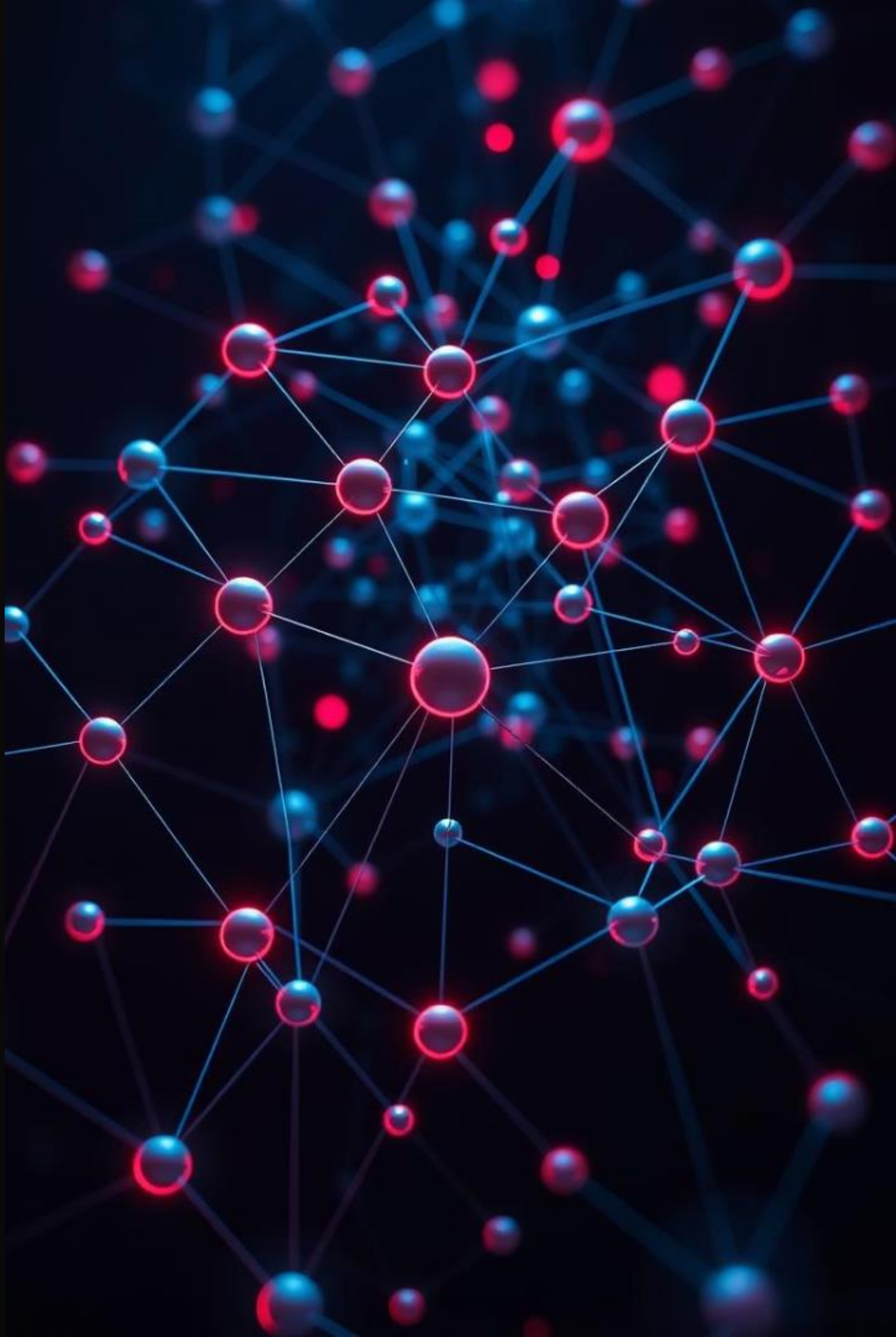
A Parallel Algorithm for Constructing Multiple Independent Spanning Trees in Bubble-Sort Networks

Authors: Shih-Shun Kao, Ralf Klasing, Ling-Ju Hung, Chia-Wei Lee, Sun-Yuan Hsieh. Published in the Journal of Parallel and Distributed Computing, 2023. This work proposes a fully parallel algorithm for constructing independent spanning trees in bubble-sort networks, solving an open problem.

By:
Absir Ahmed – 22i0915
Rihab Rabbani – 22i1345
Fahd Ahmad – 22i1131



Background: Bubble-Sort Networks & ISTs

Bubble-Sort Network

- Vertices: all permutations of {1,2,...,n}
- Edges: swap adjacent elements
- Connectivity: n-1
- Diameter: n(n-1)/2

Independent Spanning Trees

- Rooted at identity permutation
- Vertex-disjoint paths for fault tolerance
- Applications: secure message distribution

Problem and Motivation

Prior Work

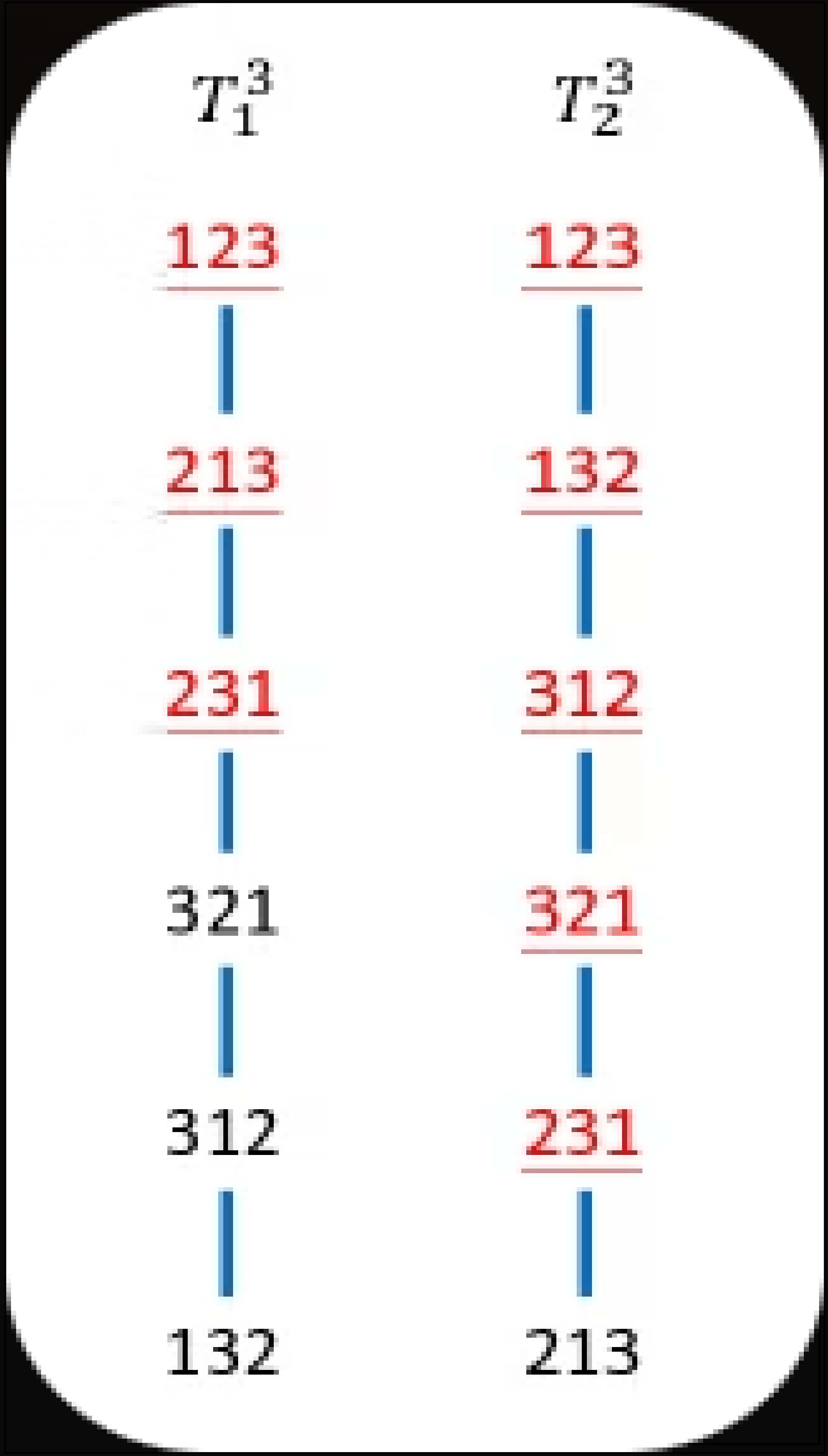
Recursive IST algorithm (Kao et al., 2019), not parallelizable.

Open Problem

Devise a parallel algorithm for ISTs in bubble-sort networks.

Motivation

Enable scalable, fault-tolerant, and secure routing via parallelism.



Proposed Algorithm

Algorithm 1

Algorithm 1 constructs n-1 ISTs rooted at identity permutation.

Non-recursive and fully parallelized.

Computation

Inverse permutation is done in O(n) time.

FindPosition(v) function and Swap(v,u) function determine the parent in constant time.

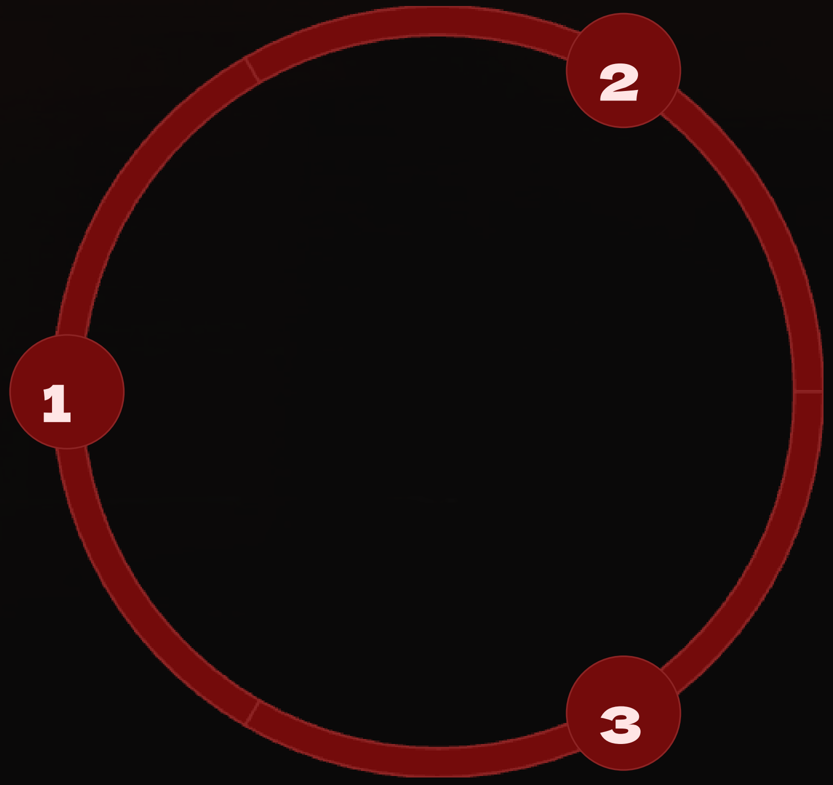
Algorithm Overview

Preprocess each vertex to compute inverse permutation set.
- For each vertex v != 1_n and three Ttn, apply the reules based on the last symbol v_n to determine the parent
- Use FindPosition and Swap to determine the parent.

Case C

(v_n = j)

For j that belongs to {1, 2, ..., n-2}, general case



Case A

(v_n = n)

Handles root connections

Case B

(v_n = n - 1)

Manages transitions to (n)

Correctness & Complexity

1

Correctness

Each Tt^n forms a valid spanning tree.

2

Vertex-Disjoint Paths

Paths in different trees are vertex-disjoint.

3

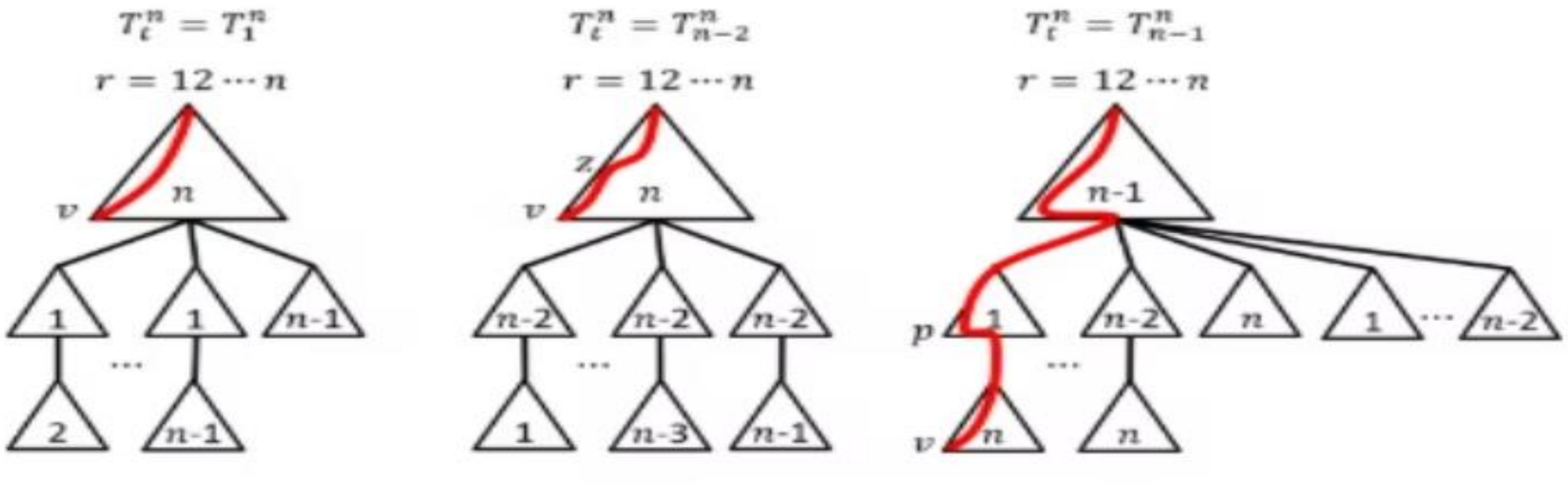
Complexity

O(1) per vertex per tree, total O(n·n!).

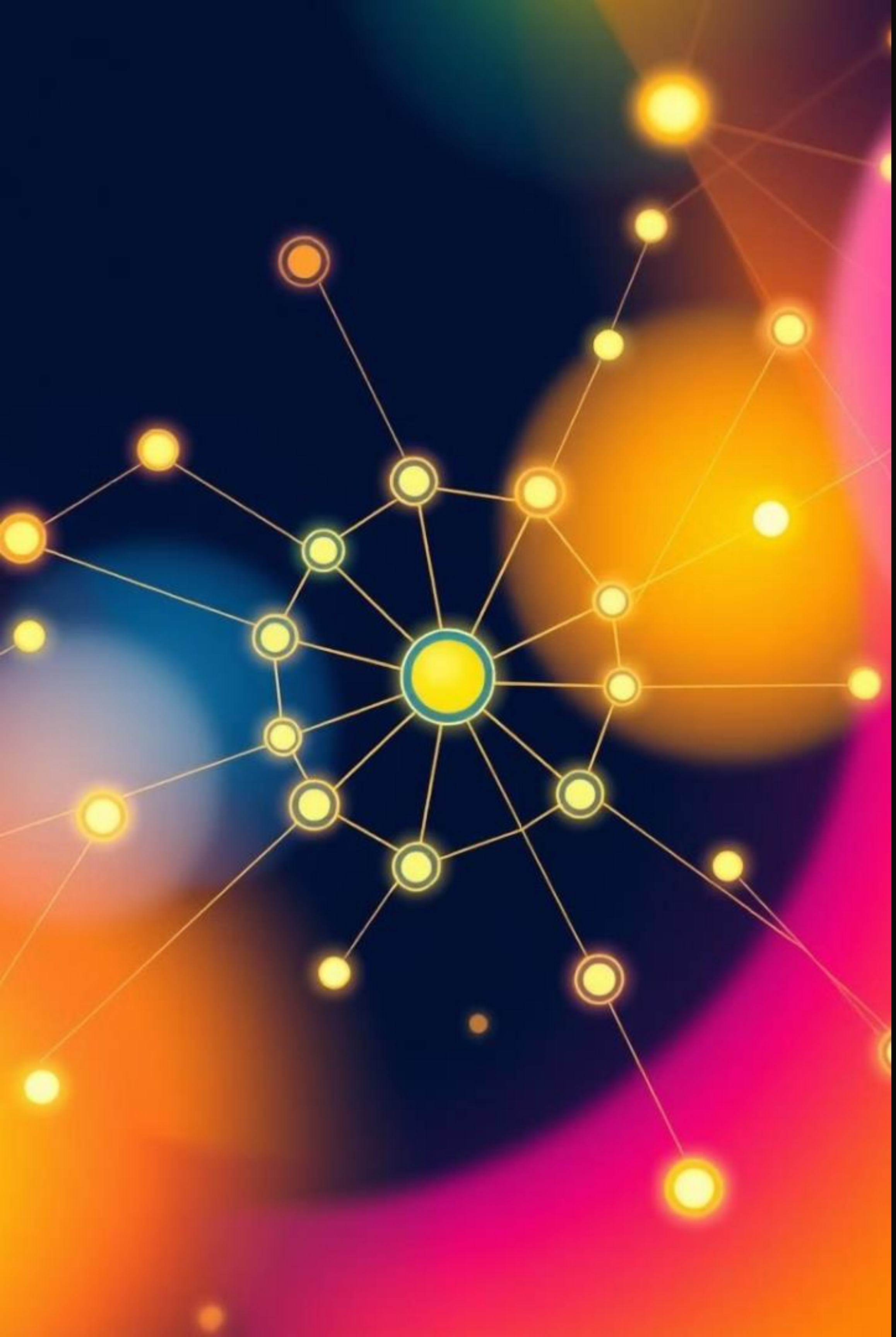
4

IST Height

At most n(n+1)/2 - 1.



(a) (b) (c)



Key Contributions

Non-Recursive Algorithm

Parent1() computes parent in $O(1)$ time, fully parallelizable.

Optimal Time Complexity

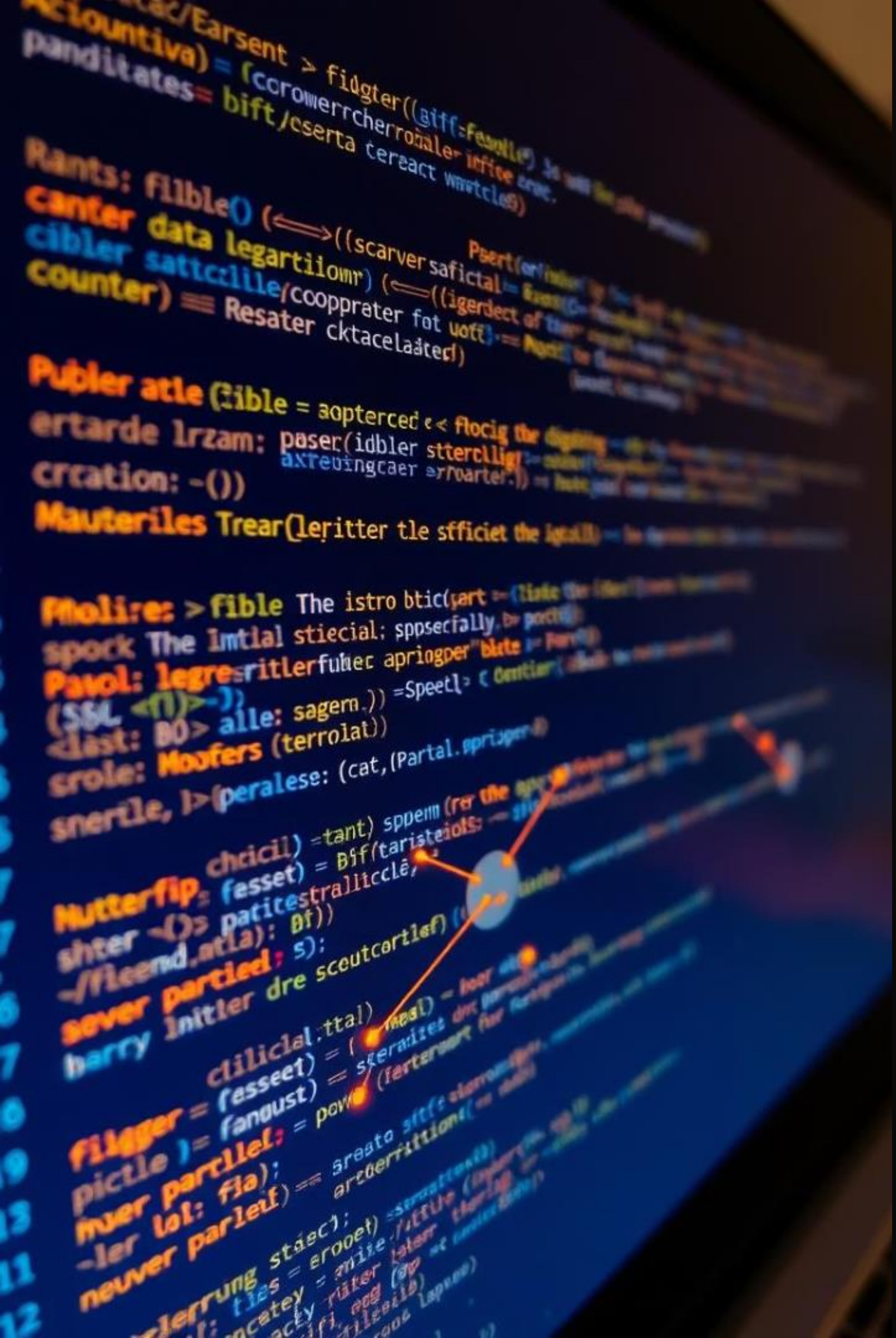
Total $O(n \cdot n!)$, matches lower bound.

IST Height

At most $n(n-1)/2 + n-1$.

Correctness

Case analysis ensures vertex-disjoint paths.



Proposed Parallelization Strategy

MPI (Inter-Node)

Partition vertices using METIS, assign to nodes.

OpenMP (Intra-Node)

Threads process vertex-tree pairs independently.

METIS Partitioning

Minimize edge cuts, balance load across nodes.

Practical Implications

Fault Tolerant and Secure Networks

Improve fault tolerance and security

Scalability and Speedup

Scalable due to parallel design and efficient

Energy Efficient

Helps build low-power parallel hardware

Relevance to Project

Implements the concepts of parallel and distributed computing

Conclusion

1 Parallel Algorithm

Presents a parallel, non-recursive algorithm. It constructs n–1 ISTs in Bn.

2 Optimal Time

Achieves $O(n \cdot n!)$ time complexity. Constant work per vertex.

3 Scalable Parallelism

Works with MPI, OpenMP, and METIS. Designed for scalable parallelism.

4 Impactful

Enhances security and fault tolerance. Solves a key open problem.