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

Shih-Shun Kao, Ralf Klasing, Ling-Ju Hung, Chia-Wei Lee, Sun-Yuan Hsieh

April 19, 2025

Introduction

Objective: Propose a non-recursive, fully parallelized algorithm for constructing $n-1$ Independent Spanning Trees (ISTs) in bubble-sort networks B_n.

- Published in *Journal of Parallel and Distributed Computing, 2023*
- Solves an open problem from Kao et al. (2019)

Background

Bubble-Sort Network (B_n):

- Cayley graph with vertex set as permutations of $\{1, 2, \ldots, n\}$ ($n!$ vertices)
- Edges connect permutations differing by swapping adjacent elements
- Properties:
  - Connectivity: $n - 1$
  - Diameter: $\frac{n(n-1)}{2}$

Independent Spanning Trees (ISTs):

- Trees rooted at the identity permutation $12 \ldots n$
- Vertex-disjoint paths to the root in different trees
- Applications: Fault-tolerant communication, secure message distribution

Problem and Motivation

Prior Work (Kao et al., 2019):

- Recursive algorithm for constructing ISTs in B_n
- Constant amortized time per vertex
- Hard to parallelize due to recursion

Open Problem: Develop a parallel algorithm for IST construction in bubble-sort networks.

Motivation:

- Enhance fault tolerance and security in interconnection networks
- Achieve scalability for large-scale networks via parallelism

Key Contributions

- **Non-Recursive Algorithm:** Parent1 computes the parent in each of the $n-1$ ISTs in constant time.
- **Fully Parallelizable:** Each vertex computes its parent independently.
- **Time Complexity:** Total complexity $\mathcal{O}(n \cdot n!)$, asymptotically optimal.
- **Height of ISTs:** At most $D(B_n) + n - 1 = \frac{n(n-1)}{2} + n - 1$.
- **Correctness:** Proven via case analysis ensuring unique, vertex-disjoint paths.
- **Solves Open Problem:** Parallel construction of ISTs in B_n.

Algorithm Overview

Algorithm 1: Parent1(v, t, n**)**

- **Input:** Vertex v, tree index $t \in \{1, \ldots, n-1\}$, dimension n.
- **Output:** Parent of v in tree T_t^n.

Steps:

- Compute $r(v)$, the rightmost out-of-place symbol.
- Apply swapping rules based on cases (v_n).
  - $v_n = n$: Rules (1.1)–(2).
  - $v_n = n-1$: Rules (3)–(4).
  - $v_n = j \in \{1, \ldots, n-2\}$: Rules (5)–(6).

Correctness and Complexity

Correctness:

- Each T_t^n forms a valid spanning tree.
- Paths in different trees are vertex-disjoint.

Complexity:

- Per vertex, per tree: $\mathcal{O}(1)$.
- Total: $\mathcal{O}(n \cdot n!)$.
- Preprocessing: $\mathcal{O}(n \cdot n!)$.

Height Analysis: Max height $\leq \frac{n(n+1)}{2} - 1$.

Parallelization Strategy

MPI (Inter-Node):

- Use METIS for balanced graph partitioning (minimize edge cuts).
- Each node processes a subset of vertices.
- Communicate boundary data using `MPI_Isend/Irecv`.

OpenMP (Intra-Node):

- Parallelize vertex processing with `#pragma omp parallel for`.
- Dynamic scheduling for load balancing.

Future Work

- Extend to other networks (e.g., (n, k)-bubble-sort, butterfly).
- Optimize IST height below $D + n - 1$.
- Test scalability on large n using distributed clusters.
- Explore GPU-based parallelism (e.g., CUDA).
- Integrate ISTs into real-world network protocols.

Conclusion

Summary:

- Novel parallel algorithm for constructing $n-1$ ISTs in B_n.
- Fully parallelizable with MPI, OpenMP, and METIS.
- Solves open problem; enhances fault tolerance and secure communication.

Next Steps:

- Implement and benchmark.
- Extend to other Cayley graphs.