College of Engineering Trivandrum

Department of Computer Science and Engineering

PECST495 Advanced Data Structures
Research activity 1

Topic: Rotation distance in trees *Submission date: 19 Dec 2025*

1 Problem description

Given two rooted binary trees T_1 and T_2 with the same number of nodes and same inorder sequence, the *rotation distance* between them, $d(T_1, T_2)$, is defined as the minimum number of rotations needed to transform T_1 into T_2. Rotation distance gives a way to measure how "structurally different" two binary trees are. It provides insight into the cost and limits of balancing algorithms (like AVL or red-black trees), because rotations are their only restructuring operation. It also arises naturally in geometric/combinatorial structures such as associahedra and polygon triangulations, and serves as a useful metric for comparing tree-shaped data in classification or structural analysis.

2 Literature review

Read some research papers to explore more on the rotation distance. A few suggestions:

[1] D. D. Sleator, R. E. Tarjan, and W. P. Thurston, "Rotation distance, triangulations, and hyperbolic geometry," in *Symposium on the Theory of Computing*, 1986, pp. 122–13 rotation distance, triangulations, and hyperbolic geometry.

[2] P. Dehornoy, "On the rotation distance between binary trees," *Advances in Mathematics*, vol. 223, no. 4, pp. 1316–1355, 2010.

[3] A. S. K. M. Anoop and J. Sarma, "On rotation distance of rank bounded trees," *Fundamenta Informaticae*, vol. 191, no. 2, pp. 79–104, 2024.

These are only sample papers. Don't limit your reading to just three.

3 Empirical exploration with Python

1. Generate all binary trees on n nodes. Assume $3 \leq n \leq 7$. Number of binary trees on n nodes grows exponentially with n. [Recall Catalan numbers!]
   - use recursive generation
   - represent trees as nested tuples
2. Implement the rotations.
3. Construct the *Rotation Graph* using the `networkx` package.
   - each vertex represents a binary tree with n nodes
   - two trees are connected by an edge if one can be turned into the other by a single rotation

   Such a graph is called the *associahedron graph* and can be used for **classification** in ML.
4. Visualize the graph using `matplotlib`.
5. Compute rotation distance using BFS on the above graph.
6. Represent the rotation distances as the *distance matrix*.

4 Runtime Benchmarking

For a fixed n, measure the execution time for

1. generating all trees of size n
2. constructing the rotation graph
3. computing the distance matrix

Create a `csv` file for recording the execution times. Also include a column for number of trees in your `csv` file.

5 Memory Usage Profiling

For each n, record the memory usage for

1. storing the trees
2. storing the rotation graph
3. storing the distance matrix

Create a `csv` file for these stats.

6 GitHub Submissions

Create a private GitHub repository with the following directories/files:

1. Source files
    - Python source code
2. Results
    - `csv` files and the matlab plot
3. `README.md`
    - include details as to how to run your code, sample commands, dependencies if any
4. `requirements.txt`
    - auto generate this using the `pip freeze` command
    - this will help recreating the run environment on another machine

7 LaTeX Submissions

Submit a report to include:

1. literature review summary
    - motivation for the work
    - methodology and results for each paper you reviewed
    - a comparison table summarizing results across papers
2. tabulations of execution time benchmarking and memory usage profiling
3. evaluation metrics
    - for every n, compute the rotation graph diameter. Check whether the computed diameter matches the known theoretical bounds [1]

8 Optional Exercise

After computing the rotation distance matrix, you can explore the geometric structure of the tree space by performing clustering. Hierarchical or k-means (or even, k-medoids) methods can be employed. You can also visualize the tree embeddings via **MDS** or **t-SNE**.