SCAMPP and BSCAMPP

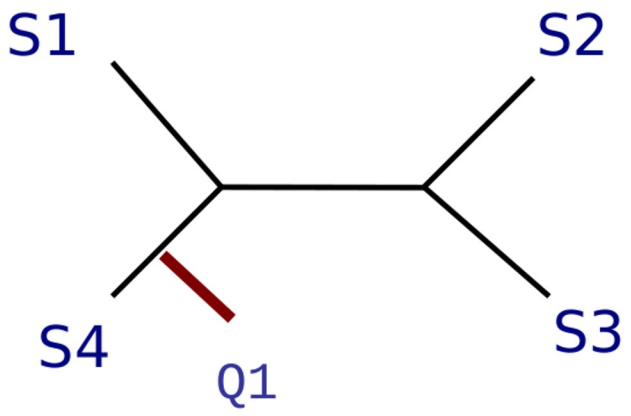
Presenter: Chengze Shen
Siebel School of Computing and Data Science
University of Illinois Urbana-Champaign

Building large trees

- NP-hard optimization
  - Maximum-likelihood tree estimation methods are accurate but costly
- Streaming data
  - New data come in → re-estimating the phylogenetic tree is costly
  - **Solution:** we directly add new data to the existing tree
    - phylogenetic placement methods

Phylogenetic placement

```
S1  = -AGGCTATCACCTGACCTCCA-AA
S2  = TAG-CTATCAC--GACCGC--GCA
S3  = TAG-CT-------GACCGC--GCT
S4  = TAC----TCAC--GACCGACAGCT
Q1  = --------T-A--AAAC--------
```



Phylogenetic placement

- Maximum likelihood methods (e.g., pplacer and EPA-ng)
- Distance-based (e.g., APPLES-2)
- Alignment-free (e.g., App-SpaM)
    - Does not require any alignment
- Best accuracy from pplacer and EPA-ng
    i. Runtime and memory issues when the tree is very large
    ii. Numerics for large trees (negative infinity values)

SCAMPP [1] and BSCAMPP [2] are designed to scale pplacer and EPA-ng to large backbone trees, with high accuracy and reduced runtime

[1] **Wedell, Eleanor, Yirong Cai, and Tandy Warnow.** "SCAMPP: Scaling Alignment-Based Phylogenetic Placement to Large Trees." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2022, 1–1. https://doi.org/10.1109/TCBB.2022.3170386.

[2] **Wedell, Eleanor, Chengze Shen, and Tandy Warnow.** "BSCAMPP: Batch-Scaled Phylogenetic Placement on Large Trees." *IEEE Transactions on Computational Biology and Bioinformatics*, 2025, 1–14. https://doi.org/10.1109/TCBBIO.2025.3562281.

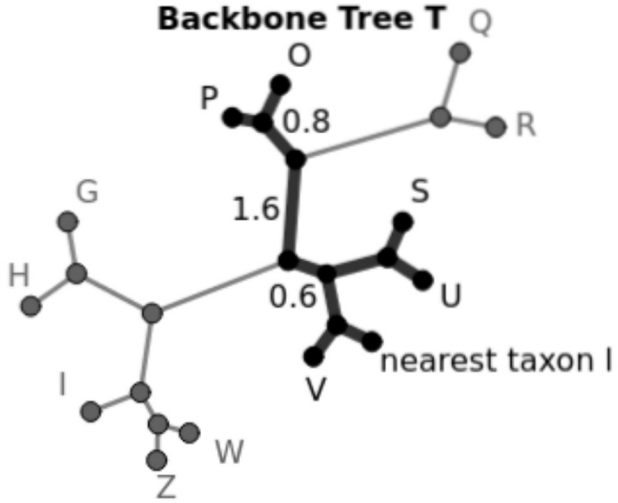
SCAMPP/BSCAMPP for phylogenetic placement

- **Divide the placement problem into multiple sub-problems**, enabling better scalability.

method	description
SCAMPP	creates a subproblem for *each new sequence*.
BSCAMPP	creates a subproblem for *a group of new sequences*.
Both	solve each subproblem with pplacer or EPA-ng.

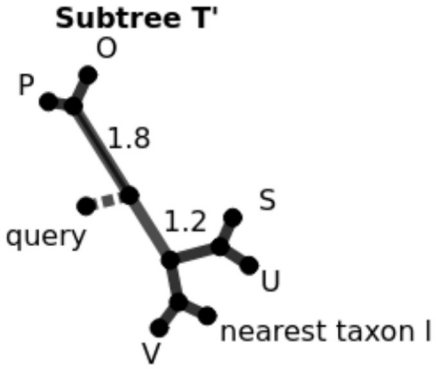
SCAMPP pipeline

Stage 1



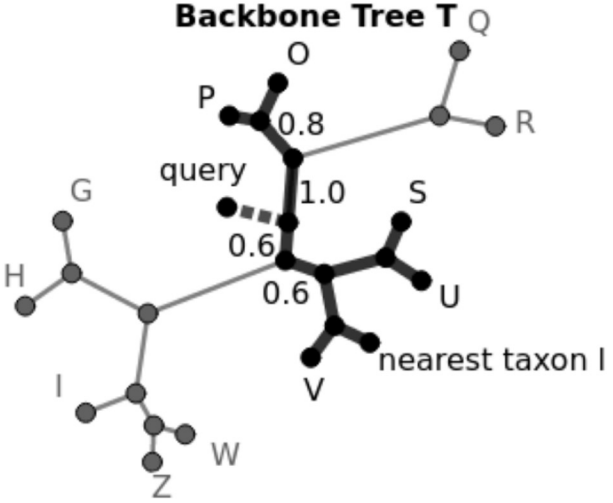
A new sequence X is assigned to its closest leaf

Stage 2



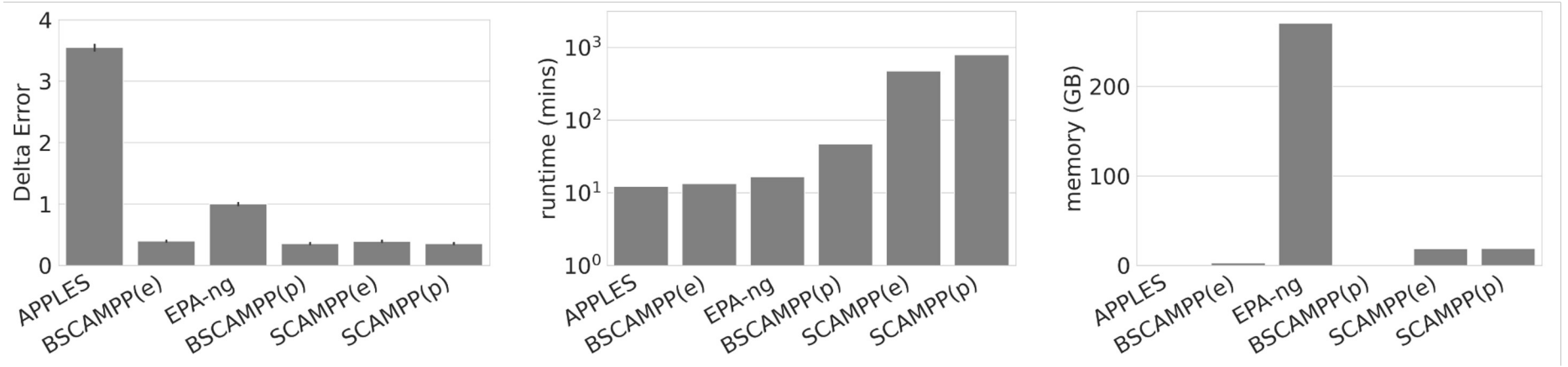
Subtree extracted from the selected leaf; X is placed to the subtree.

Stage 3



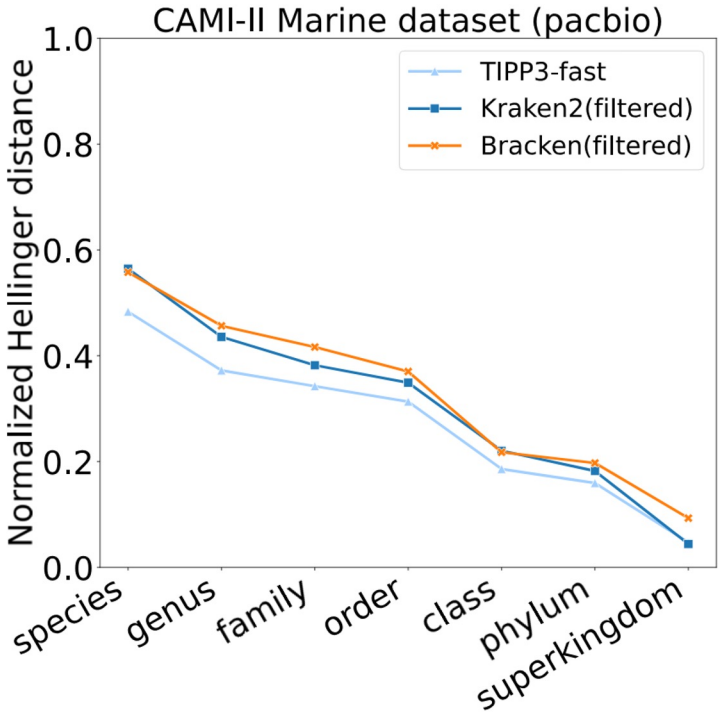
Placement of X transits to the backbone tree.

Comparison to other placement methods



> Placing 20,000 fragmentary new sequences into a 180,000-leaf phylogenetic tree.
> BSCAMPP(e) denotes that it is run with EPA-ng ("(p)" for pplacer).
(left) placement error, lower is better
(middle) runtime usage
(right) memory usage

Impact of using BSCAMPP - abundance profiling



- TIPP3-fast [1] uses BSCAMPP to place metagenomic reads onto taxonomic trees
- Higher profiling accuracy compared to Kraken2 and Bracken.

[1] Shen, Chengze, Eleanor Wedell, Mihai Pop, and Tandy Warnow. “TIPP3 and TIPP3-Fast: Improved Abundance Profiling in Metagenomics.” *PLOS Computational Biology* 21, no. 4 (April 4, 2025): e1012593. https://doi.org/10.1371/journal.pcbi.1012593.

SCAMPP/BSCAMPP code

BSCAMPP and SCAMPP - Two Scalable Phylogenetic Placement Methods and Frameworks

pypi v1.0.7 | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | build passing | license GPL | CHANGELOG

Table of Contents

1. Overview
2. Installation
3. Usage
4. Example Code and Data

Overview

- **Inputs**
  - i. Reference tree to place sequences into.
  - ii. Alignment of reference sequences (protein or nucleotide).
  - iii. Alignment of query sequences (can be combined with ii.).
  - iv. Tree info file.
    - (EPA-ng as base method), RAxML-ng info file, typically with suffix `.bestModel`.
    - (pplacer as base method), RAxML-ng or FastTree log file containing model parameters.
- **Output**
  - i. Placement results of query sequences in the reference tree in `.jplace` format.

- Implemented with Python and C++ (requires **cmake** installed)
- Available on GitHub and PyPI: https://github.com/ewedell/BSCAMPP

Installation and Usage: SCAMPP/BSCAMPP

```
# 1. install with pip (--user if no root access)
pip install bscampp [--user]

# 2. Four binary executables will be installed. The first time
#    running any will create a config file at
#    ~/.bscampp/main.config that resolves the links to all
#    external software (e.g., epa-ng, pplacer)

# ---- BSCAMPP functions
bscampp [-h]    # or
run_bscampp.py [-h]

# ---- SCAMPP functions
scampp  [-h]    # or
run_scampp.py
```

I will demonstrate some of the use cases in the following slide

Scenarios to use: BSCAMPP

1. Scenario 1: placing new sequences into an existing tree
    - Note: all commands require a model file that describes the input tree
        i. E.g., log file from the RAxML-ng/FastTree-2 tree estimation

```
run_bscampp.py -i [raxml best model] -t [existing tree] -a [alignment file]
```

1. Scenario 2: changing the base placement method (EPA-ng to pplacer)

```
run_bscampp.py -i [logfile from either RAxML/FastTree] -t [existing tree] -a [alignment file] --placement-method pplacer
```

Sample logging

1. Scenario 1: placing new sequences into an existing tree

```
run_bscampp.py -i [raxml best model] -t [existing tree] -a [alignment file]
```

```
[23:47:29] pipeline.py (line 34):  WARNING: Initializing ProcessPoolExecutor...
[23:47:29] functions.py (line 89):     INFO: Reading in input data...
[23:47:29] functions.py (line 147):     INFO: Time to read in input data: 0.0764616369997384 seconds
[23:47:29] functions.py (line 157):     INFO: Computing closest leaves for query sequences...
[23:47:29] functions.py (line 222):     INFO: Time to compute closest leaves: 0.11823938100133091 seconds
[23:47:29] functions.py (line 231):     INFO: Adding query votes to the placement tree...
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 100
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 72
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 50
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 38
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 26
[23:47:29] functions.py (line 257):     INFO: queries left to assign: 25
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 22
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 17
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 11
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 9
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 5
[23:47:30] functions.py (line 257):     INFO: queries left to assign: 1
[23:47:30] functions.py (line 331):     INFO: Time to assign queries to subtrees: 0.3356298080034321 seconds
[23:47:30] functions.py (line 412):     INFO: Performing placement on each subtree with epa-ng...
[23:47:30] functions.py (line 428):     INFO: Submitting jobs for subtree placement...
[23:47:31] functions.py (line 507):     INFO: - Subtree 6/12 with 1 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 7/12 with 3 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 3/12 with 12 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 4/12 with 11 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 5/12 with 12 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 1/12 with 12 queries
[23:47:32] functions.py (line 507):     INFO: - Subtree 2/12 with 20 queries
[23:47:33] functions.py (line 507):     INFO: - Subtree 12/12 with 1 queries
[23:47:33] functions.py (line 507):     INFO: - Subtree 8/12 with 6 queries
[23:47:33] functions.py (line 507):     INFO: - Subtree 11/12 with 3 queries
[23:47:33] functions.py (line 507):     INFO: - Subtree 10/12 with 5 queries
[23:47:33] functions.py (line 507):     INFO: - Subtree 9/12 with 14 queries
[23:47:33] functions.py (line 584):     INFO: Final number of subtrees used: 12
[23:47:33] functions.py (line 594):     INFO: Time to place queries to subtrees: 3.7278890250017866 seconds
[23:47:33] functions.py (line 603):     INFO: Writing aggregated placements to local...
[23:47:33] functions.py (line 614):     INFO: Time to build final jplace file: 0.0022617619979428127 seconds
[23:47:33] pipeline.py (line 77):  WARNING: Shutting down ProcessPoolExecutor...
[23:47:33] pipeline.py (line 79):  WARNING: ProcessPoolExecutor shut down.
[23:47:33] pipeline.py (line 83):     INFO: Removing temporary files...
[23:47:33] pipeline.py (line 180):     INFO: - Removed tmp0
[23:47:33] pipeline.py (line 88):     INFO: BSCAMPP completed in 4.279889469995396 seconds...
```

Sample output

1. Scenario 1: placing new sequences into an existing tree

```
run_bscampp.py -i [raxml best model] -t [existing tree] -a [alignment file]
```

Output to directory (default) → `bscampp_output/*`

Output placement file → `bscampp_output/bscampp_result.jplace`

Summary

SCAMPP/BSCAMPP

- Maximum-likelihood phylogenetic placement methods on very large trees (200,000 sequences and more)
- Scales to large data than other maximum-likelihood methods (pplacer, EPA-ng)
- Higher accuracy than other types of phylogenetic placement methods
- BSCAMPP: much faster than SCAMPP and about as accurate

This work was supported by U.S. National Science Foundation grants 2006069 and 2316233 (to Tandy Warnow).