Adaptive search heuristic for ML phylogenetic tree inference based on the predicted difficulty of dataset.

Anastasis Togkousidis, Alexandros Stamatakis
Introduction
Introduction

Input MSA → Phylogenetic Tree Inference → Binary tree
Introduction

How do we solve this difficult problem?

Input

MSA

Phylogenetic Tree Inference

Binary tree
Introduction

Input
MSA

RAxML-NG
ML inference

Binary tree
Introduction

Input MSA → RAxML-NG ML inference → Binary tree

Waiting for RAxML-NG to finish
**Introduction**

- **Input MSA** → **RAxML-NG ML inference** → **Binary tree**
- **Pythia** → **Difficulty score** → **IDEA**
Introduction

Input MSA

RAxML-NG
ML inference

Difficulty score

Pythia

Binary tree

Adaptive RAxML-NG

IDEA
Background
The concept of difficulty

- For some datasets, independent ML tree searches starting from different trees, converge to a **single - or topologically similar - tree(s)**
- This implies a **single**, well distinguishable, **globally optimal peak** on the likelihood surface, associated with a single tree topology
- We say that these datasets exhibit a “**clear phylogenetic signal**”
The concept of difficulty

- Other datasets yield **topologically highly distinct, yet equally likely trees**
- These trees are also **statistically indistinguishable** based on IQ-TREE 2 significance tests
- This implies a “**rugged**” likelihood surface, with **multiple local optima** associated with contradicting topologies
Comments

- These two examples demonstrate two extreme-case dataset types
- There is a whole spectrum of in-between dataset cases (for example datasets with multiple local optima, albeit only a small proportion of them being significantly better)
- This diverse behavior is essentially quantified by pythia
Difficulty prediction (Haag et al.)

Input
MSA

→

Pythia

→

0.0
Easy

0.5
Intermediate

1.0
Difficult
Difficulty prediction (Haag et al.)

Input MSA → Pythia

Difficulty prediction (Haag et al.)
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.

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy MSAs</td>
<td>Similarly in terms of likelihood score and topological accuracy</td>
</tr>
<tr>
<td>Intermediate</td>
<td>RAxML-NG and IQ-TREE 2 infer significantly better trees</td>
</tr>
<tr>
<td>Difficult MSAs</td>
<td>All tools perform similarly in terms of likelihood score</td>
</tr>
</tbody>
</table>

On difficult MSAs, all tools perform similarly but only in terms of likelihood score.
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](image)
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**
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
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
SPR round

- Sequence of SPR moves
- Hill-climbing - greedy heuristic
- Two types of SPRs round in 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**

- 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

  ![Diagram](Diagram.png)

  - 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

And so on ....
Fast SPR round

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

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.

List of LH scores:
- LH_0
- LH_1

LH_1
Fast SPR round

- In Fast 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

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

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.

List of LH scores:
- $LH_0$
- $LH_1$

Local branch length optimization $LH_1$
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. LH₁
2. LH₂
3. LH₃

... Best LH
Adaptive heuristic in RAxML-NG

● We modify (with respect to difficulty):  
  ○ the number of ML tree searches  
  ○ the radius in Slow SPR round

● We further introduce:  
  ○ The NNI moves
Adaptive heuristic in RAxML-NG

Number of random/MP starting trees

Difficulty

SLOW-SPR radius over difficulty

Difficulty
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)

We present the results from filtered datasets
Experimental pipeline
Density histograms of empirical/simulated MSAs over 10 difficulty intervals

Difficulty score distribution
Likelihood score comparison

- Log-likelihood difference metric ($LD$):
  \[ LD = LH_S - LH_A \]

- Relative log-likelihood difference metric ($RLD$):
  \[ RLD = \frac{LH_S - LH_A}{|LH_S|} \]

- 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 - 10 difficulty intervals**

- **empirical**
- **simulated**

**Difficulty Intervals**

- [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]
Future work
Future work

- Concerning our future work, we intend to:
  - More focus on heuristics
  - Experiment with statistical tests for early termination of the tree search
  - Efficient parallelization of adaptive RAxML-NG
  - Insert checkpoints
  - Import it into RAxML-NG
For now

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

- Manuscript under preparation:
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)*

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

- 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) ← BLO (tree)  
  (tree, LnL) ← MPO (tree)  
}  // Easy and difficult datasets start with an NNI Round
if difficulty < 0.3 OR difficulty > 0.7 then
  (tree, LnL) ← NNI (tree)
  (tree, LnL) ← 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

```plaintext
// First stage, Fast-SPR + NNI
sprRad ← 5, step ← 5, rf ← ∞, 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 > ε )  // 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 > \epsilon), 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 $\varepsilon$ 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 $\varepsilon$.
- 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-partitioned 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**)
Distribution of absolute LH differences between standard and adaptive search - Empirical data

Likelihood score comparison

76
Likelihood score comparison

Distribution of absolute LH differences between standard and adaptive search - Simulated data
## Speedups

<table>
<thead>
<tr>
<th>Difficulty</th>
<th>Empirical</th>
<th>Simulated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Av.S</td>
<td>Std.S</td>
</tr>
<tr>
<td>[0.0, 0.1)</td>
<td>12.91</td>
<td>5.92</td>
</tr>
<tr>
<td>[0.1, 0.2)</td>
<td>7.66</td>
<td>3.98</td>
</tr>
<tr>
<td>[0.2, 0.3)</td>
<td>3.81</td>
<td>2.87</td>
</tr>
<tr>
<td>[0.3, 0.4)</td>
<td>2.33</td>
<td>1.73</td>
</tr>
<tr>
<td>[0.4, 0.5)</td>
<td>1.95</td>
<td>1.41</td>
</tr>
<tr>
<td>[0.5, 0.6)</td>
<td>1.89</td>
<td>0.7</td>
</tr>
<tr>
<td>[0.6, 0.7)</td>
<td>2.33</td>
<td>1.81</td>
</tr>
<tr>
<td>[0.7, 0.8)</td>
<td>3.56</td>
<td>2.32</td>
</tr>
<tr>
<td>[0.8, 0.9)</td>
<td>6.71</td>
<td>4.37</td>
</tr>
<tr>
<td>[0.9, 1.0)</td>
<td>14.17</td>
<td>6.45</td>
</tr>
</tbody>
</table>