PDC
PRESENTATION

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

i220839 M Talha
i221295 Ahmad Haroon

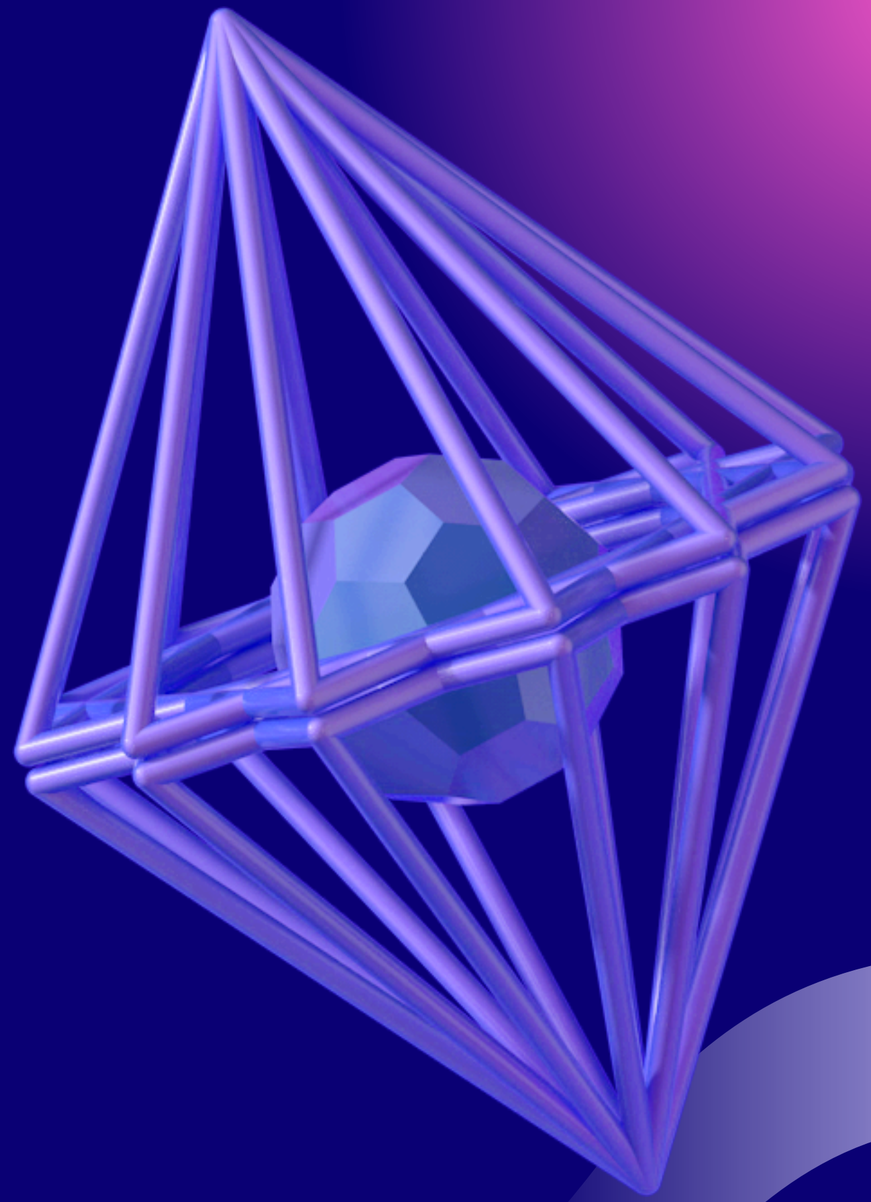
BSN

What is a Bubble Sort Network (BSN)?

- A BSN of order n has n! nodes.
- Each node represents a unique permutation of n elements.
- An edge connects nodes that differ by a single adjacent swap.
- Example: From 1234 to 1243 (swap position 3)

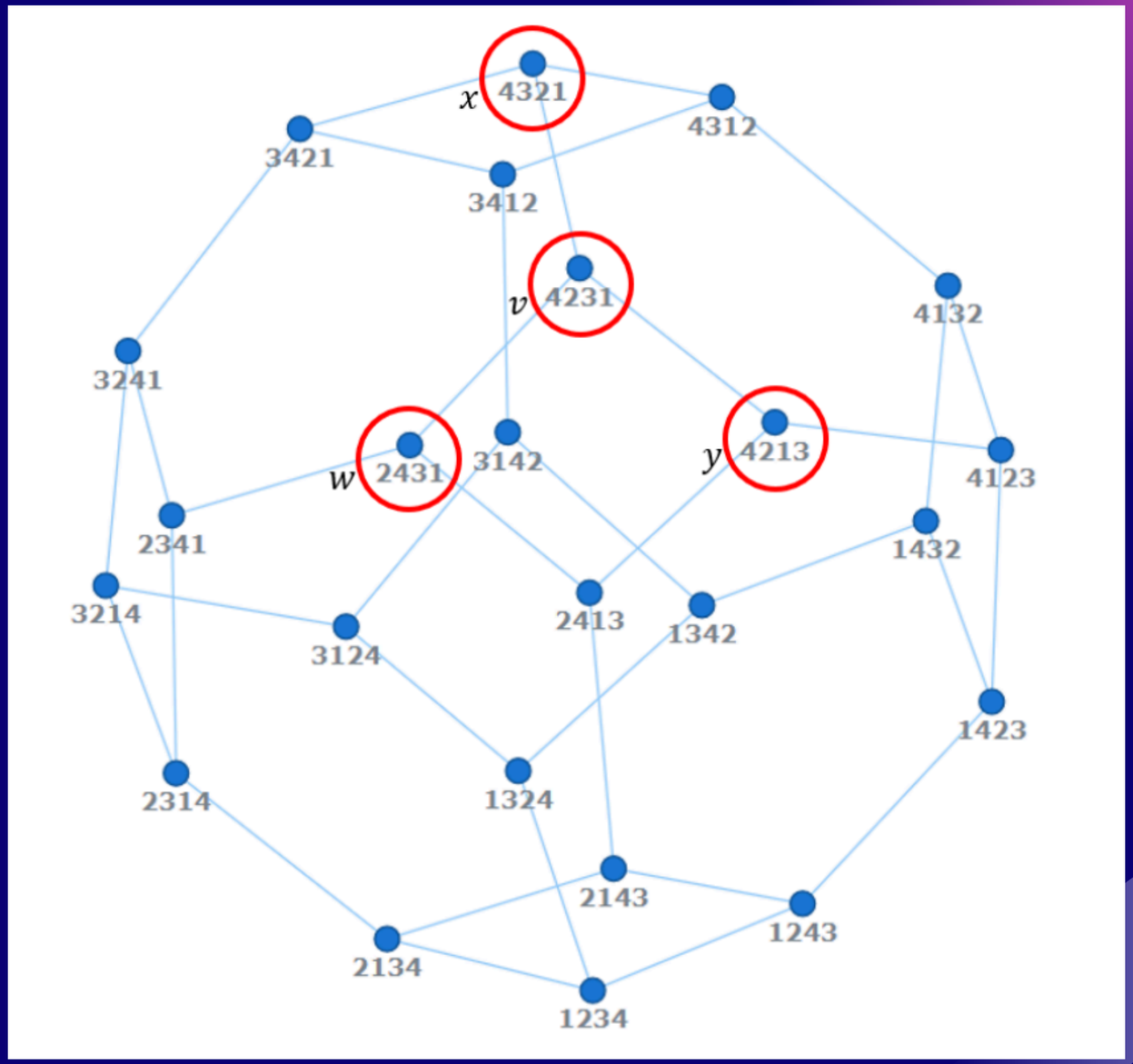
Why BSN?

- Models fault-tolerant interconnection networks.
- Suitable for parallel computing tasks.

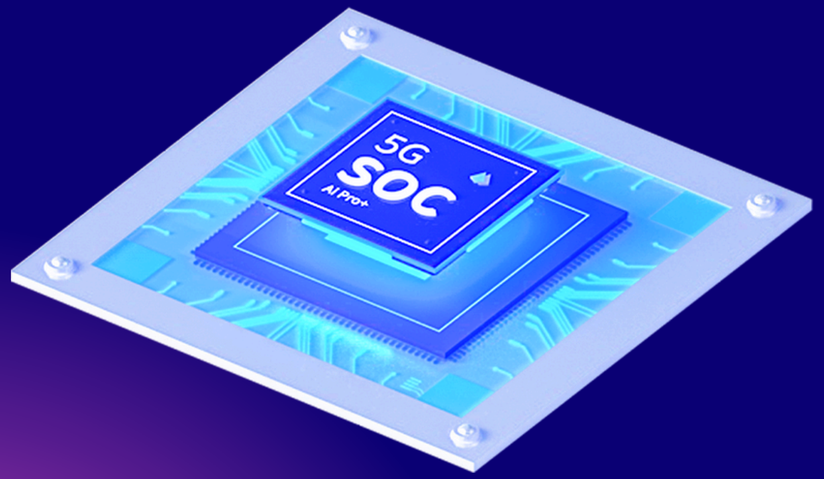


BSN DIAGRAM

4321
(ROOT 1234)



PROBLEM STATEMENT



Problem Statement

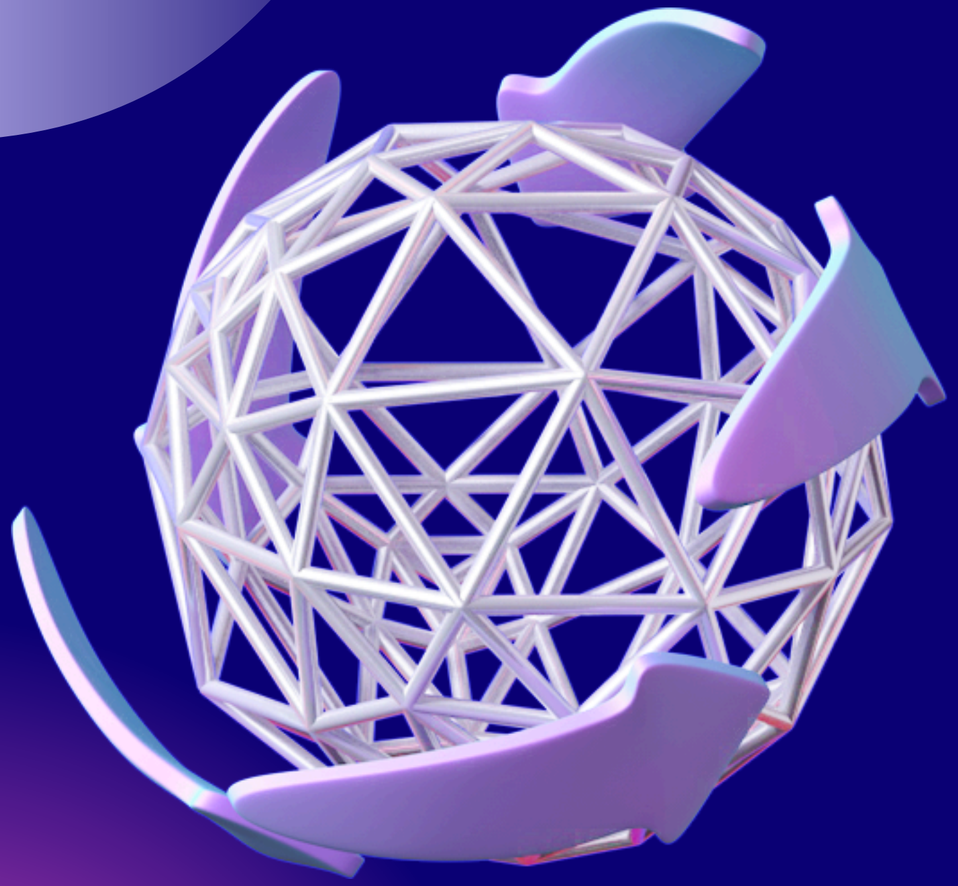
Problem: if edge deletes in BSN then how to communicate ?
Solution: Construct n Independent Spanning Trees (ISTs) in BSN(n)

Challenges:

- Large graph size (n! nodes)
- Avoiding edge overlap between ISTs
- Efficient parallel construction for fault-tolerant routing

KEY CONTRIBUTIONS

1. Proposes a parallel algorithm to construct n ISTs in BSN(n)
2. Ensures all ISTs are:
    - Edge-disjoint (means independent)
    - Rooted at the identity permutation (e.g., 1234)
    - Span all n! nodes
3. Reduces communication overhead
4. Supports efficient, fault-tolerant routing in networks



INDEPENDENT SPANING TREE

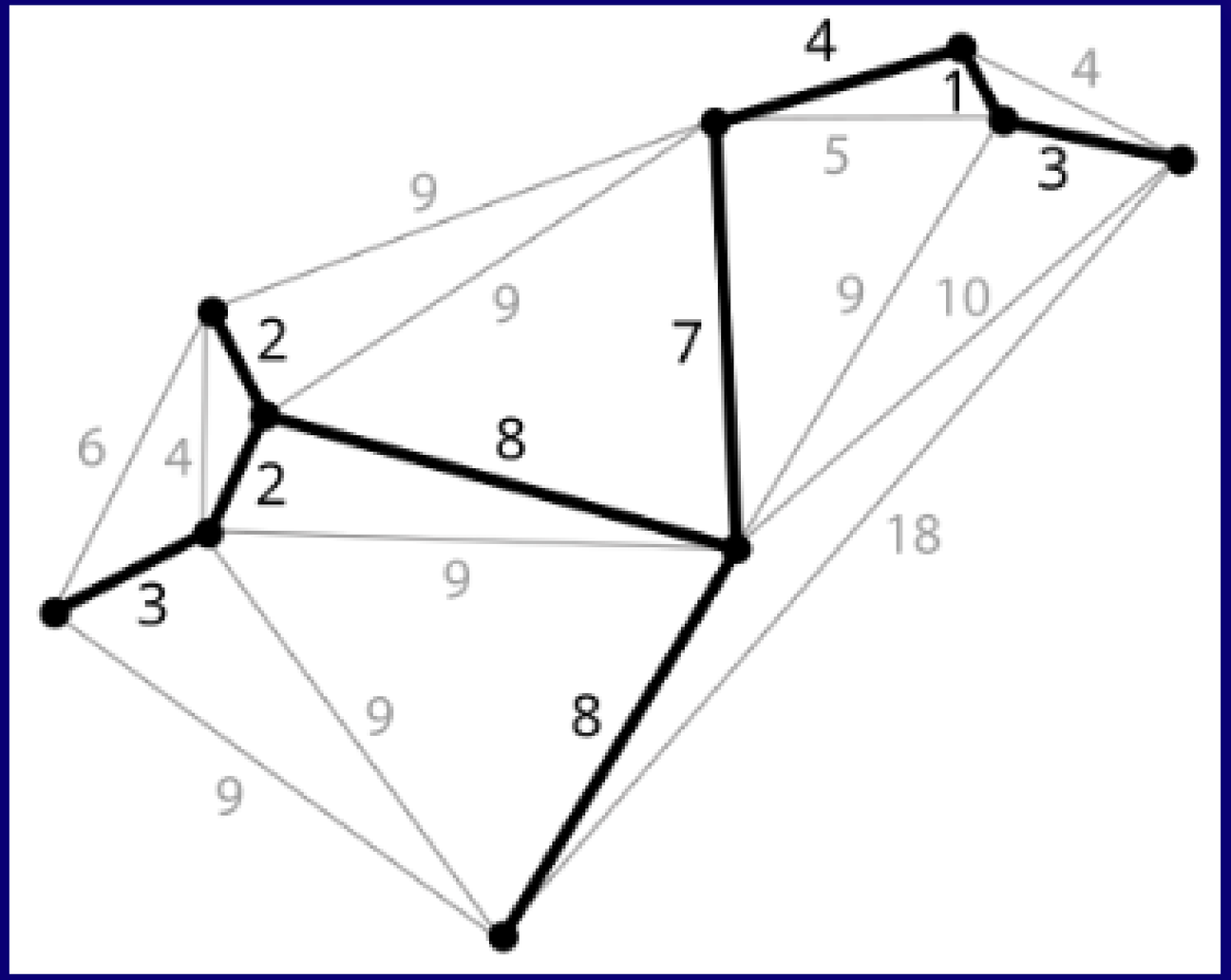
1. A spanning tree that covers all nodes without cycles
2. In ISTs:

- All trees share the same root
- Trees are edge-disjoint
- Each node is reachable via unique paths

Why ISTs?

- Enable parallel routing
- Improve network reliability

INDEPENDENT
SPANING TREE



ALGORITHM
OVERVIEW

Parallel Construction of ISTs:

- Number of ISTs = n (order of BSN)
- Rooted at the identity permutation
- Each IST_k uses adjacent swaps at position k
- Constructed in parallel, each tree uses distinct swap rules

Result:

- n edge-disjoint spanning trees

ALGORITHM OVERVIEW

Input: n (size of permutation)
Output: n independent spanning trees IST_1, IST_2, ..., IST_n

for i = 1 to n in parallel do
    initialize IST_i with root node (identity permutation)
  mark root node as visited
  for each node in BSN do
    if node is unvisited in IST_i then
      generate child nodes by adjacent swaps (except at position i)
      add child to IST_i
      mark child as visited
    end if
  end for
end for

BSN
EXAMPLE

BSN(4):

- Total Nodes = 4! = 24
- Total ISTs = 4

IST Root: 1234 (identity)
Leaf nodes: Non-deterministic, differ in each tree
Construction: Based on swap position [1..n-1]

Useful for:

- Parallel routing
- Load balancing

PARALLELIZATION
STRATEGY

1. MPI (Inter-node parallelism):
   - Distribute construction of IST_1 to IST_n across different processes
   - Each MPI node builds one IST independently

2. OpenMP (Intra-node parallelism):
   - Parallelize operations like swap computations and node generation within each MPI process

3. METIS:
   - Used for graph partitioning before tree construction
   - Ensures load balancing among MPI processes

TOOLS & TECHS

Technologies Used:

- MPI: Message Passing Interface for distributed computing
- OpenMP: Shared memory parallelism within a node
- METIS: Graph partitioning tool for efficient workload distribution
- GitHub: Version control and progress tracking

Implementation Language: C/C++ with MPI and OpenMP

KEY TAKEAWAYS

- Constructed n edge-disjoint ISTs in BSN(n)
- Parallel strategy reduces time and enhances scalability
- MPI + OpenMP + METIS integration proposed
- Useful in routing, fault tolerance, and parallel graph processing

KEY TAKEAWAYS

- Time Complexity per IST=O((n−1)×n!)=O(n×n!)
- Locality Optimization: [OpenMP]
- Parallelism: One processor per IST [MPI]
- Graph Partitioning with METIS: (if the network is very large)
    a. You can partition the full BSN into subgraphs and assign each partition to a processor.
- Synchronous partitioning by METIS
- No PageRank is used in BSN.

QUEATIONS?

! OEMW

MEOW !

THANK YOU MEOW !