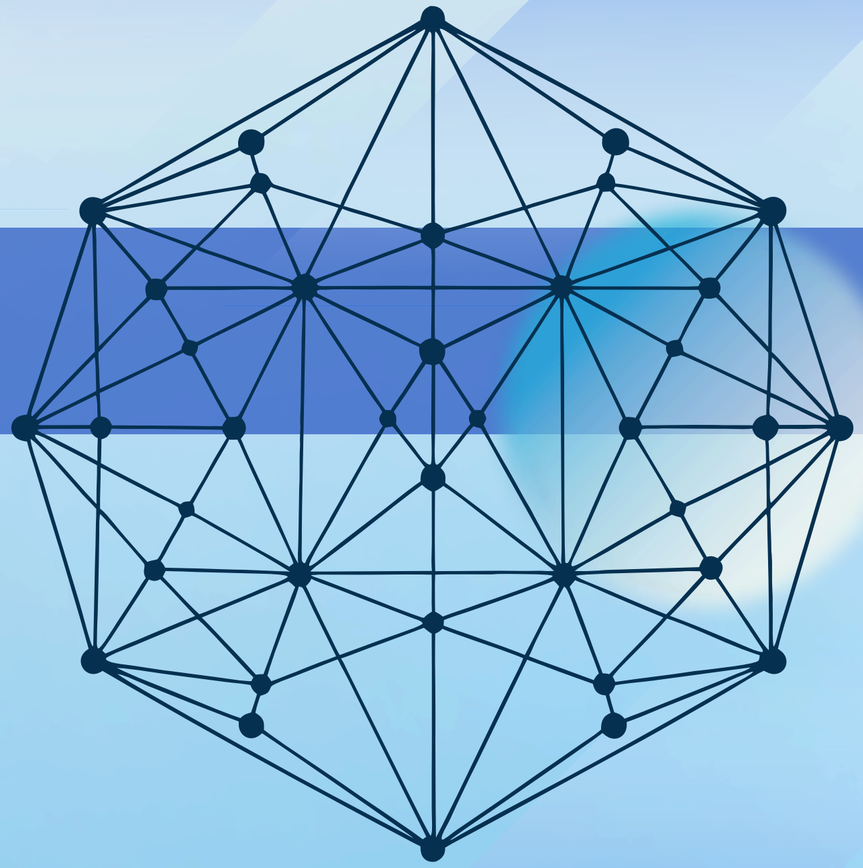
IST in bubble sort networks



Syed Muhmmad **Dilawar** Rasool Shah, 22i-0884
Muhammad **Junaid** Asrar, 22i-0770
Sardar **Abdullah** Ashfaq, 22i-1149

Research Paper Summary

•- Paper: "A Parallel Algorithm for Constructing Multiple Independent Spanning Trees in Bubble-Sort Networks"

•- Key Goals:

• • Construct n-1 ISTs rooted at the same vertex in Bn
• • Ensure constant-time parent computation
• • Enable full parallelization
• • Achieve optimal time complexity: O(n · n!)

Problem Background

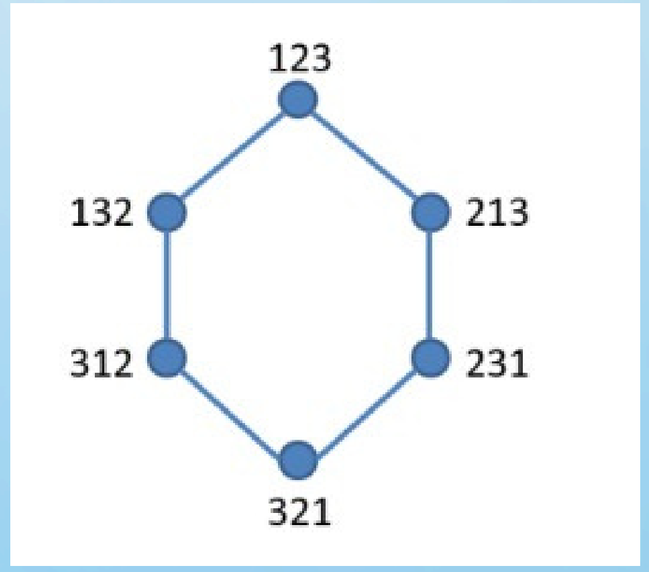
•- Independent Spanning Trees (ISTs) ensure fault-tolerant and secure communication.

•- Multiple disjoint paths are vital in network reliability.

•- Constructing ISTs in structured networks like Bubble-Sort Networks (Bn) is challenging.

•- Previous algorithms were recursive and hard to parallelize.

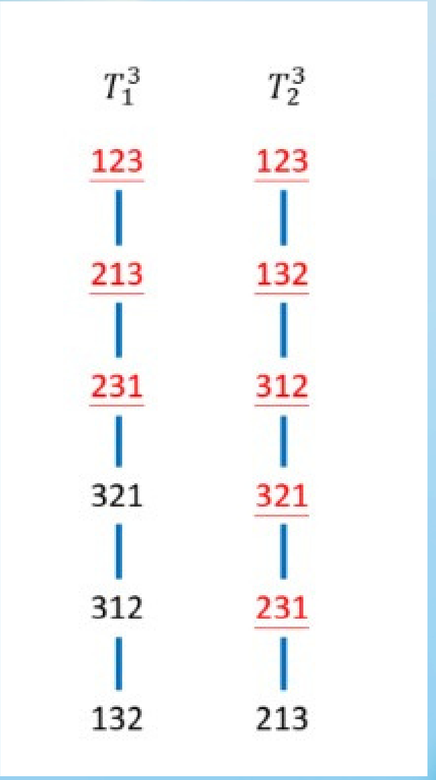
Bubble-Sort Network (Bn)



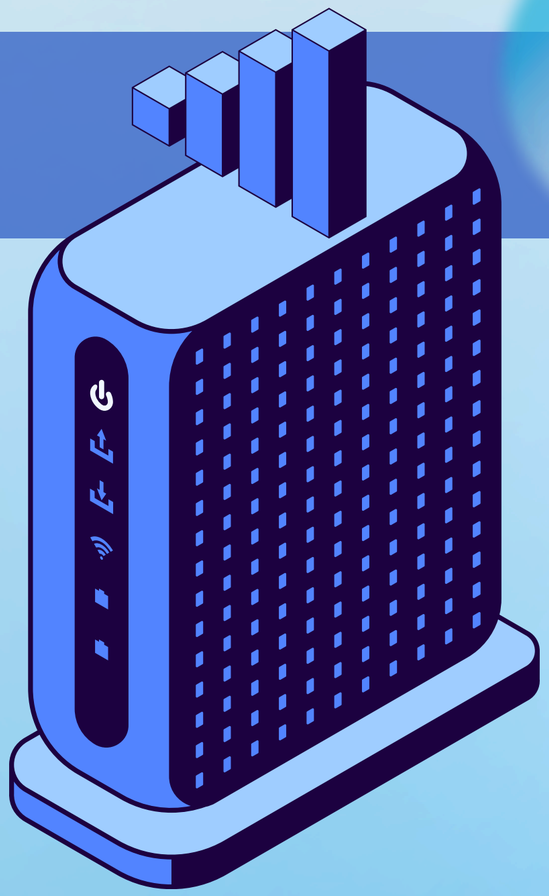
- A Cayley graph over permutations of {1, 2, ..., n}
- Each vertex: one permutation
- Edges: swap of adjacent elements
- Vertices: n!, Degree: n-1, Diameter: n(n-1)/2

What Are ISTs?

- •- A set of k spanning trees rooted at the same node
- •- Paths from any node to root in different trees are vertex and edge disjoint (except endpoints)
- •- Useful in fault-tolerant broadcasting and secure routing



The proposed algorithm



- Non-recursive, rule-based construction
- Two key helper functions:
  - Swap(v, x) - swaps x to correct position
  - FindPosition(v) - identifies position to fix

Each vertex independently determines parent for each tree

If the last element equals n ($v_n = n$):

- Rule (1): If $t \neq n-1$, then p = FindPosition(v)
- Rule (2): Otherwise, p = Swap(v, v_{n-1})

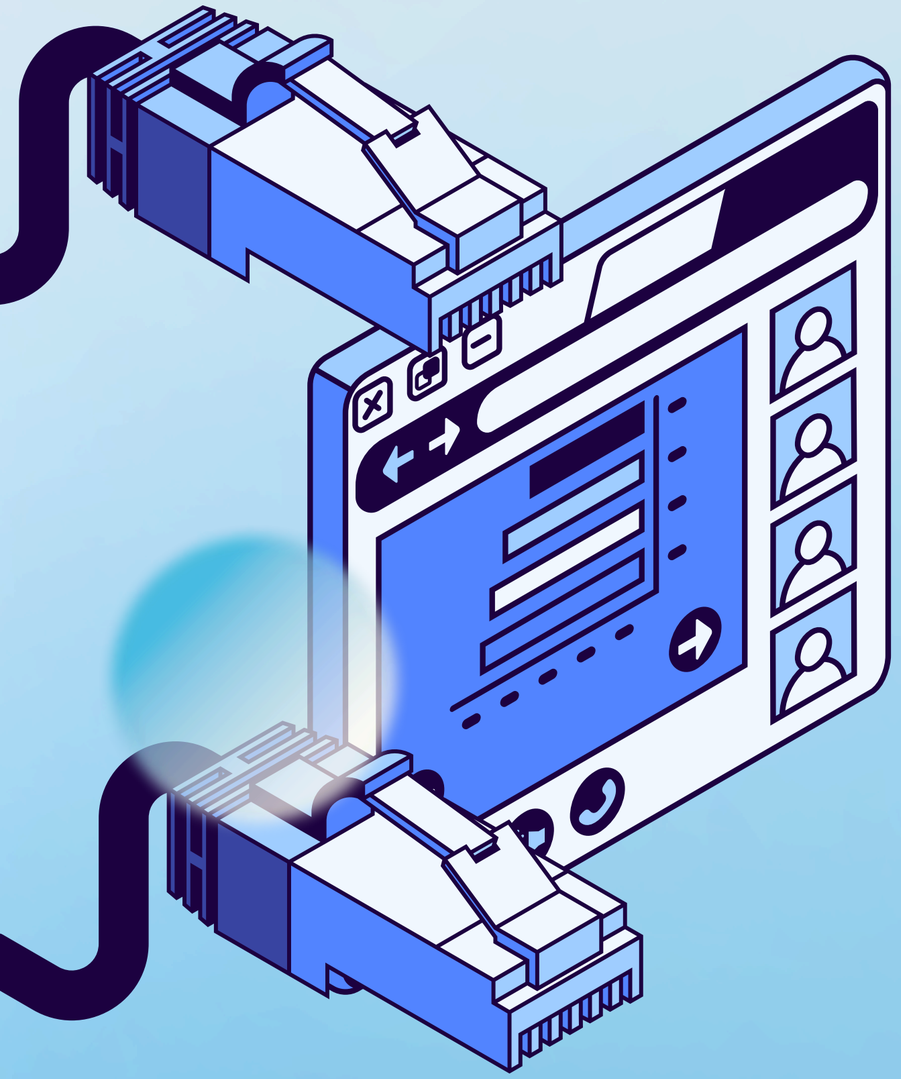
If the last element equals n-1 AND second-to-last equals n ($v_n = n-1$, $v_{n-1} = n$, and Swap(v,n) $\neq 1_n$):

- Rule (3): If $t = 1$, then p = Swap(v, n)
- Rule (4): Otherwise, p = Swap(v, $t-1$)

For all other cases:

- Rule (5): If the last element equals t ($v_n = t$), then p = Swap(v, n)
- Rule (6): Otherwise, p = Swap(v, t)

v	t	v_4	Rule	p	v	t	v_4	Rule	p
1234	-	-	-	-	3124	1 2 3	4	(1.3) (1.2) (2)	3214 1324 3142
1243	1 2 3	3	(6) (6) (5)	2143 1423 1234	3142	1 2 3	2	(6) (5) (6)	3412 3124 1342
1324	1 2 3	4	(1.3) (1.2) (2)	3124 1234 1342	3214	1 2 3	4	(1.2) (1.3) (2)	2314 3124 3241
1342	1 2 3	2	(6) (5) (6)	3142 1324 1432	3241	1 2 3	1	(5) (6) (6)	3214 3421 2341
1423	1 2 3	3	(6) (6) (5)	4123 1432 1243	3412	1 2 3	2	(6) (5) (6)	3421 3142 4312
1432	1 2 3	2	(6) (5) (6)	4132 1342 1423	3421	1 2 3	1	(5) (6) (6)	3241 3412 4321
2134	1 2 3	4	(1.2) (1.1) (2)	1234 2314 2143	4123	1 2 3	3	(6) (6) (5)	4213 4132 1423
2143	1 2 3	3	(3) (4) (4)	2134 2413 1243	4132	1 2 3	2	(6) (5) (6)	4312 1432 4123
2314	1 2 3	4	(1.2) (1.3) (2)	2134 3214 2341	4213	1 2 3	3	(6) (6) (5)	4231 4123 2413
2341	1 2 3	1	(5) (6) (6)	2314 3241 2431	4231	1 2 3	1	(5) (6) (6)	2431 4321 4213
2413	1 2 3	3	(6) (6) (5)	2431 4213 2143	4312	1 2 3	2	(6) (5) (6)	4321 3412 4132
2431	1 2 3	1	(5) (6) (6)	2341 4231 2413	4321	1 2 3	1	(5) (6) (6)	3421 4312 4231



Our Parallelization Strategy

- MPI for inter-node graph partitioned execution
- OpenMP for intra-node shared-memory parallelism
- METIS for graph partitioning before MPI allocation
- OpenMP for intra-node shared-memory parallelism

Expected Benefits & Challenges

•Benefits:

•– Scalable parallel execution

•– Fault-tolerant and secure path generation

•– Reduced computation time due to parallel design

Challenges

•– Efficient METIS partitioning

•– Hybrid sync and load balancing

Phase 2 roadmap

- Implement Algorithm using MPI + OpenMP
- Graph generation and METIS integration
- Evaluate weak and strong scaling
- Use MPI Analyzer for profiling

Thank You!

Questions?