Faster Shortest Path Computation for Traffic Assignment

Boshen Chen

Supervised by: Dr. Andrea Raith, Olga Perederieieva

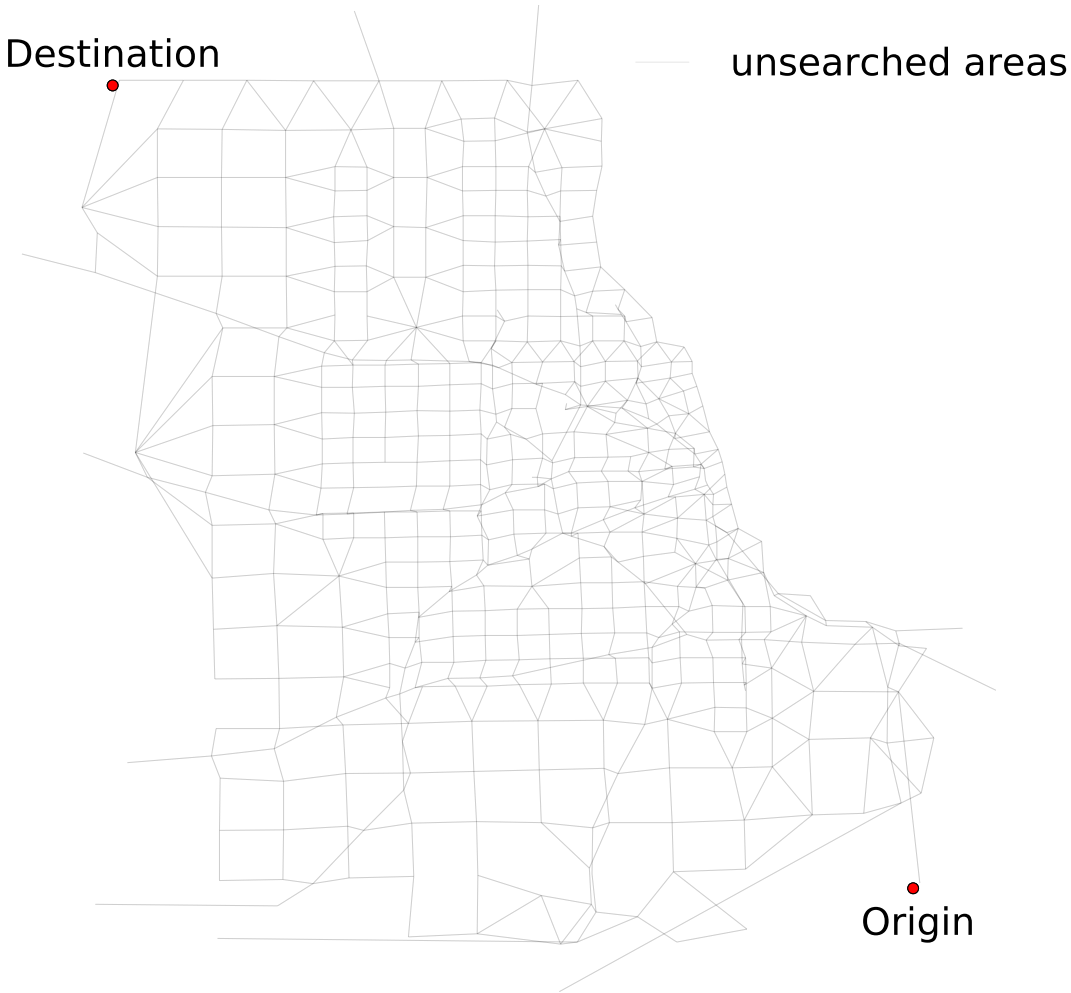
Department of Engineering Science
University of Auckland

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

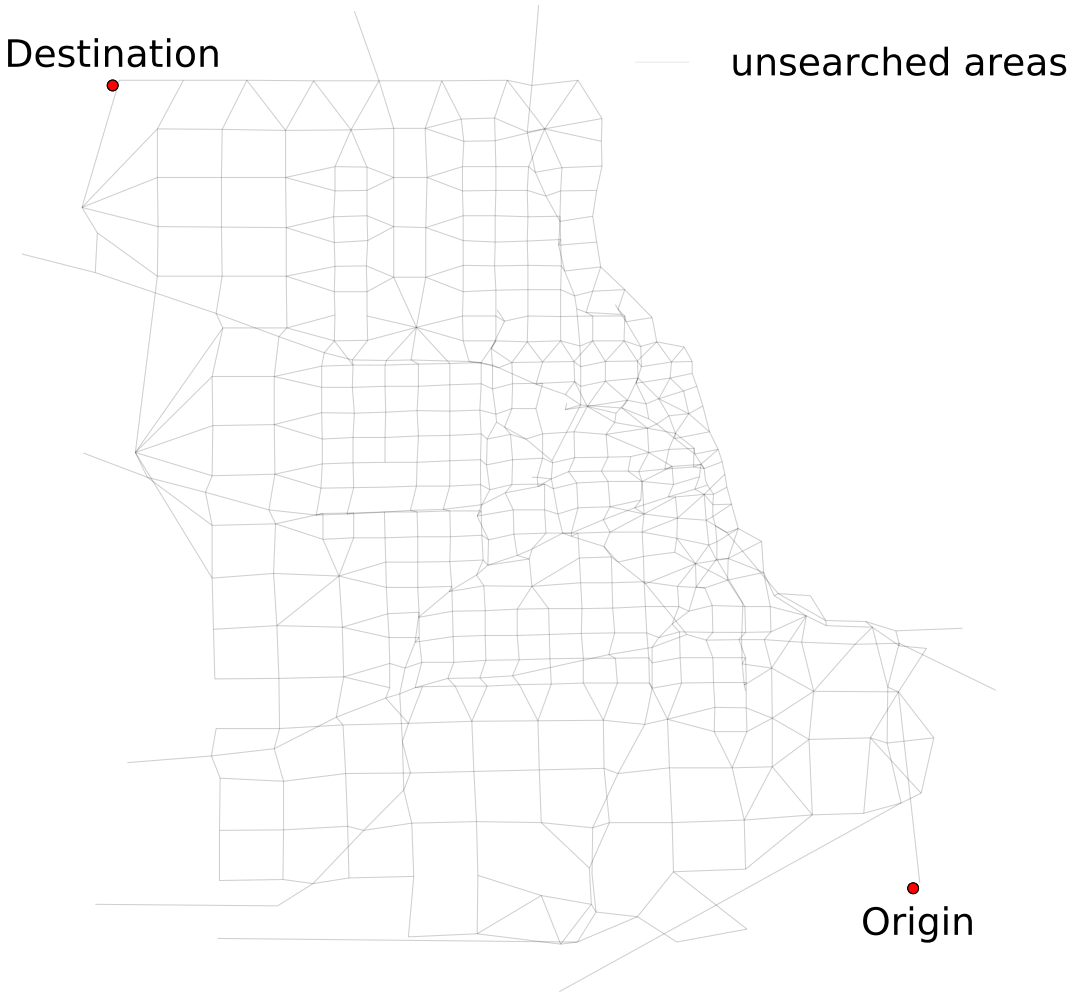
Traffic Assignment

- transportation network with supply and demand nodes
- minimise travel times
- arcs have non-linear travel times for capturing congestion effects

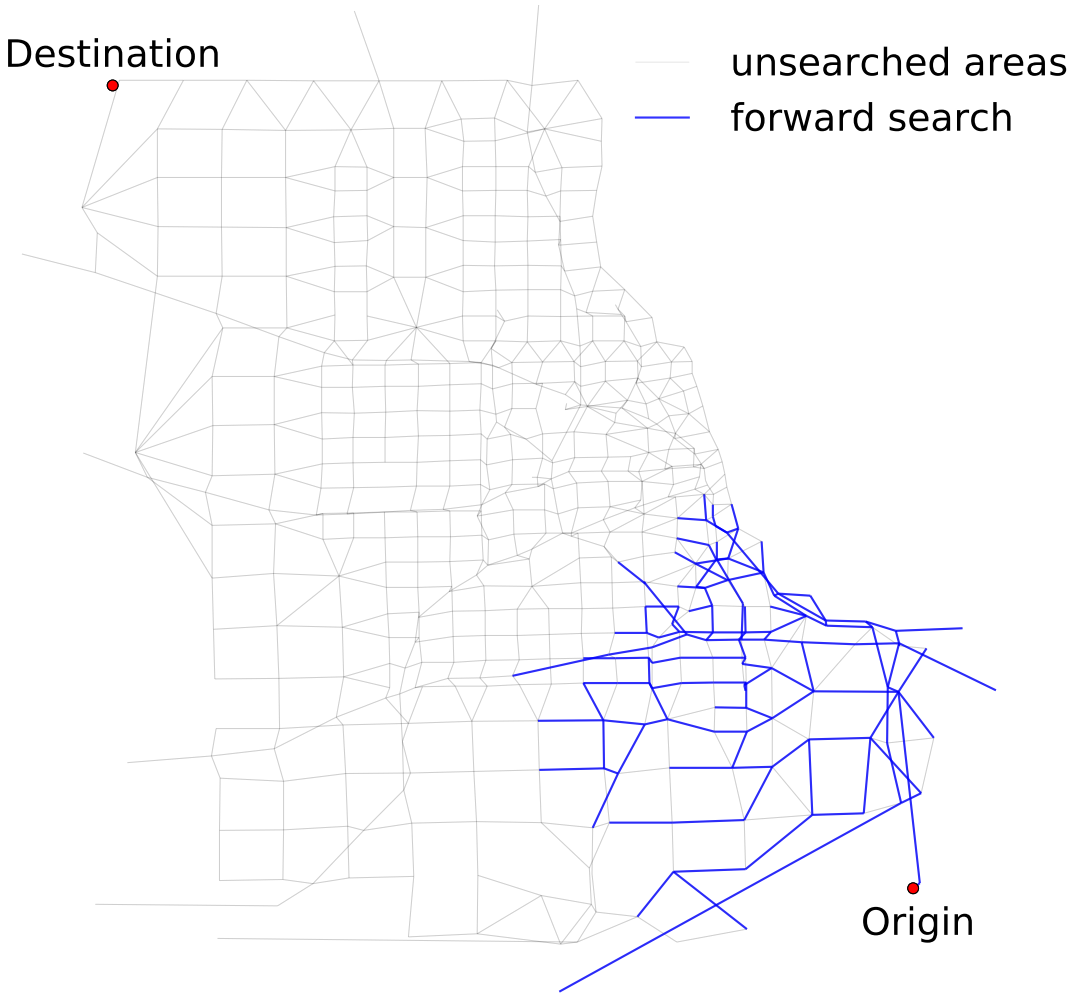
The Graph - 93,135 Origin-Destination Pairs



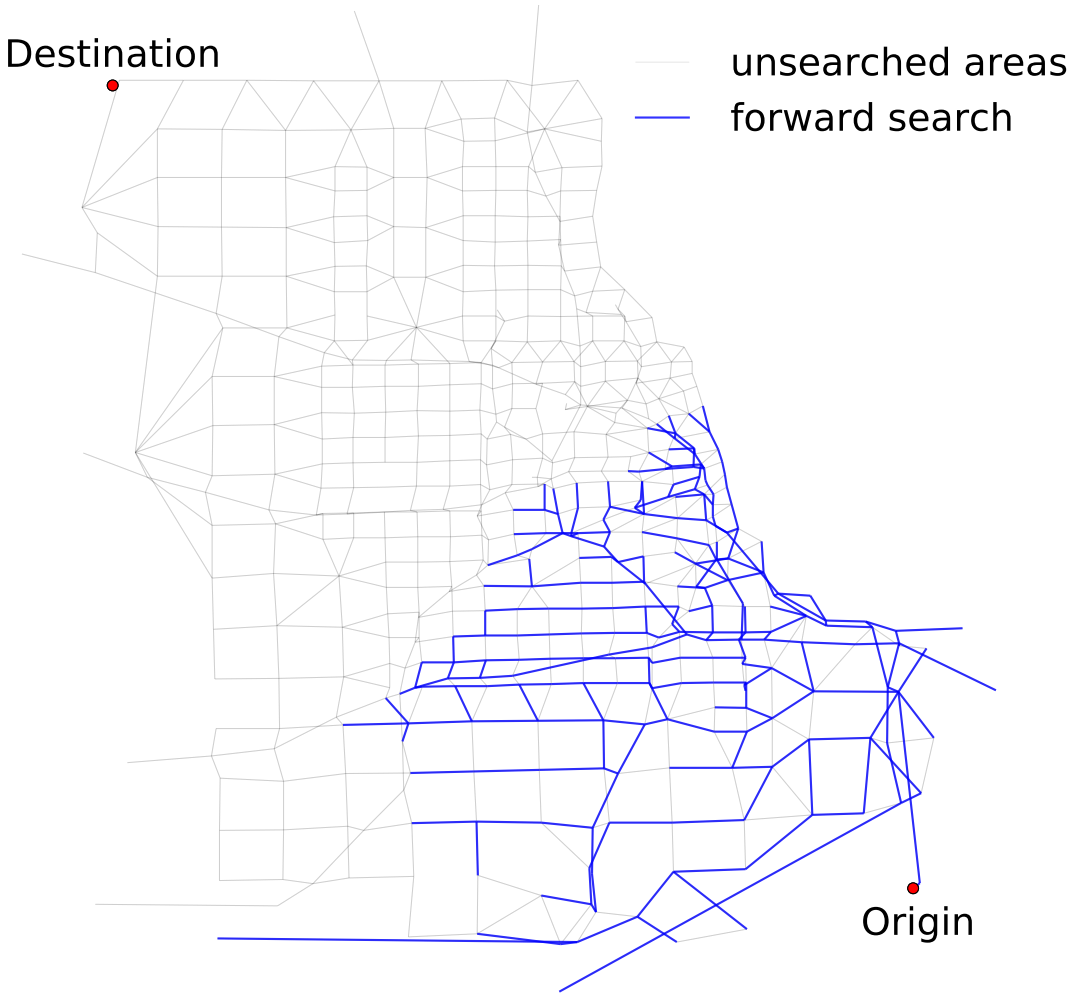
Dijkstra's Algorithm



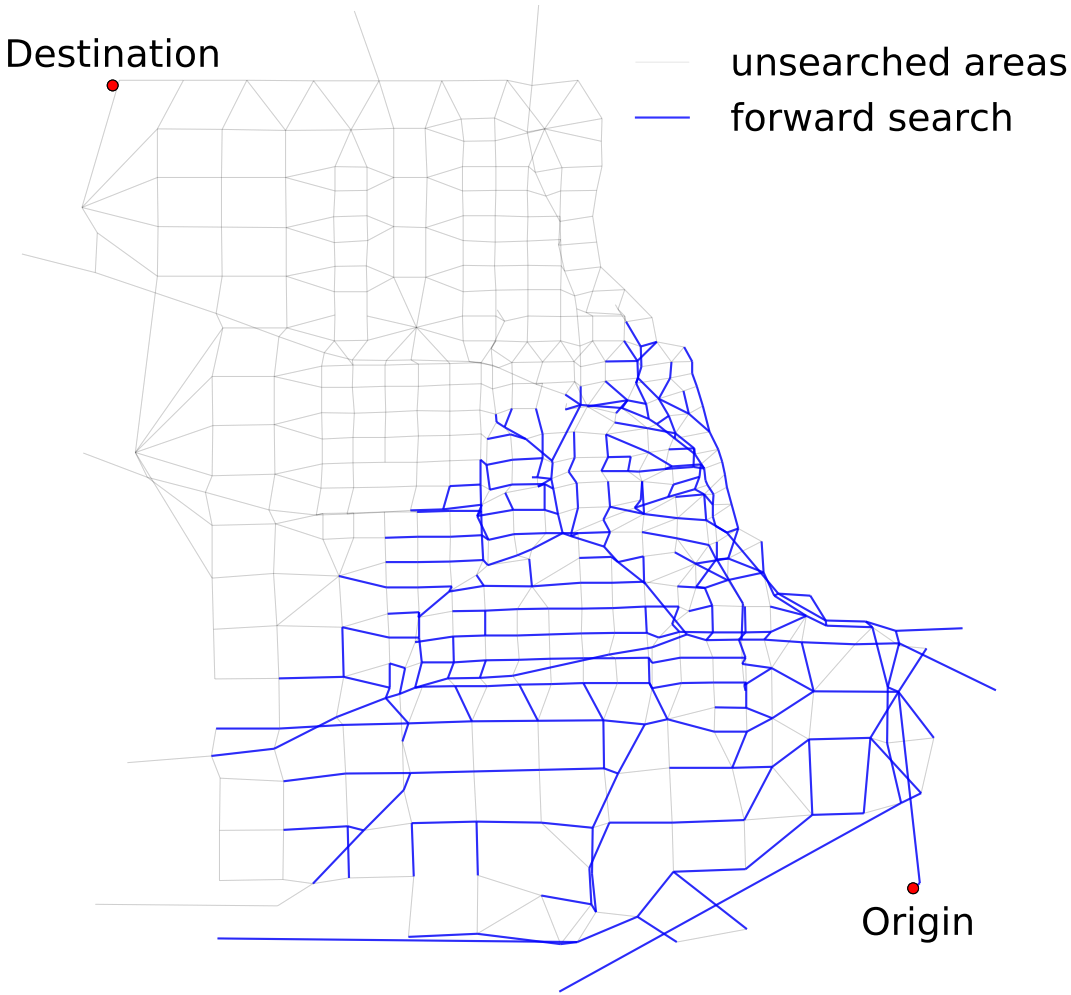
Dijkstra's Algorithm



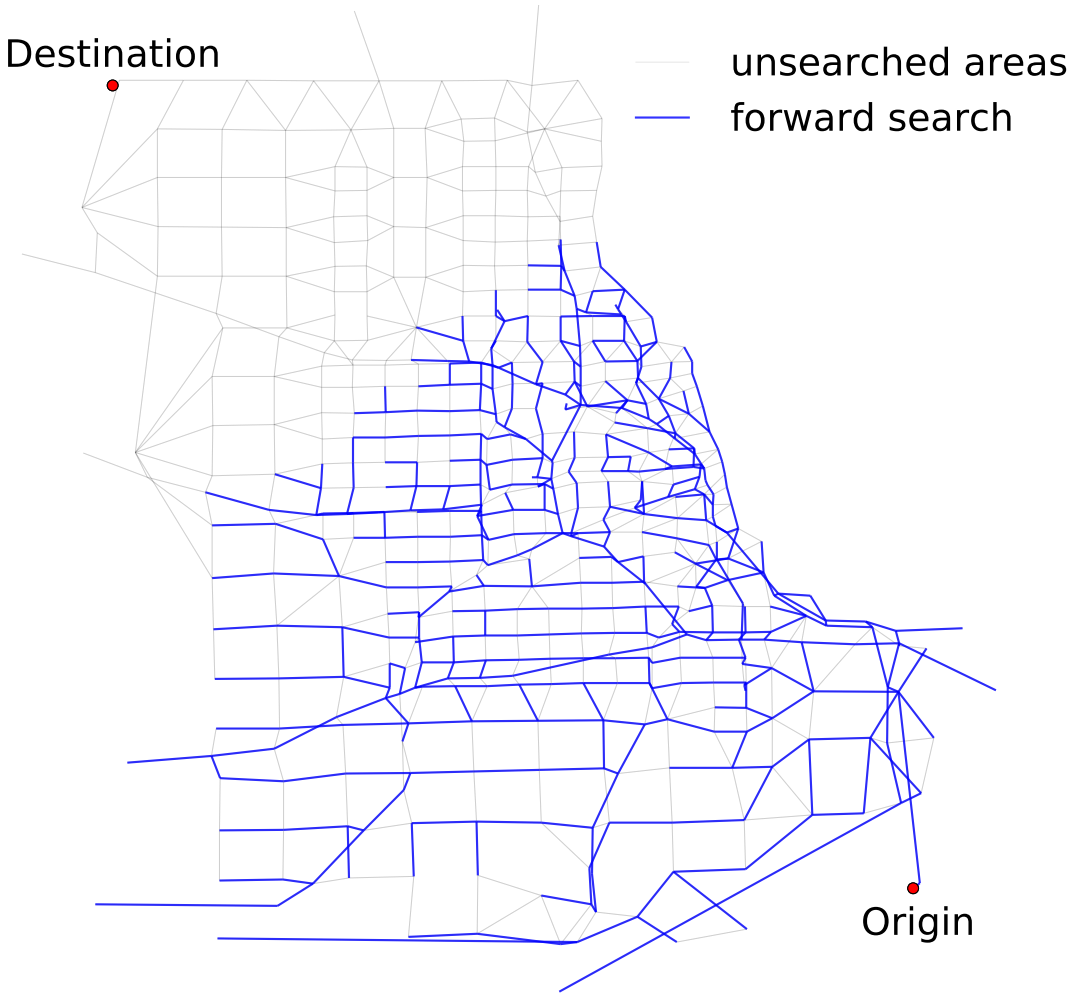
Dijkstra's Algorithm



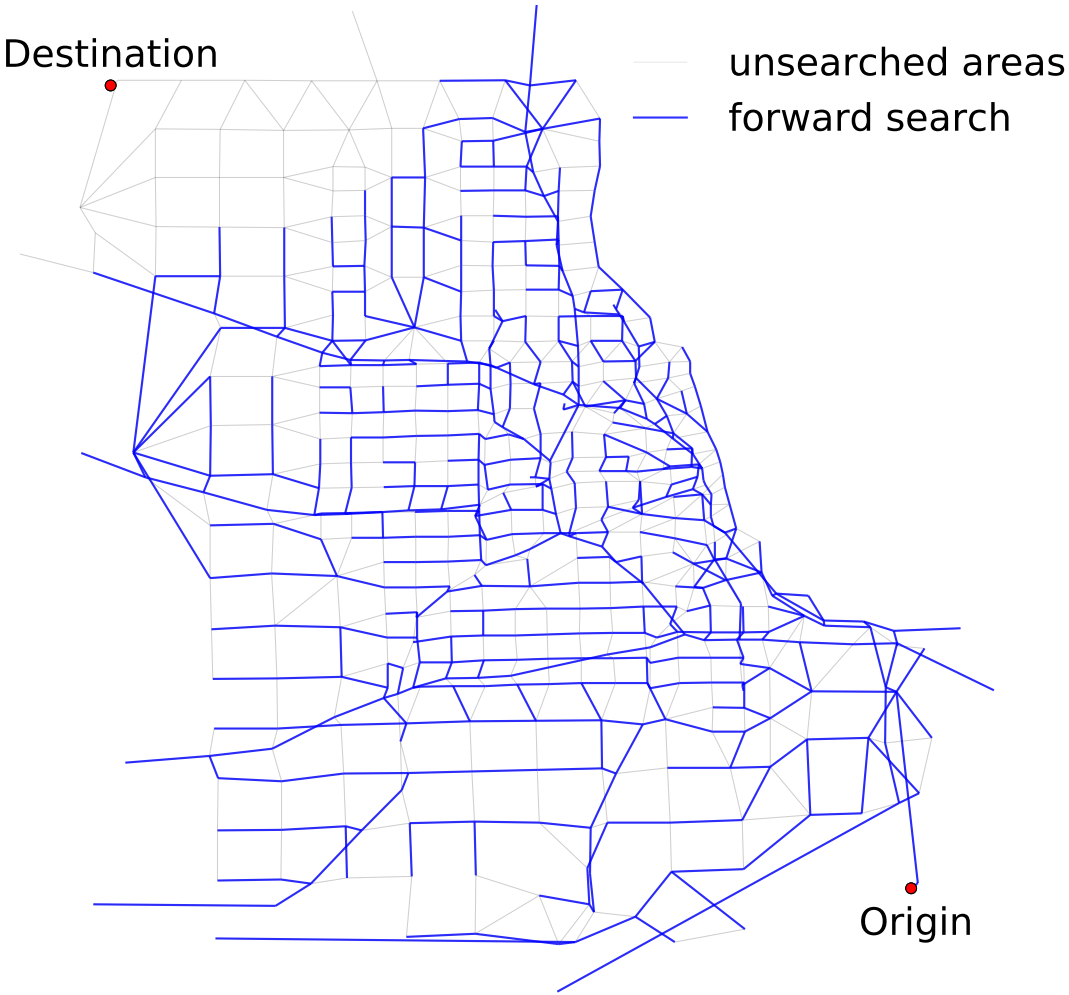
Dijkstra's Algorithm



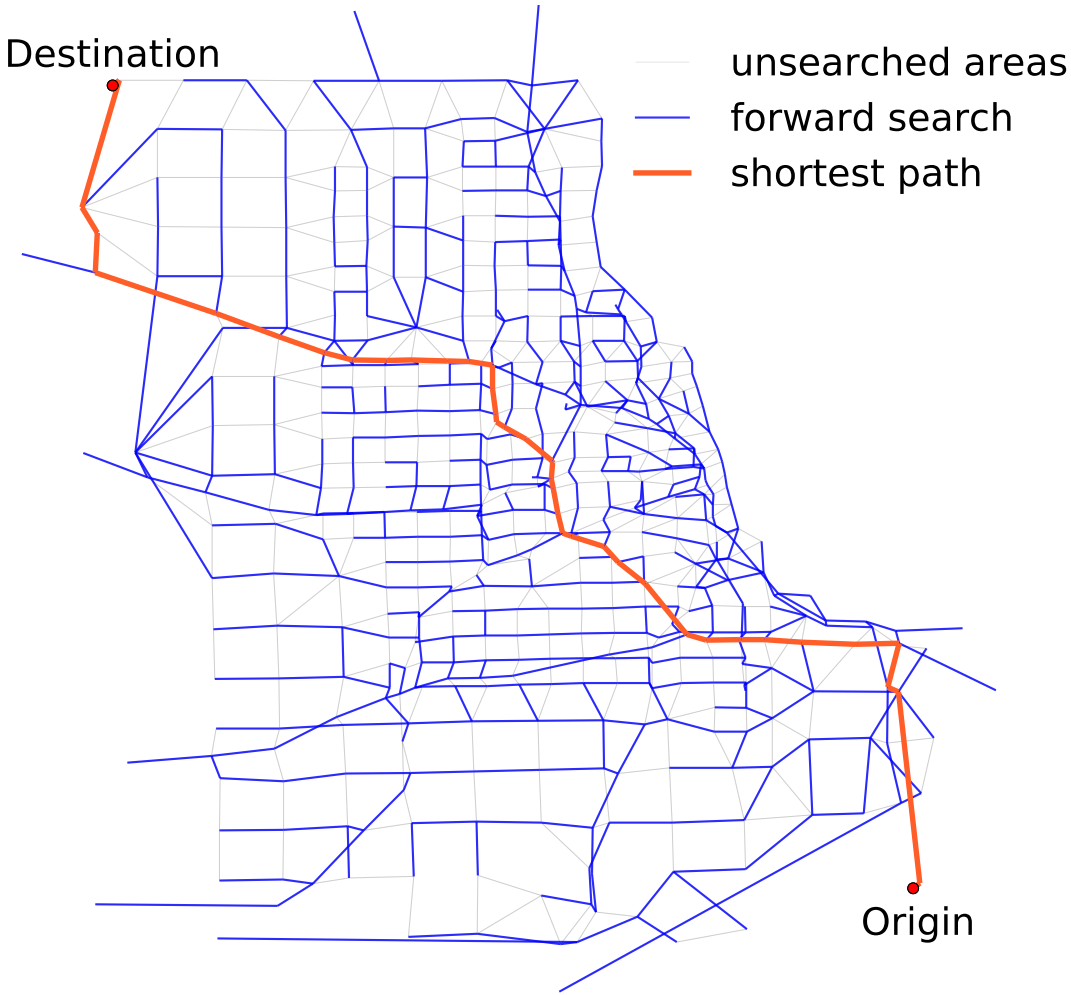
Dijkstra's Algorithm



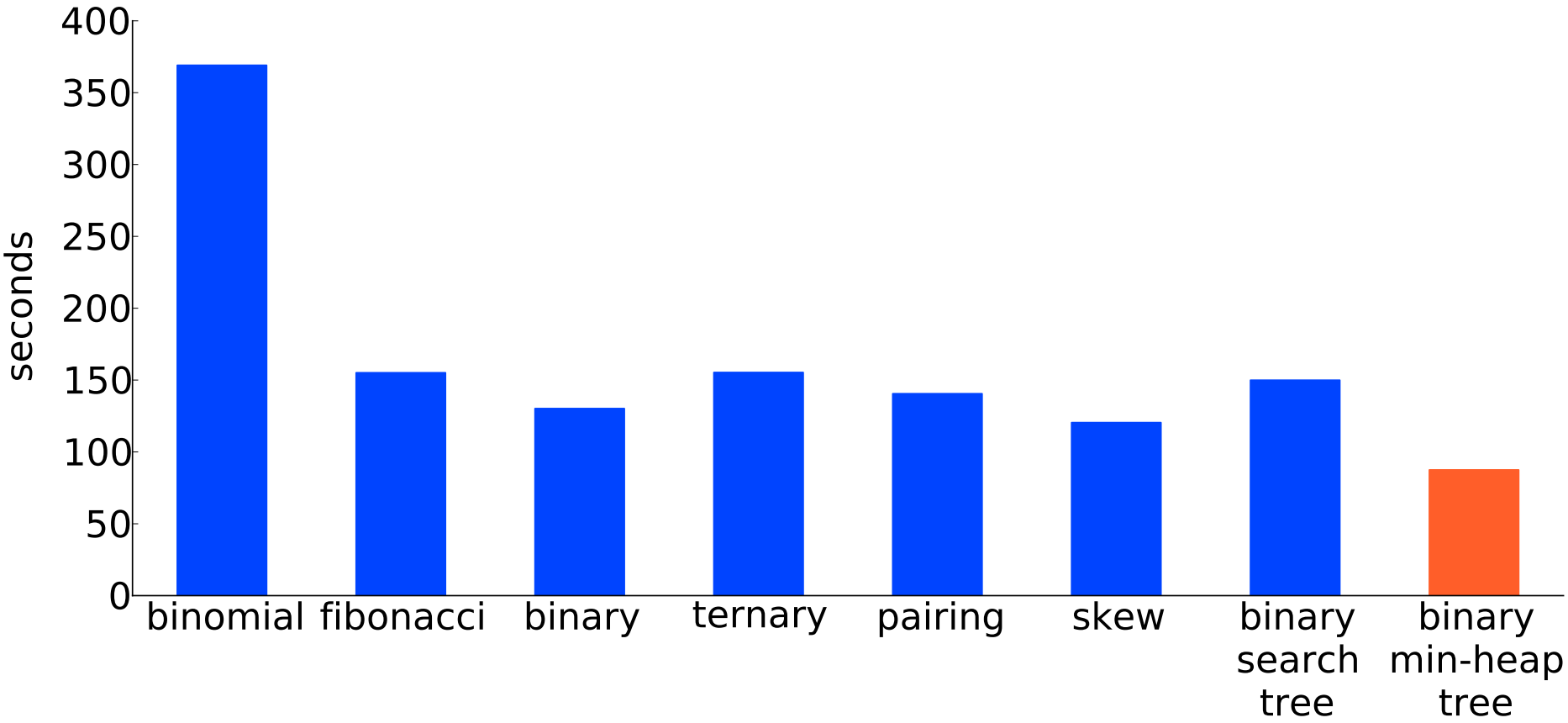
Dijkstra's Algorithm



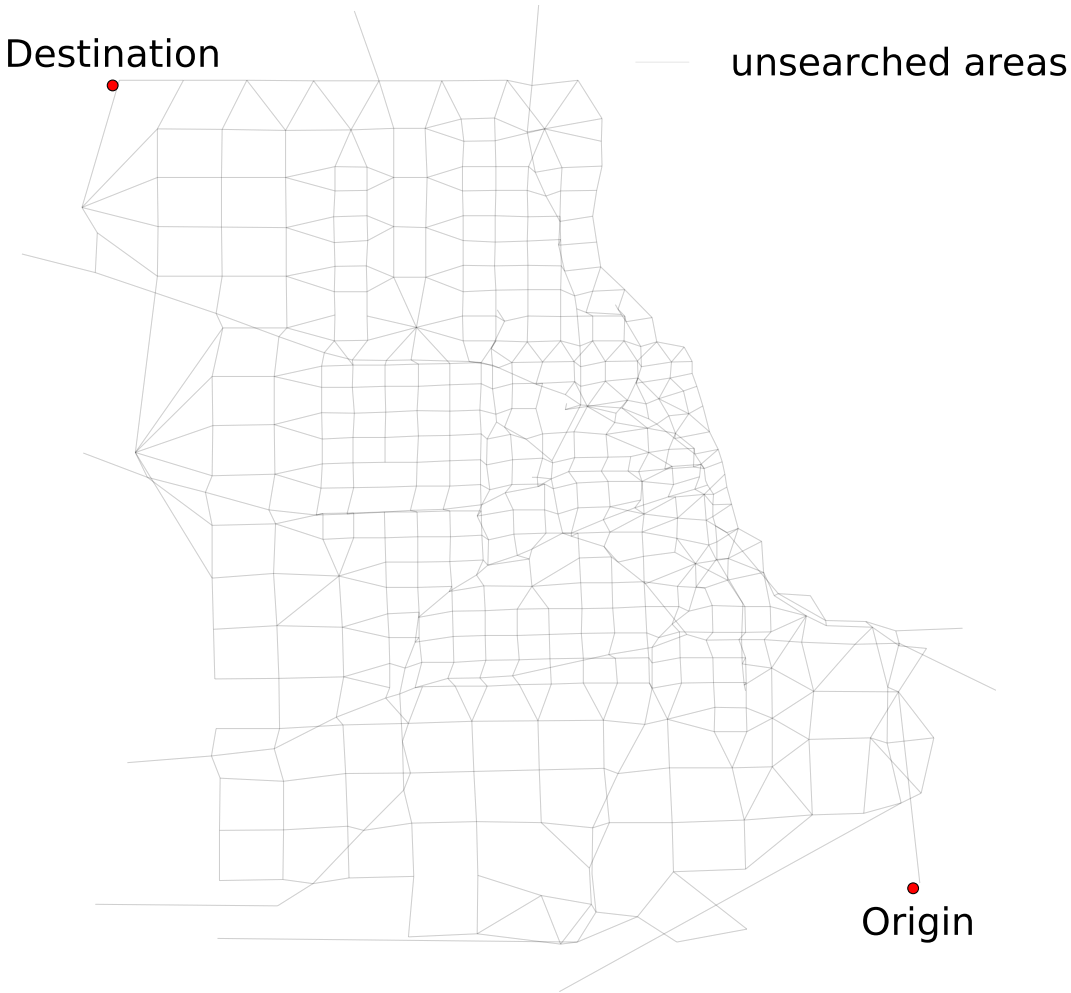
Dijkstra's Algorithm



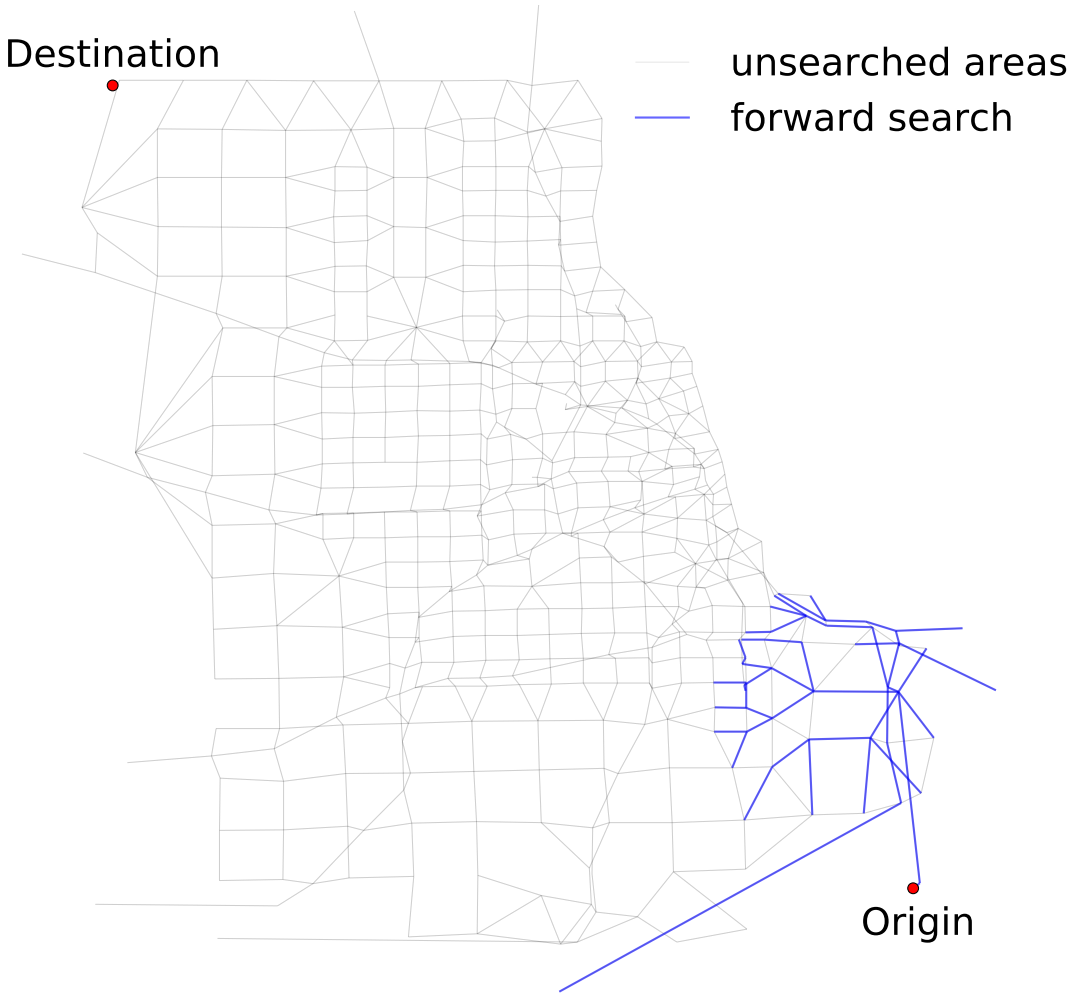
Priority queues results



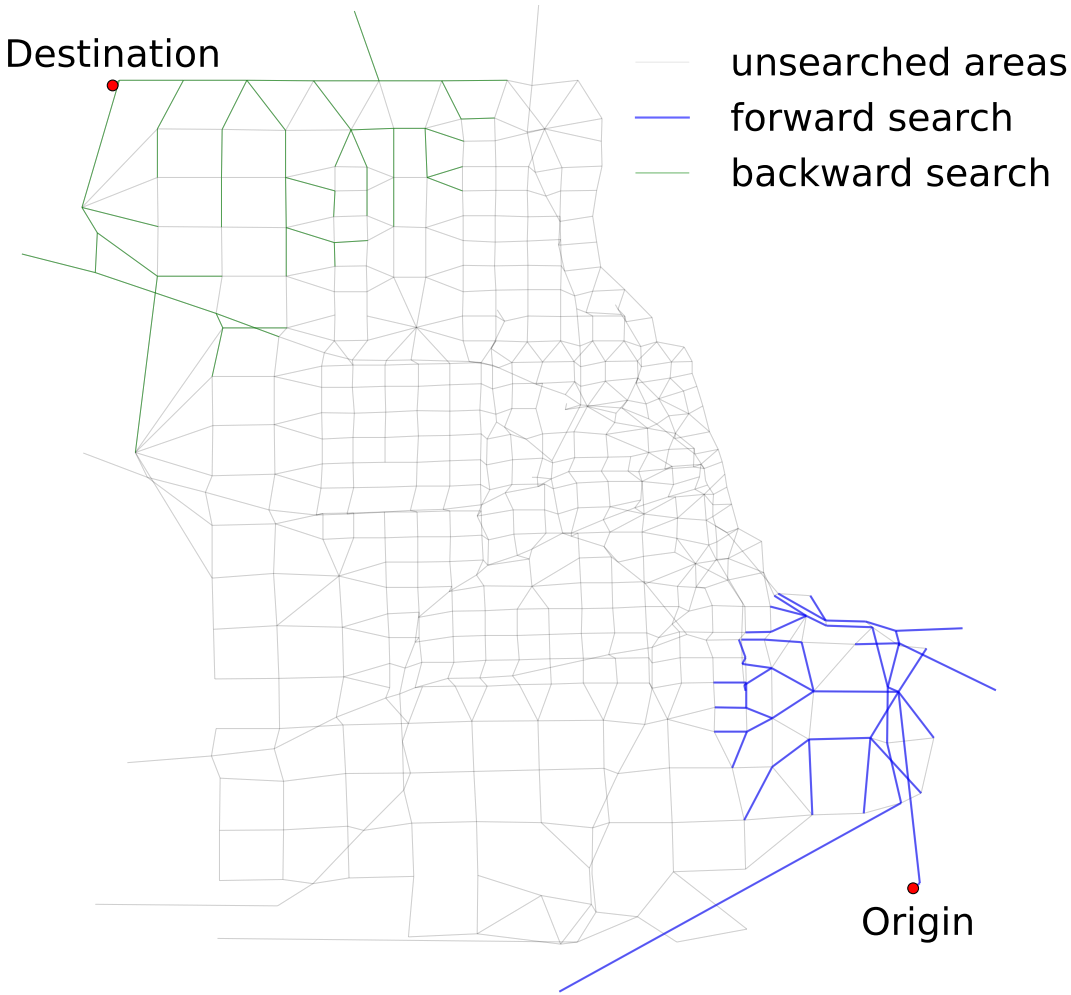
Bidirectional Dijkstra's Algorithm



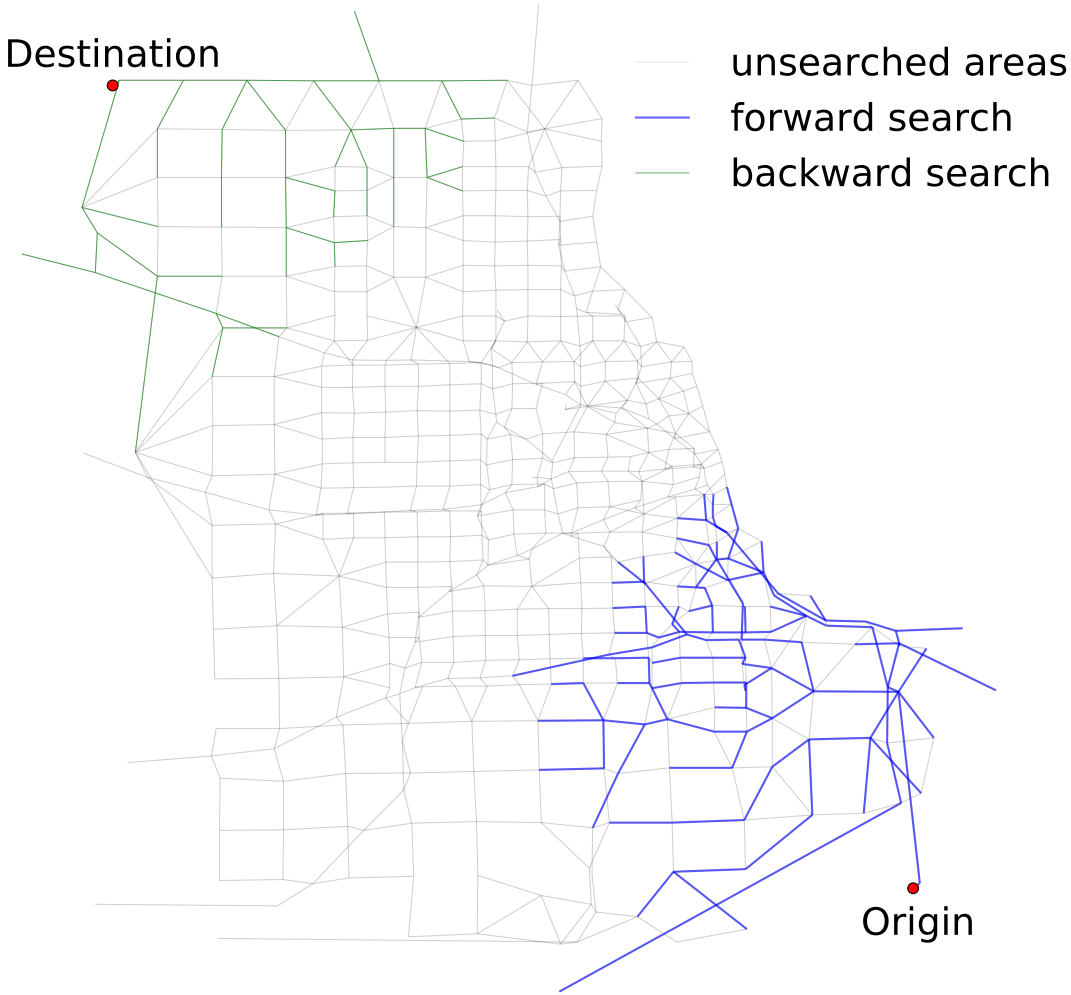
Bidirectional Dijkstra's Algorithm



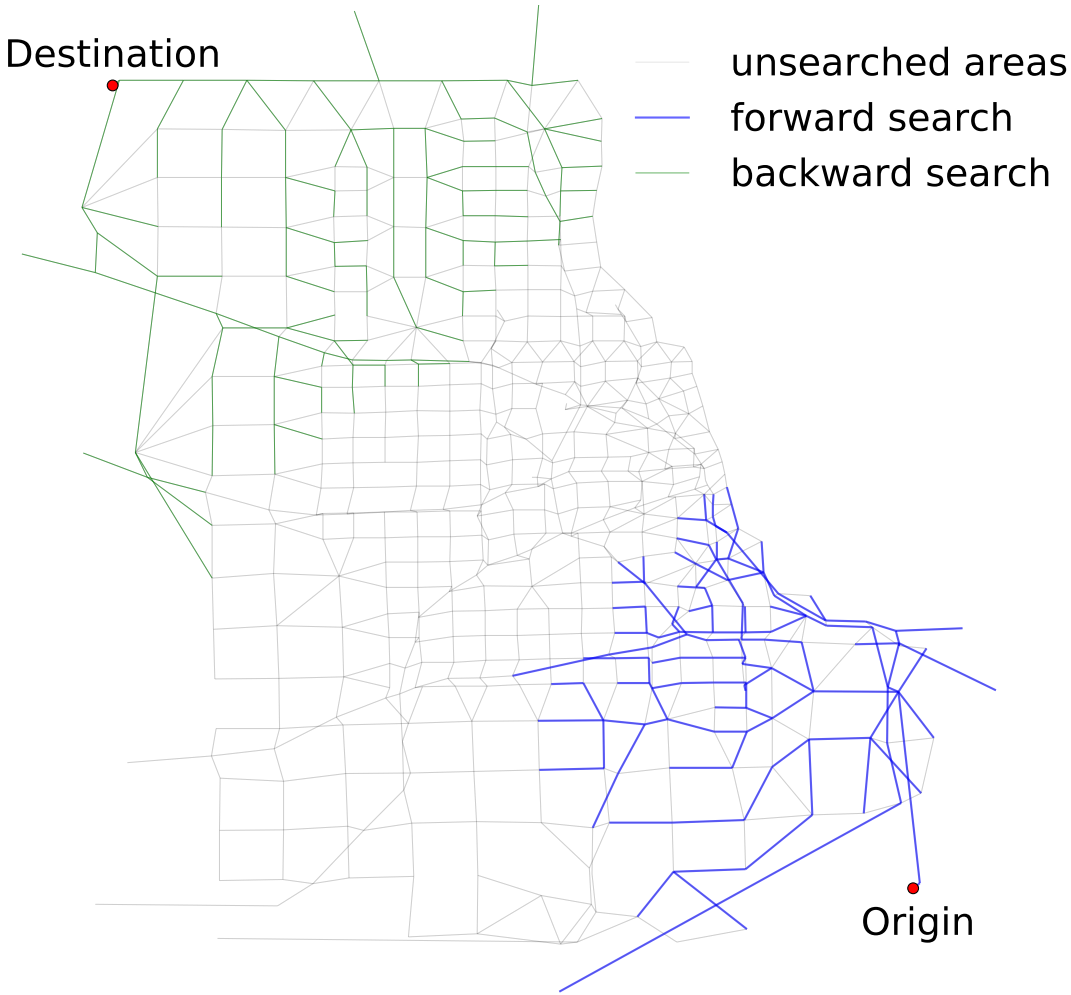
Bidirectional Dijkstra's Algorithm



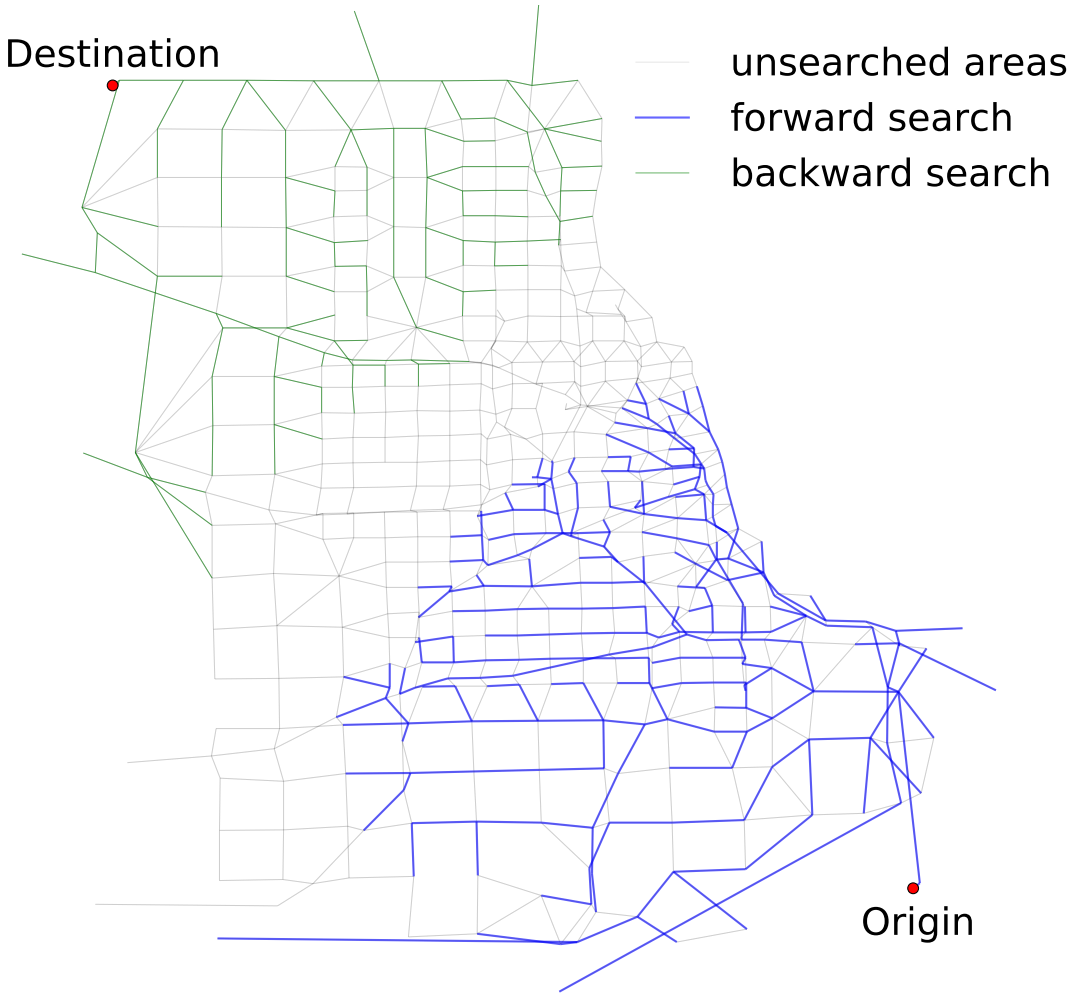
Bidirectional Dijkstra's Algorithm



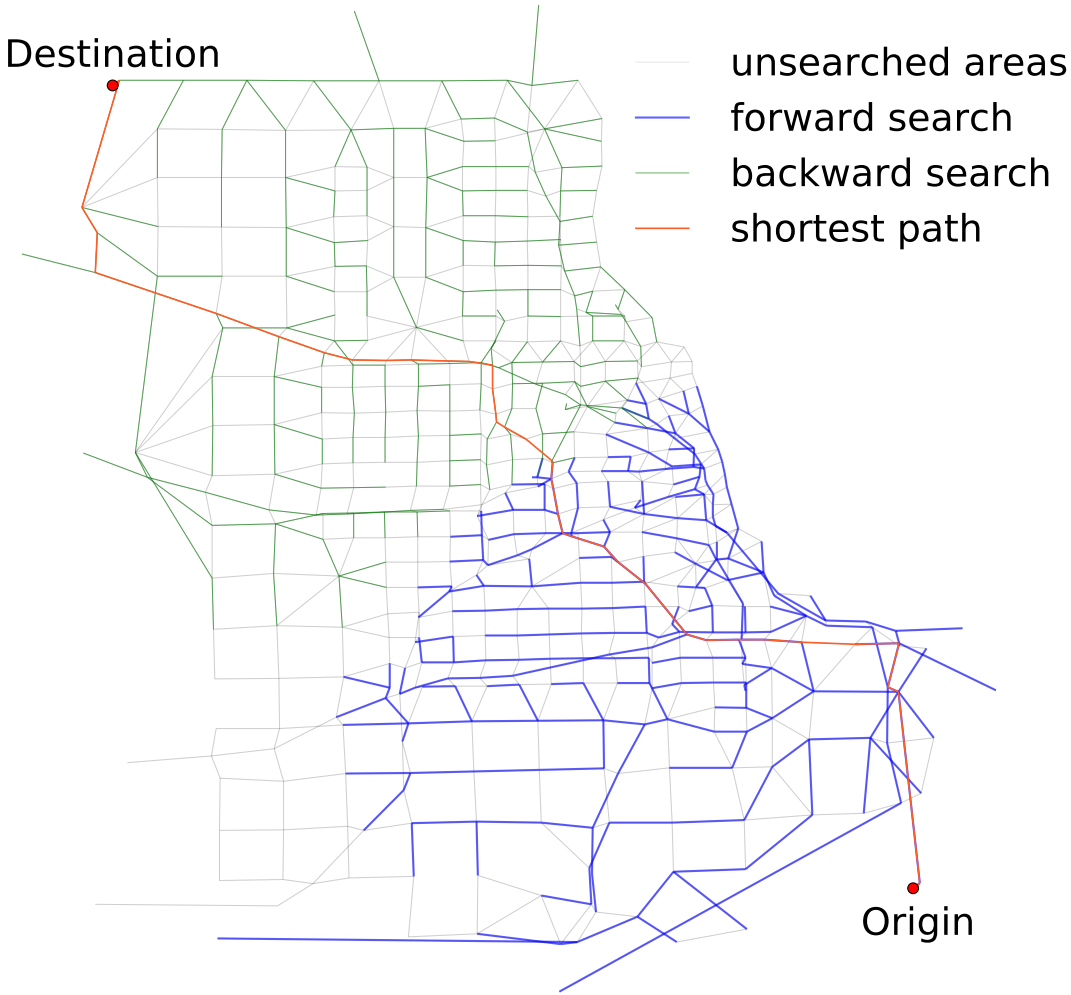
Bidirectional Dijkstra's Algorithm



Bidirectional Dijkstra's Algorithm

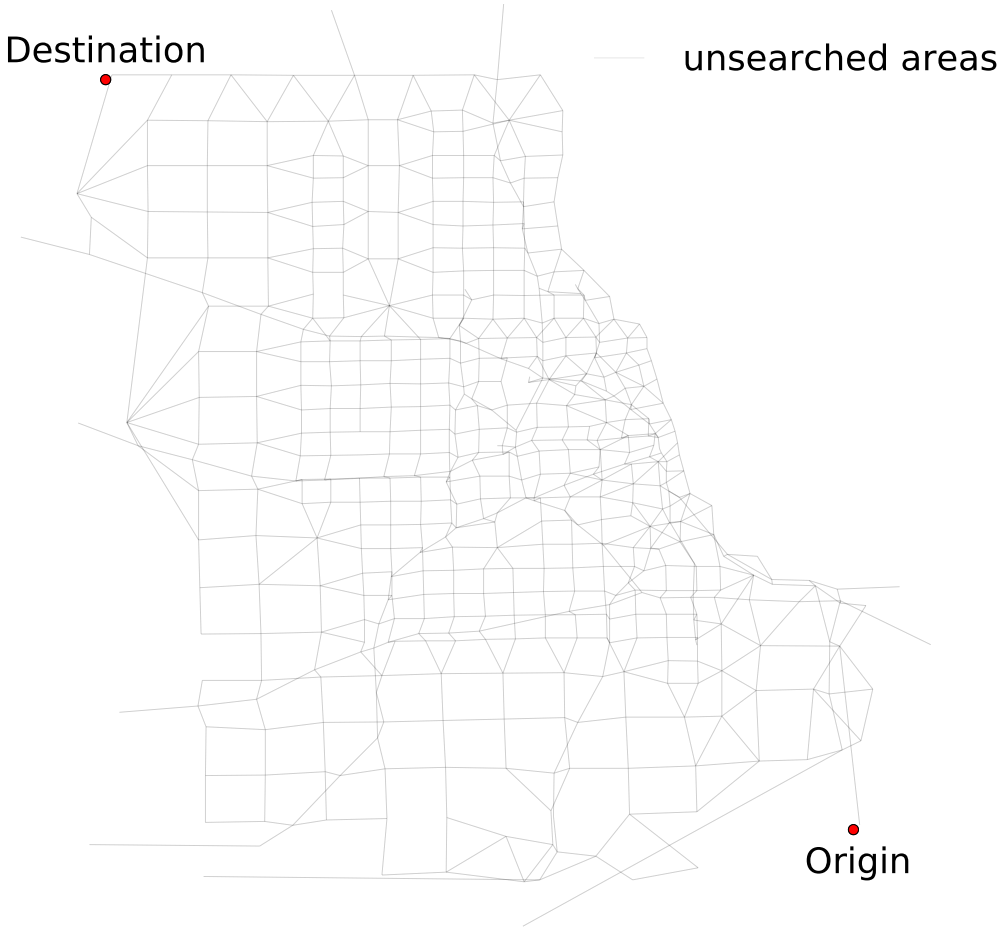


Bidirectional Dijkstra's Algorithm



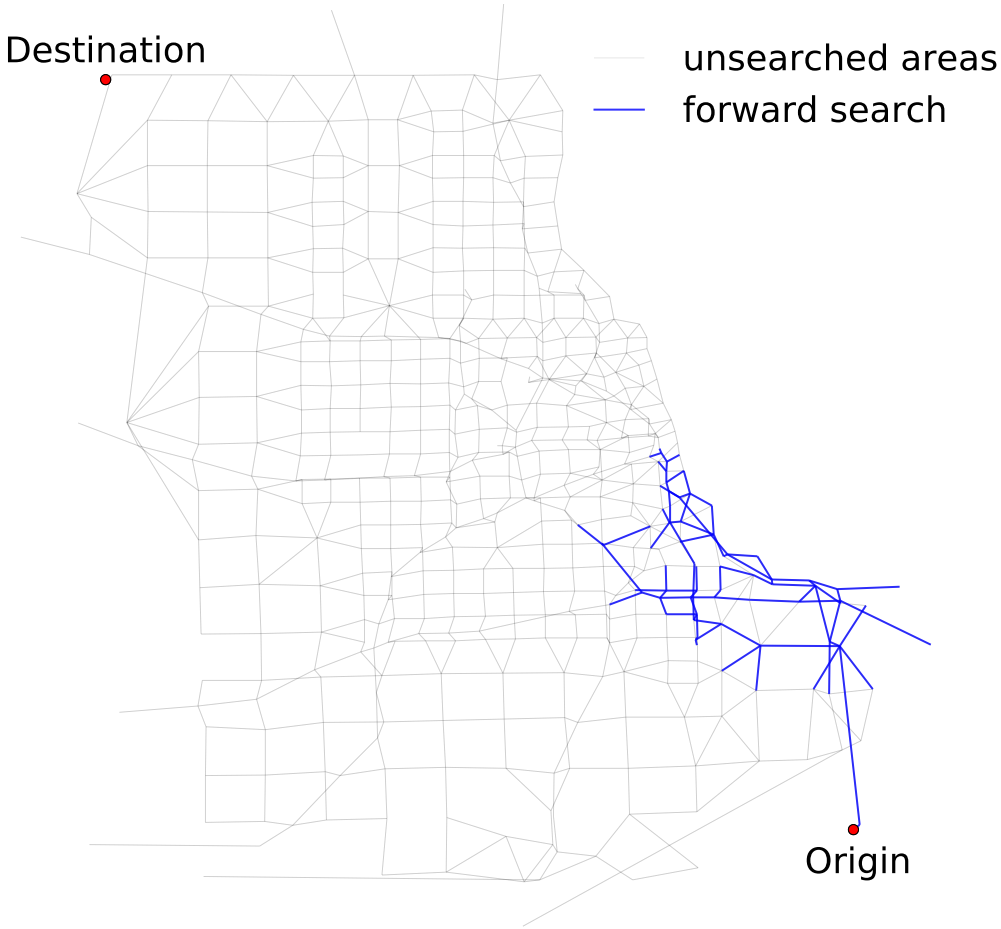
A* Search (Best-First Search)

- Visit the next node that is on the expected shortest path.



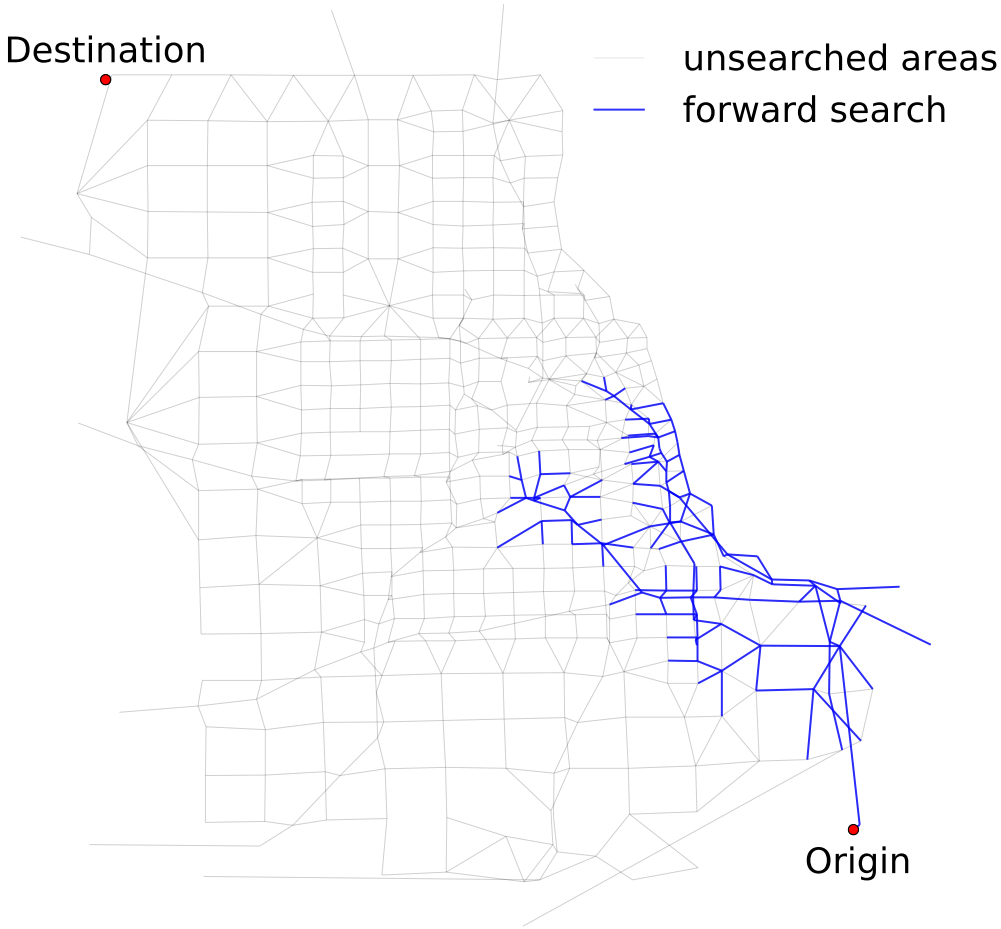
A* Search (Best-First Search)

- Visit the next node that is on the expected shortest path.



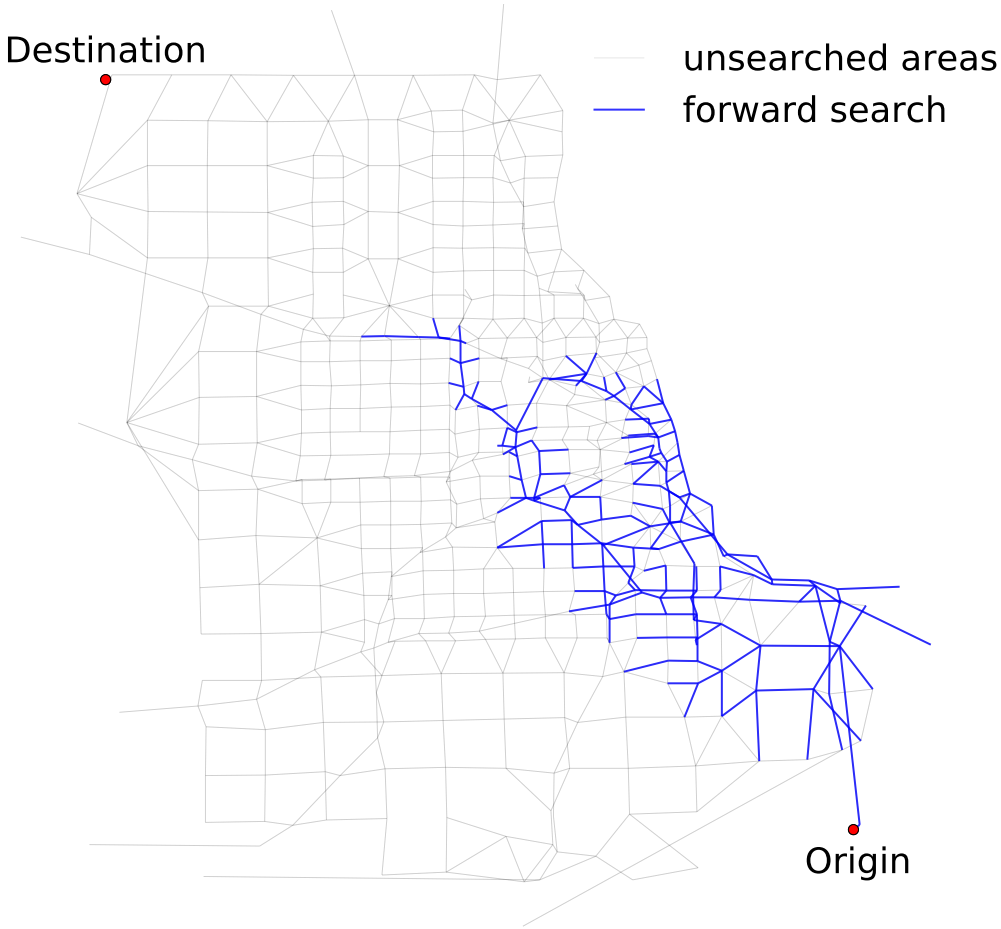
A* Search (Best-First Search)

- Visit the next node that is on the expected shortest path.



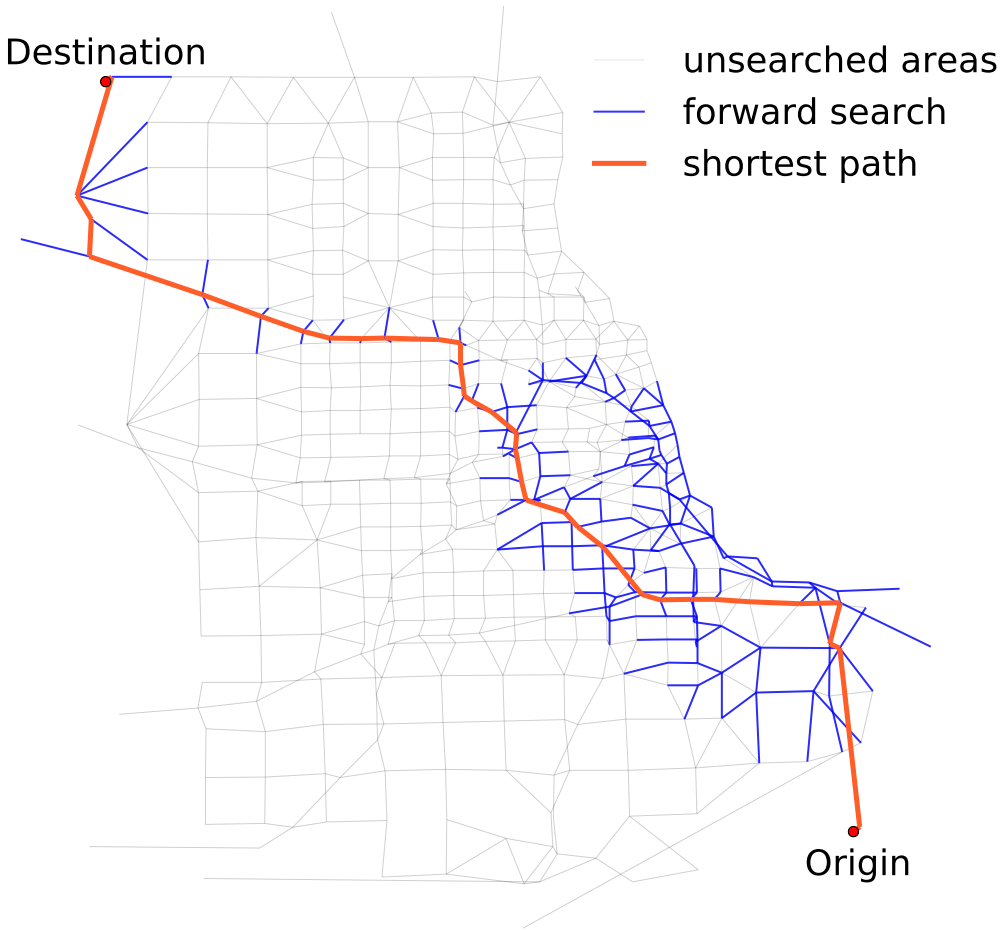
A* Search (Best-First Search)

- Visit the next node that is on the expected shortest path.

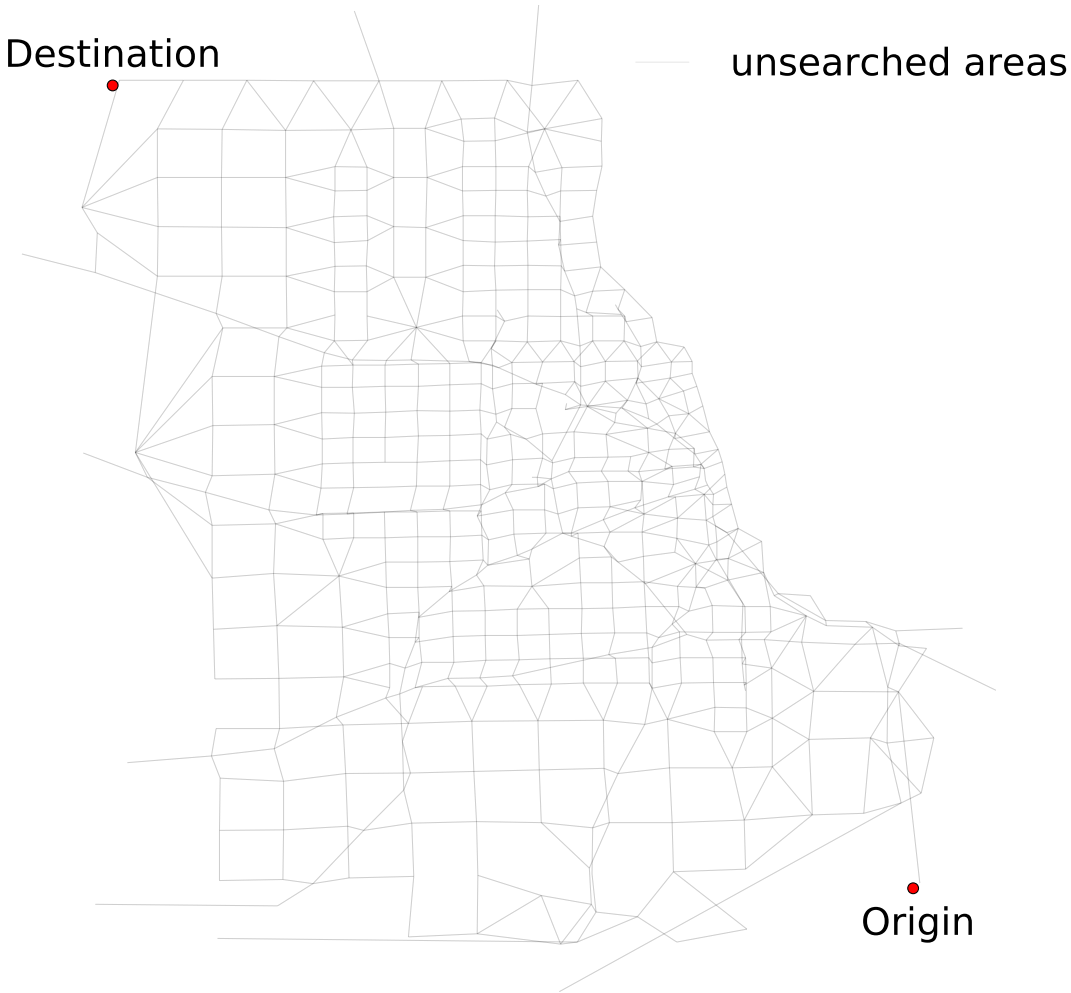


A* Search (Best-First Search)

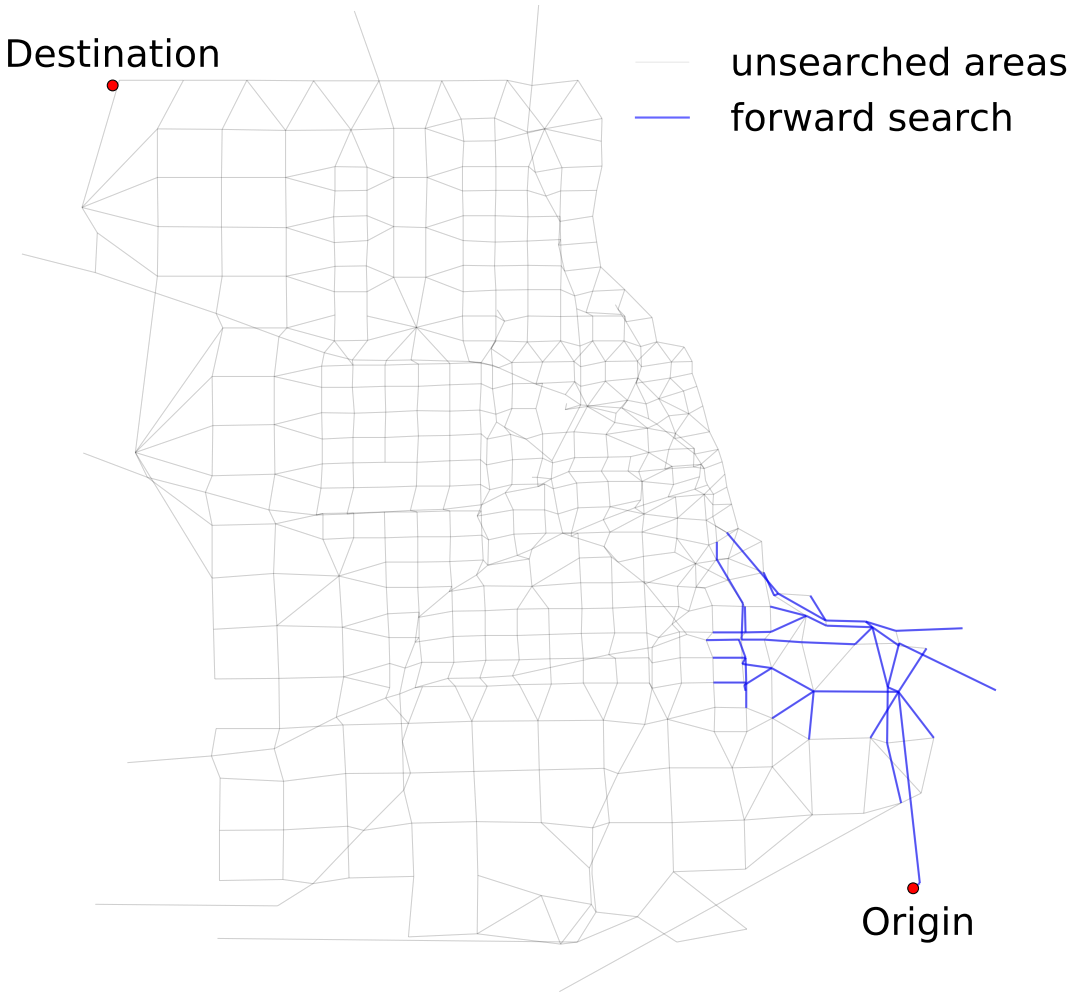
- Visit the next node that is on the expected shortest path.



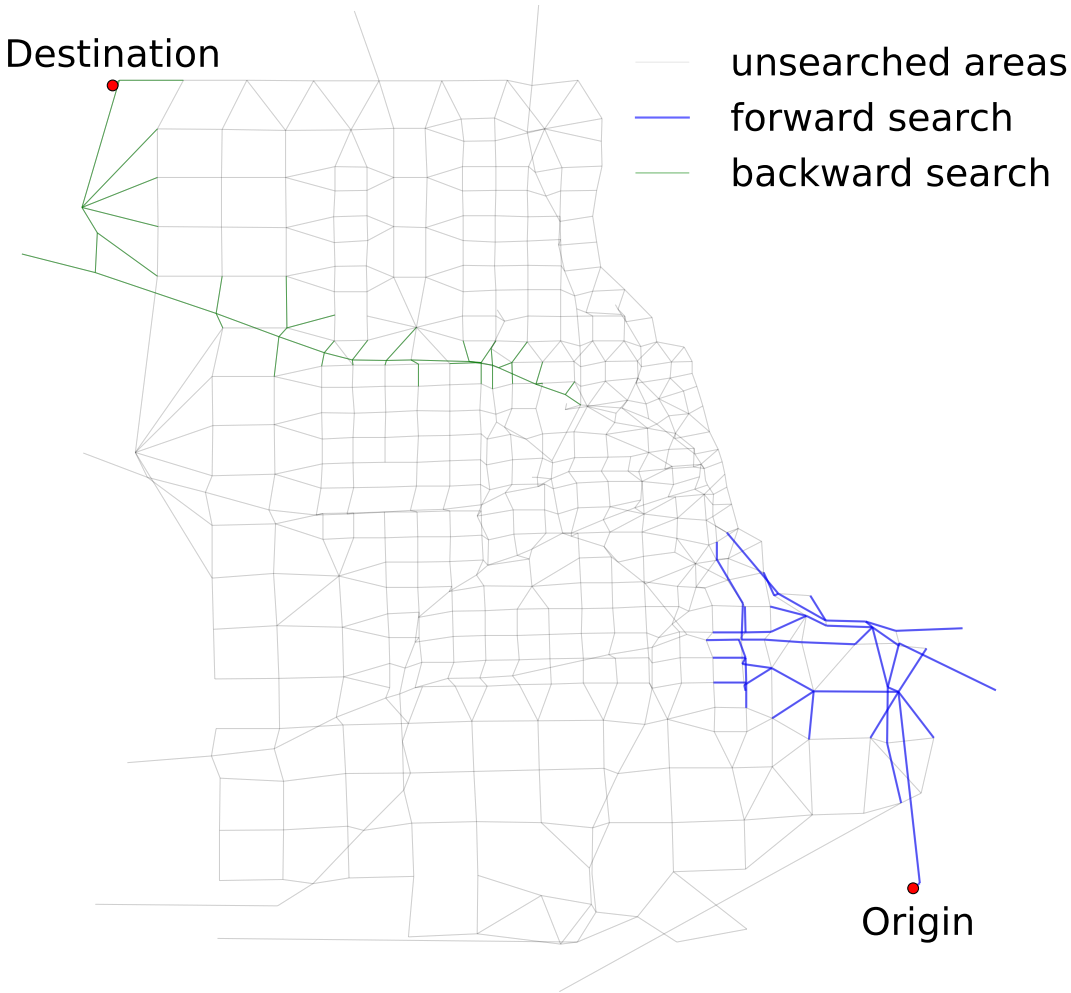
Bidirectional A* search



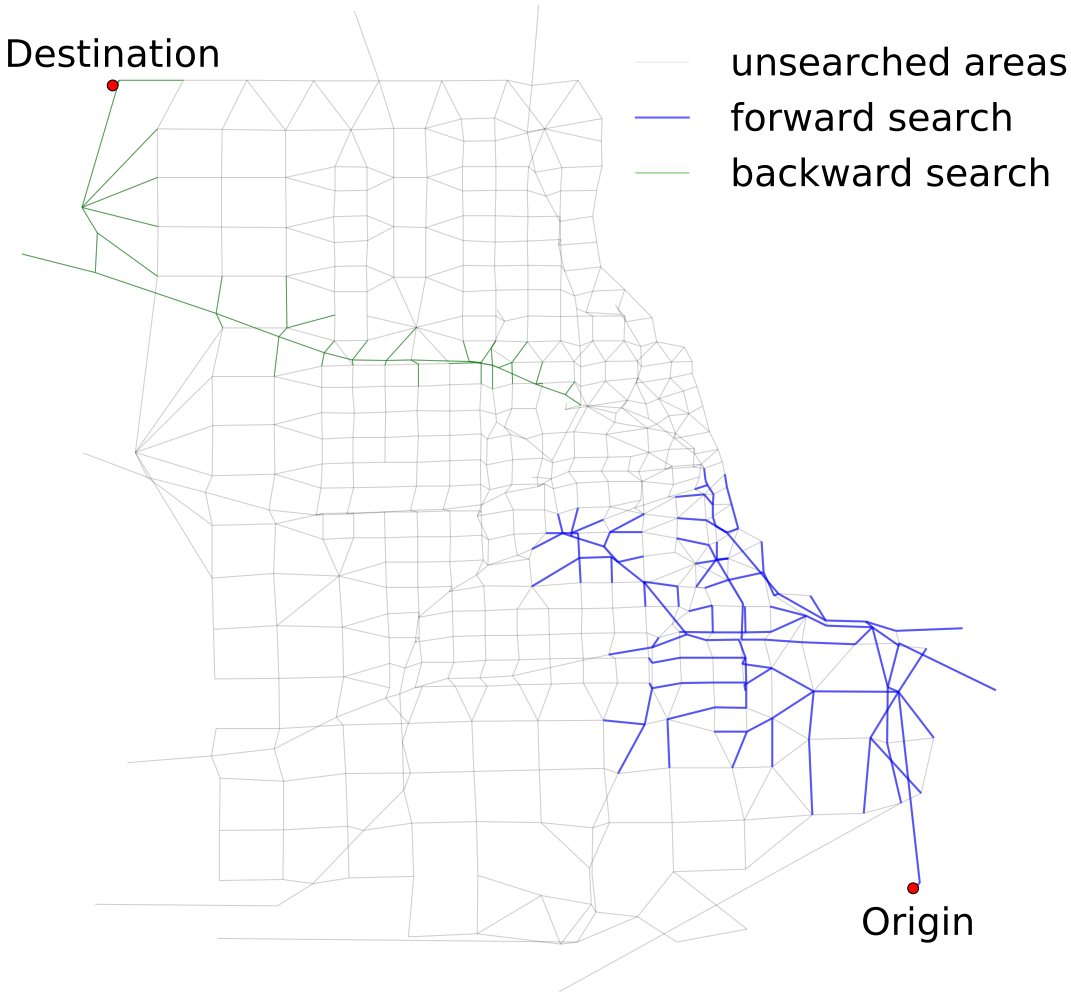
Bidirectional A* search



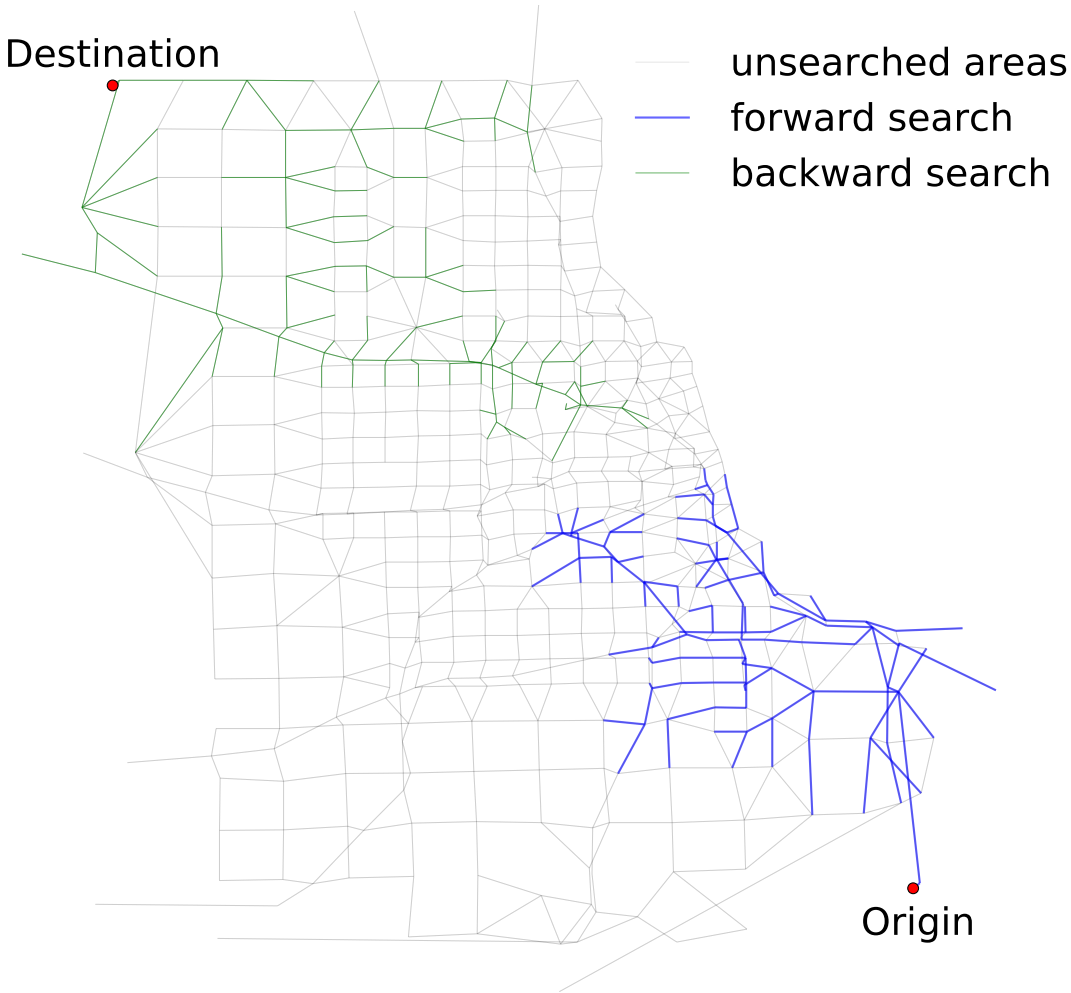
Bidirectional A* search



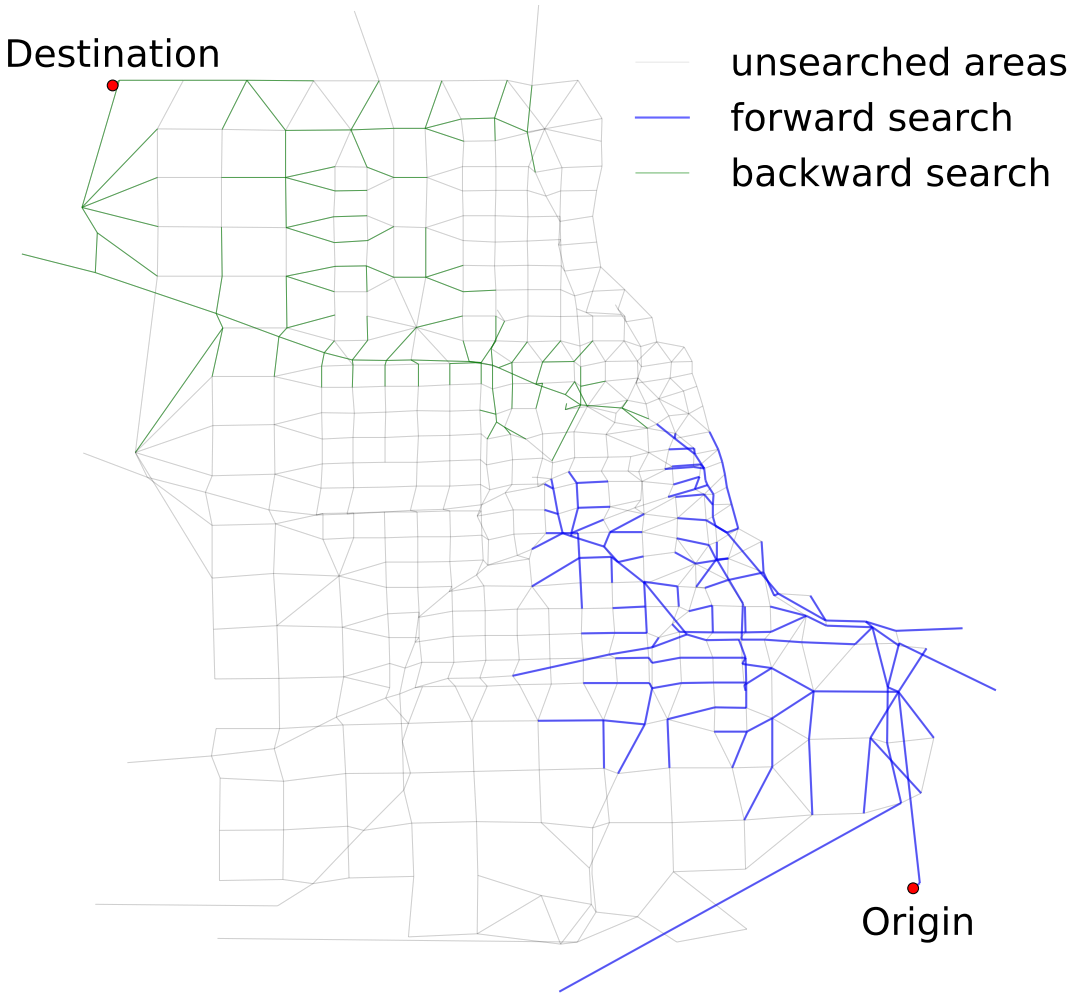
Bidirectional A* search



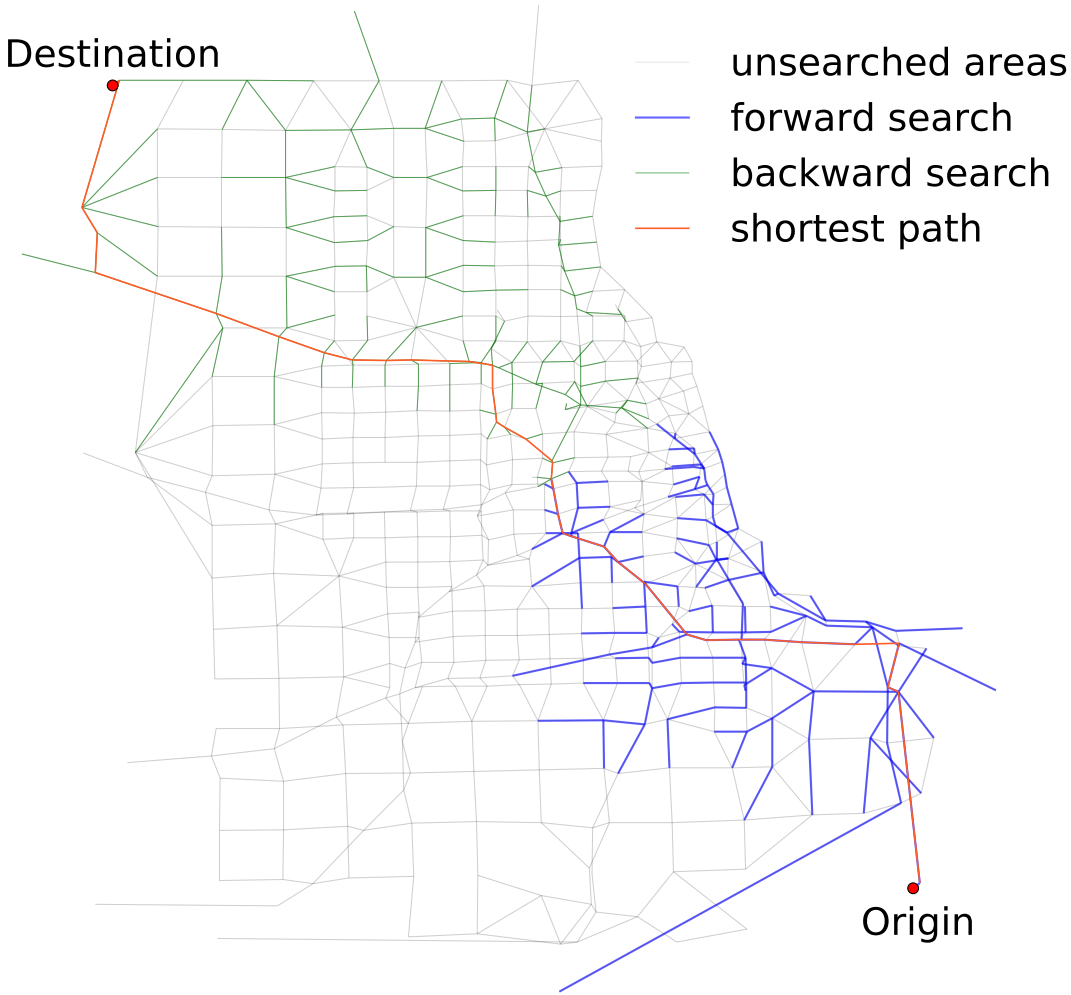
Bidirectional A* search



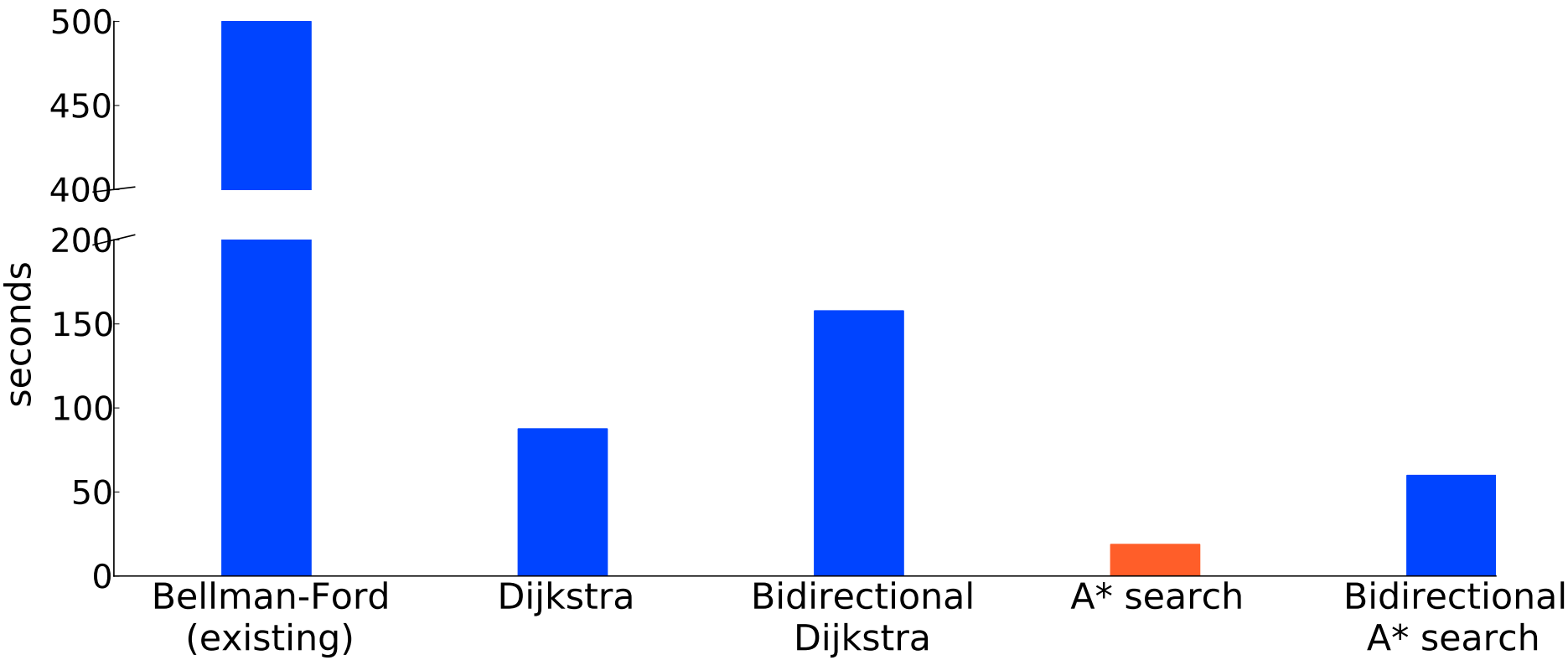
Bidirectional A* search



Bidirectional A* search



Shortest path algorithm results

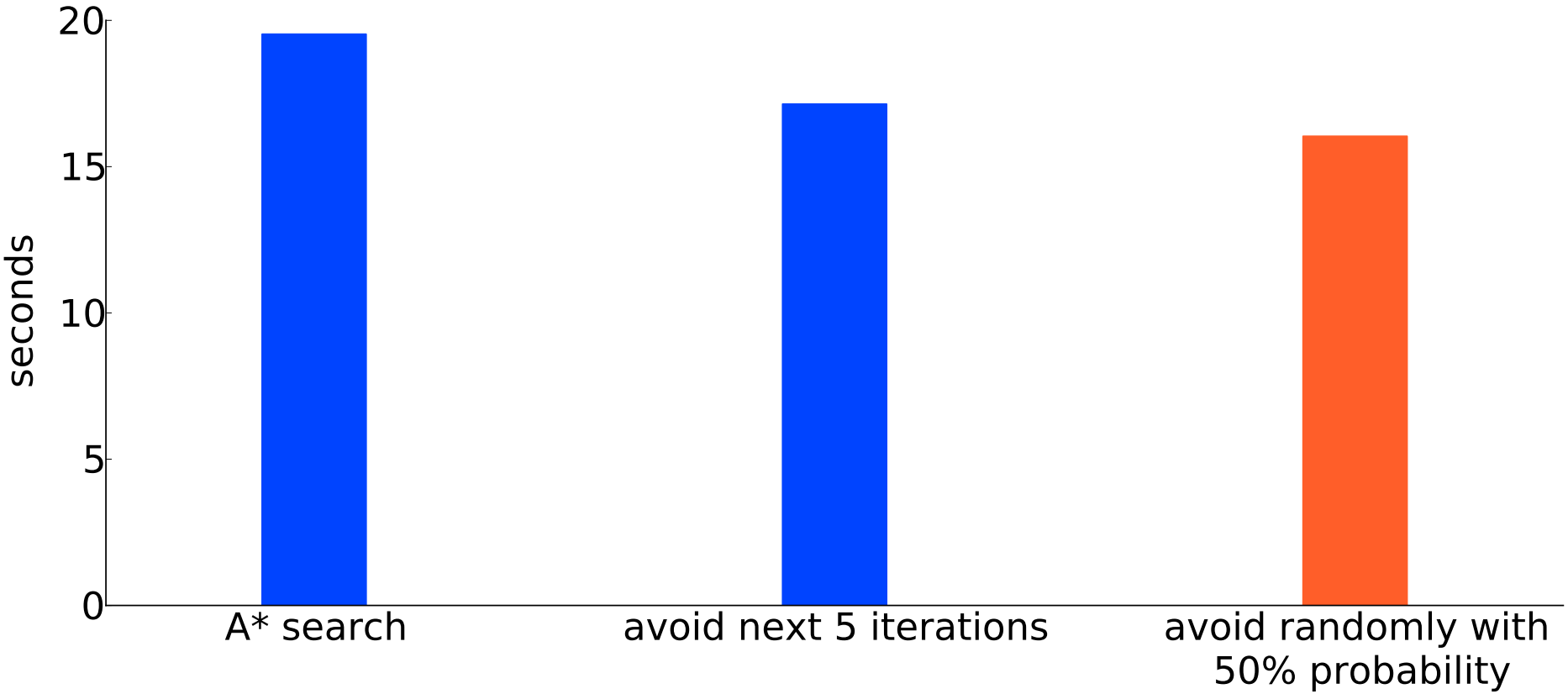


Avoiding shortest path calculations in traffic assignment

- In PE, some shortest path calculations can be avoided between iterations to speed up the overall performance
- The shortest path from the previous iteration can be re-used to avoid the calculation in the current iteration

1. avoid the next few iterations if the shortest paths of the previous two iterations are identical
2. randomly avoid the next shortest path calculation in the hope that the shortest path of previous and current iteration are identical

Avoiding shortest path calculation results



Conclusions

- 30 times faster than the existing implemented Bellman-Ford algorithm

Future Work

- preprocessing
  - multi-thread on GPU
  - use the avoiding stratey on similar algorithms that solve the traffic assignment problem