Hierarchical Parallel A* Algorithm

Abhishek Subramaniam

CSE 633: Parallel Algorithms

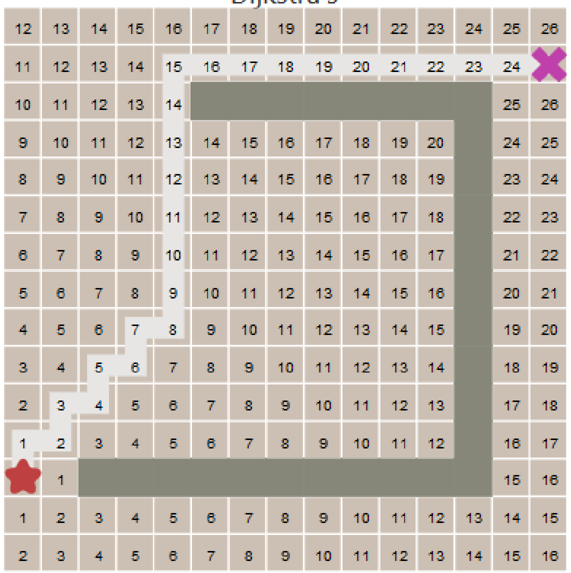
Instructor: Dr. Russ Miller

University at Buffalo The State University of New York

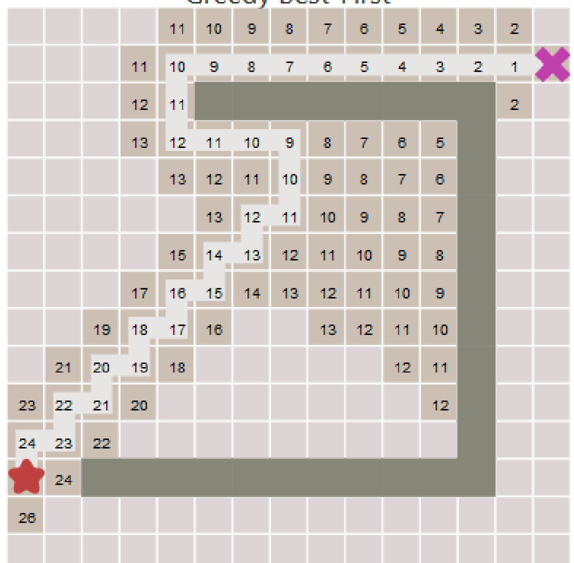
A* algorithm

- A* algorithm is a popular pathfinding algorithm used in game theory and navigation.
- It makes use of a heuristic cost function to find the solution quickly.
- The heuristic cost function uses the sum of two parameters, ‘current cost’ and ‘predicted cost’ to calculate the optimal path.
- All relevant nodes are kept in 2 lists, ‘visited’ and ‘next’.
- The implementation resembles a dynamic programming approach, with the heuristic cost function determining the order in which the nodes are visited.
- Examples of admissible heuristic functions to minimize distance include Manhattan distance and Euclidean distance.

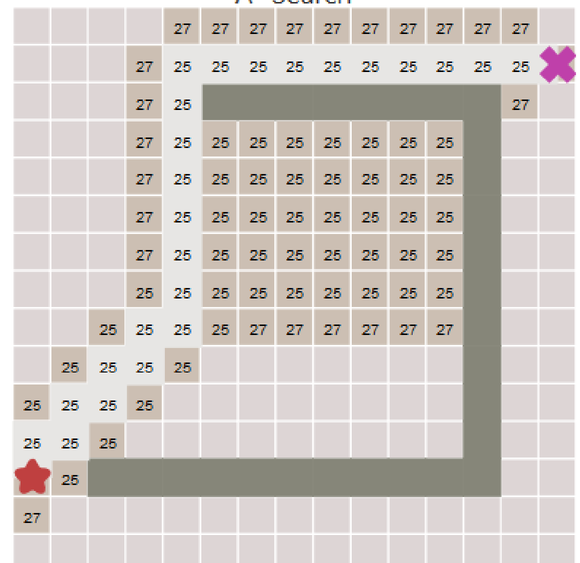
Dijkstra's



Greedy Best-First

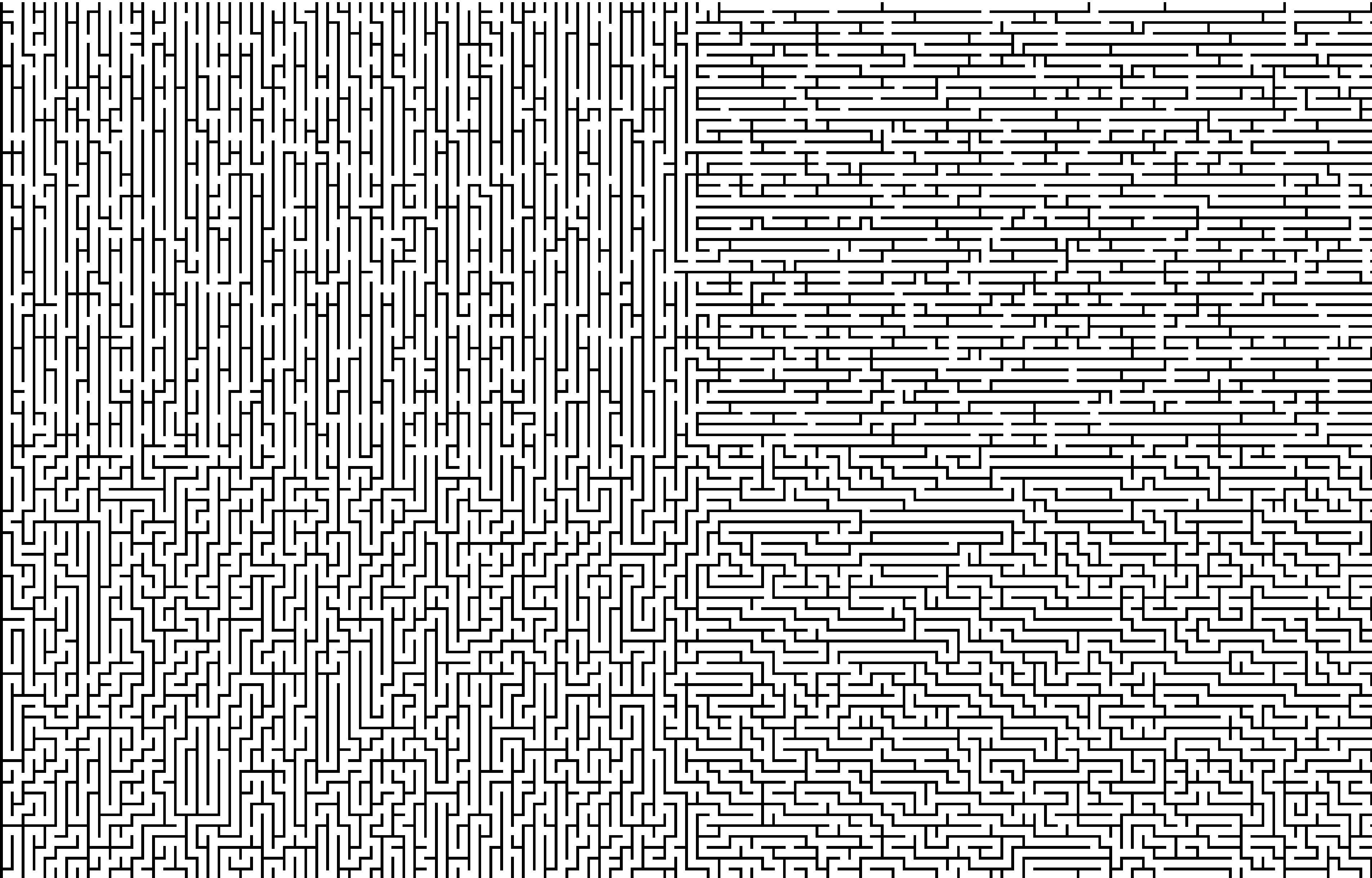


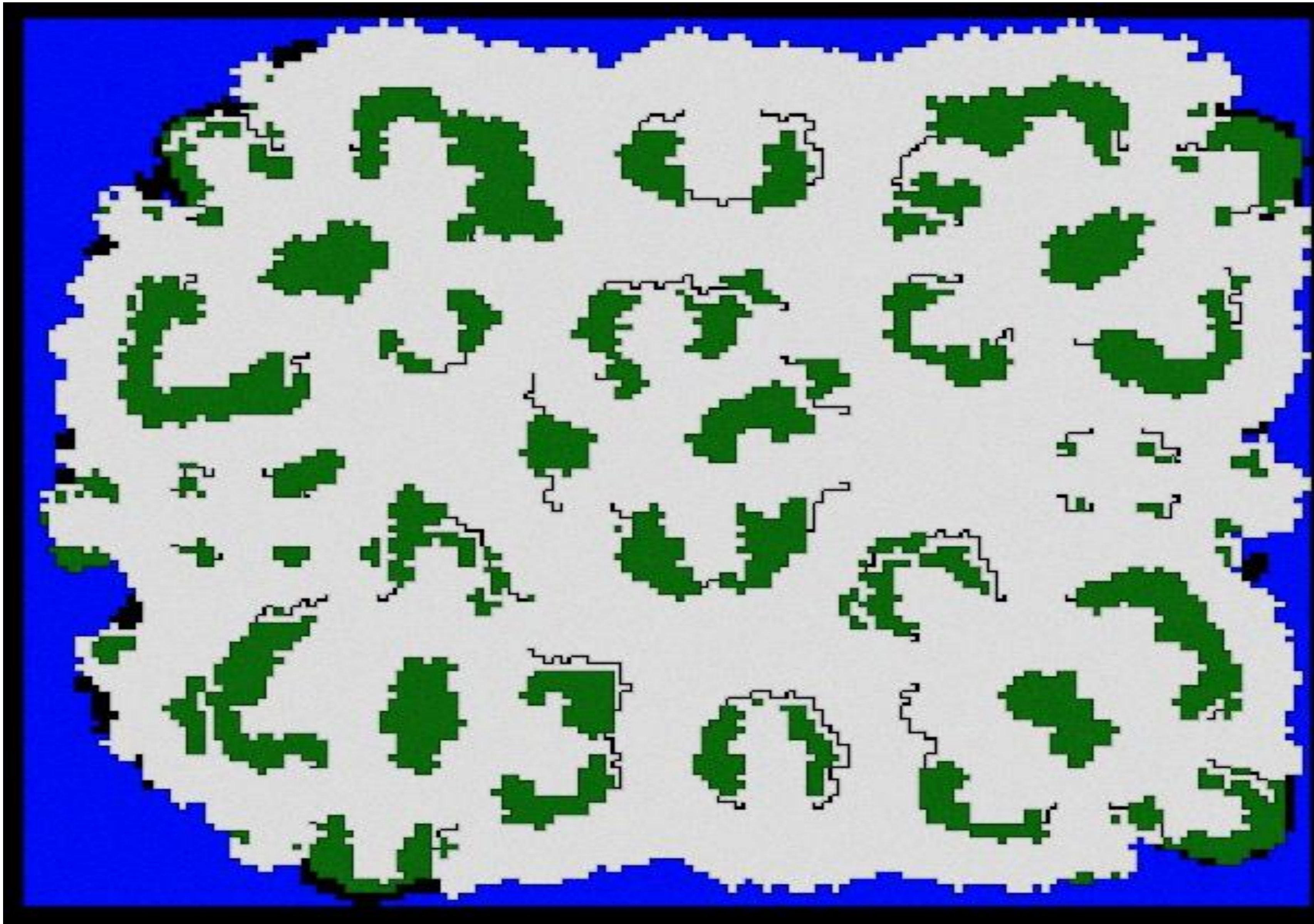
A* Search



Hierarchical Parallel A* Algorithm

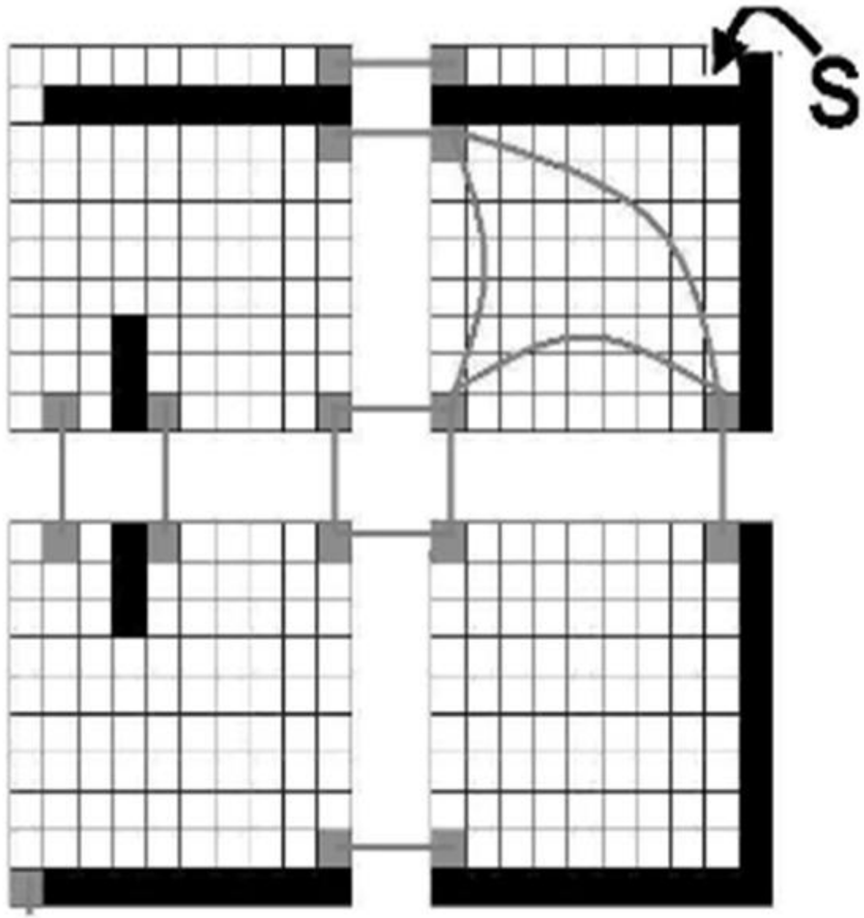
- The graph is broken down into equal size chunks.
- Each processor is assigned a chunk.

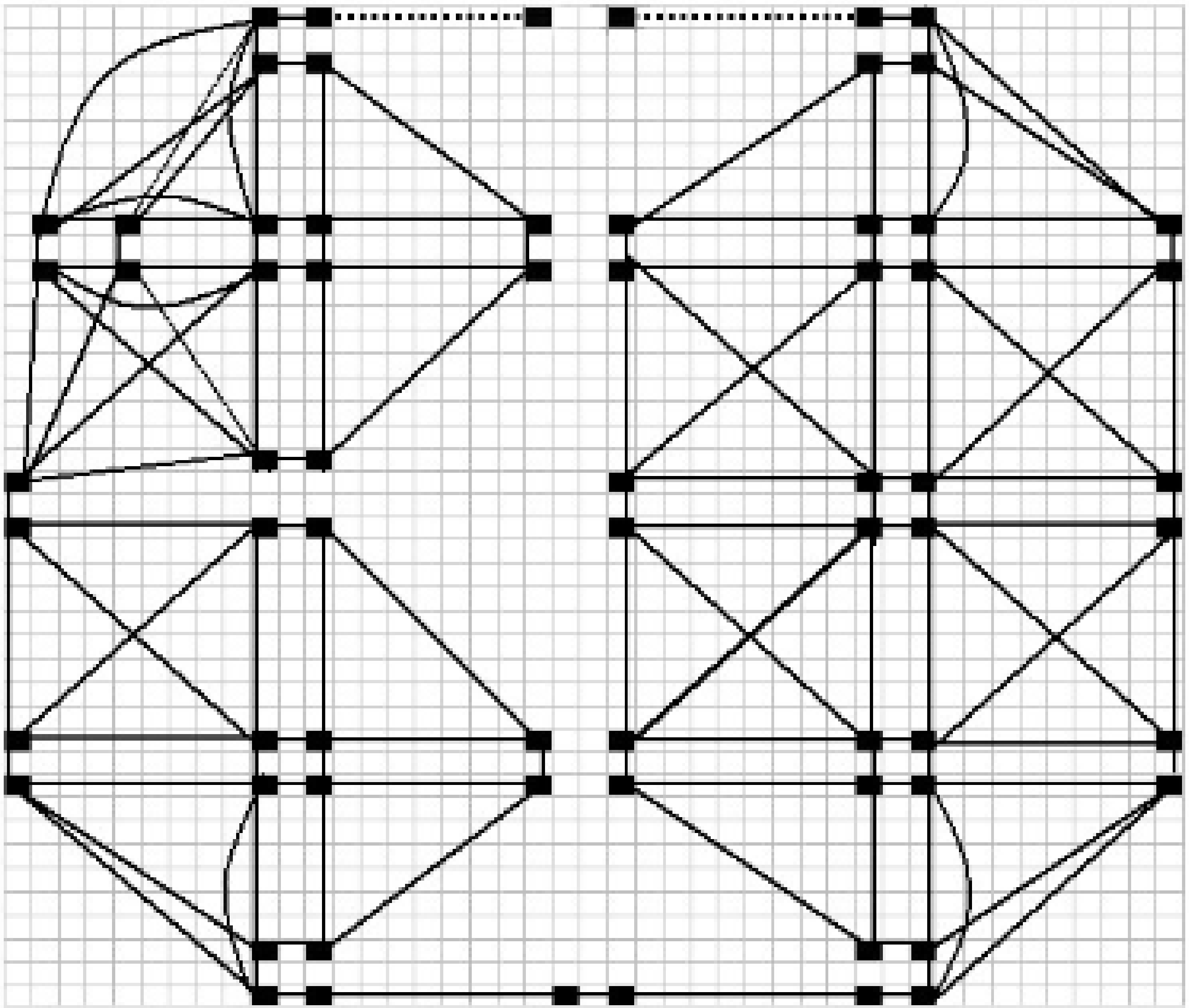




Hierarchical Parallel A* Algorithm

- The graph is broken down into equal size chunks.
- Each processor is assigned a chunk.
- The processor then finds the entry/exit nodes for the chunk assigned to it.
- It calculates the actual cost of traversal for each combination of entry/exit nodes.


S

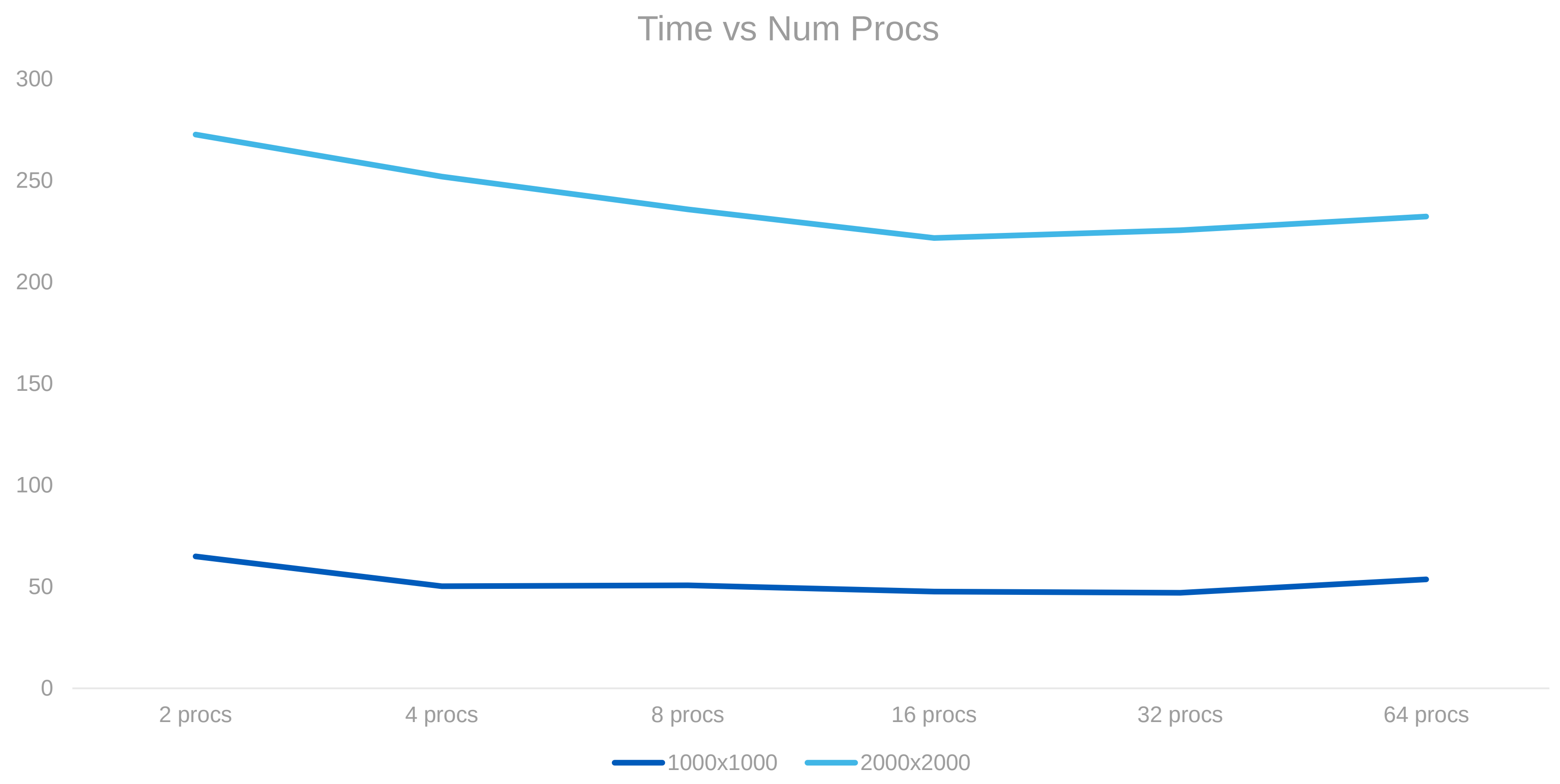


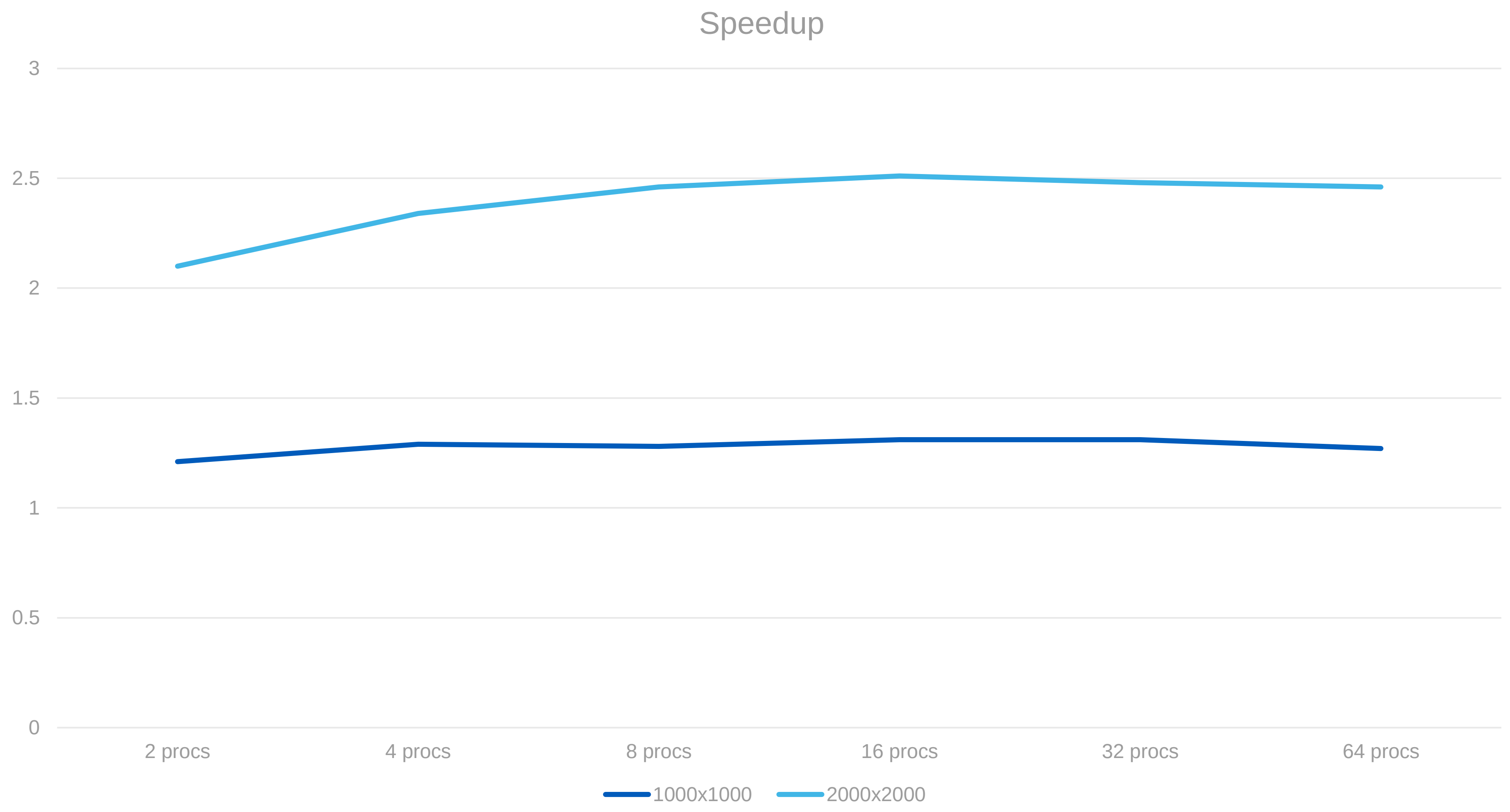
Hierarchical Parallel A* Algorithm

- The graph is broken down into equal size chunks.
- Each processor is assigned a chunk.
- Each processor then finds the entry/exit nodes for the chunk assigned to it.
- They calculate the actual cost of traversal for each combination of entry/exit nodes.
- Estimate the average cost of horizontal or vertical traversal through the chunk and sends the value to the master.
- The master broadcasts the average cost of traversal through each chunk.
- Run A* algorithm on the node containing the starting node and all other nodes.
- When one solution is found, broadcast the cost to all nodes.
- Run until cutoff.

Results

	A	B	C	D	E	F	G
1	Graph size	2 procs	4 procs	8 procs	16 procs	32 procs	64 procs
2	10000	64.88	50.26	50.58	47.58	47.04	53.56
3	40000	272.49	251.81	235.67	221.56	225.43	232.16


Time vs Num Procs
300
250
200
150
100
50
0
2 procs
4 procs
8 procs
16 procs
32 procs
64 procs
1000x1000
2000x2000


Speedup
3
2.5
2
1.5
1
0.5
0
2 procs
4 procs
8 procs
16 procs
32 procs
64 procs
1000x1000
2000x2000

Factors influencing the results

- Size of the graph.
- Choice of algorithm to find the pairwise shortest path within a chunk.
- Nature of the graph.
- Unexpected behavior of code.

Path Error

	A	B	C	D	E	F
1	-1.411911491	-1.623252616	-1.512489497	-1.932625468	-1.888065592	-2.030657195

- Percentage error

- RMS error = 1373.225
- Which is a 1.748% deviation from the optimal path

Future scope

- Run path smoothing algorithms on the obtained path.
- Find better estimates block length for Manhattan distance.
- Remove ‘optimizations’ and check time.
- Run the algorithm on actual maps.

References

- Near Optimal Hierarchic Pathfinding, Adi Botea, Martin Muller, Jonathan Schaeffer, https://webdocs.cs.ualberta.ca/~mmueller/ps/hpastar.pdf
- Implementation of Parallel Path Finding in a Shared Memory Architecture, David Cohen and Matthew Dallas, https://pdfs.semanticscholar.org/9201/badbffa25a272852e05401cedf68f8043a23.pdf
- Fringe Search: Beating A* at Pathfinding on Game Maps, Yngvi Bjornsson, Markus Enzenberger, Robert C. Holte and Jonathan Schaeffe, https://webdocs.cs.ualberta.ca/~games/pathfind/publications/cig2005.pdf
- Implementation of A*, RedBlob Games, https://www.redblobgames.com/pathfinding/a-star/implementation.html
- Game maps, Nathan Sturtevant, http://www.movingai.com/benchmarks/