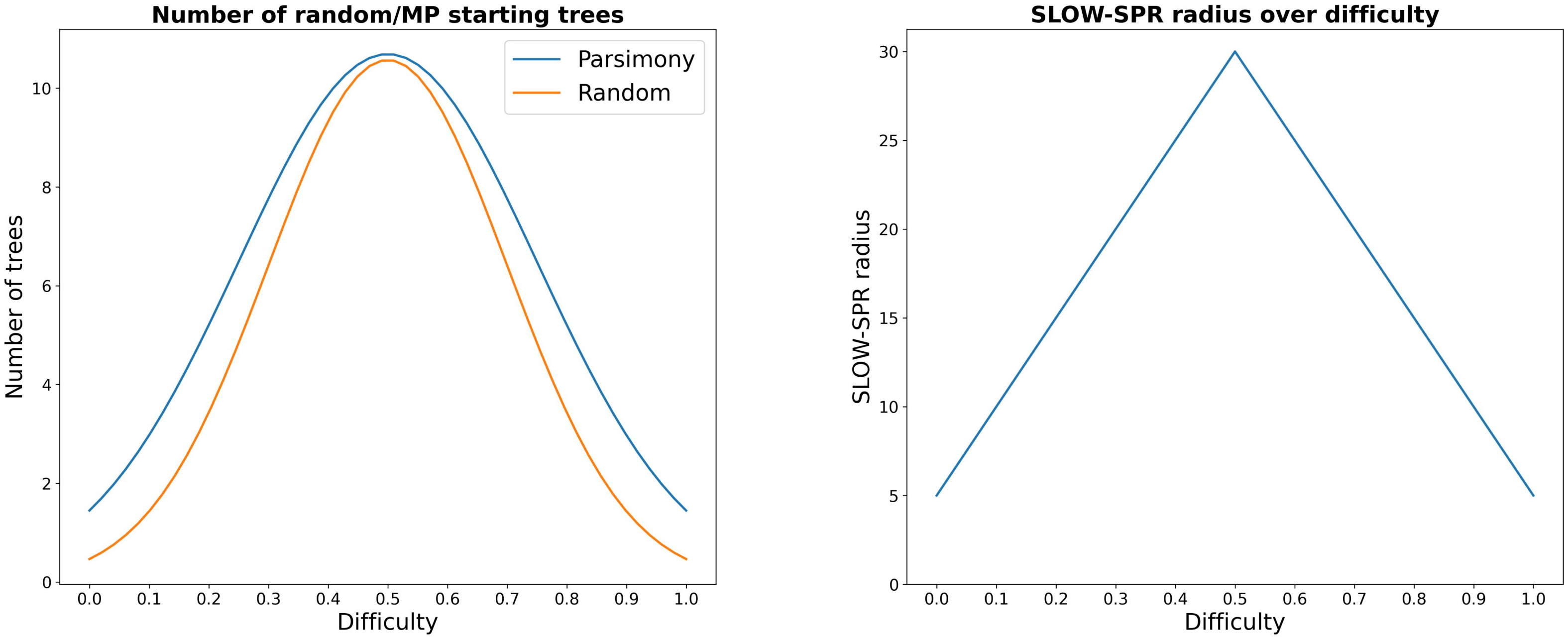
Best LH

Adaptive heuristic in RAxML-NG

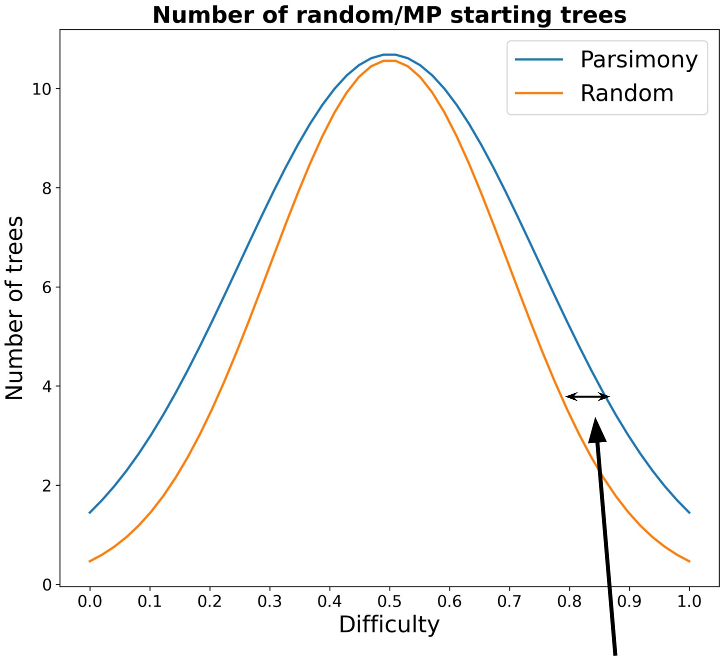
- We **modify** (with respect to difficulty) :
    - the **number** of **ML tree searches**
    - the radius in Fast/Slow SPR rounds

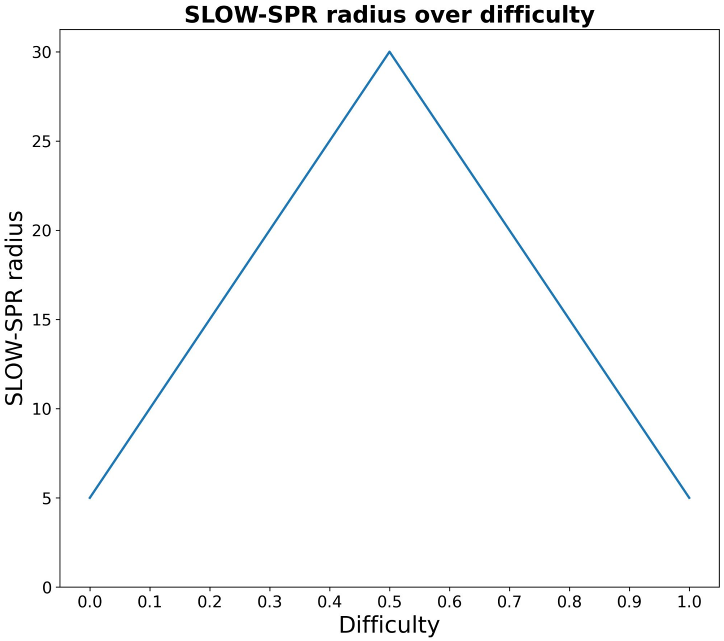
- We further **introduce**:
    - The **NNI moves**

Adaptive heuristic in RAxML-NG



Adaptive heuristic in RAxML-NG

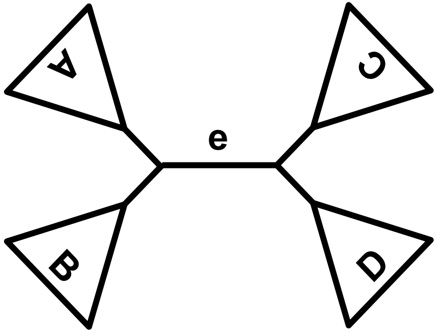




The MP starting trees curve is wider, due to the observation made by Morel *et al.*

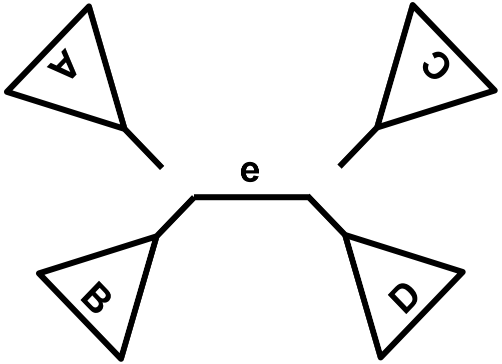
Nearest Neighbor Interchange (NNI) move

- Around a central branch *e*



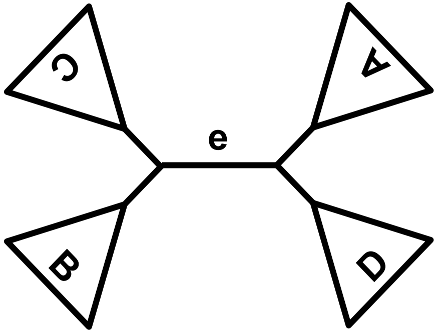
Nearest Neighbor Interchange (NNI) move

- Around a **central** inner **branch *e***
- Pick one subtree from each side



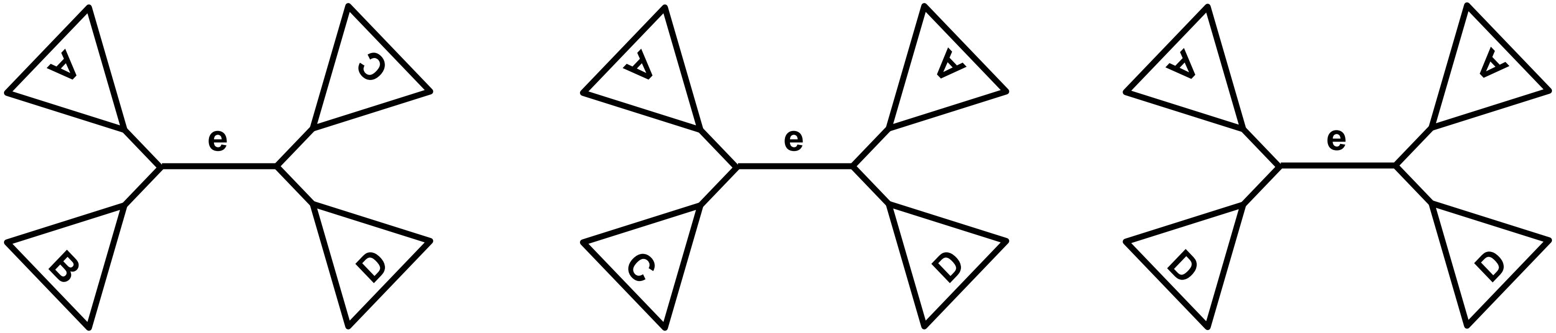
Nearest Neighbor Interchange (NNI) move

- Around a **central** inner **branch *e***
- Pick one subtree from each side
- And **interchange** them



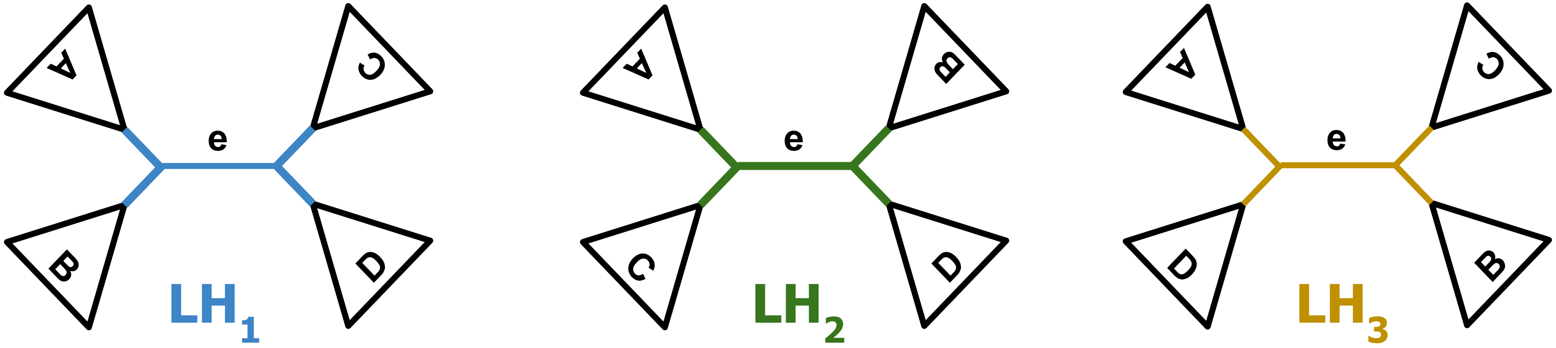
NNI round

- Sequence of NNI moves (hill-climbing/greedy heuristic)
- For each inner branch *e*, we check all three neighboring NNI topologies



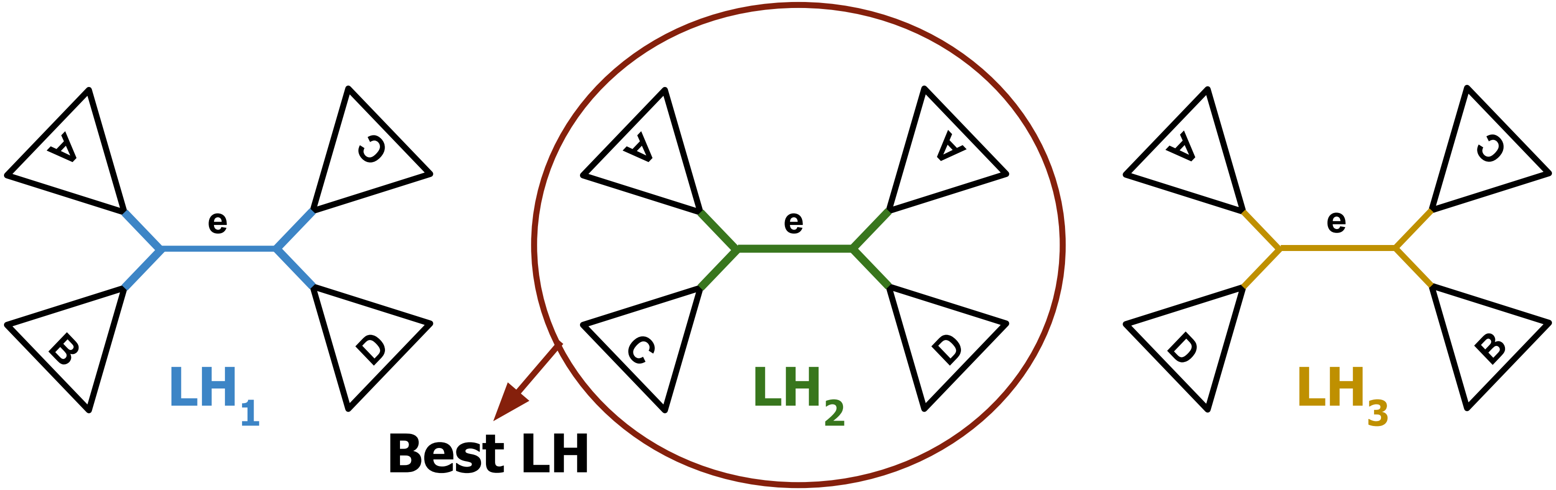
NNI round

- Sequence of NNI moves (hill-climbing/greedy heuristic)
- For each inner branch *e*, we check all three neighboring NNI topologies
- We optimize the five central branches and calculate the likelihoods



NNI round

- Sequence of NNI moves (hill-climbing/greedy heuristic)
- For each inner branch *e*, we check all three neighboring NNI topologies
- We optimize the five central branches and calculate the likelihoods
- Accept the best-scoring topology. Proceed to adjacent inner branches



Adaptive heuristic in RAxML-NG

- We **alternate** between **SPR** and **NNI rounds**
- The alternation between SPR and NNI rounds achieves faster likelihood convergence, while maintaining the accuracy
- On **easy** and **difficult** datasets, we **begin** with an **NNI round**, since the probability of rapid convergence on those datasets is comparatively high

Results

Experimental setup

Experimental setup

- ~10,000 empirical - 5,000 simulated datasets
- Empirical data from TreeBASE [DNA/AA, single/multi partitioned]
- Simulated data from Höhler *et al.* (RAxML Grove reference trees)

Experimental setup

- ~10,000 empirical - 5,000 simulated datasets
- Empirical data from TreeBASE [DNA/AA, single/multi partitioned]
- Simulated data from Höhler *et al.* (RAxML Grove reference trees)

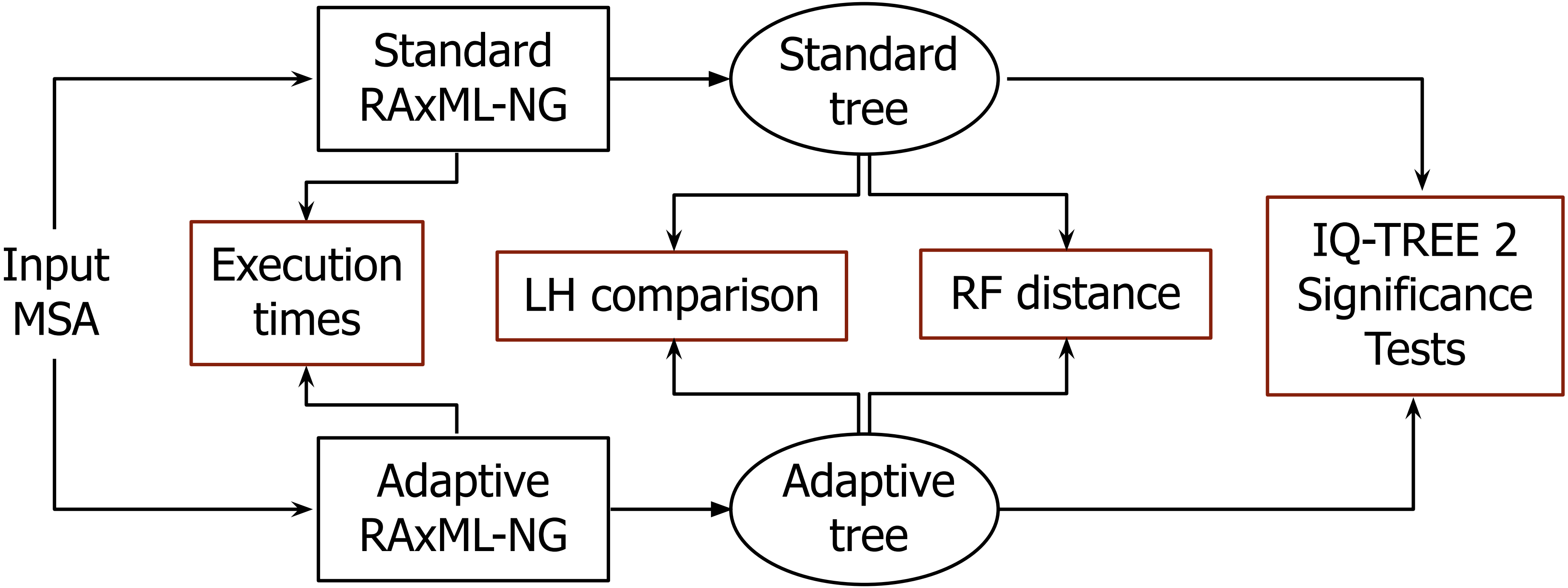
10,000 empirical
5,000 simulated

Filtering

9,515 empirical
5,000 simulated

We present the results
from filtered datasets

Experimental pipeline



Input
MSA

Standard
RAxML-NG

Adaptive
RAxML-NG

Standard
tree

Adaptive
tree

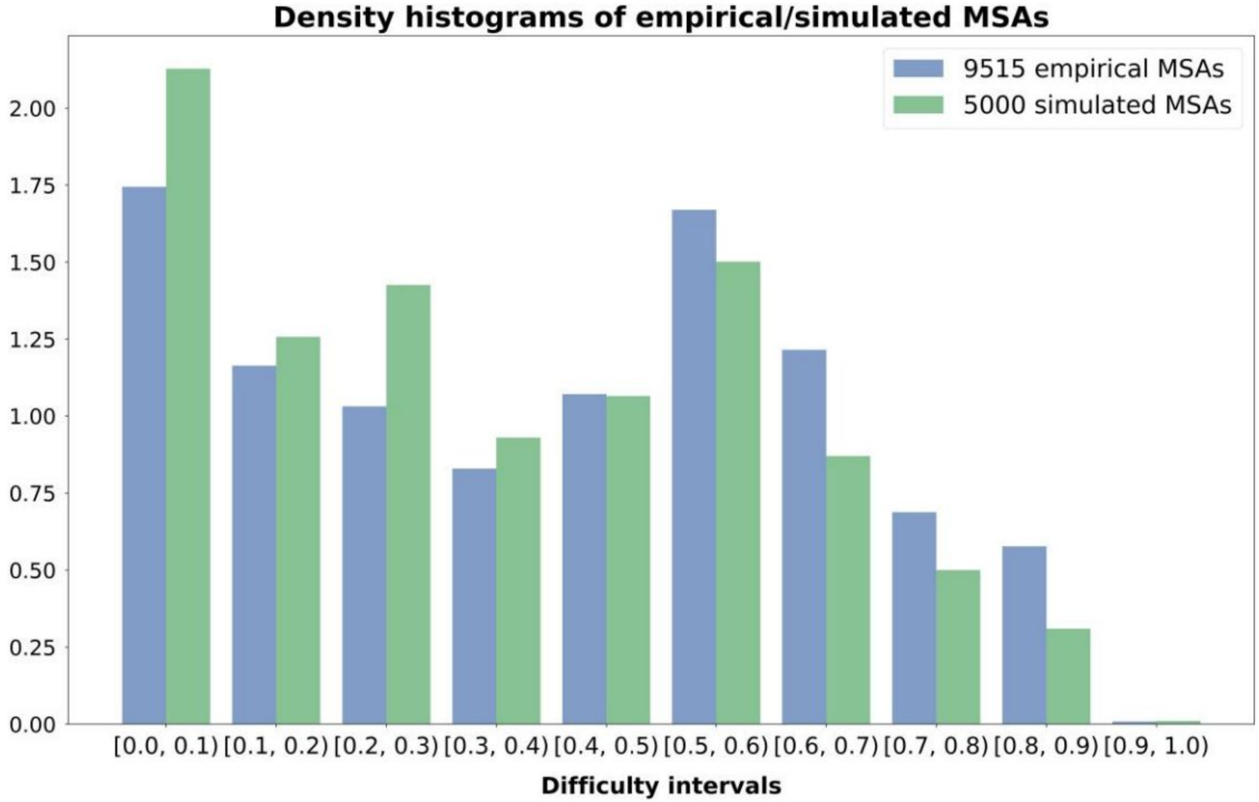
Execution
times

LH comparison

RF distance

IQ-TREE 2
Significance
Tests

Difficulty score distribution



Likelihood score comparison

- Log-likelihood difference metric (**LD**):

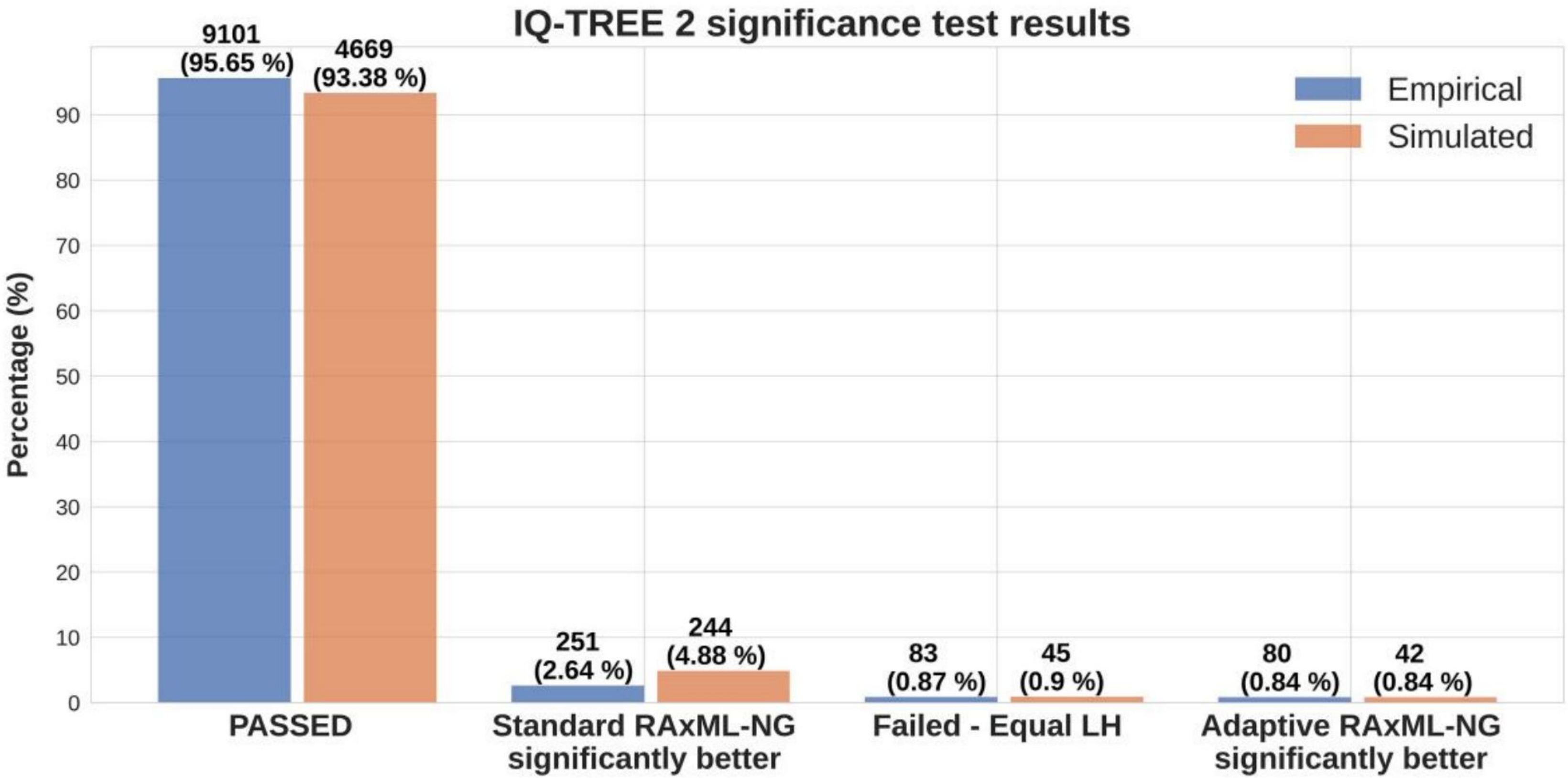
$$\boxed{LD \;=\; LH_S - LH_A}$$

- Relative log-likelihood difference metric (**RLD**):

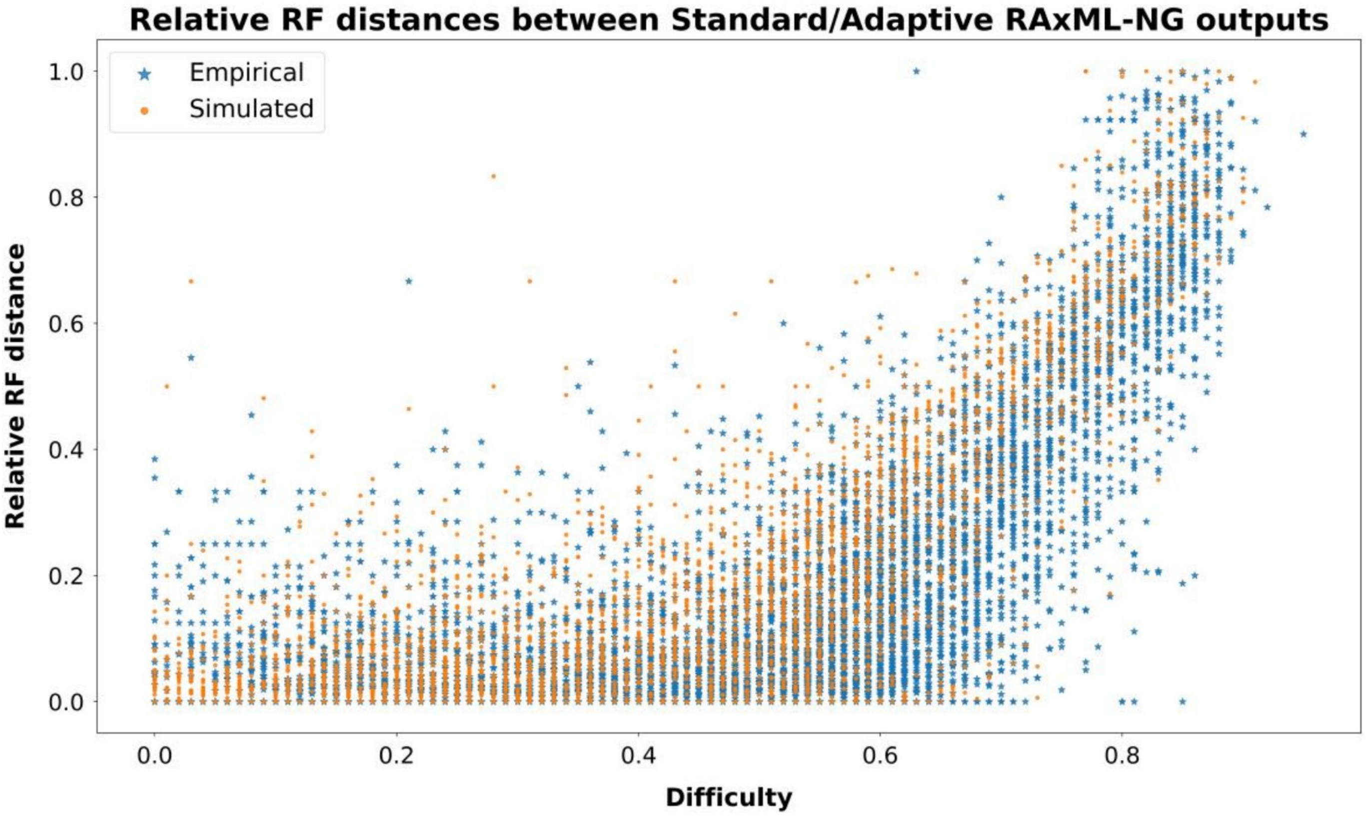
$$\boxed{RLD = \frac{LH_S - LH_A}{|LH_S|}}$$

- In cases the adaptive tree has higher LH, LD<0 and RLD<0
- In **~98%** of empirical/simulated data, **LD < 2 LHU**
- In ~**99%** of the cases (on empirical/simulated data) **RLD<10^{-3}** (0.1%), while in **all** cases, **RLD<10^{-2}** (1%)

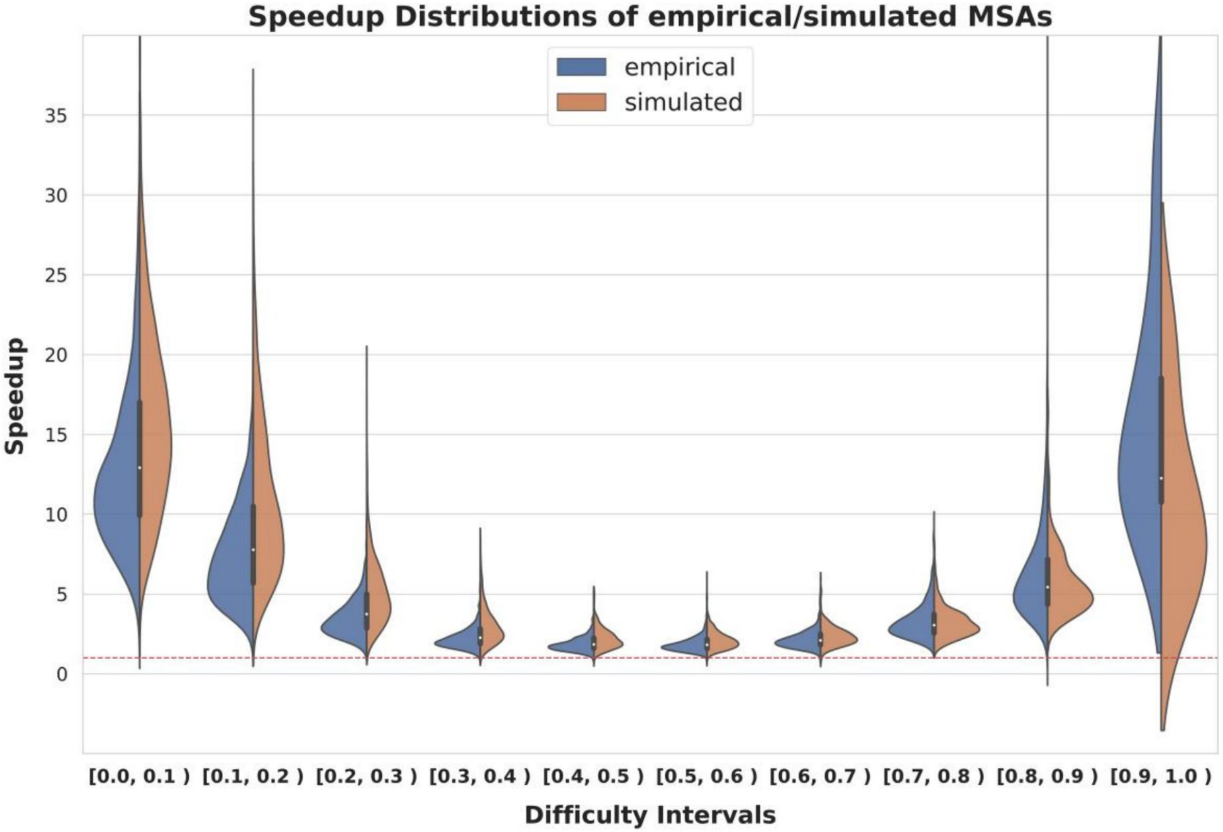
IQ-TREE 2 significance tests



Relative RF distances



Speedups


Speedup Distributions of empirical/simulated MSAs
empirical
simulated
Speedup
35
30
25
20
15
10
5
0
[0.0, 0.1)
[0.1, 0.2)
[0.2, 0.3)
[0.3, 0.4)
[0.4, 0.5)
[0.5, 0.6)
[0.6, 0.7)
[0.7, 0.8)
[0.8, 0.9)
[0.9, 1.0)
Difficulty Intervals

Future work

Future work

- Concerning our future work, we intend to:
    - Play a little bit with adaptive RAxML-NG heuristic
    - Experiment with **statistical tests** for **early termination** of the tree search
    - Consider the **sequence error rate** during the inference

Availability

- Adaptive RAxML-NG is currently available under GNU GPL:
  - https://github.com/togkousa/raxml-ng/tree/adaptive

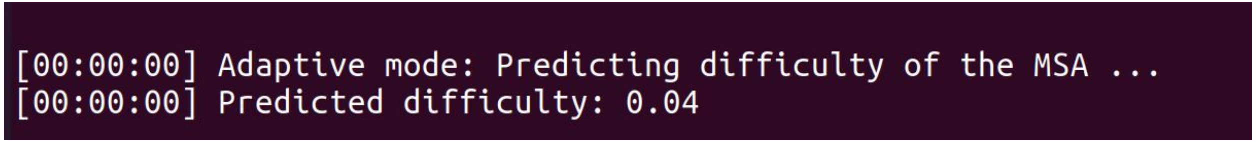
- Manuscript:
  - Togkousidis, A., Kozlov, O. M., Haag, J., Höhler, D., & Stamatakis, A. (2023). Adaptive RAxML-NG: Accelerating Phylogenetic Inference under Maximum Likelihood using Dataset Difficulty. Molecular Biology and Evolution, 40(10), msad227.

Demo

- In order to execute the adaptive version of RAxML-NG, the user needs to invoke the `--adaptive` option in the command line:

```
./bin/raxml-ng --adaptive --threads 4 --msa ../seminar-datasets/easy/alignment.phy --model LG --seed 0 --nofiles
```

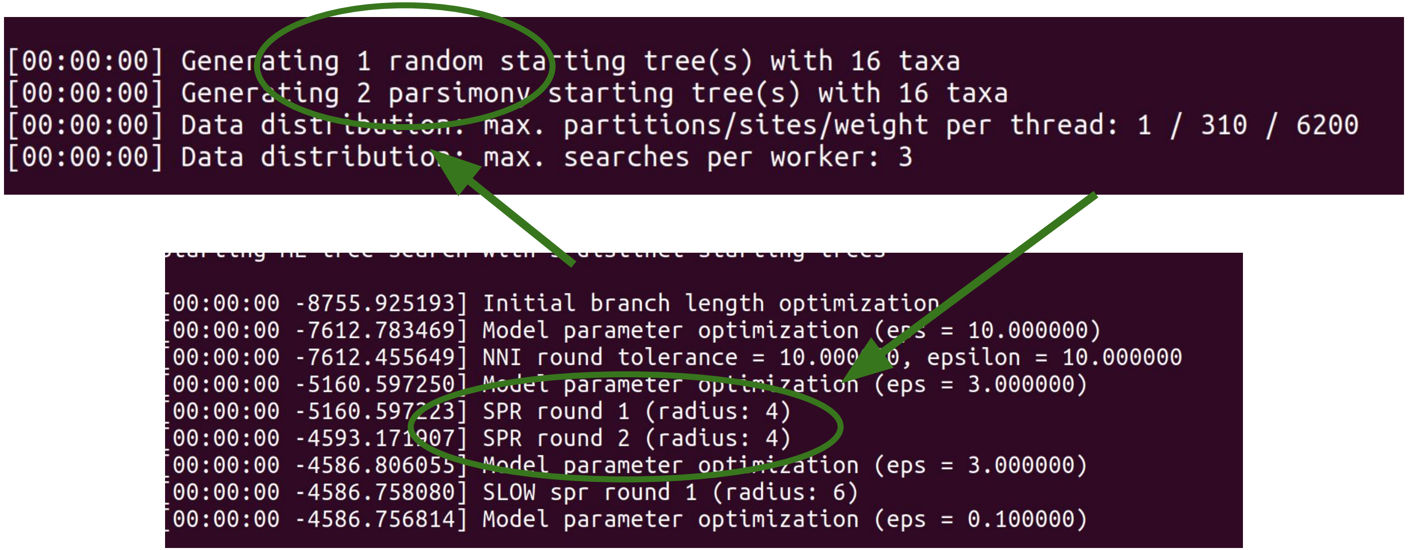
- The predicted difficulty is printed out directly at the beginning of the execution:



```
[00:00:00] Adaptive mode: Predicting difficulty of the MSA ...
[00:00:00] Predicted difficulty: 0.04
```

Demo

- The number of independent tree searches and the SPR radius are modified according to the predicted difficulty score:

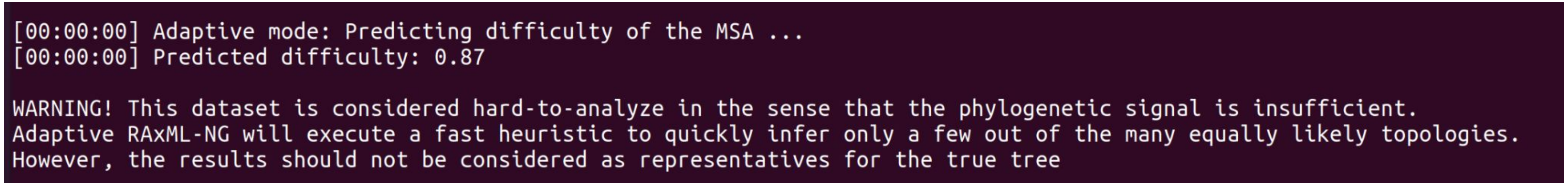


```
[00:00:00] Generating 1 random starting tree(s) with 16 taxa
[00:00:00] Generating 2 parsimony starting tree(s) with 16 taxa
[00:00:00] Data distribution: max. partitions/sites/weight per thread: 1 / 310 / 6200
[00:00:00] Data distribution: max. searches per worker: 3
```

```
[00:00:00 -8755.925193] Initial branch length optimization
[00:00:00 -7612.783469] Model parameter optimization (eps = 10.000000)
[00:00:00 -7612.455649] NNI round tolerance = 10.000000, epsilon = 10.000000
[00:00:00 -5160.597250] Model parameter optimization (eps = 3.000000)
[00:00:00 -5160.597223] SPR round 1 (radius: 4)
[00:00:00 -4593.171907] SPR round 2 (radius: 4)
[00:00:00 -4586.806055] Model parameter optimization (eps = 3.000000)
[00:00:00 -4586.758080] SLOW spr round 1 (radius: 6)
[00:00:00 -4586.756814] Model parameter optimization (eps = 0.100000)
```

Demo

- In case the predicted difficulty score is higher than 0.7, a warning is printed to inform the user that the phylogenetic signal in the input MSA is insufficient



```
[00:00:00] Adaptive mode: Predicting difficulty of the MSA ...
[00:00:00] Predicted difficulty: 0.87

WARNING! This dataset is considered hard-to-analyze in the sense that the phylogenetic signal is insufficient.
Adaptive RAxML-NG will execute a fast heuristic to quickly infer only a few out of the many equally likely topologies.
However, the results should not be considered as representatives for the true tree
```

Thank you

BACK UP
SLIDES

Difficulty prediction (Haag *et al.*)

- Recently, Haag *et al.* proposed a definition for the difficulty of analyzing an MSA
- The **difficulty score** essentially **quantifies** the amount of phylogenetic **signal** on a given MSA
- Difficulty score is a real number between **0.0 (easy MSAs)** and **1.0 (difficult MSAs)**
- They also implemented and published **Pythia**, a Random-Forest Regressor able to accurately predict the difficulty score of an MSA

Summary of their work

- Quantified difficulty on ~3,000 empirical MSAs from TreeBASE. For each dataset they:
    - conducted 100 ML RAxML-NG tree searches
    - extracted the *plausible tree set (PST)*

$$\boxed{Difficulty = \frac{1}{5}\left(RF_{all} + RF_{pl} + \frac{N_{all}^{*}}{N_{all}} + \frac{N_{pl}^{*}}{N_{pl}} + \left(1 - \frac{N_{pl}}{N_{all}}\right)\right)}$$

- Trained and tested Pythia on the inferred difficulty labels

Using Pythia

- Pythia uses **eight features** to represent each dataset as a data point
- Six of them are dataset’s attributes (fast-to-compute)
- For the remaining two features, Pythia conducts **100 MP tree inferences**
- The two remaining features are attributes of the MP output tree set
- **Predicting the difficulty** is on average **five times faster than a single ML tree inference** in RAxML-NG

Beyond Pythia

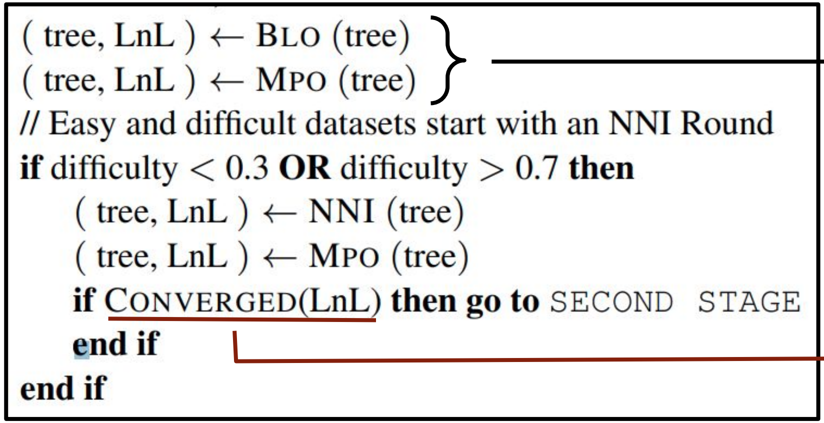
- Based on the difficulty score, one can classify MSAs into **easy**, **intermediate** and **difficult** (hard/hopeless) to analyze.
- Our suggestion:
  - Easy datasets (Difficulty < 0.3)
  - Intermediate (0.3 ≤ Difficulty ≤ 0.7)
  - Difficult datasets (Difficulty > 0.7)
- The concept of difficulty provides adequate explanation for the ambiguities arising from different tool performance-assessment studies

Our idea

- The three observations made in Höhler *et al.* study indicate that one can **modify** the **thoroughness** of the tree search **heuristic based on** the predicted **difficulty** of the MSA
- On **easy** and **difficult** datasets, **fast heuristics** perform equally well
- On easy datasets, tree searches converge rapidly
- Difficult datasets are hopeless to analyze, and thus it suffices to quickly infer only a few out of the many equally likely trees, to reduce overall execution time

Heuristic step by step

- Adaptive RAxML-NG begins with a BLO and MPO round. In case the dataset is easy or difficult, it applies an NNI round + MPO. If likelihood convergence is achieved, it proceeds directly to the second stage



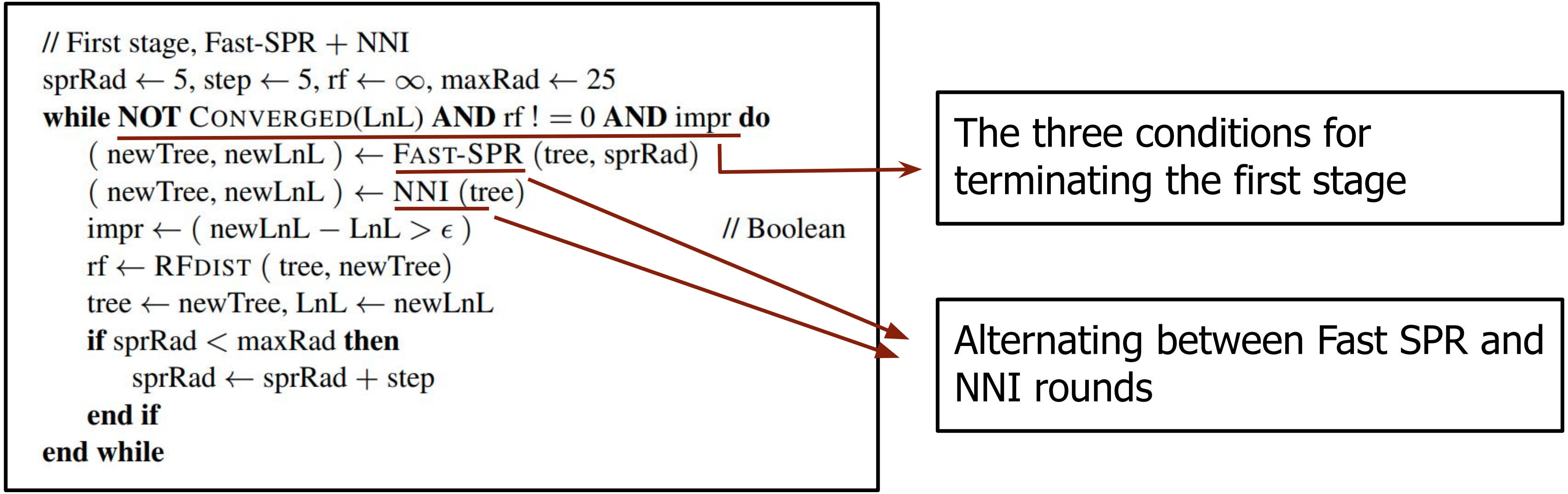
```
( tree, LnL ) $\leftarrow$ BLO (tree)
( tree, LnL ) $\leftarrow$ MPO (tree)
// Easy and difficult datasets start with an NNI Round
if difficulty $< 0.3$ OR difficulty $> 0.7$ then
    ( tree, LnL ) $\leftarrow$ NNI (tree)
    ( tree, LnL ) $\leftarrow$ MPO (tree)
    if CONVERGED(LnL) then go to SECOND STAGE
    end if
end if
```

Branch-length optimization
Model parameter optimization

Checks if likelihood convergence was achieved (1% likelihood convergence interval). If so, it skips the first stage

Heuristic step by step

- During the first stage, Fast-SPR round are alternated with NNI rounds



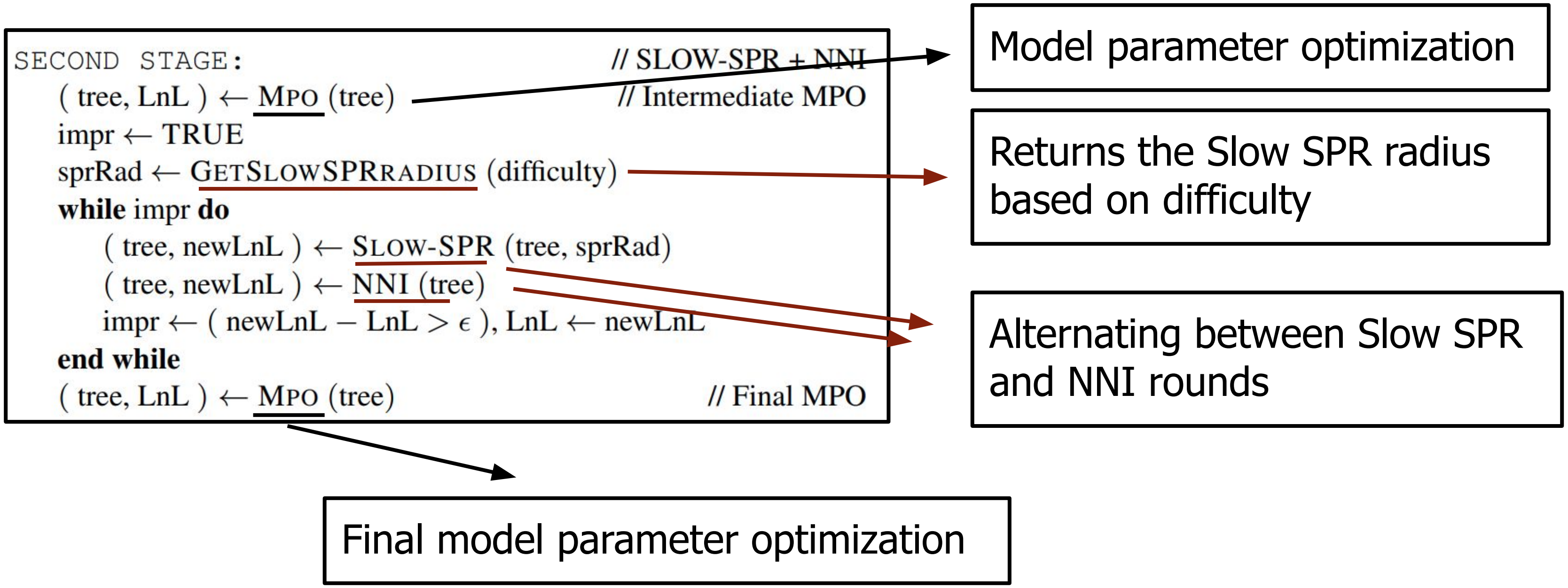
```
// First stage, Fast-SPR + NNI
sprRad ← 5, step ← 5, rf ← $\infty$, maxRad ← 25
while NOT CONVERGED(LnL) AND rf != 0 AND impr do
    ( newTree, newLnL ) ← FAST-SPR (tree, sprRad)
    ( newTree, newLnL ) ← NNI (tree)
    impr ← ( newLnL − LnL > $\epsilon$ )                // Boolean
    rf ← RFDIST ( tree, newTree)
    tree ← newTree, LnL ← newLnL
    if sprRad < maxRad then
        sprRad ← sprRad + step
    end if
end while
```

The three conditions for terminating the first stage

Alternating between Fast SPR and NNI rounds

Heuristic step by step

- During the first stage, Fast-SPR round are alternated with NNI rounds



```
SECOND STAGE:                                                // SLOW-SPR + NNI
    ( tree, LnL ) ← MPO (tree)                               // Intermediate MPO
    impr ← TRUE
    sprRad ← GETSLOWSPRRADIUS (difficulty)
    while impr do
        ( tree, newLnL ) ← SLOW-SPR (tree, sprRad)
        ( tree, newLnL ) ← NNI (tree)
        impr ← ( newLnL − LnL > ϵ ), LnL ← newLnL
    end while
    ( tree, LnL ) ← MPO (tree)                               // Final MPO
```

Model parameter optimization

Returns the Slow SPR radius based on difficulty

Alternating between Slow SPR and NNI rounds

Final model parameter optimization

Adaptive heuristic in RAxML-NG

- The heuristic is divided into **two stages**. During the **first stage**, **Fast-SPR** rounds are alternated with **NNI** rounds.
- The first stage is terminated if:
  - The likelihood improvement is less than ε *OR*
  - The RF distance between two consecutive tree topologies is 0 *OR*
  - The likelihood is less than 1% lower from the score of the best ML tree found so far from a finished tree inference (**1% likelihood convergence interval**)

Adaptive heuristic in RAxML-NG

- During the **second stage**, **Slow-SPR** rounds are alternated with **NNI** rounds.
- The second round is terminated if the likelihood improvement is less than ε
- Before and after each stage, adaptive RAxML-NG conducts Branch-Length Optimization (BLO) and Model Parameter Optimization (MPO)

Adaptive heuristic in RAxML-NG

- On **easy** and **difficult** datasets, adaptive RAxML-NG begins with an NNI round + MPO
- The probability of achieving likelihood convergence with only an NNI round is comparatively high on such datasets

Experimental setup

- We collected **10,000 empirical** MSAs from TreeBASE and used the **5,000 simulated** MSAs from Höhler *et al.* study.
- We apply some **filtering** process (which we will describe on the next slide). After filtering, we end up with **9,192 empirical** and **4,991 simulated** MSAs.
- All simulated data are single-partitoned DNA datasets
- Out of the 9,192 (final) empirical MSAs.
    - 7,769 are single-partitioned DNA
    - 614 are multi-partitioned DNA
    - 801 are single-partitioned AA
    - 8 are multi-partitioned AA

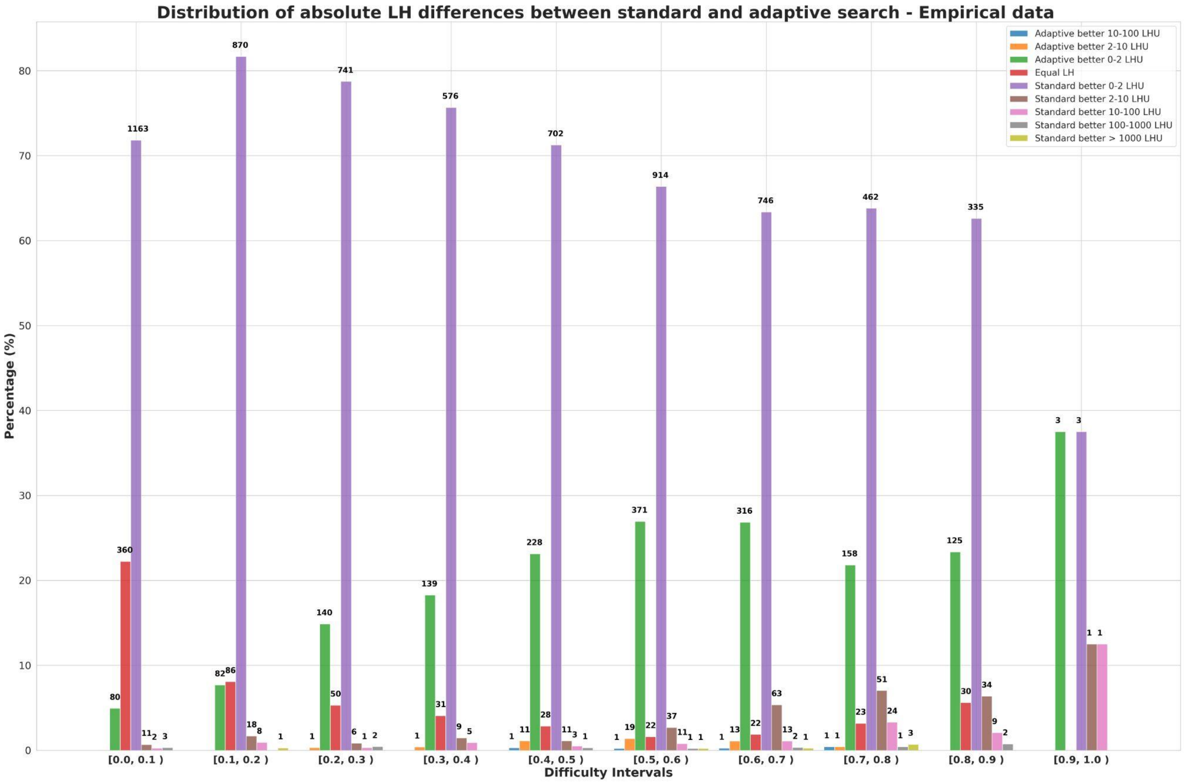
Pipeline + Filtering

- We ran both **standard** and **adaptive RAxML-NG** on each one of the datasets (sequentially, 24 hours threshold)
- We filtered out those MSAs in which either:
    - The execution of standard/adaptive RAxML-NG took longer than 24 hours *OR*
    - At least one of the RAxML-NG executions failed for whatever reason
- We ran IQ-TREE 2 significance tests (Tree Topology Tests) on all pairs of standard/adaptive output trees. Those datasets in which the execution of IQ-TREE 2 failed were also filtered out
- We end up with 9,192 empirical and 4,991 simulated MSAs

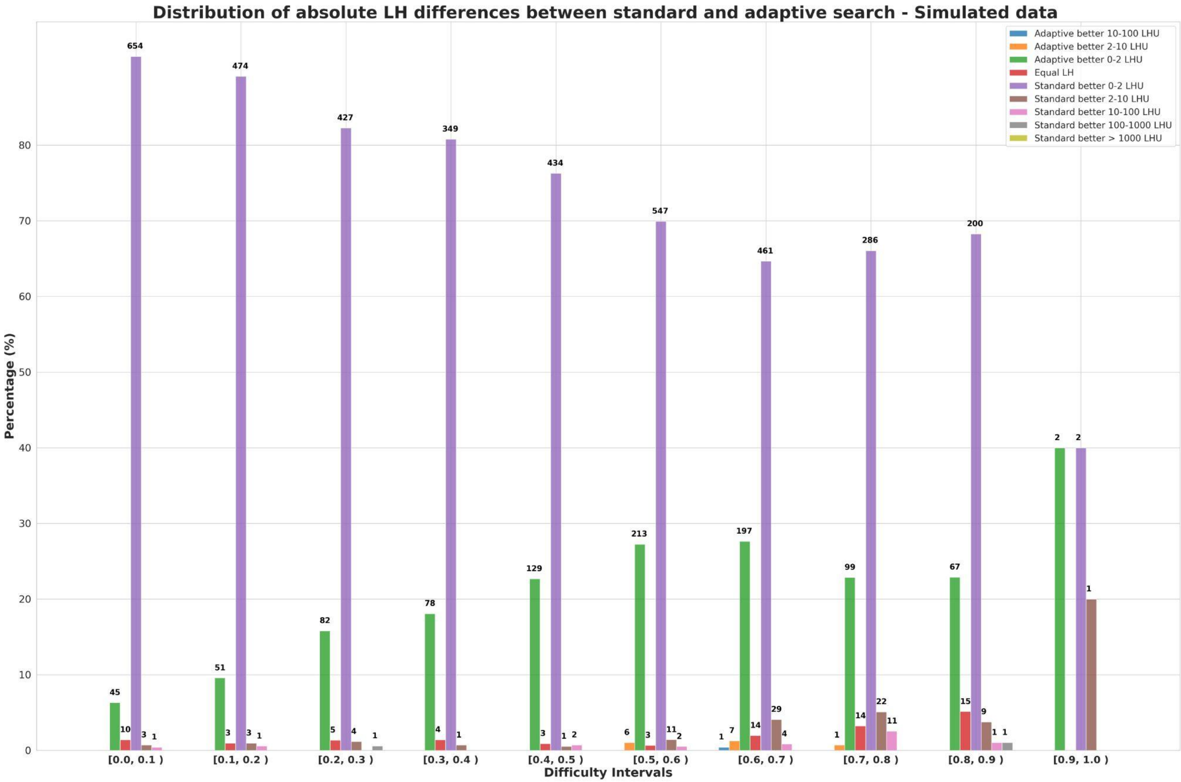
Standard/Adaptive RAxML-NG comparison

- We **compare** the two versions of RAxML-NG based on:
  - The **likelihood score** of the output tree
  - The result IQ-TREE 2 **significance tests** (We consider the two output trees to be **statistically indistinguishable** if the pair passes **all** significance tests)
  - The relative **RF-distance** between the output trees
  - The execution times (**Speedups**)

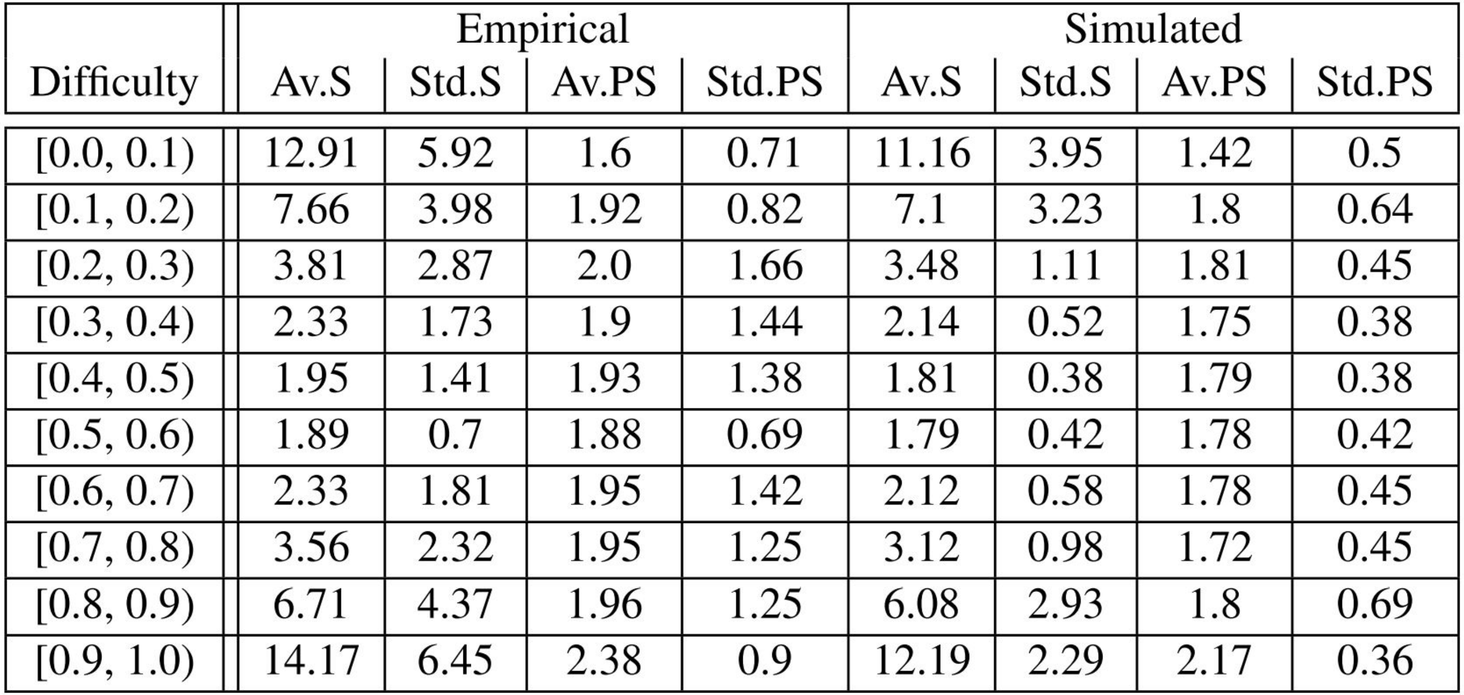
Likelihood score comparison


Distribution of absolute LH differences between standard and adaptive search - Empirical data

Likelihood score comparison



Speedups



	Empirical				Simulated			
Difficulty	Av.S	Std.S	Av.PS	Std.PS	Av.S	Std.S	Av.PS	Std.PS
[0.0, 0.1)	12.91	5.92	1.6	0.71	11.16	3.95	1.42	0.5
[0.1, 0.2)	7.66	3.98	1.92	0.82	7.1	3.23	1.8	0.64
[0.2, 0.3)	3.81	2.87	2.0	1.66	3.48	1.11	1.81	0.45
[0.3, 0.4)	2.33	1.73	1.9	1.44	2.14	0.52	1.75	0.38
[0.4, 0.5)	1.95	1.41	1.93	1.38	1.81	0.38	1.79	0.38
[0.5, 0.6)	1.89	0.7	1.88	0.69	1.79	0.42	1.78	0.42
[0.6, 0.7)	2.33	1.81	1.95	1.42	2.12	0.58	1.78	0.45
[0.7, 0.8)	3.56	2.32	1.95	1.25	3.12	0.98	1.72	0.45
[0.8, 0.9)	6.71	4.37	1.96	1.25	6.08	2.93	1.8	0.69
[0.9, 1.0)	14.17	6.45	2.38	0.9	12.19	2.29	2.17	0.36