Lawrence Livermore National Laboratory

Heuristic-based techniques for mapping irregular communication graphs to mesh topologies

Abhinav Bhatele and Laxmikant V. Kale
University of Illinois at Urbana-Champaign

LLNL-PRES-495871

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551

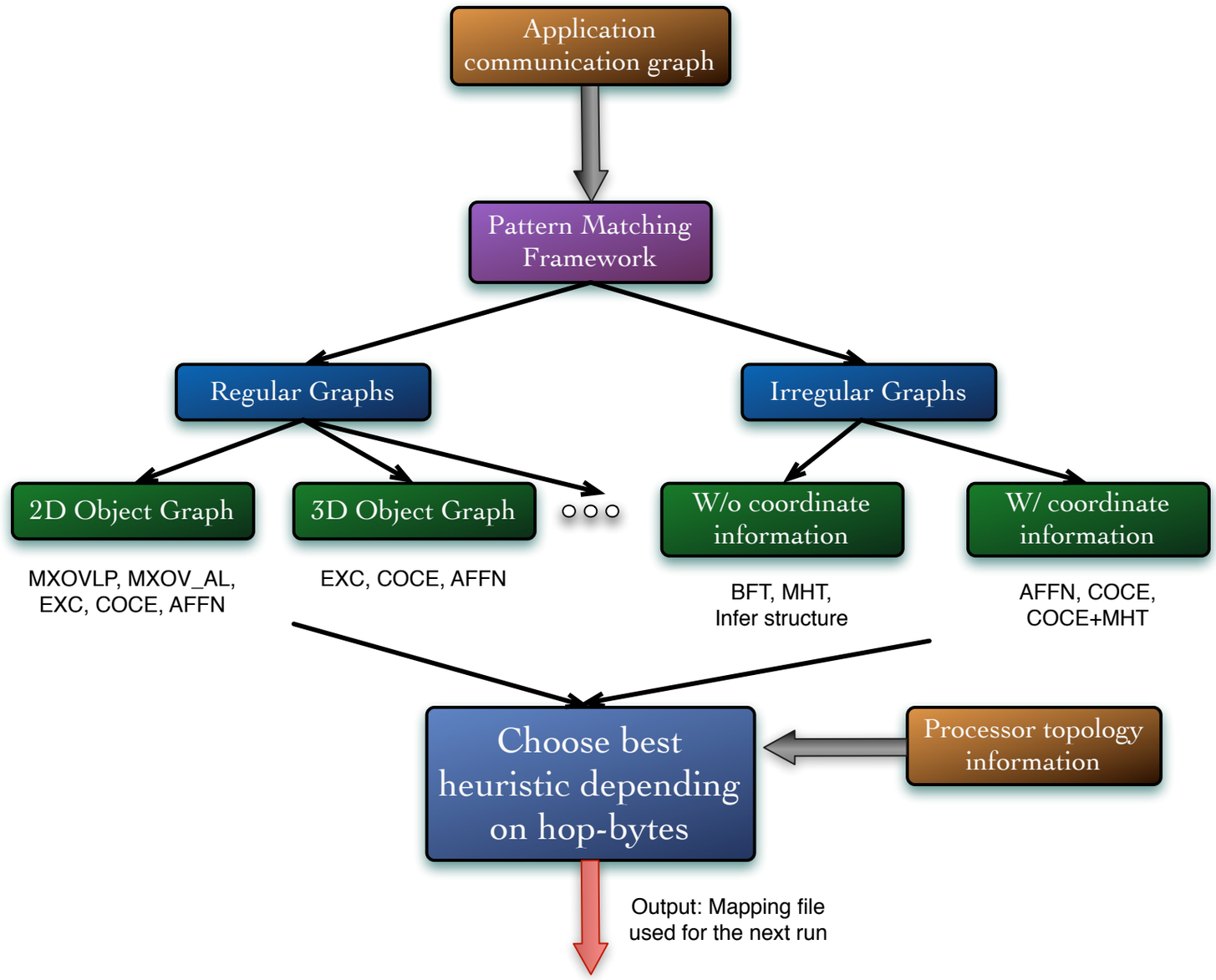
Introduction

- Various kinds of interconnect networks deployed today for supercomputers
  - mesh, fat-tree, Kautz graph, dragonfly
- Link sharing leads to contention and performance slowdowns
- Communication optimization becoming increasingly important

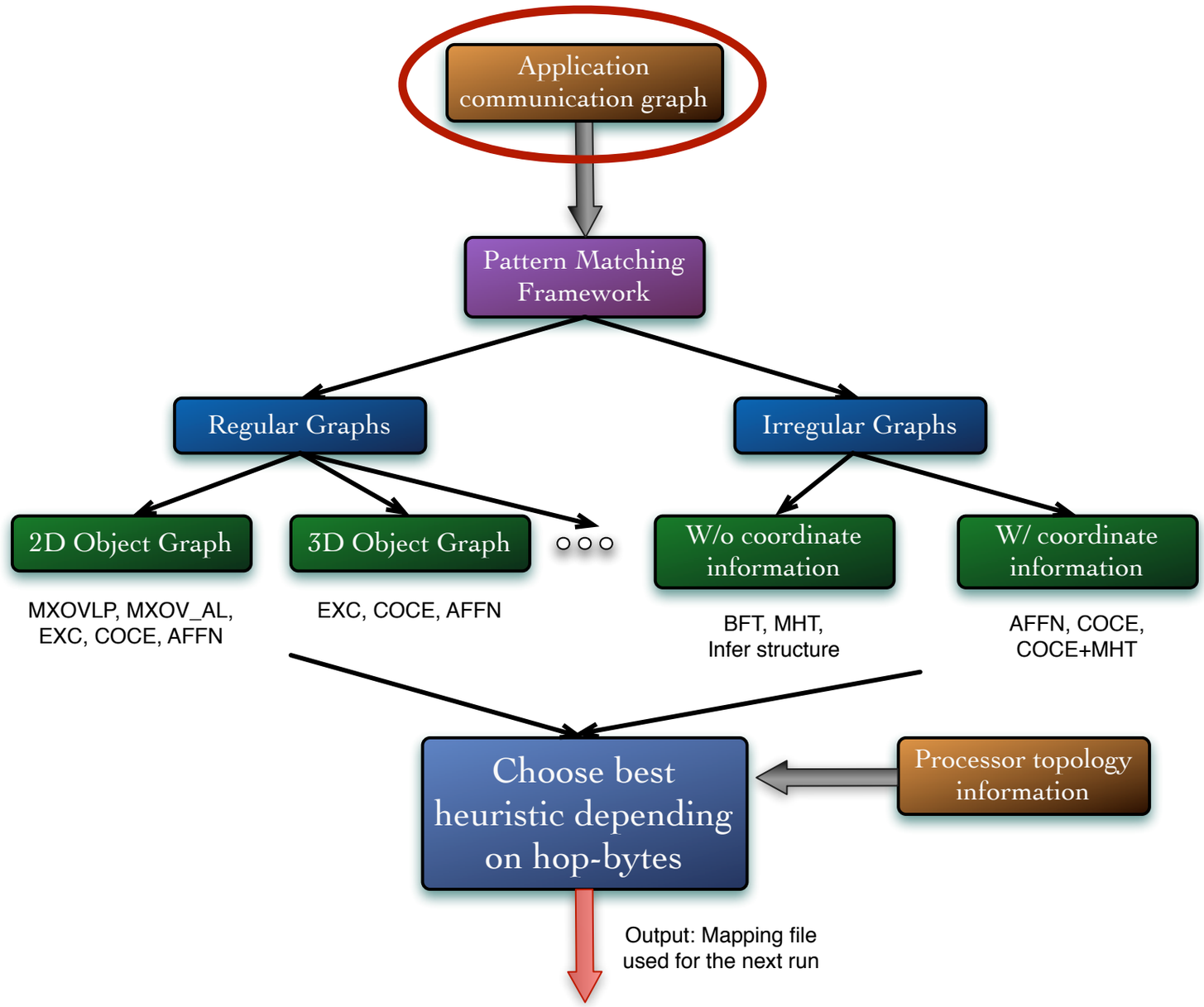
Topology aware mapping

- Mapping the communication graph of an application to the physical topology can optimize communication
- Diverse set of parameters from the application graph and the processor topology
  - one solution may not do well in all cases
- Specific heuristics for mesh topologies and regular/ irregular communication graphs

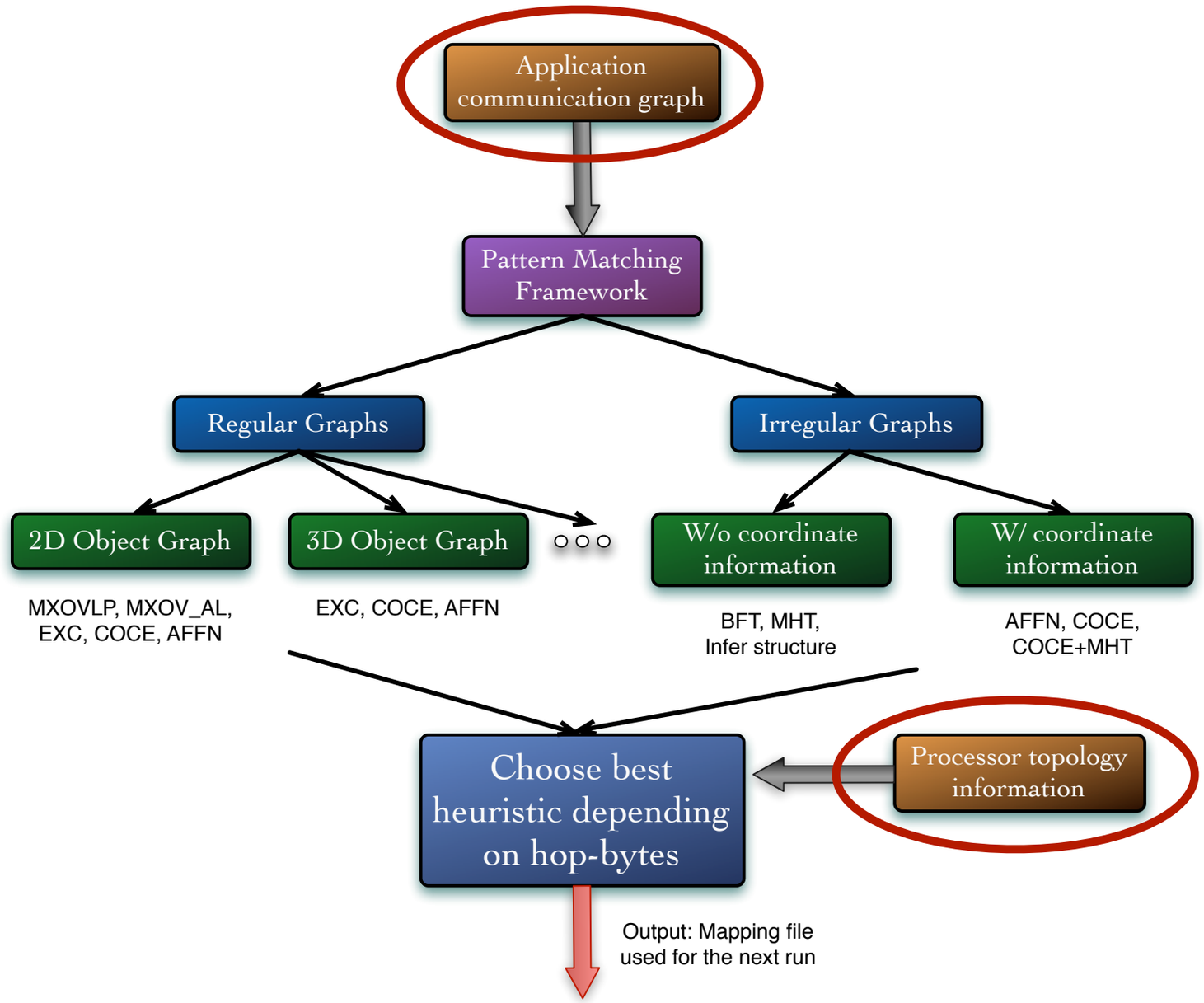
Automatic Mapping Framework



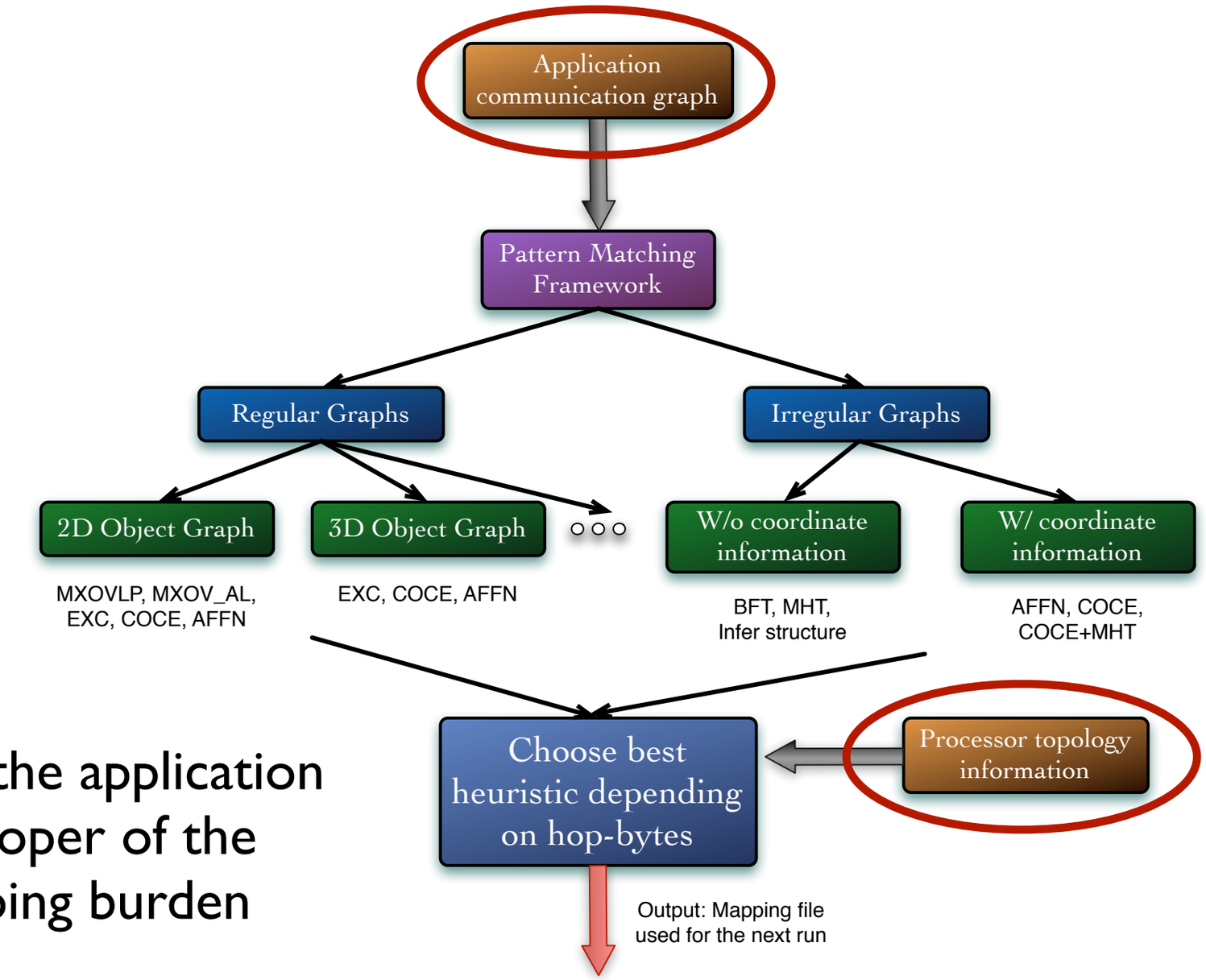
Automatic Mapping Framework



Automatic Mapping Framework

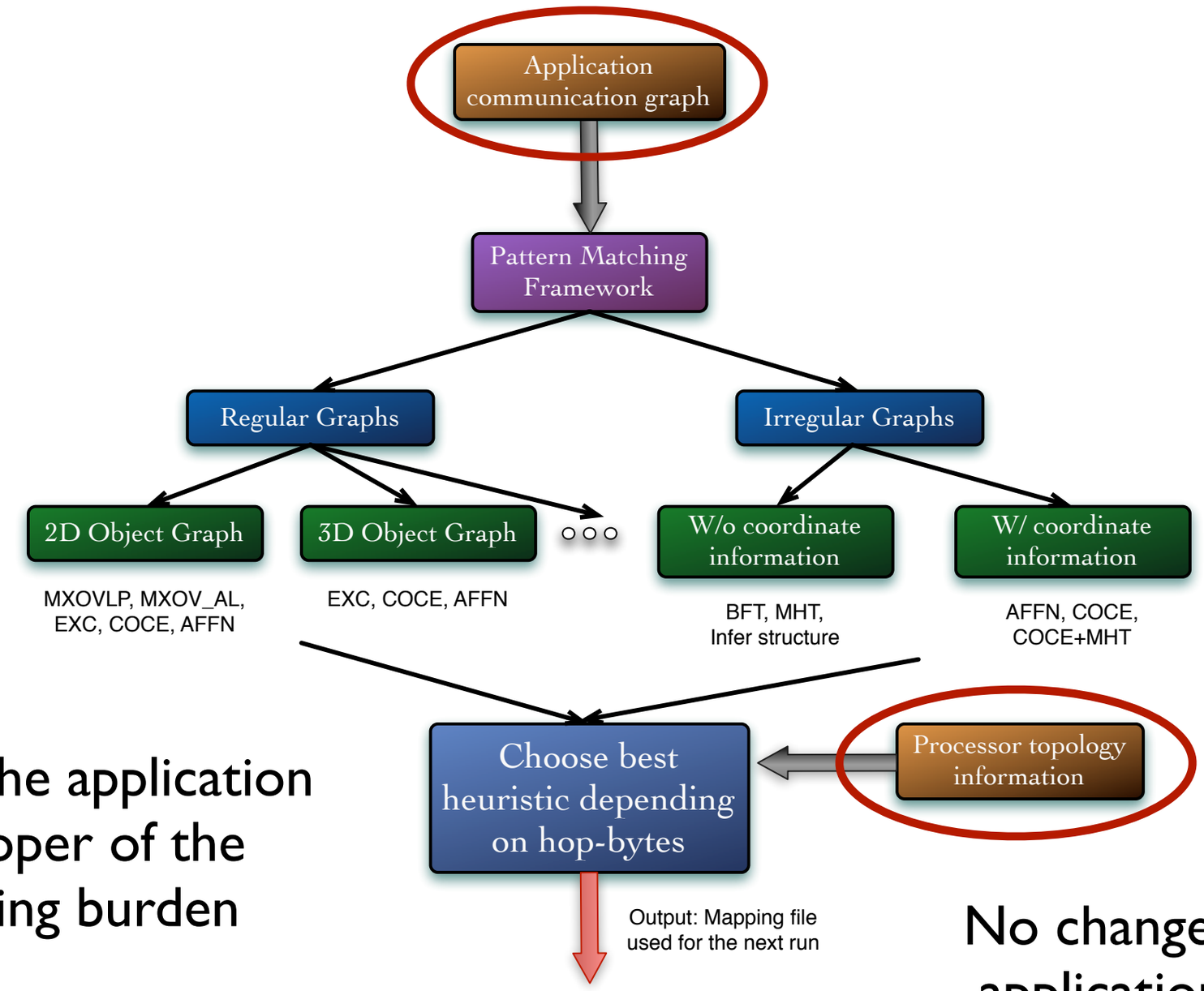


Automatic Mapping Framework



Relieve the application developer of the mapping burden

Automatic Mapping Framework



Relieve the application developer of the mapping burden

No change to the application code

Two different scenarios

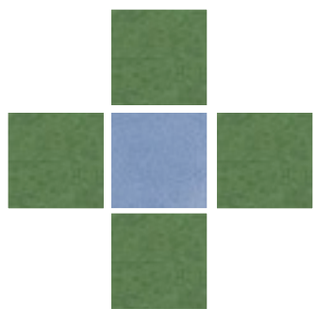
- There is no spatial information associated with the node
  - Option 1: Work without it
  - Option 2: If we know that the simulation has a geometric configuration, try to infer the structure of the graph
- We have geometric coordinate information for each node
  - Use coordinate information to avoid crossing of edges and for other optimizations

Finding the nearest available processor

- Algorithms in this paper:
  - pick a vertex in the graph to map
  - find a "desirable" processor to place it on
- Spiraling
- Quadtree

Finding the nearest available processor

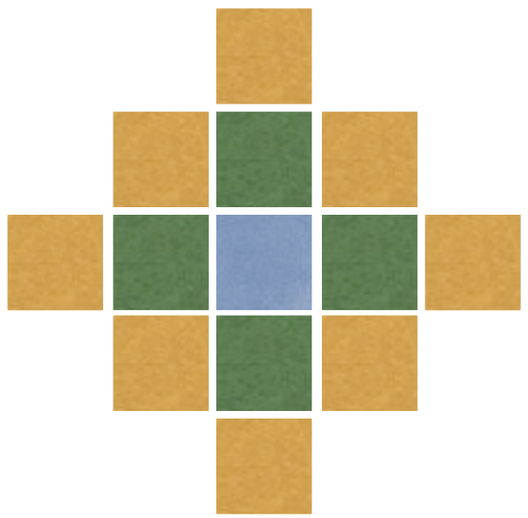
- Algorithms in this paper:
  - pick a vertex in the graph to map
  - find a "desirable" processor to place it on



- Spiraling
- Quadtree

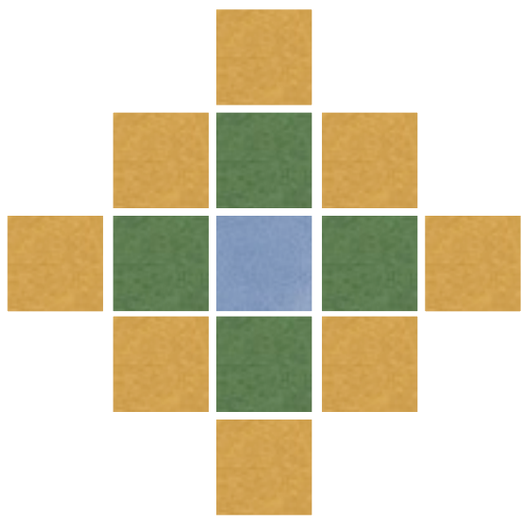
Finding the nearest available processor

- Algorithms in this paper:
  - pick a vertex in the graph to map
  - find a "desirable" processor to place it on
- Spiraling
- Quadtree

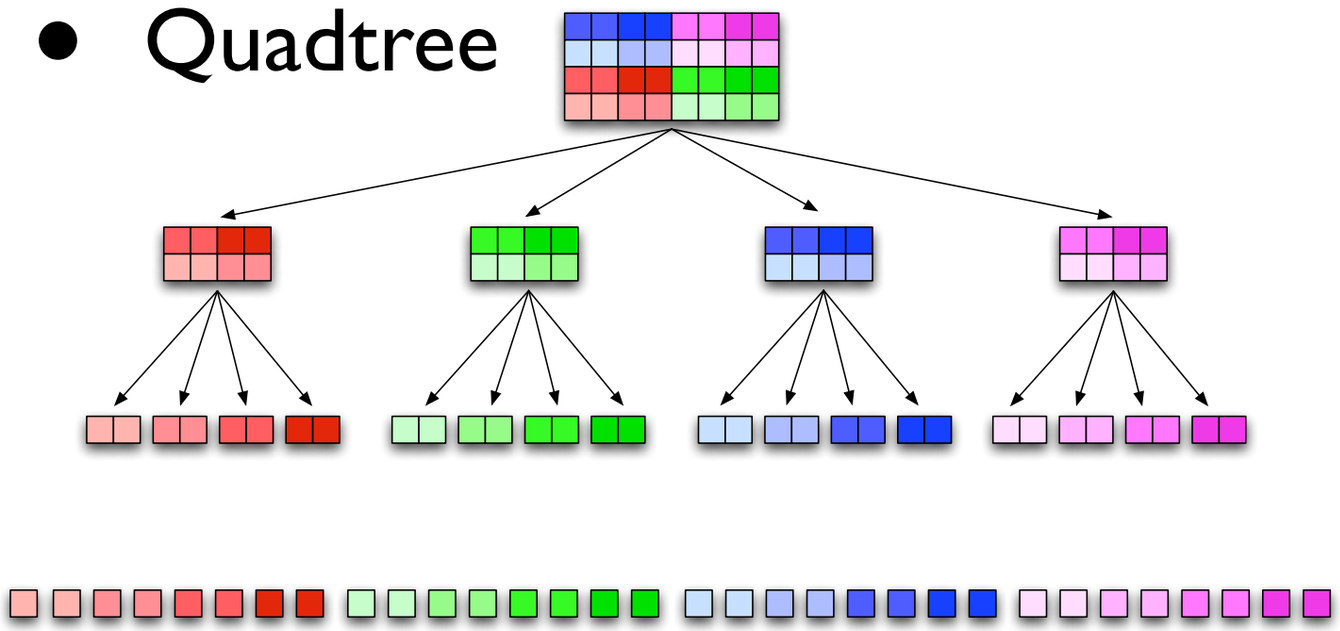


Finding the nearest available processor

- Algorithms in this paper:
  - pick a vertex in the graph to map
  - find a "desirable" processor to place it on

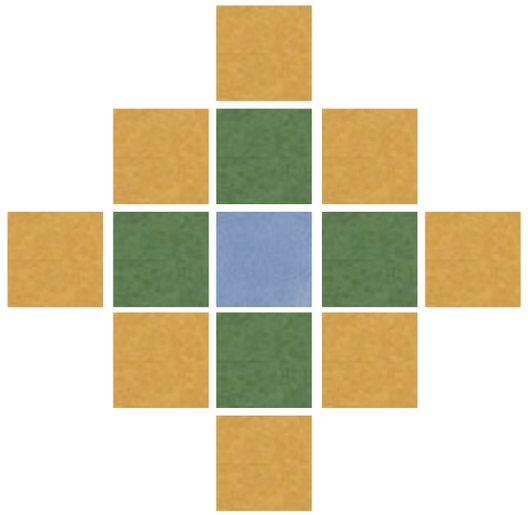


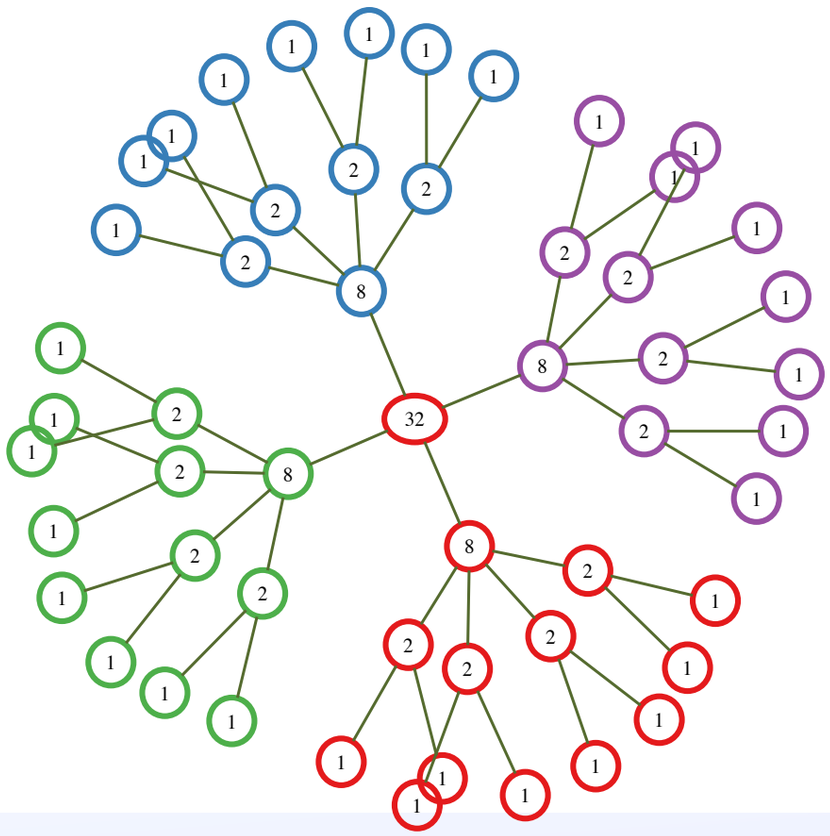
- Spiraling
- Quadtree

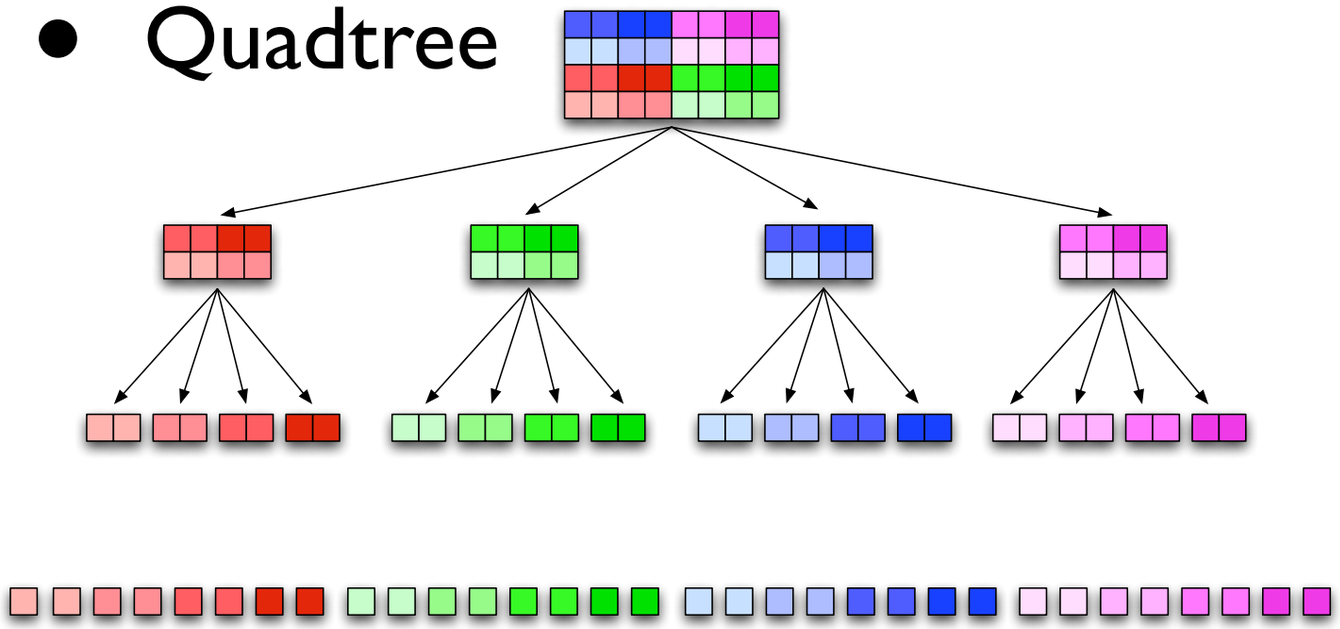


Finding the nearest available processor

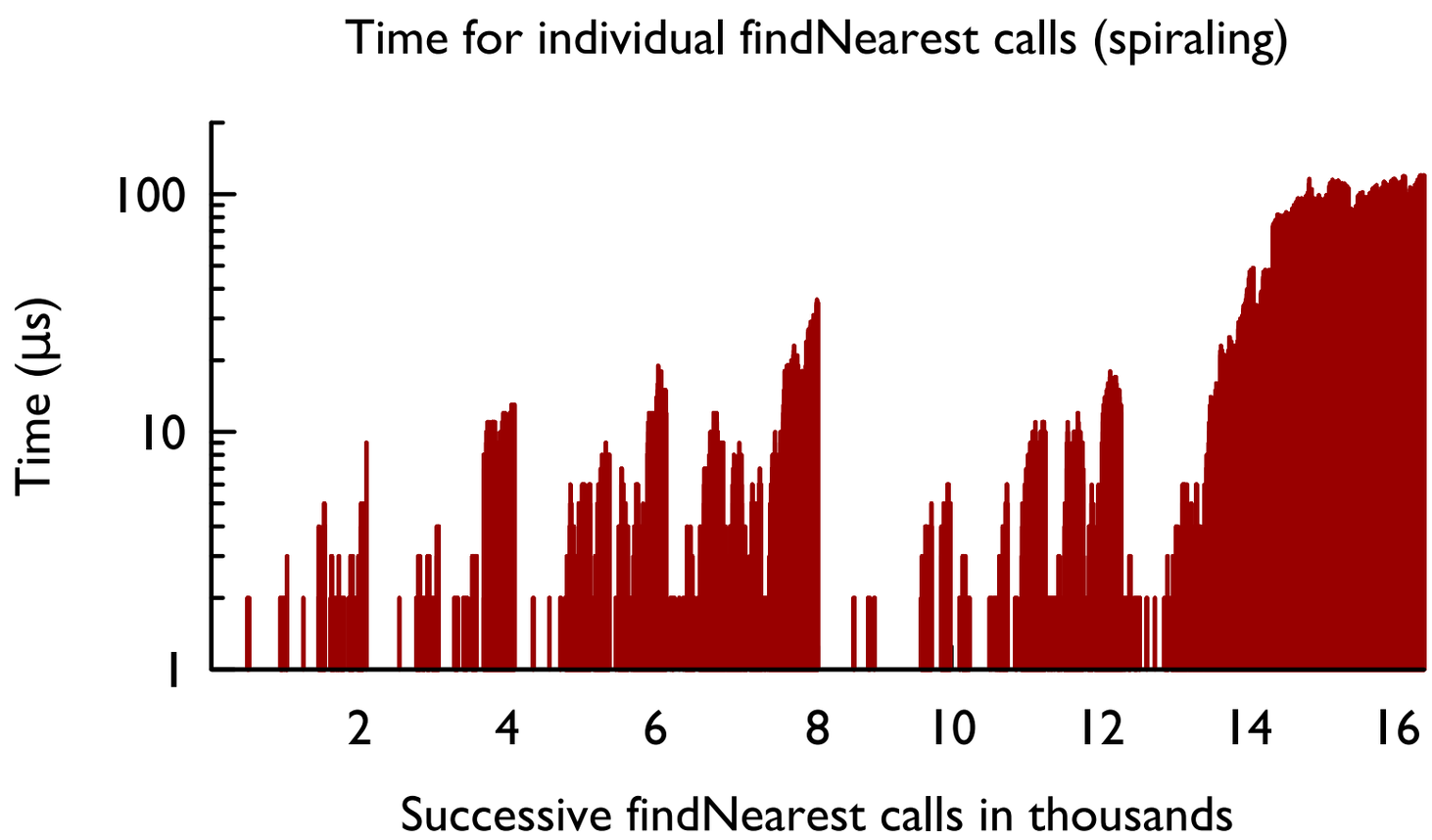
- Algorithms in this paper:
  - pick a vertex in the graph to map
  - find a "desirable" processor to place it on
- Spiraling
- Quadtree



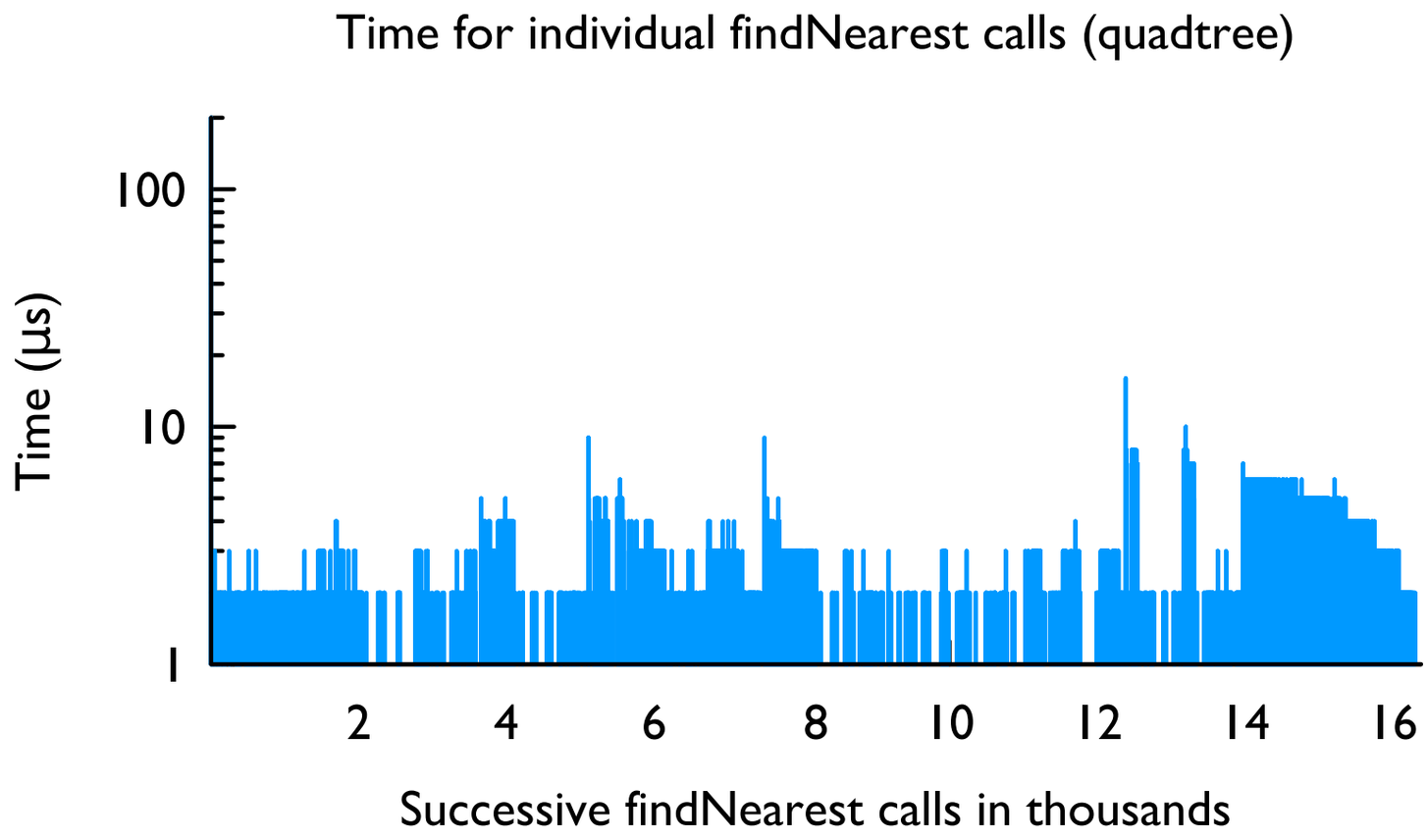




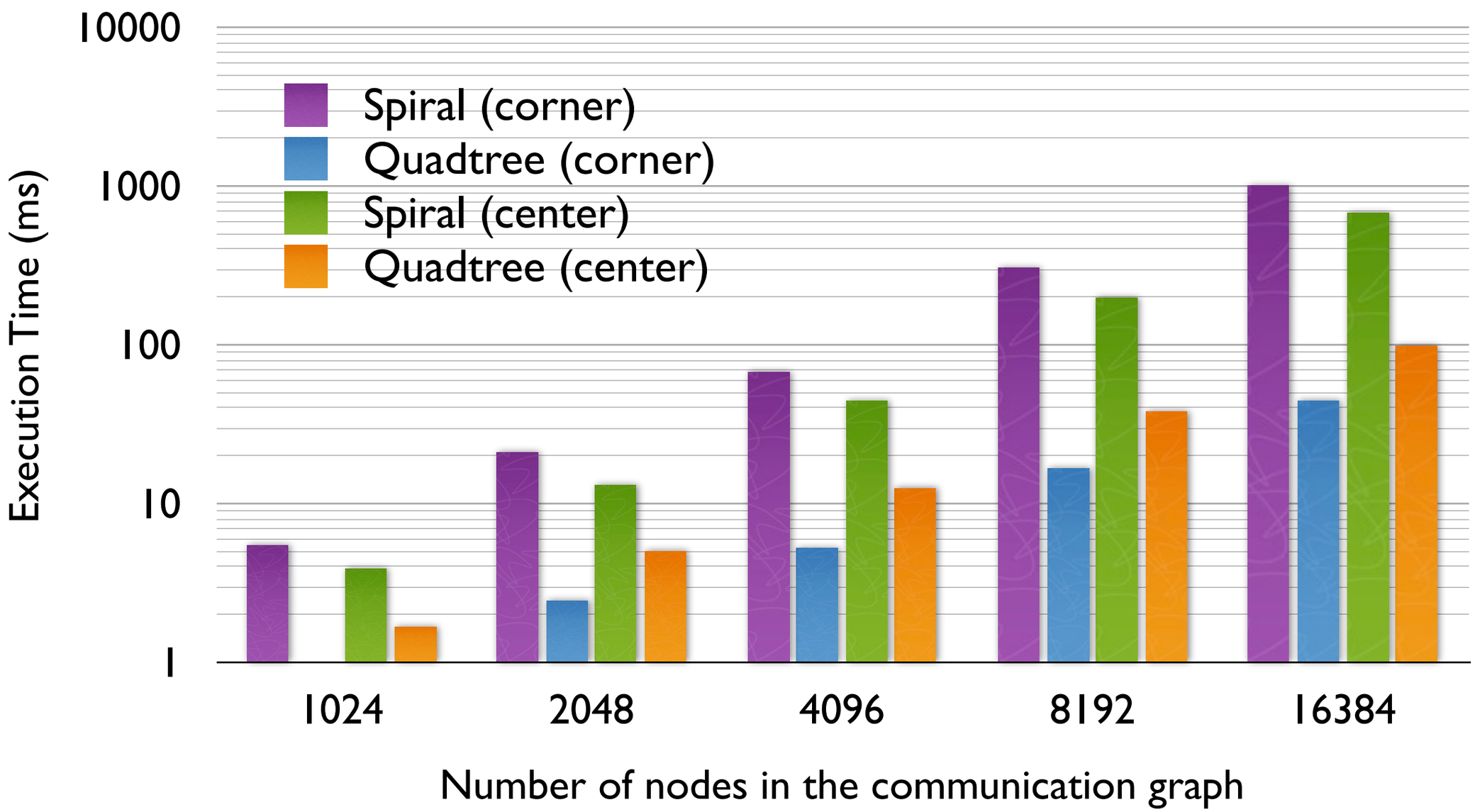
Finding the nearest available processor



Finding the nearest available processor

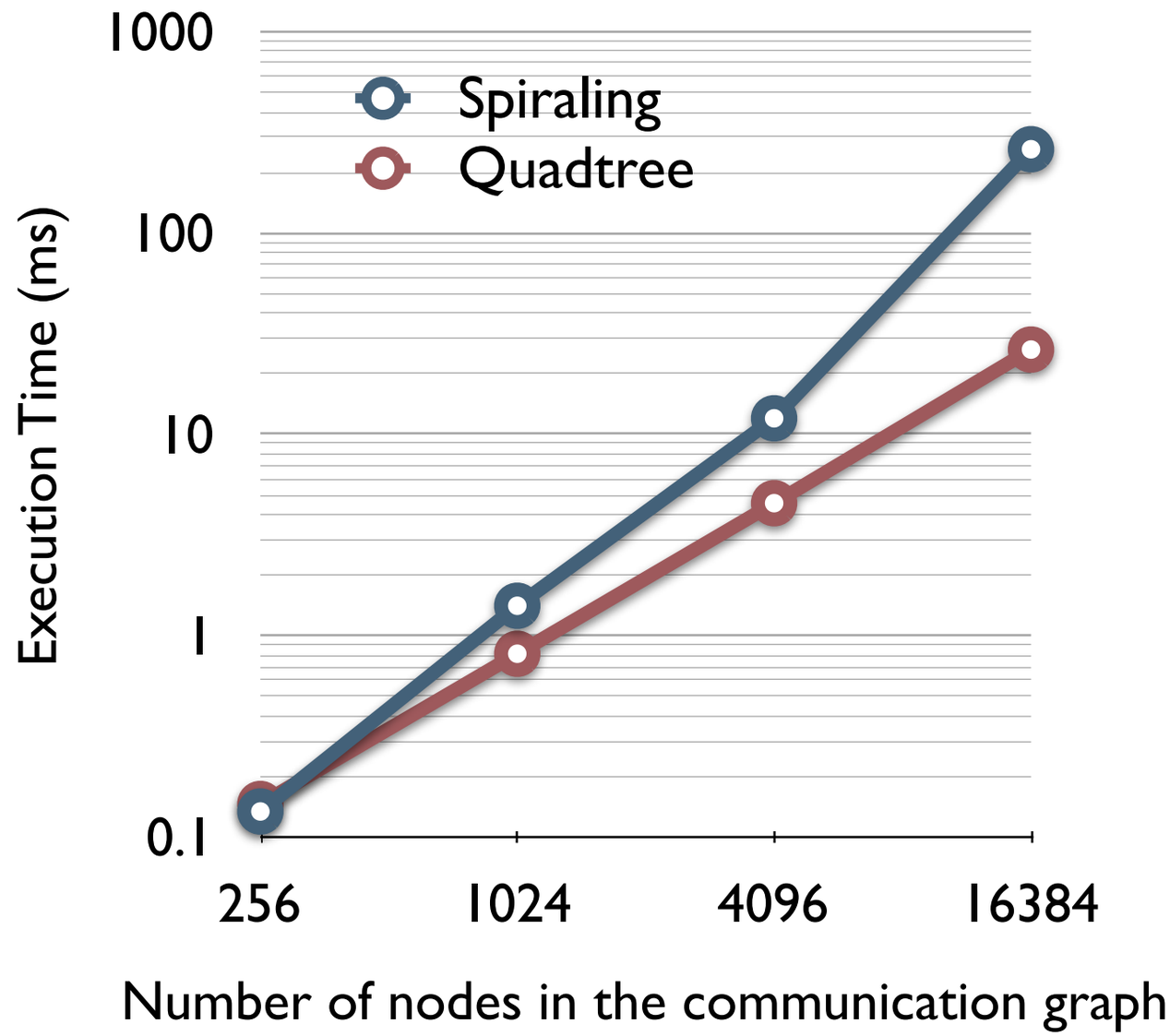


Spiraling vs Quadtree

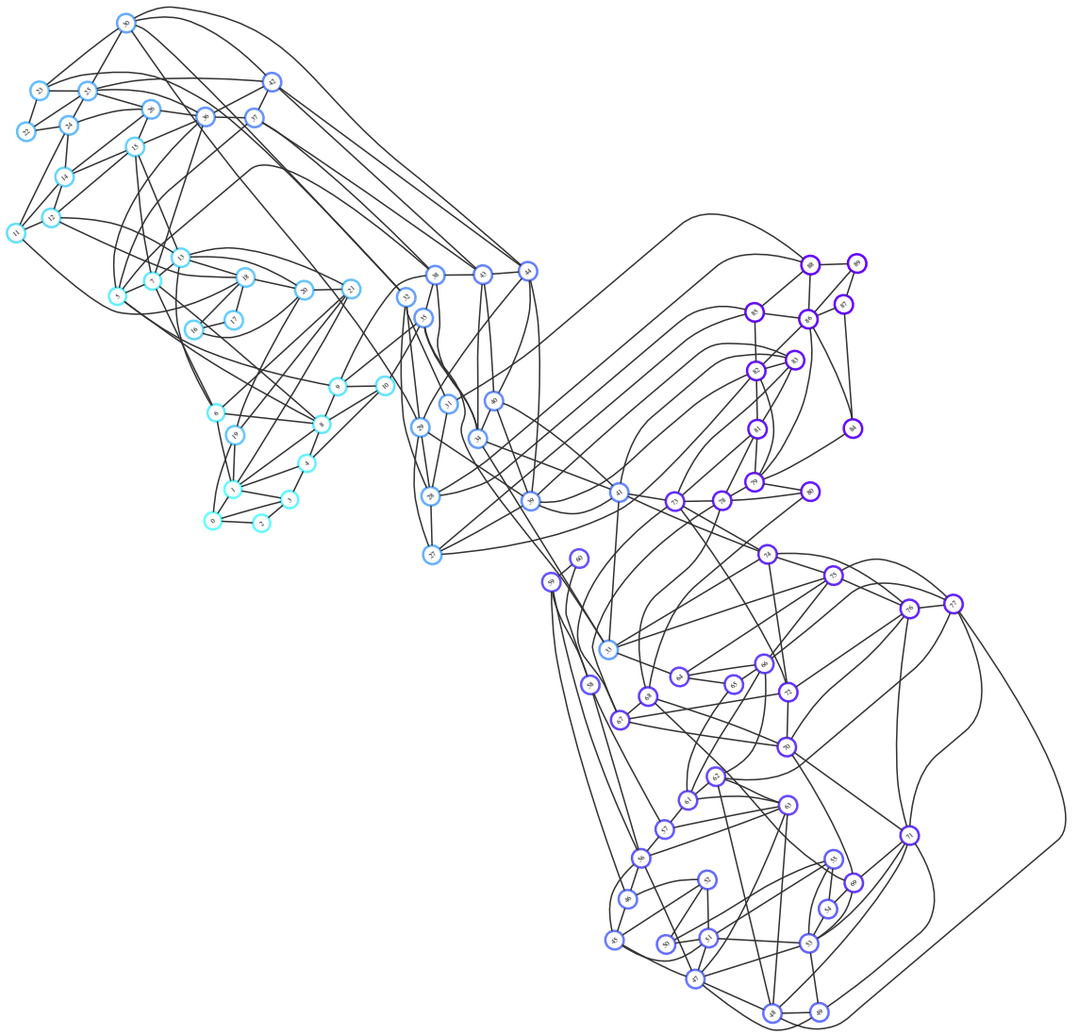


Spiraling vs Quadtree

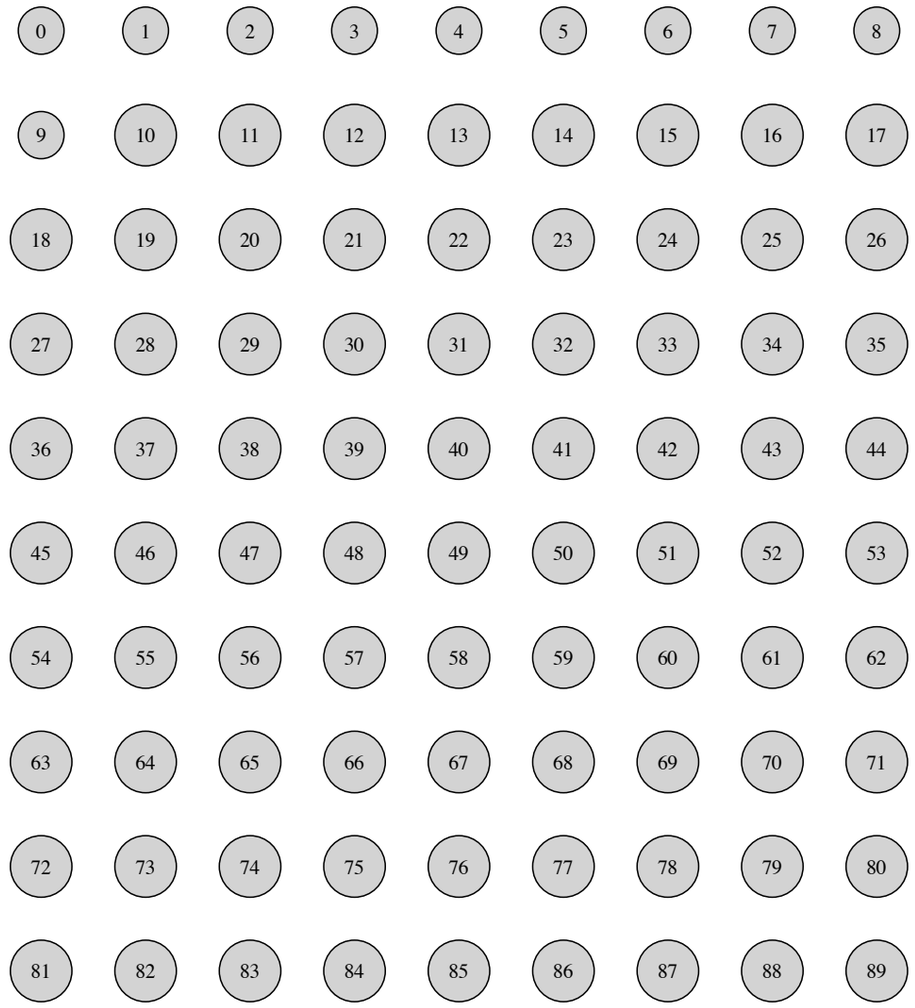
- Performance when called from AFFN: a mapping algorithm



Mapping Irregular Graphs



Object graph: 90 nodes



Processor Mesh: 10 x 9

No coordinate information

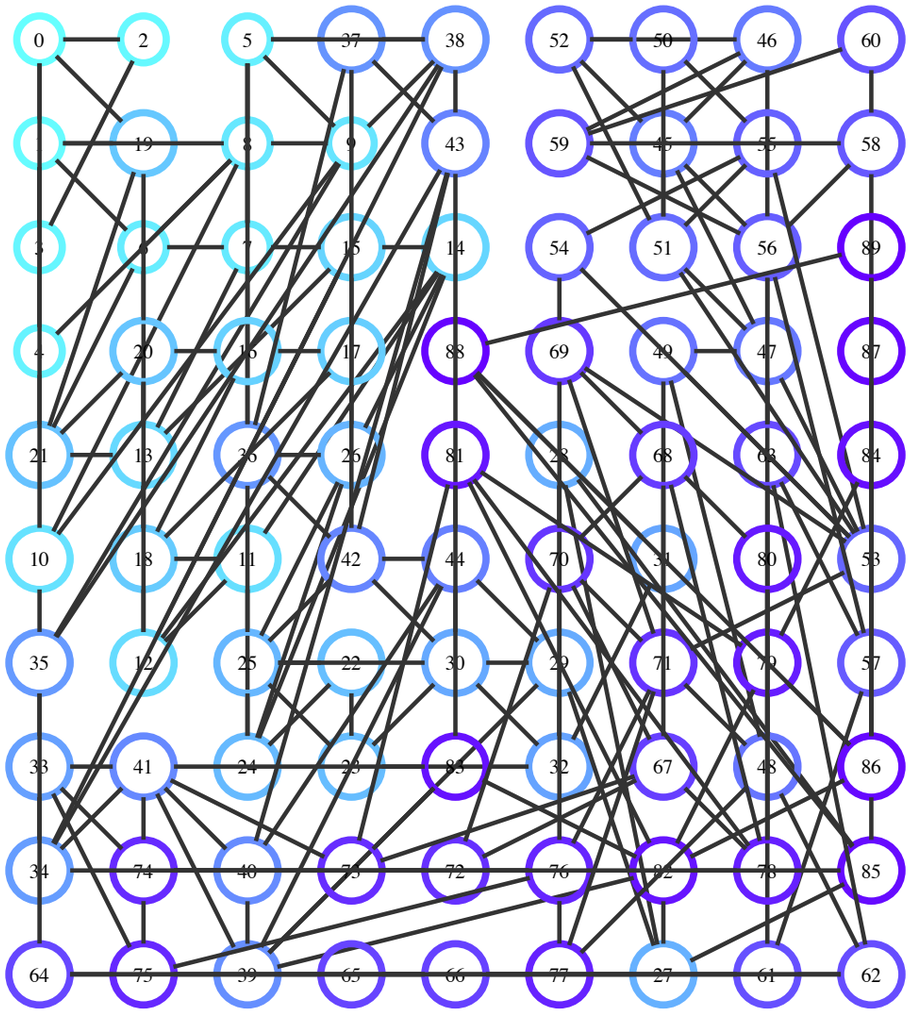
No coordinate information

- Breadth first traversal (BFT)
  - Start with a random node and one end of the processor mesh
  - Map nodes as you encounter them close to their parent

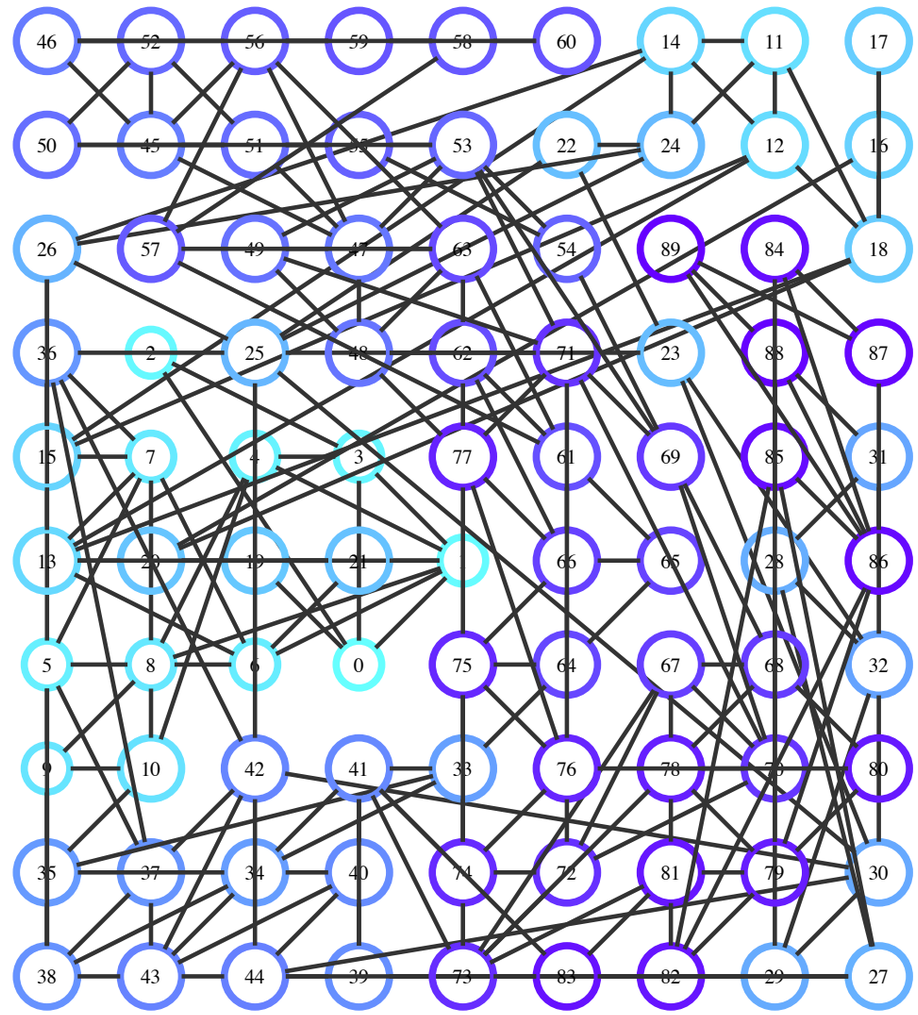
No coordinate information

- Breadth first traversal (BFT)
  - Start with a random node and one end of the processor mesh
  - Map nodes as you encounter them close to their parent
- Max heap traversal (MHT)
  - Start with a random node and one end/center of the mesh
  - Put neighbors of a mapped node into the heap (node at the top is the one with maximum number of mapped neighbors)
  - Map elements in the heap one by one around the centroid of their mapped neighbors

Mapping visualization

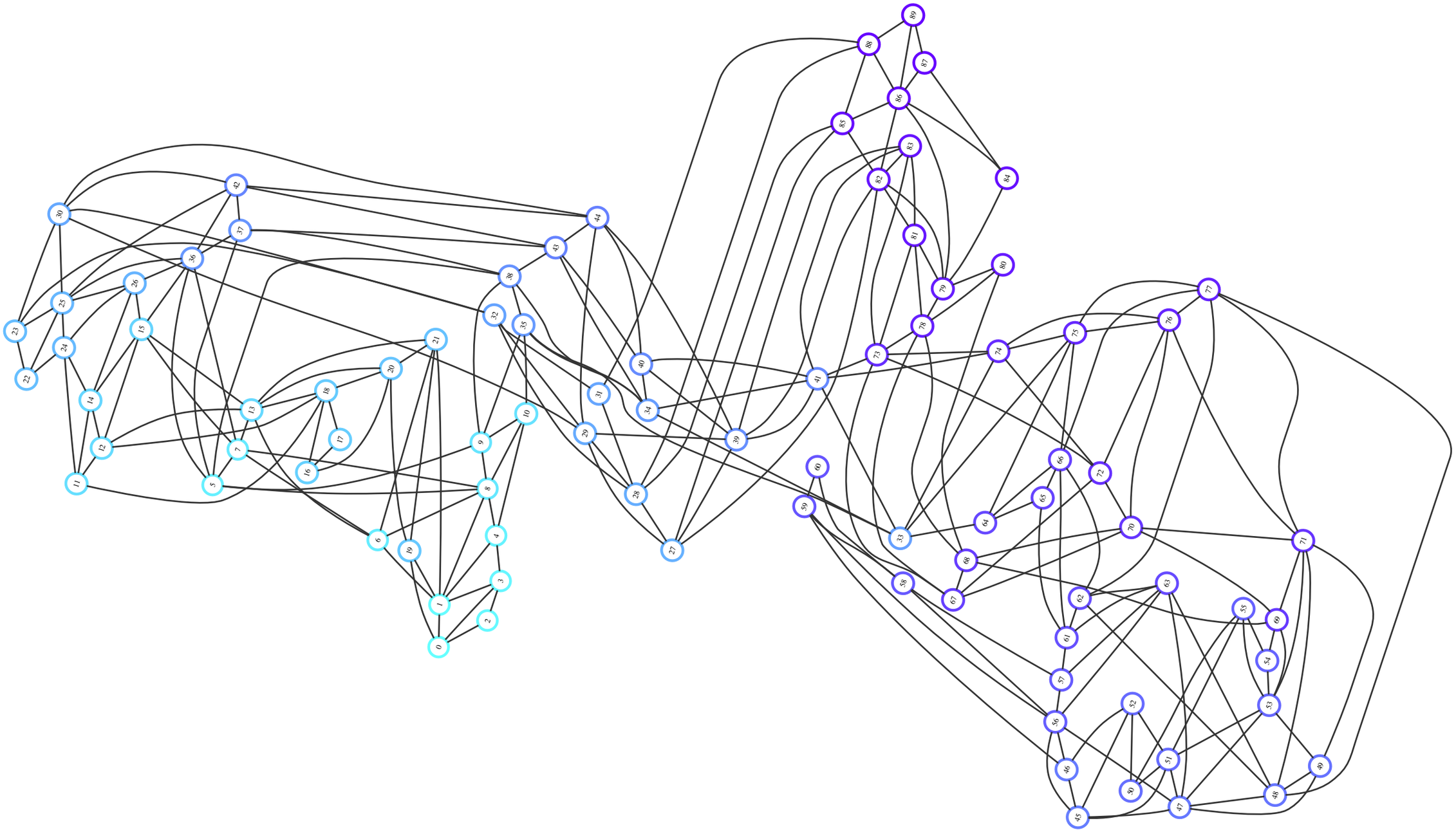


BFT: 2.89



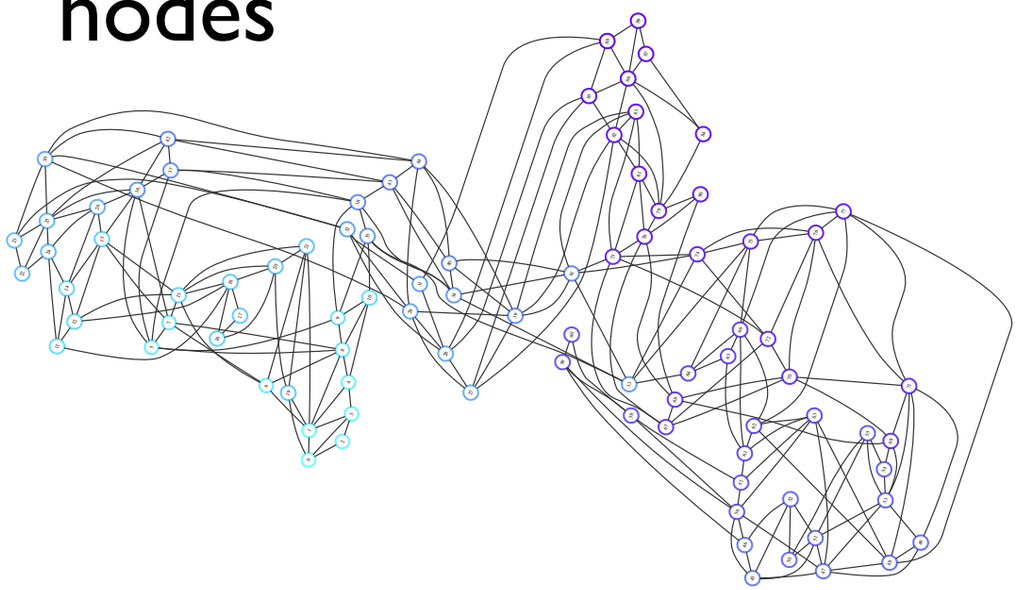
MHT: 2.69

Inferring the spatial placement



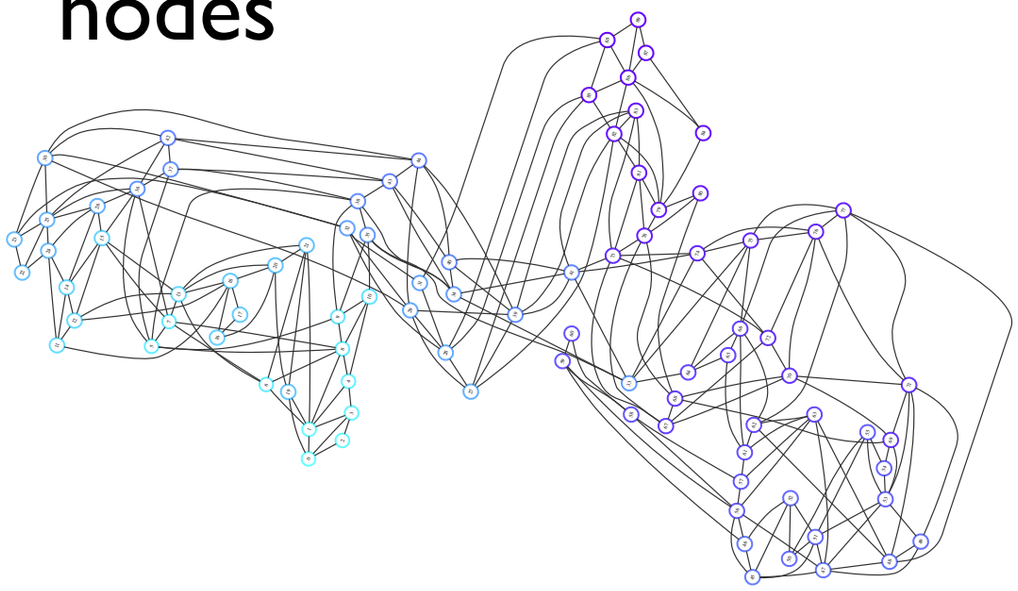
Inferring the spatial placement

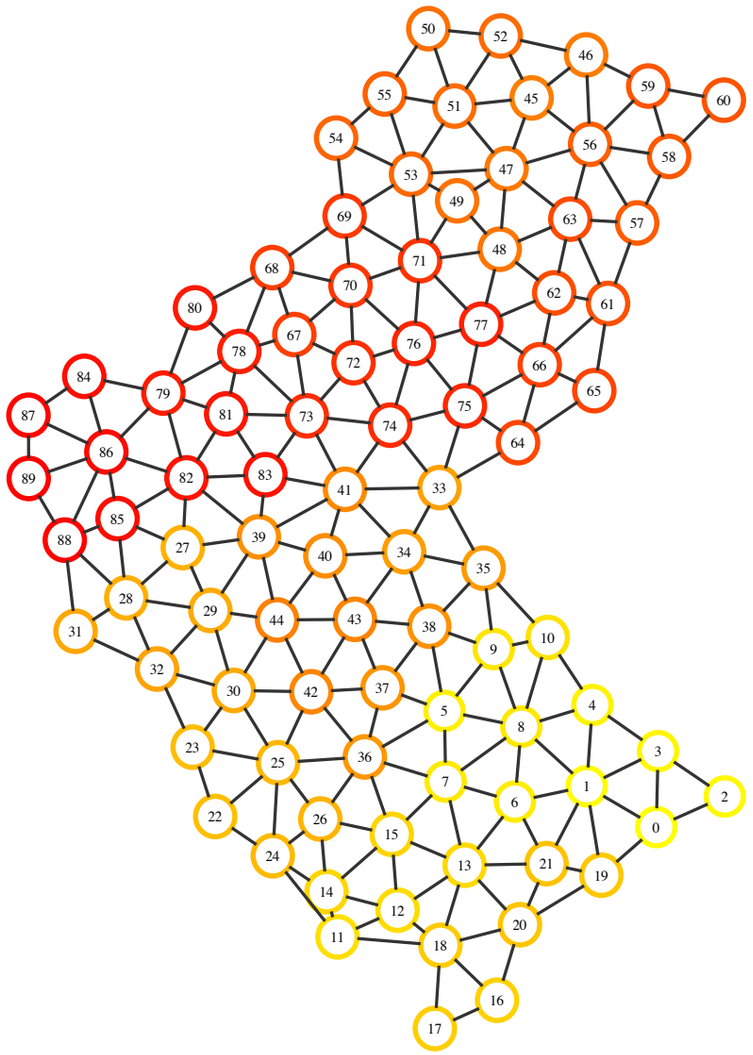
- Graph layout algorithms
  - Force-based layout to reduce the total energy in the system
- Use the graphviz library to obtain coordinates of the nodes



Inferring the spatial placement

- Graph layout algorithms
  - Force-based layout to reduce the total energy in the system
- Use the graphviz library to obtain coordinates of the nodes

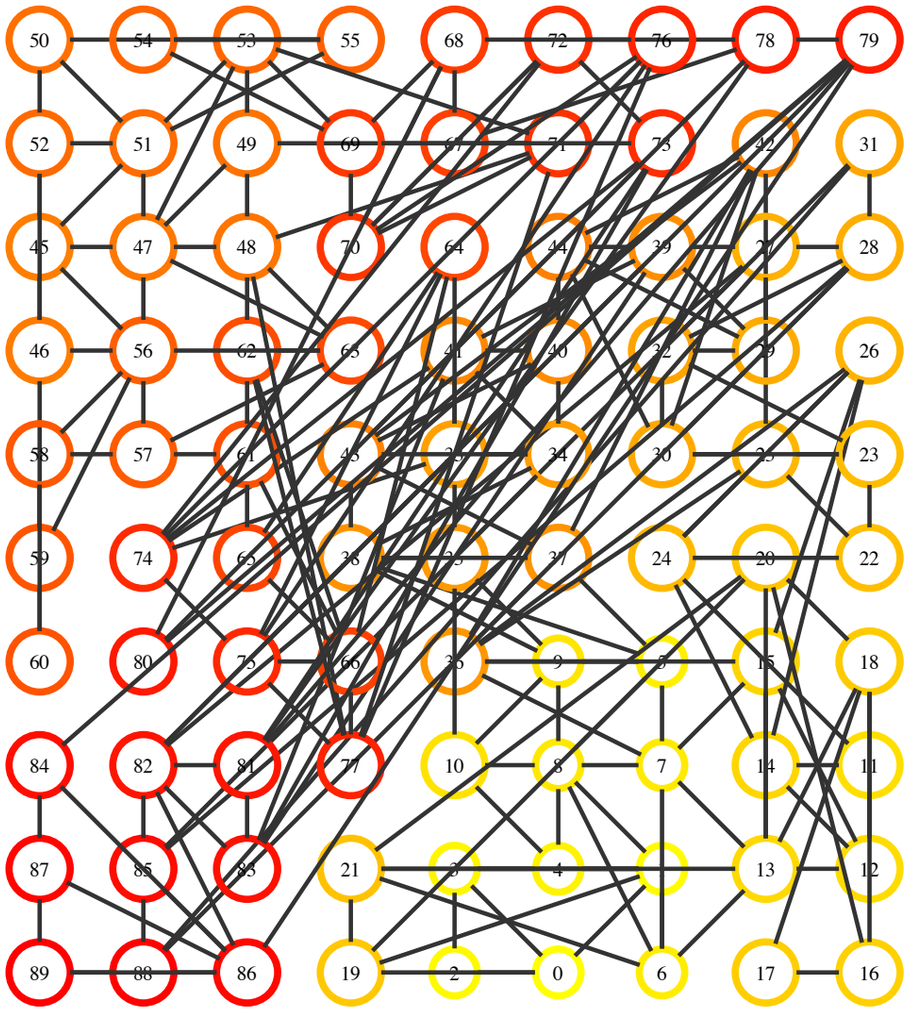




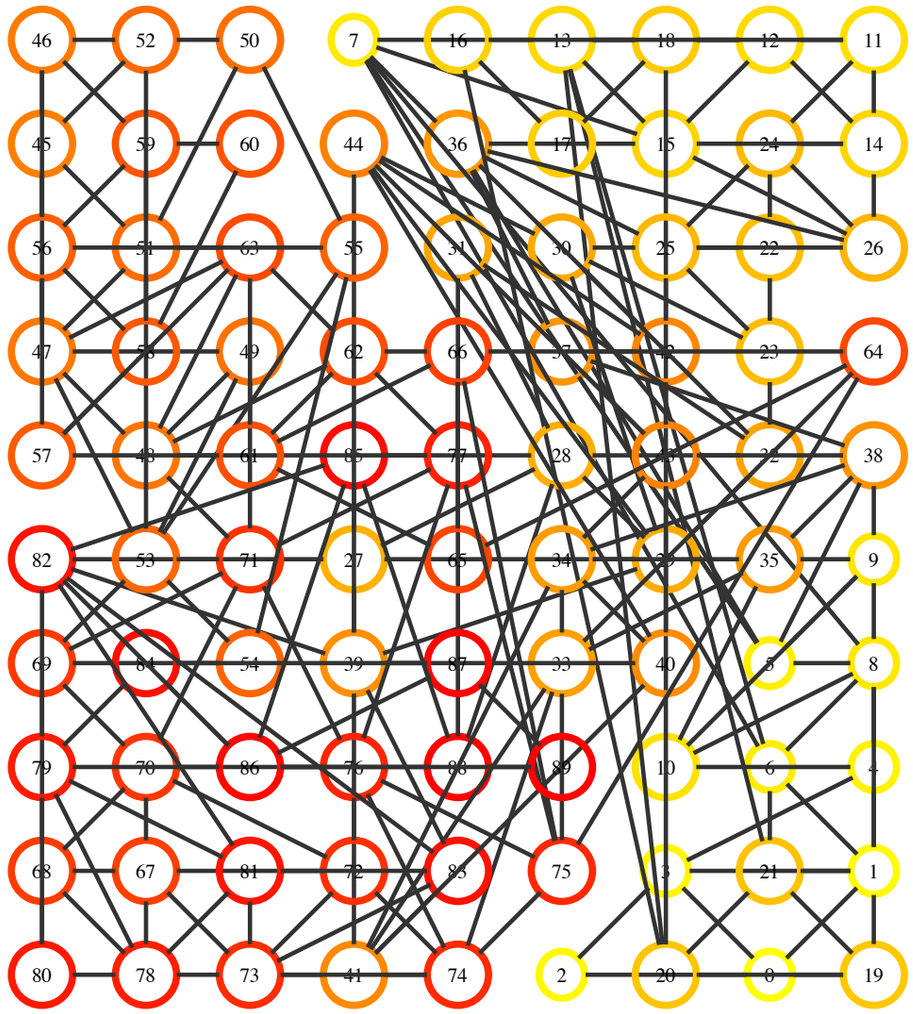
With coordinate information

- Affine Mapping (AFFN)
  - Stretch/shrink the object graph (based on coordinates of nodes) to map it on to the processor grid
  - In case of conflicts for the same processor, find the nearest available processor
- Corners to Center (COCE)
  - Use four corners of the object graph based on coordinates
  - Start mapping simultaneously from all sides
    - Place nodes encountered during a BFT close to their parents

Mapping visualization

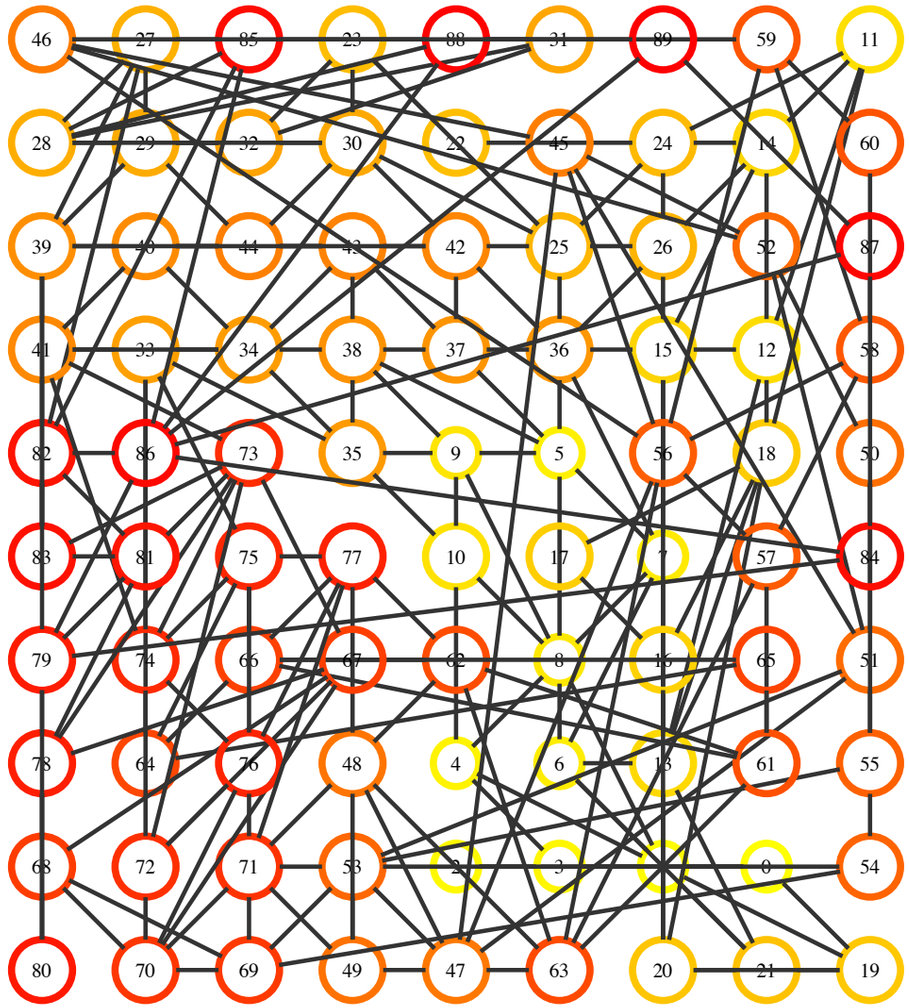


AFFN: 3.17



COCE: 2.88

- COCE+MHT Hybrid:
  - We fix four nodes at geometric corners of the mesh to four processors in 2D
  - Put neighbors of these nodes into a max heap
- Map from all sides inwards
  - Starting from centroid of mapped neighbors



COCE: 2.78

Time Complexity

Time Complexity

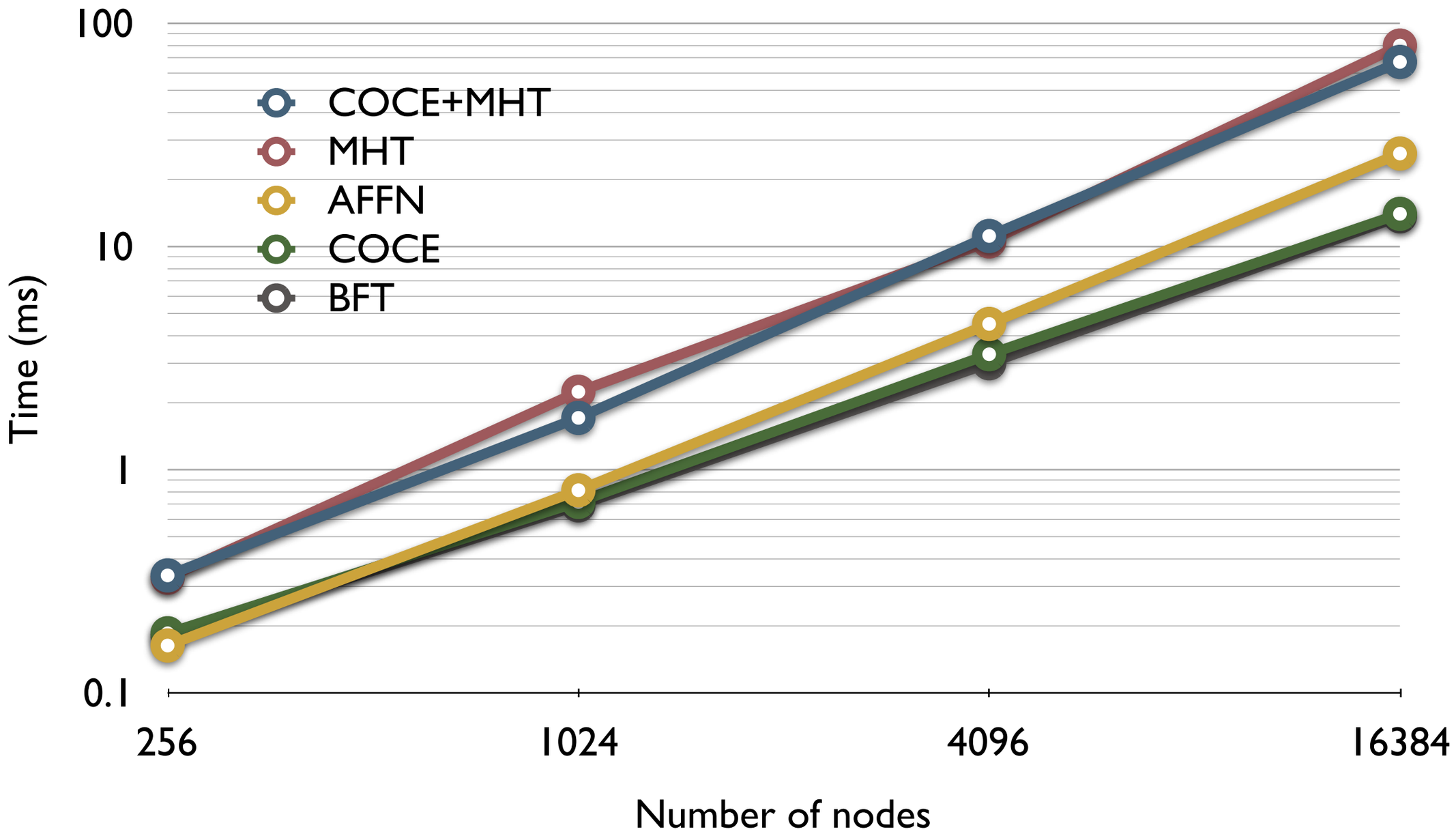
- All algorithms discussed above choose a desired processor and spiral around it to find the nearest available processor
  - Heuristics generally applicable to any topology

Time Complexity

- All algorithms discussed above choose a desired processor and spiral around it to find the nearest available processor
  - Heuristics generally applicable to any topology
- Depending on the running time of findNearest:

BFT	COCE	AFFN	MHT	COCE+MHT
O(n)	O(n)	O(n)	O(n logn)	O(n logn)
$O(n\ (\log n)^2)$	$O(n\ (\log n)^2)$	$O(n\ (\log n)^2)$	$O(n\ (\log n)^2)$	$O(n\ (\log n)^2)$

Running Time



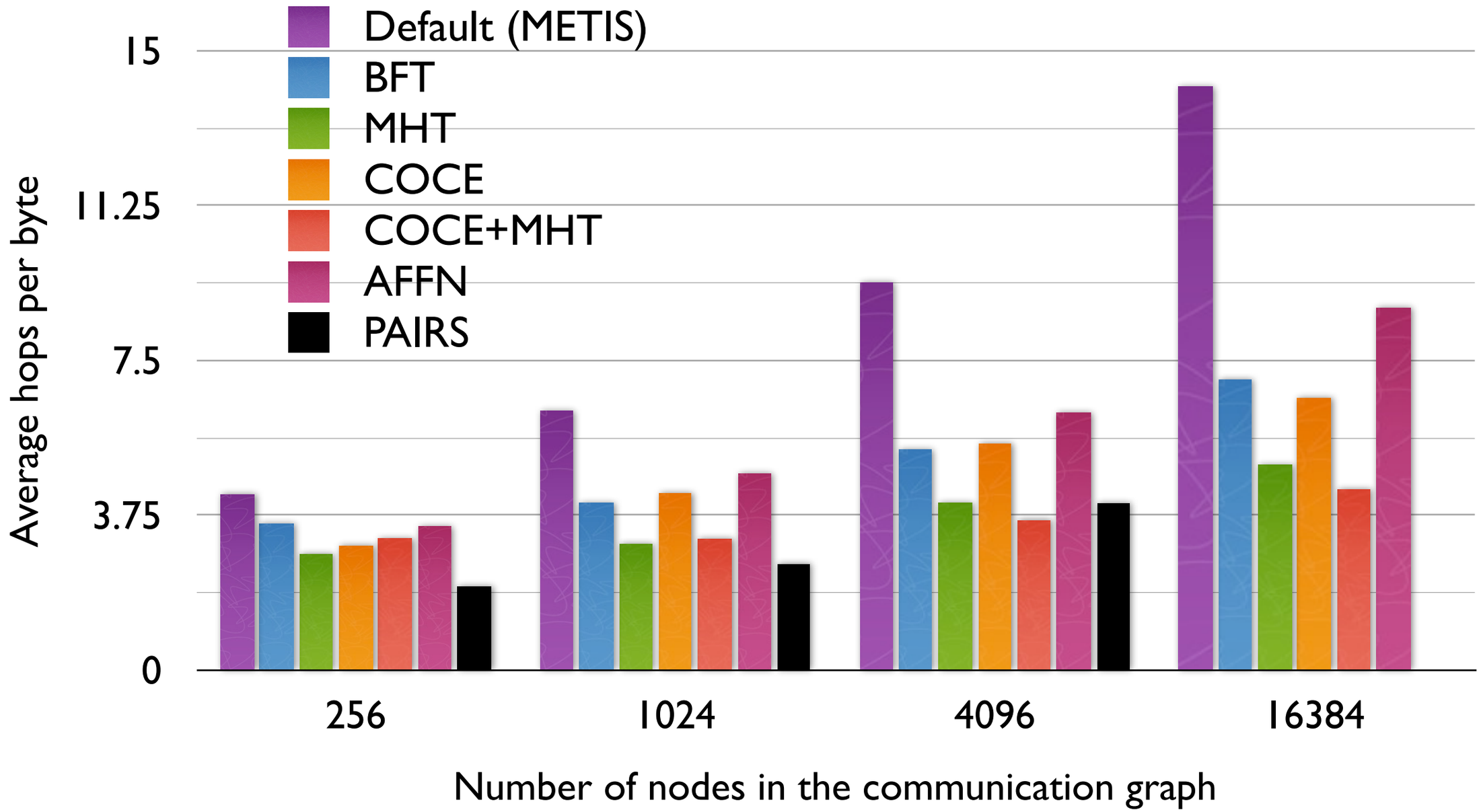
Evaluation

- Metric for comparison: hop-bytes

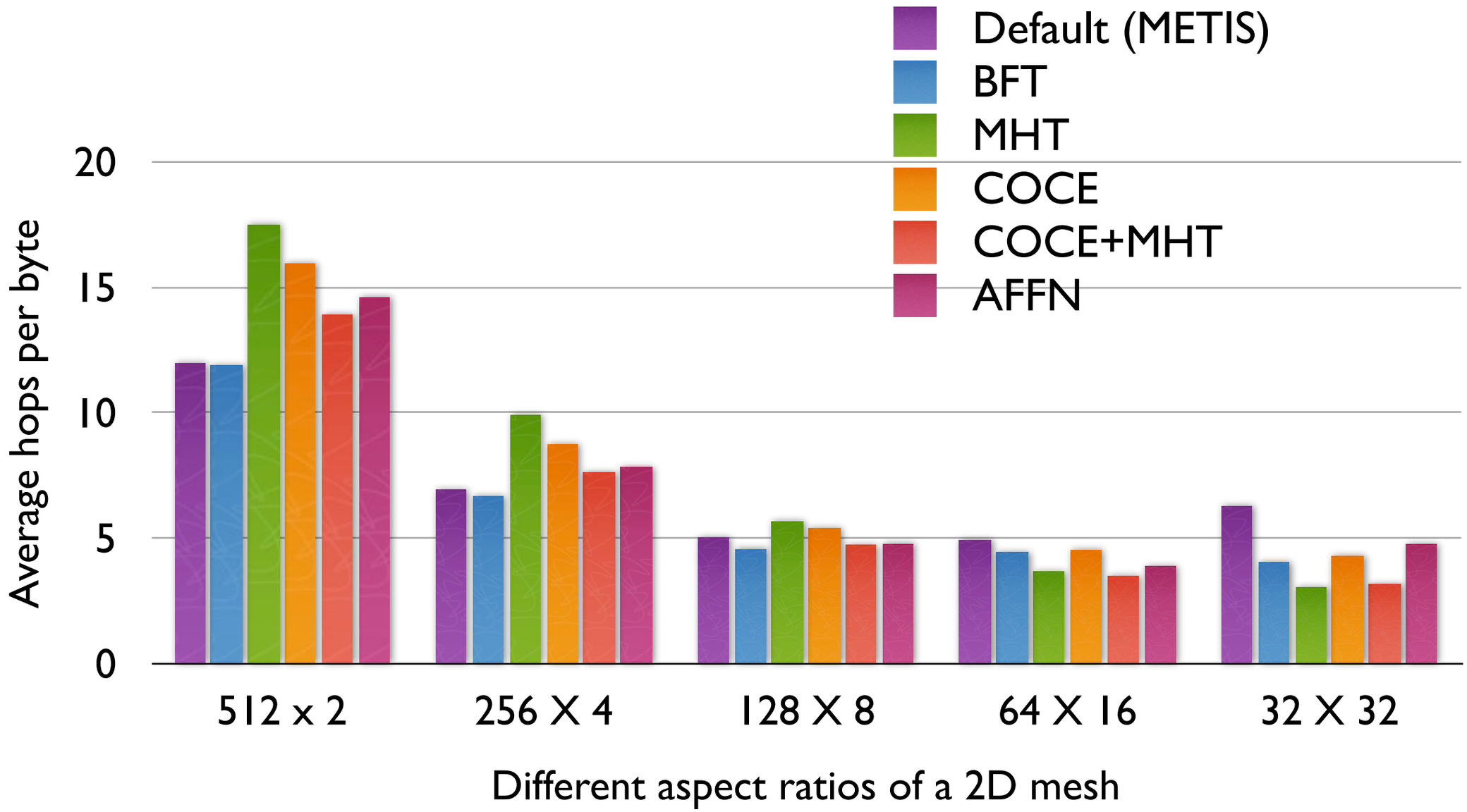
$$average\ hops\ per\ byte = \left(\sum_{i=1}^{n} d_i \times b_i\right) \div \left(\sum_{i=1}^{n} b_i\right)$$

- Indicates amount of traffic and hence contention on the network
- Previously used metric: maximum dilation

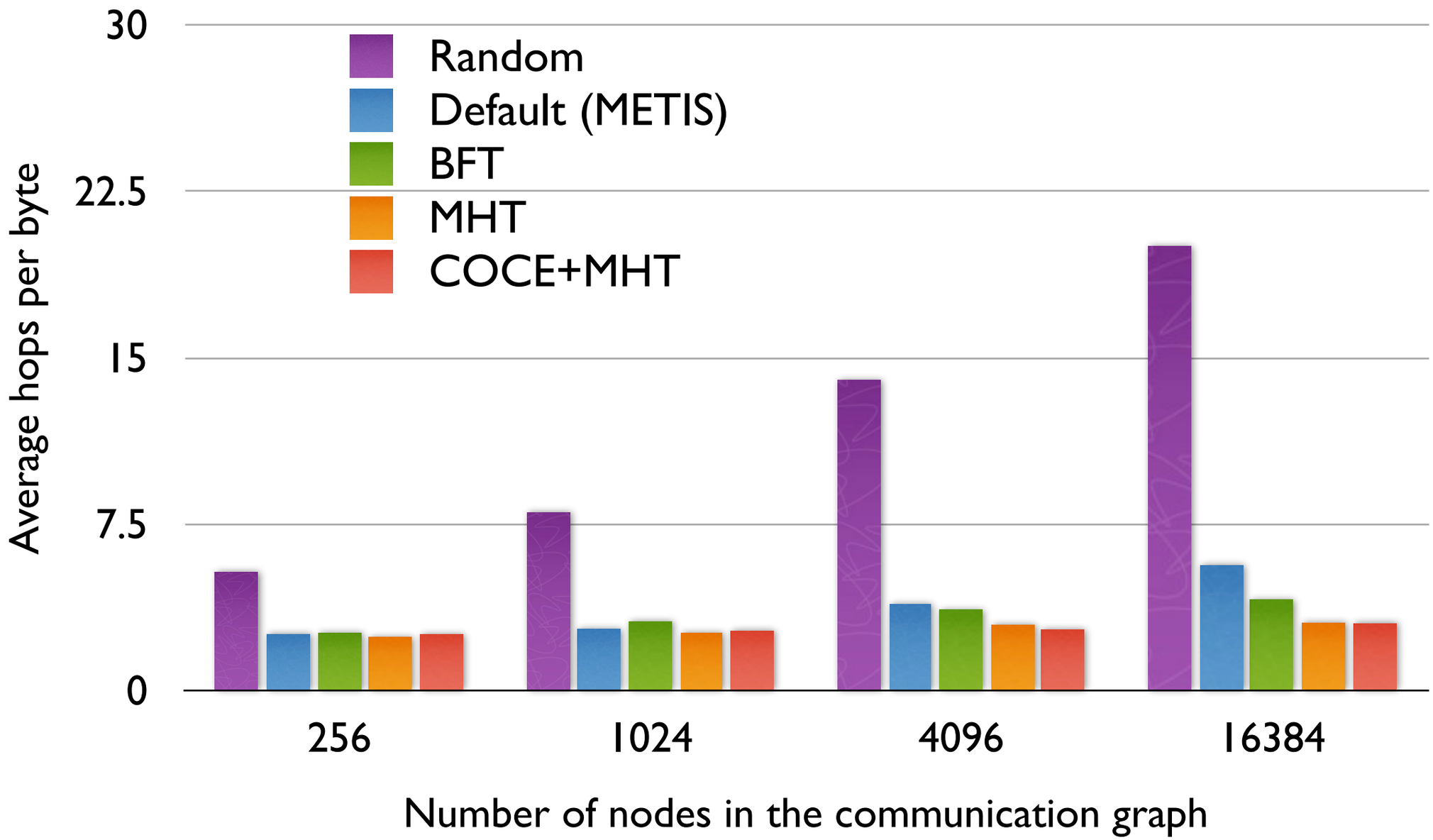
Mapping Simple2D to a 2D mesh



Mapping Simple2D to a 2D mesh



Mapping Simple2D to a 3D mesh



Summary

- Heuristics for mapping irregular graphs to mesh topologies
  - best heuristic chosen at runtime (based on hop-bytes)
- Mapping library to help the application developer
- Extensible to other topologies also

Lawrence Livermore National Laboratory

Questions?

Abhinav Bhatele, Automating Topology Aware Mapping for Supercomputers, PhD Thesis, Department of Computer Science, University of Illinois. http://hdl.handle.net/2142/16578

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551