Faster Shortest Path Computation for Traffic Assignment

Boshen Chen
Supervised by: Dr. Andrea Raith, Olga Perederieieva

Department of Engineering Science
University of Auckland

Traffic Assignment

Traffic Assignment

- transportation network with supply and demand nodes

Traffic Assignment

- transportation network with supply and demand nodes
- minimise travel times

Traffic Assignment

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

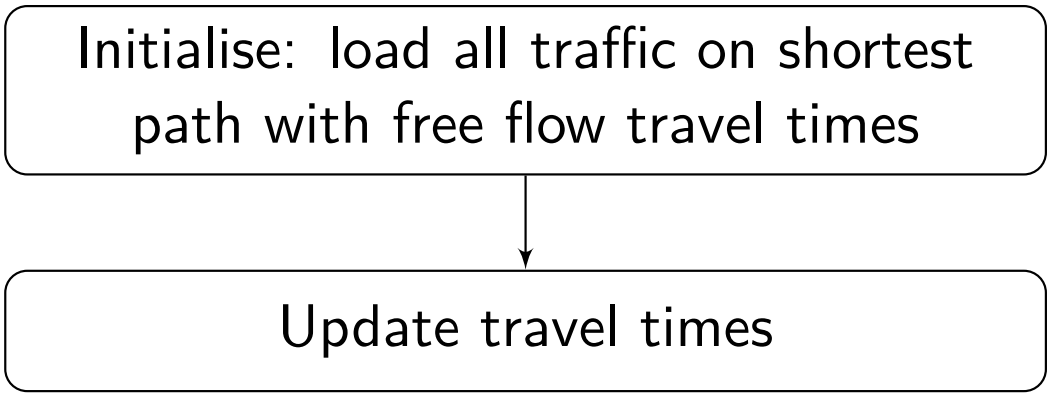
Traffic Assignment

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

Initialise: load all traffic on shortest path with free flow travel times

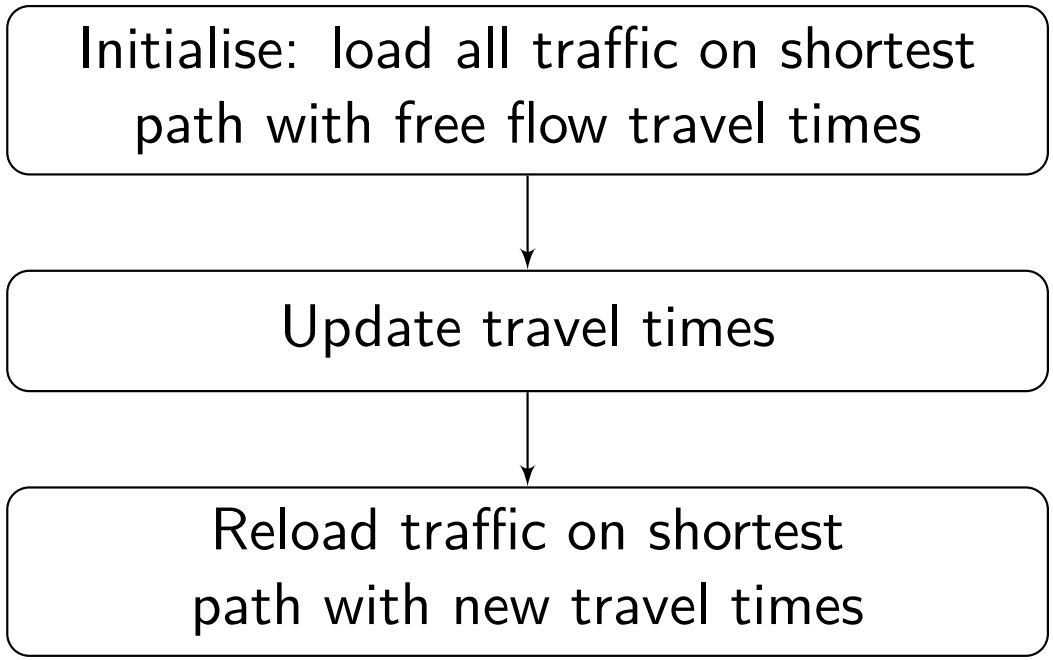
Traffic Assignment

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



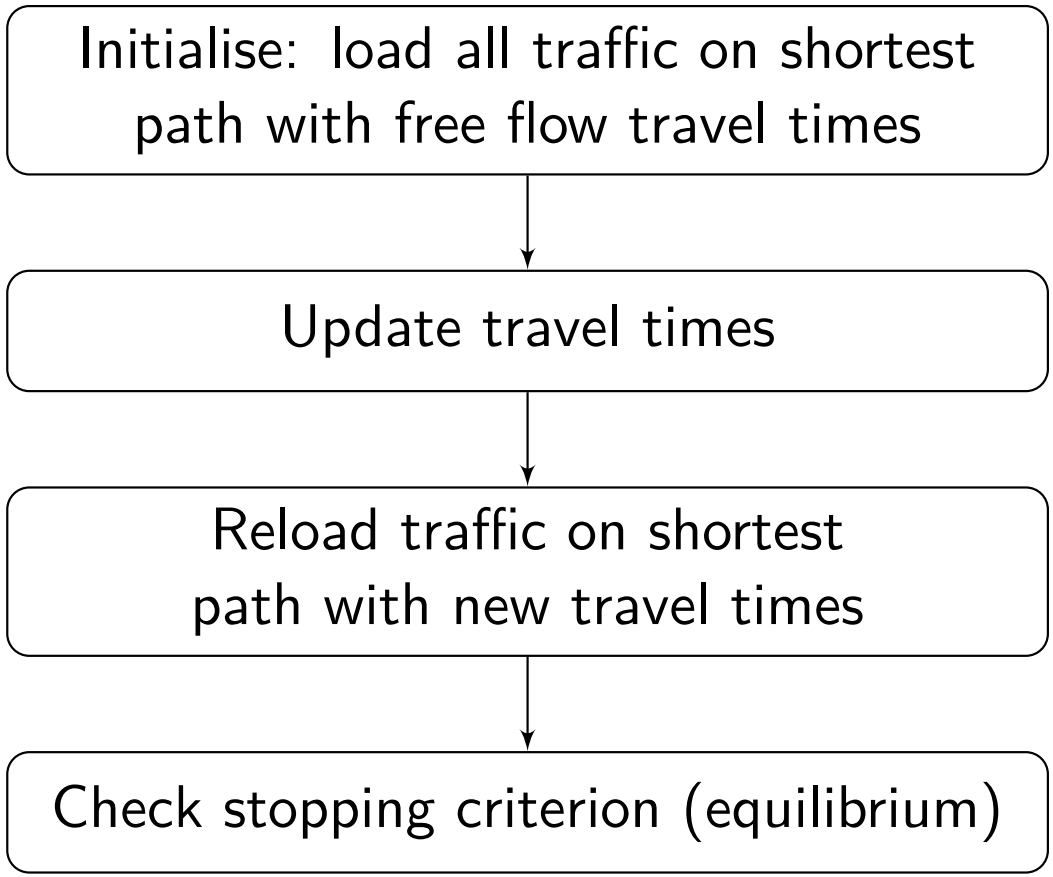
Traffic Assignment

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



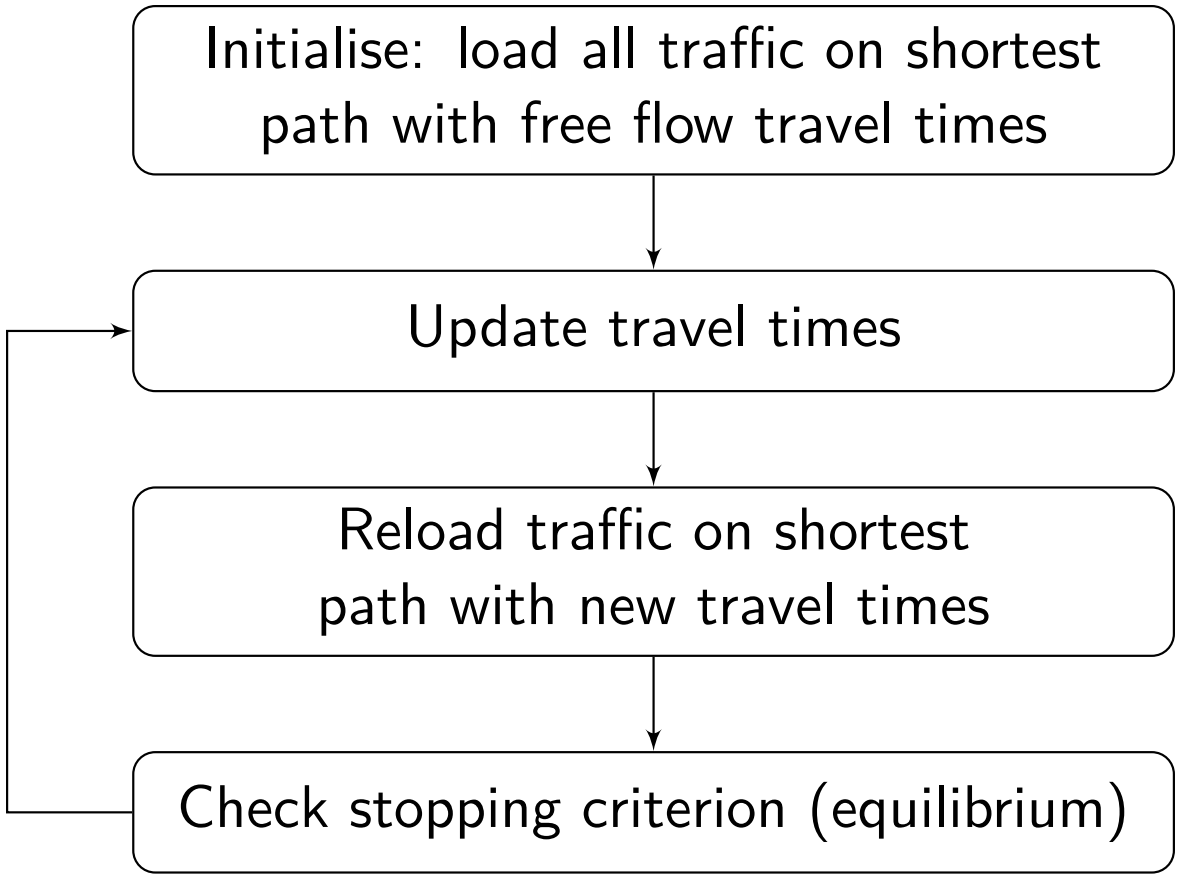
Traffic Assignment

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

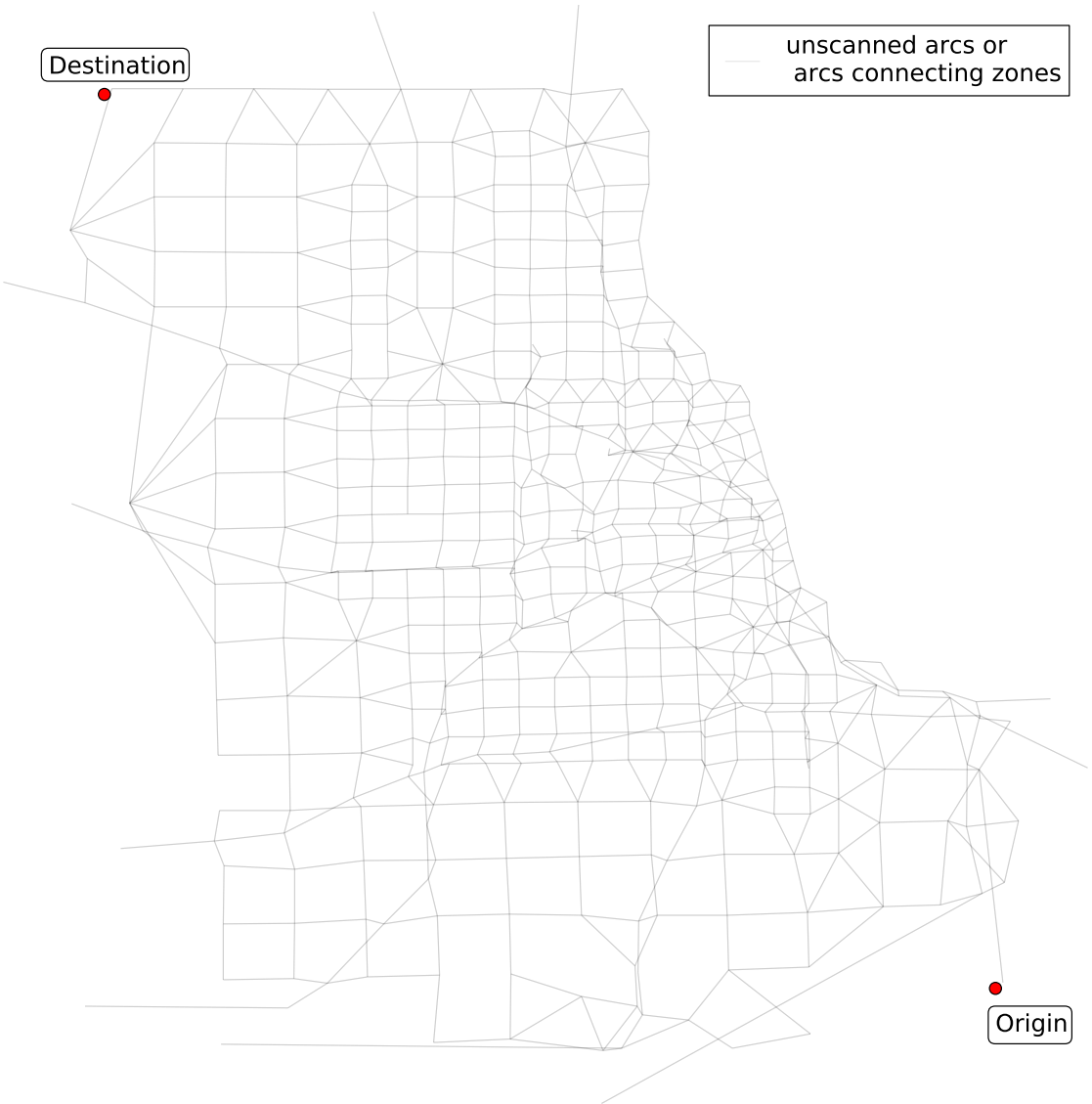


Traffic Assignment

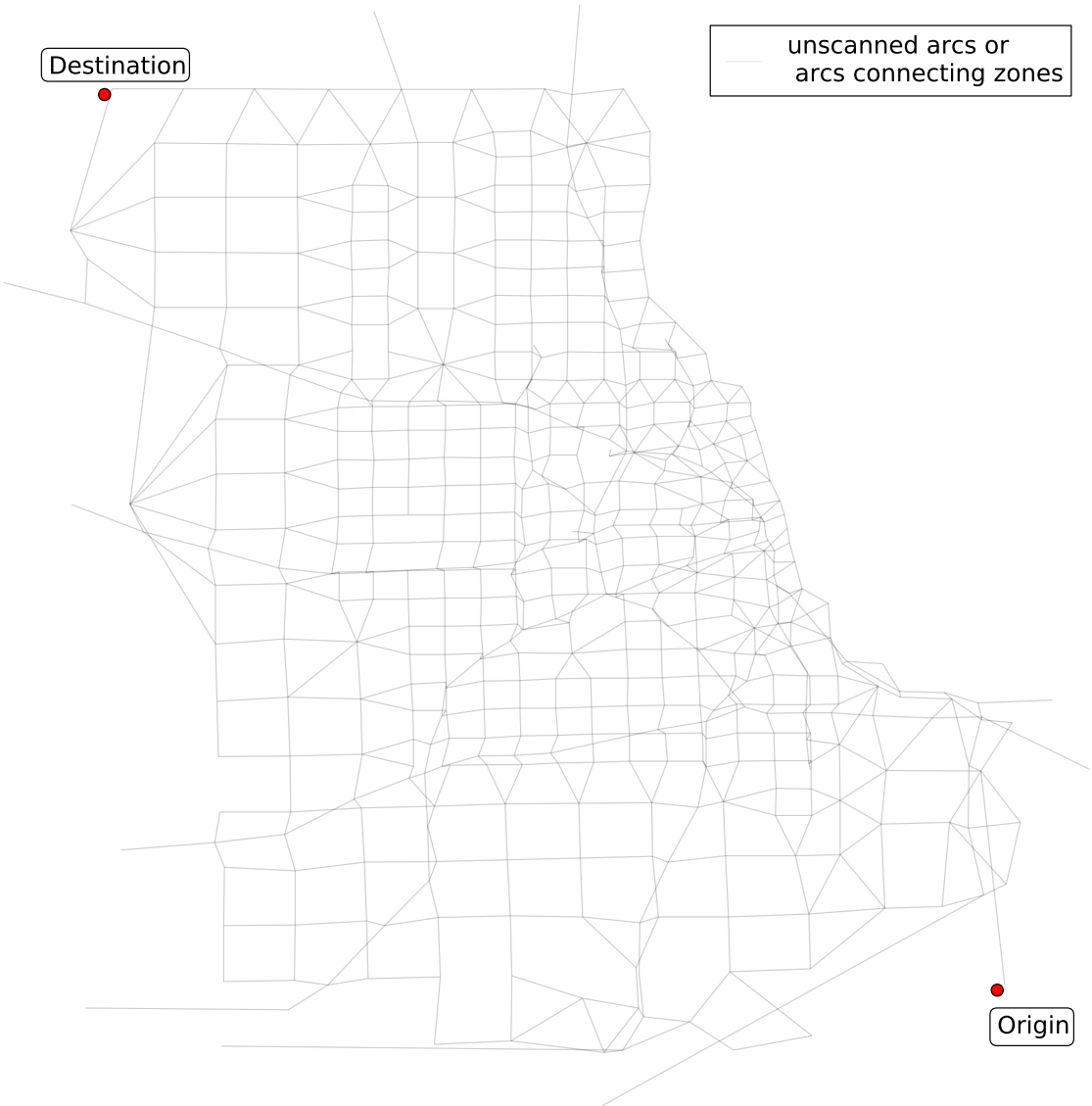
- transportation network with supply and demand nodes
- minimise travel times
- arcs have non-linear travel times for capturing congestion effects



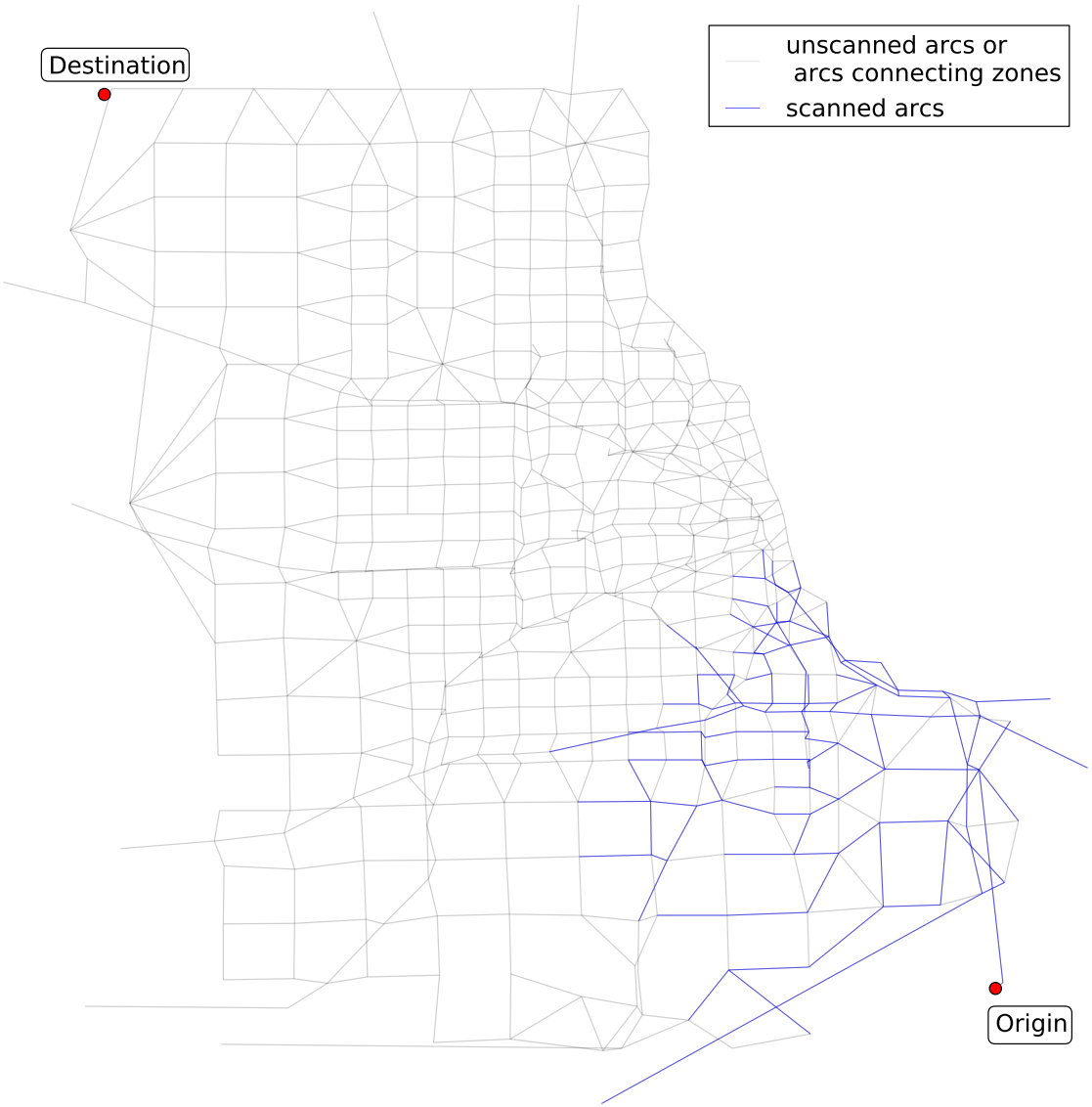
The Graph - 93,135 Origin-Destination Pairs



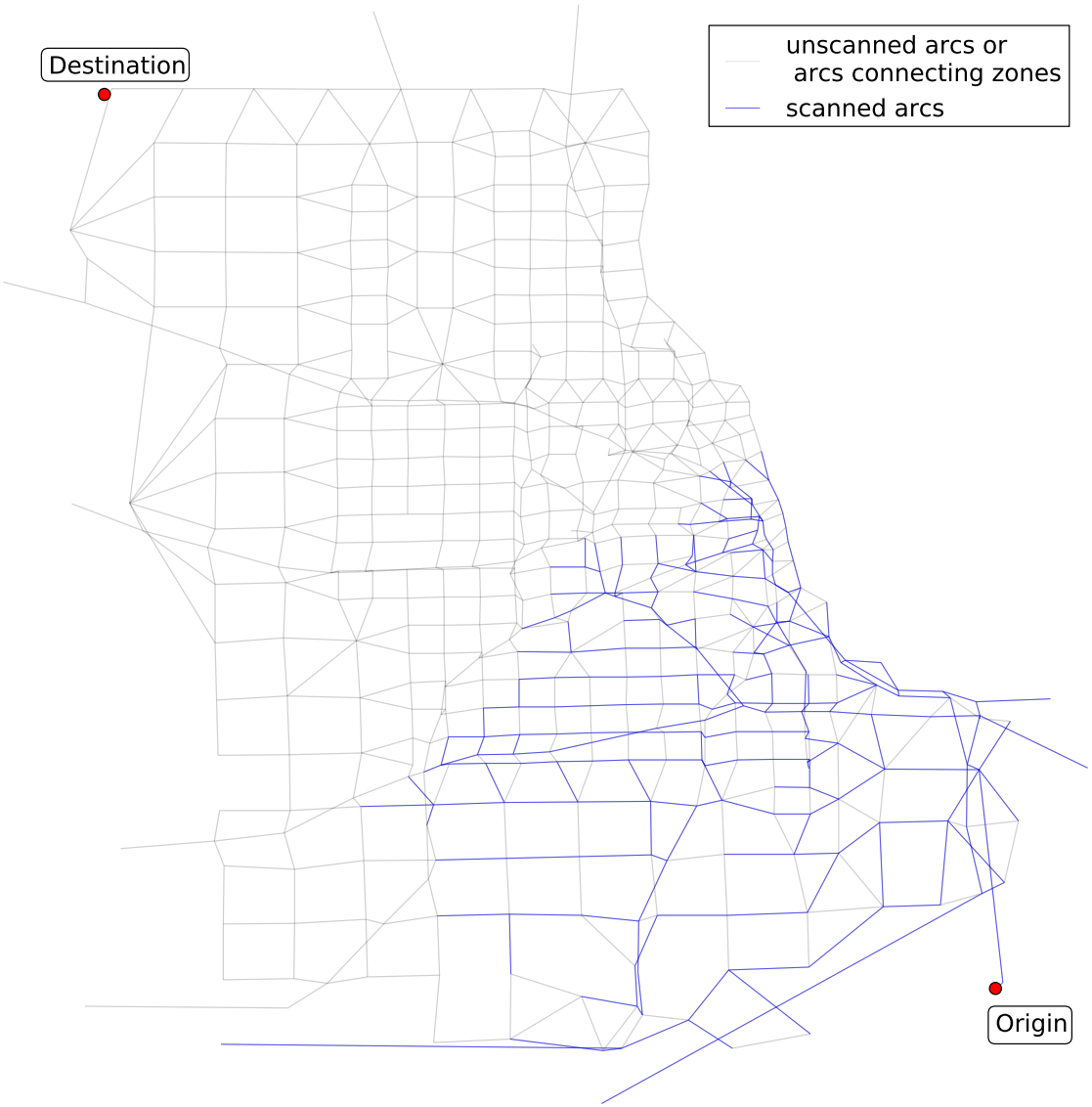
Dijkstra's Algorithm



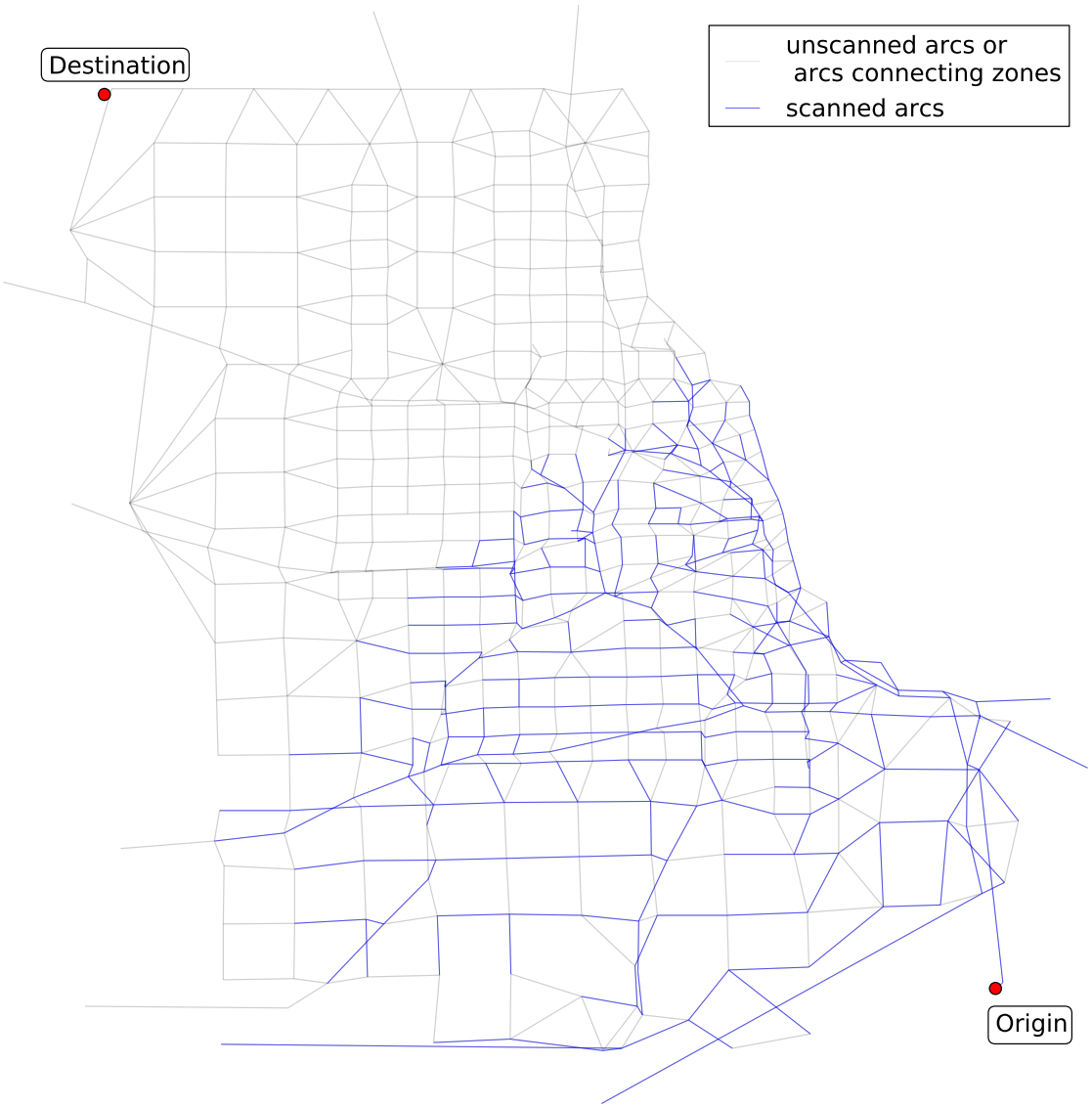
Dijkstra's Algorithm



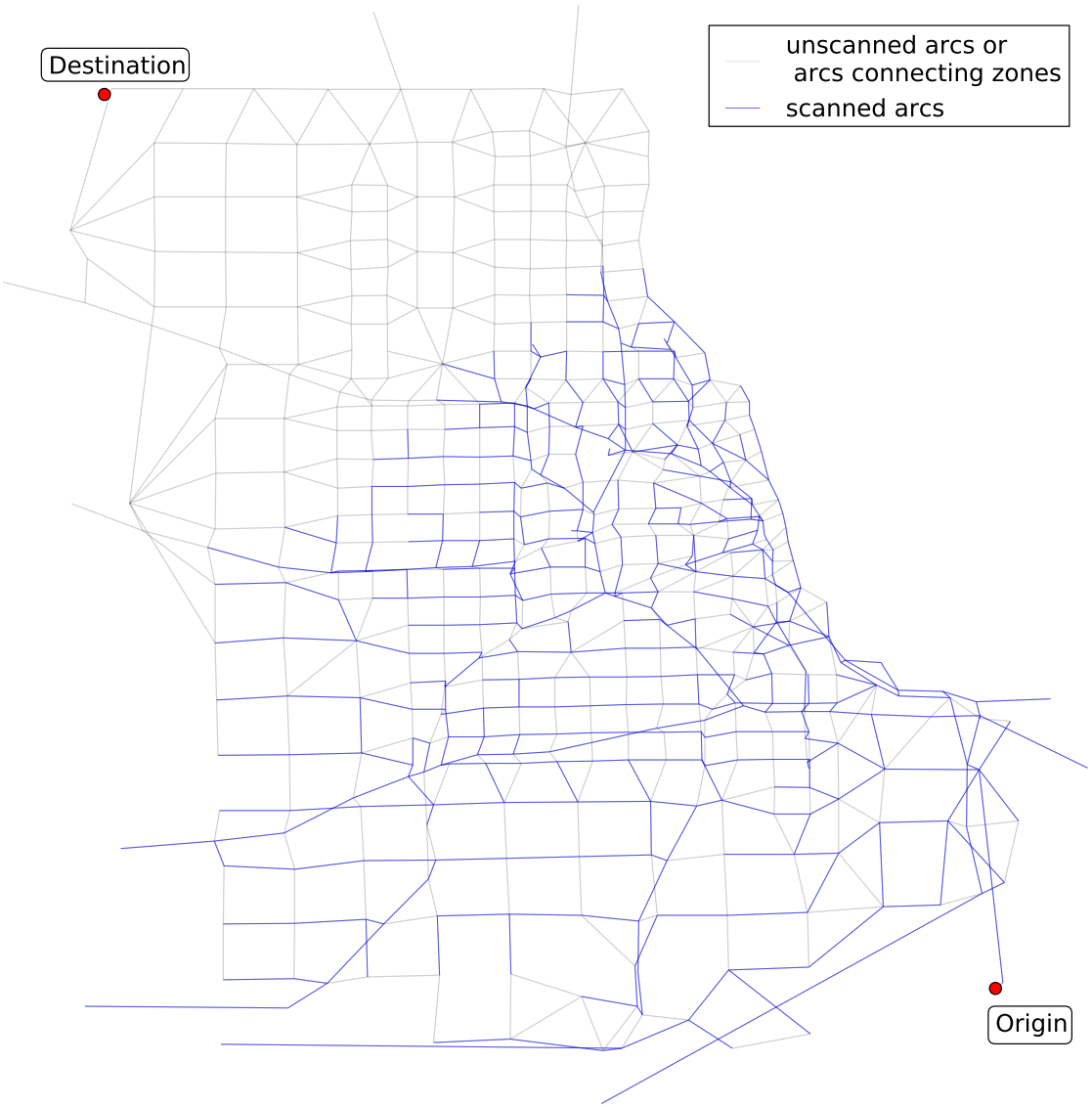
Dijkstra's Algorithm



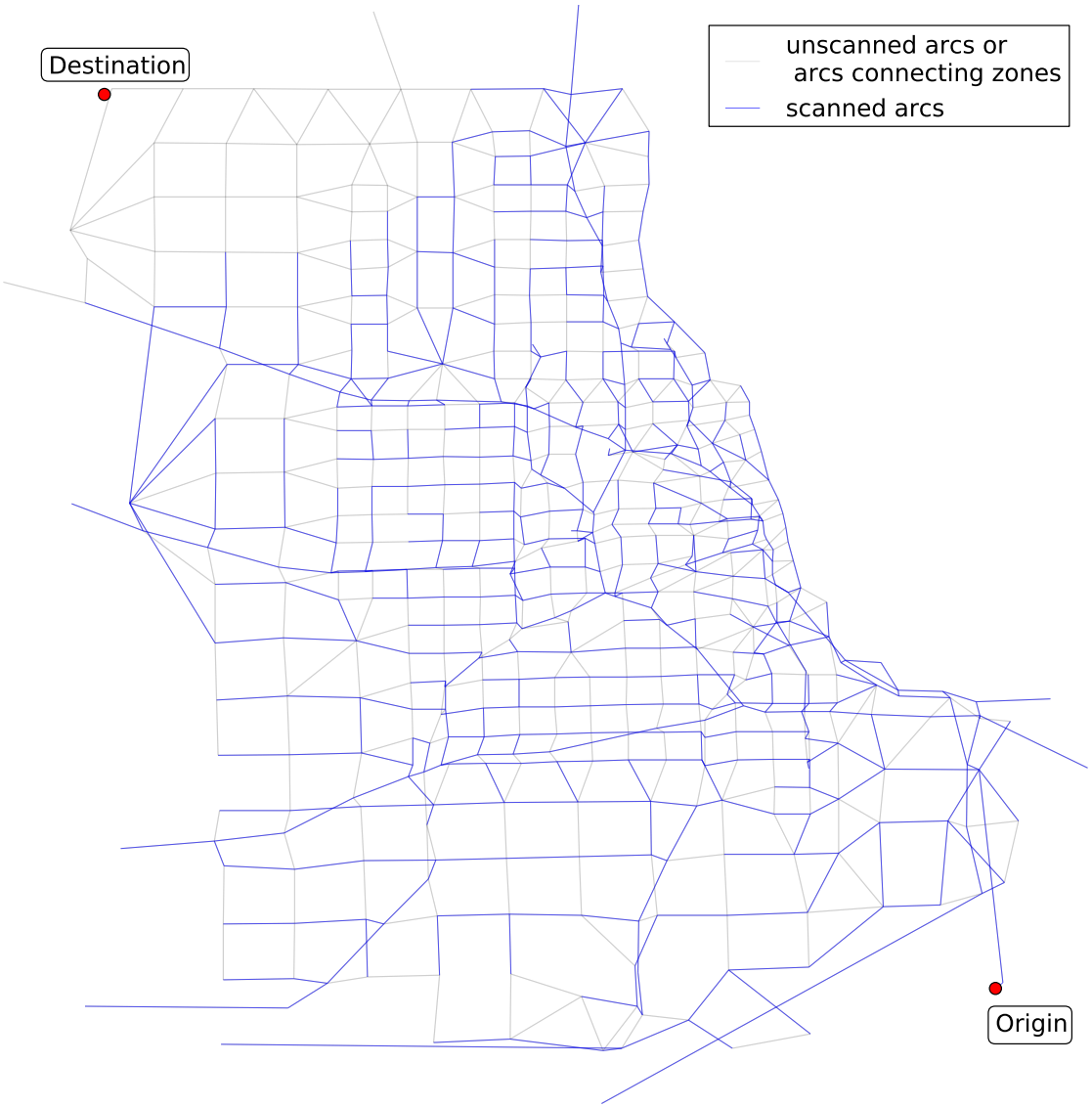
Dijkstra's Algorithm



Dijkstra's Algorithm



Dijkstra's Algorithm

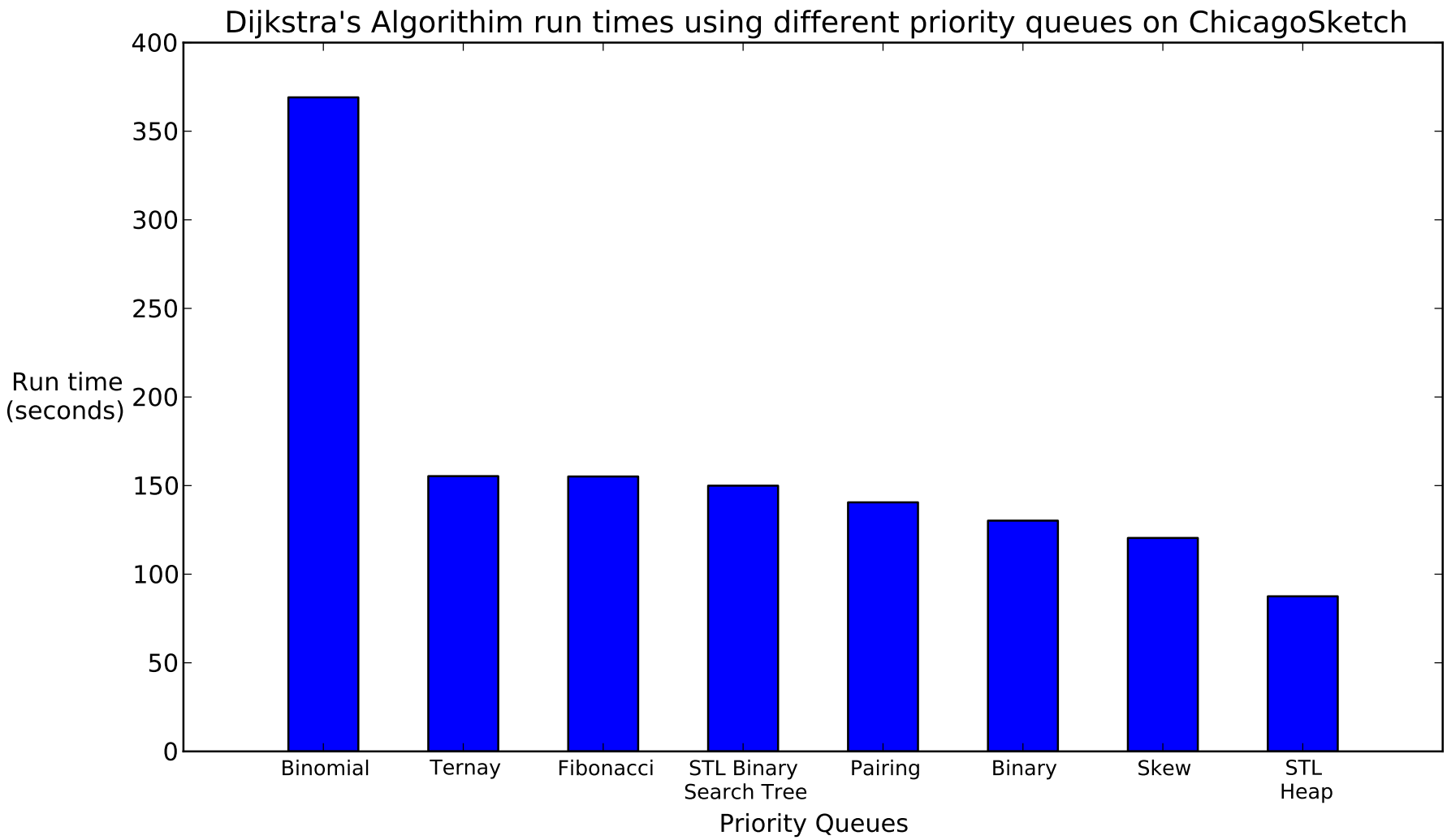


Dijkstra's Algorithm

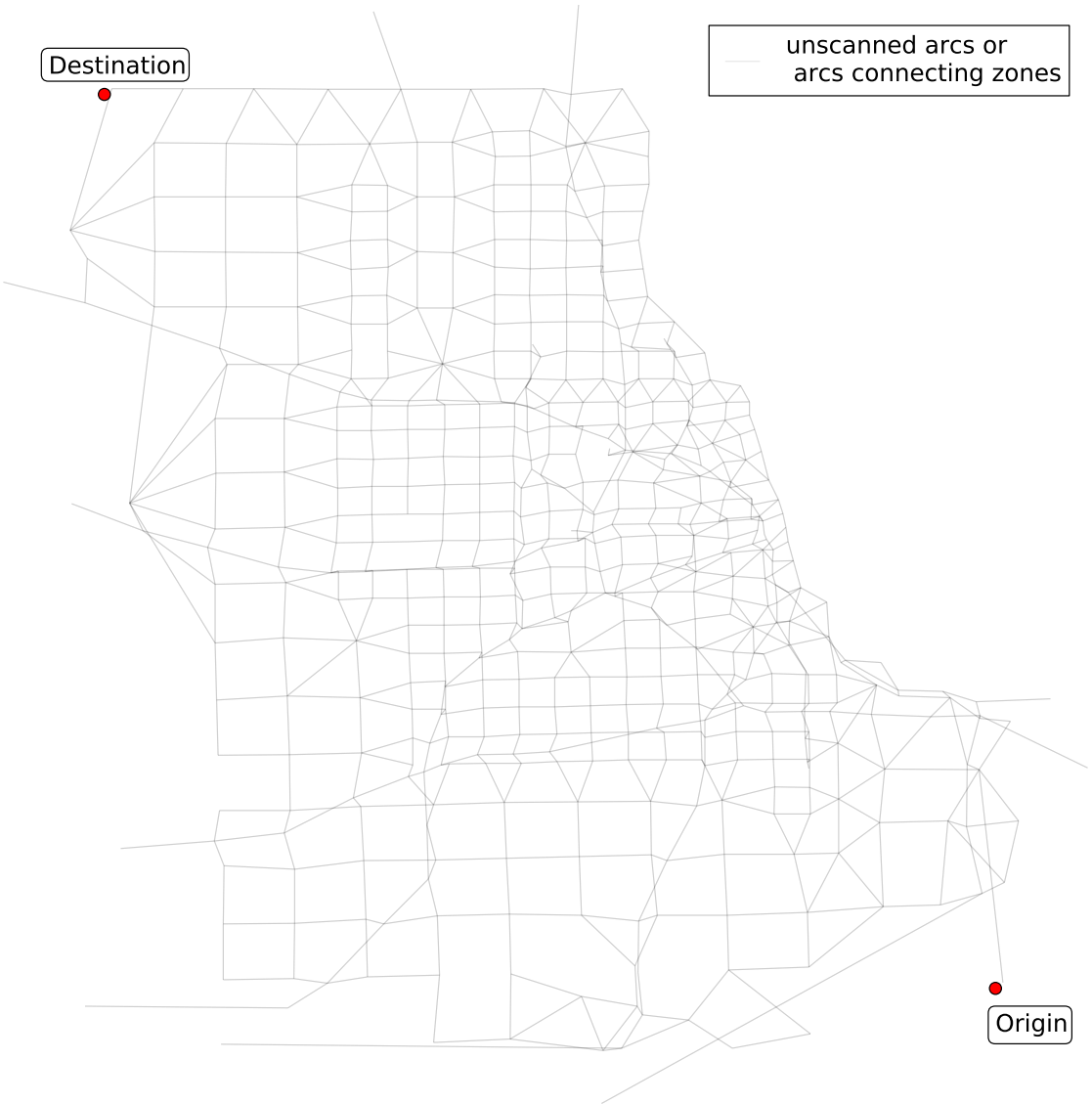
Dijkstra's Algorithm - Priority Queue

- pointer based Heap (C++ boost library)
  - Binomial
  - Pairing
  - Binary
  - Ternary
  - Skew
  - Fibonacci
- (red-black) binary search tree (C++ STL <set>)
- array based Heap (C++ STL <priority_queue>)

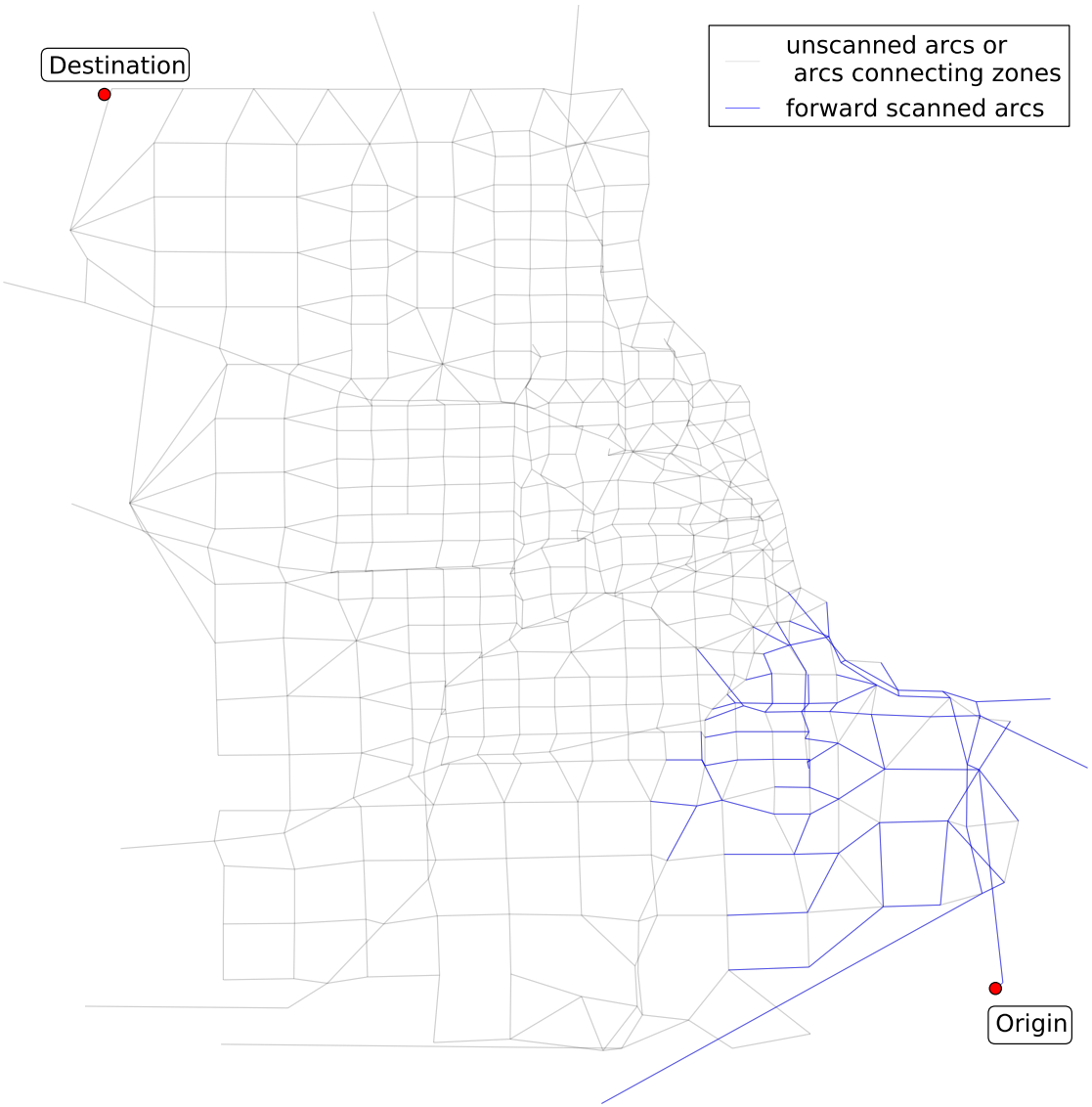
Dijkstra's Algorithm - Priority Queue



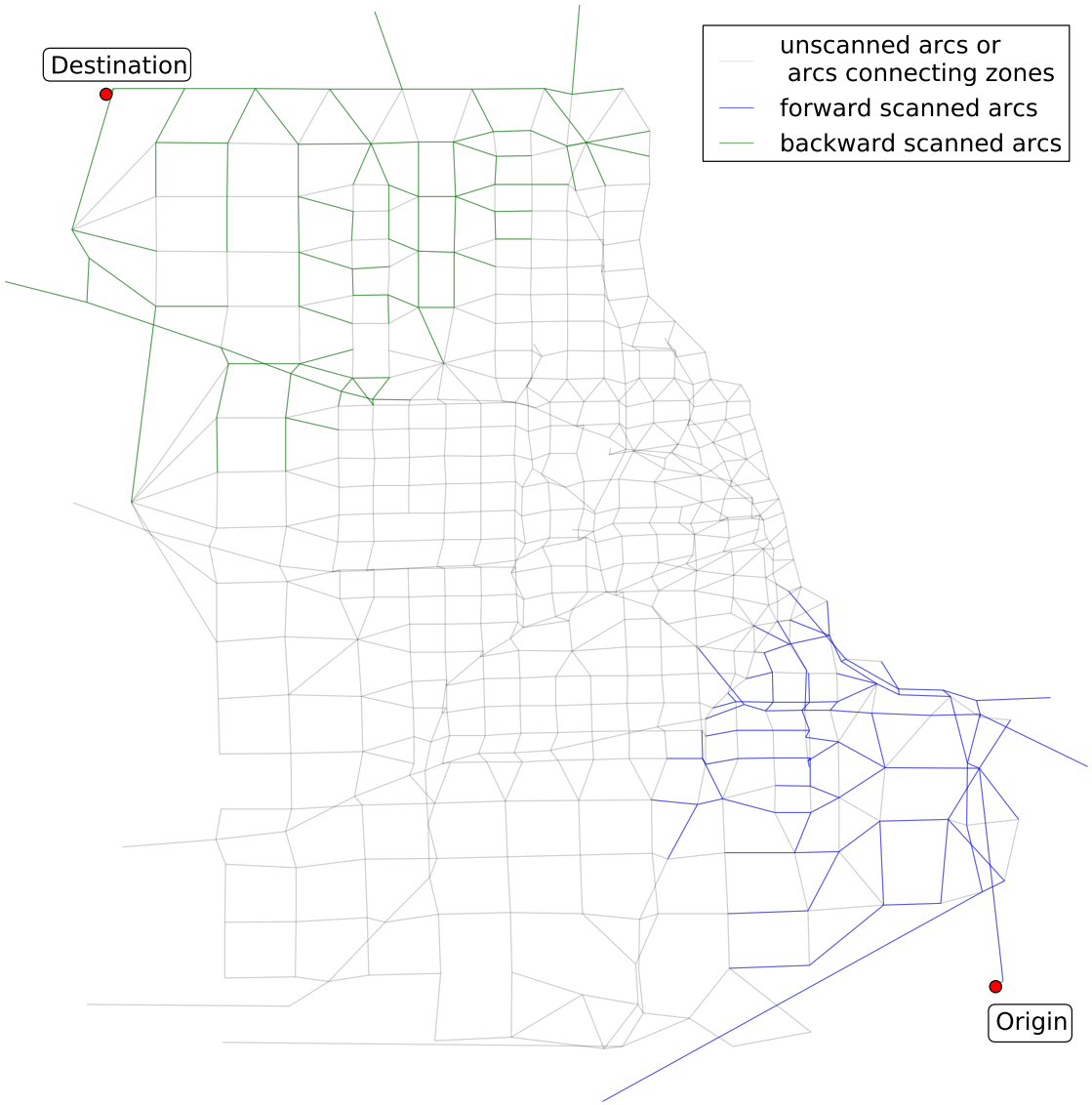
Bidirectional Dijkstra's Algorithm



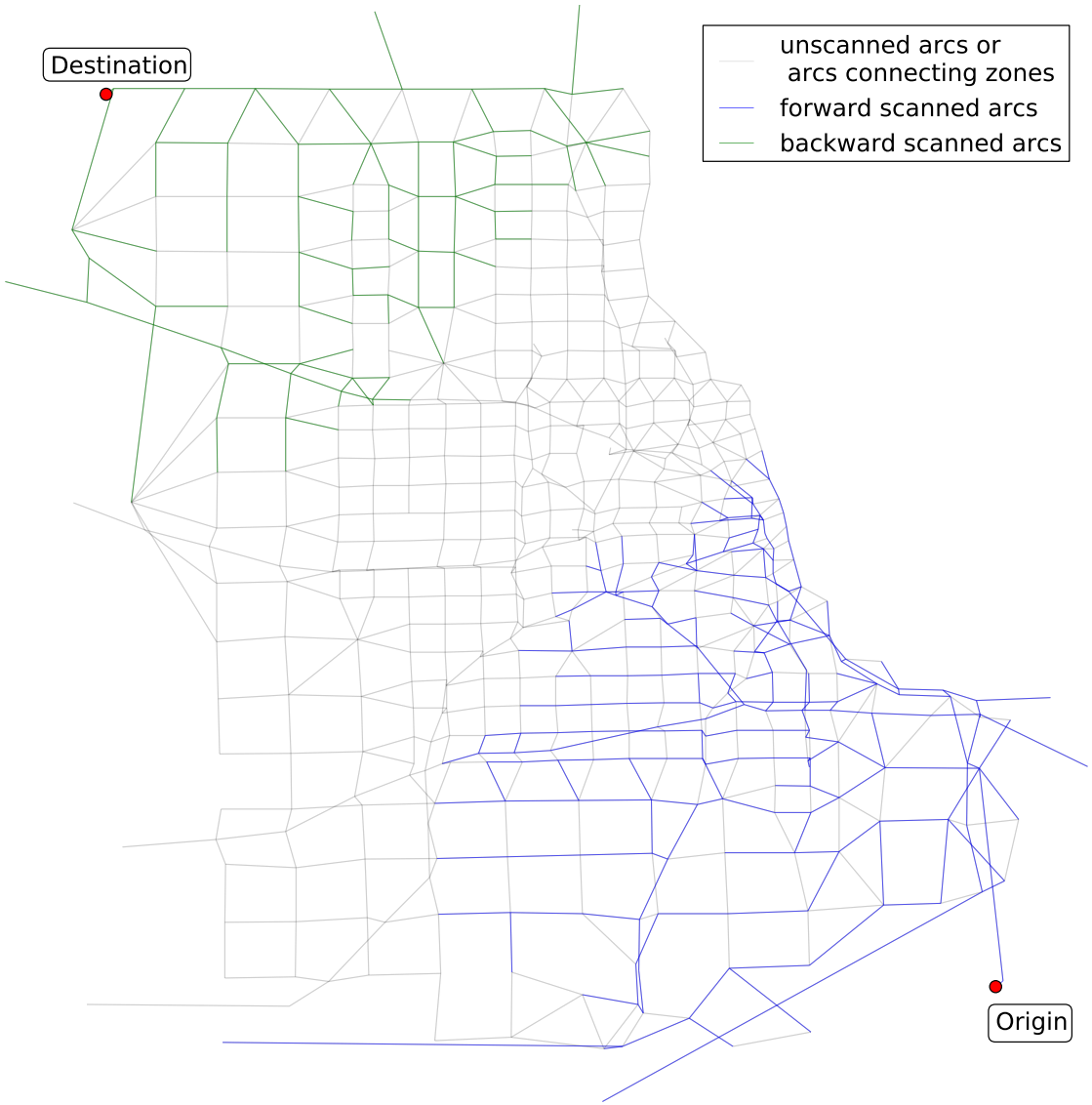
Bidirectional Dijkstra's Algorithm



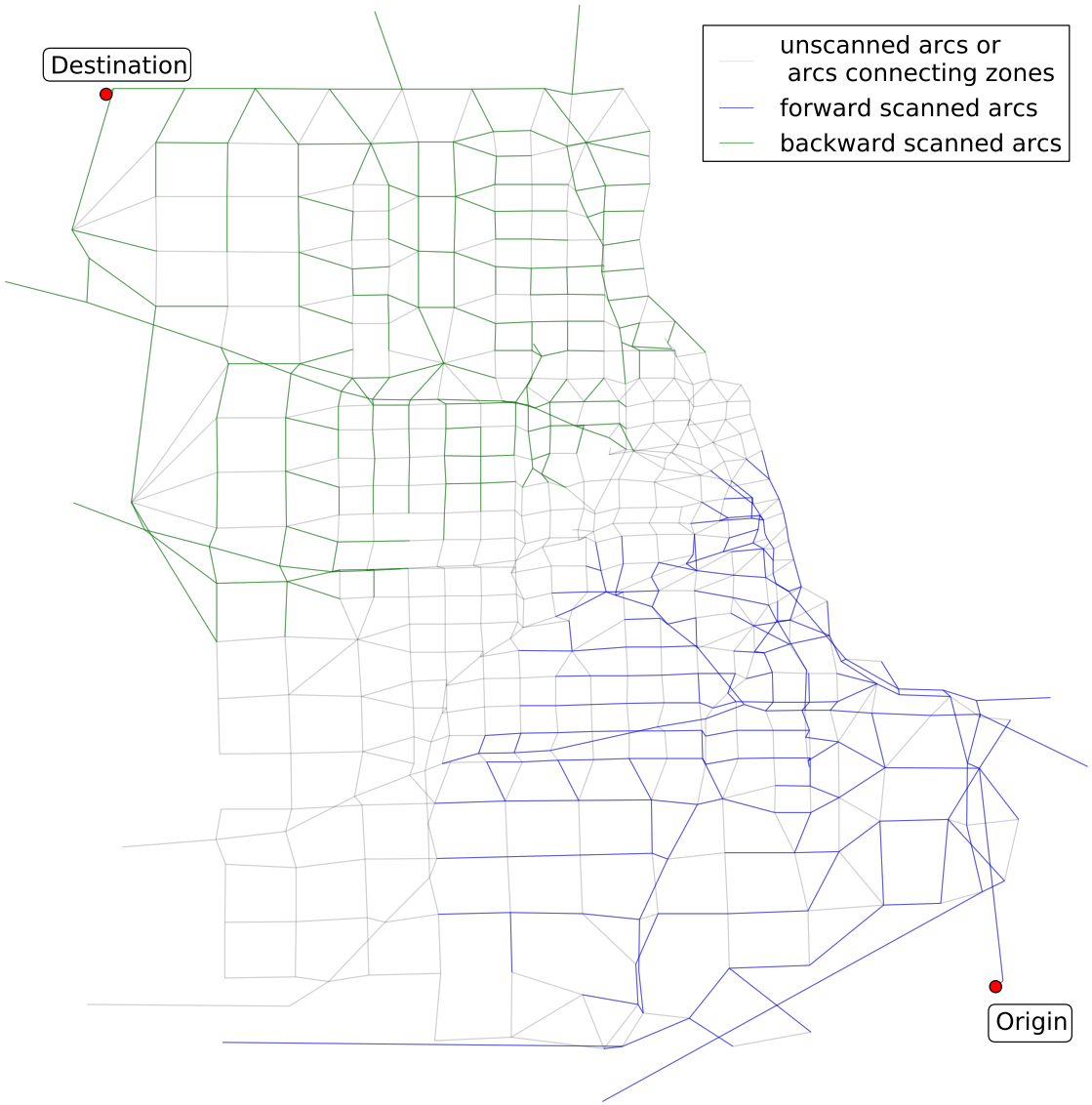
Bidirectional Dijkstra's Algorithm



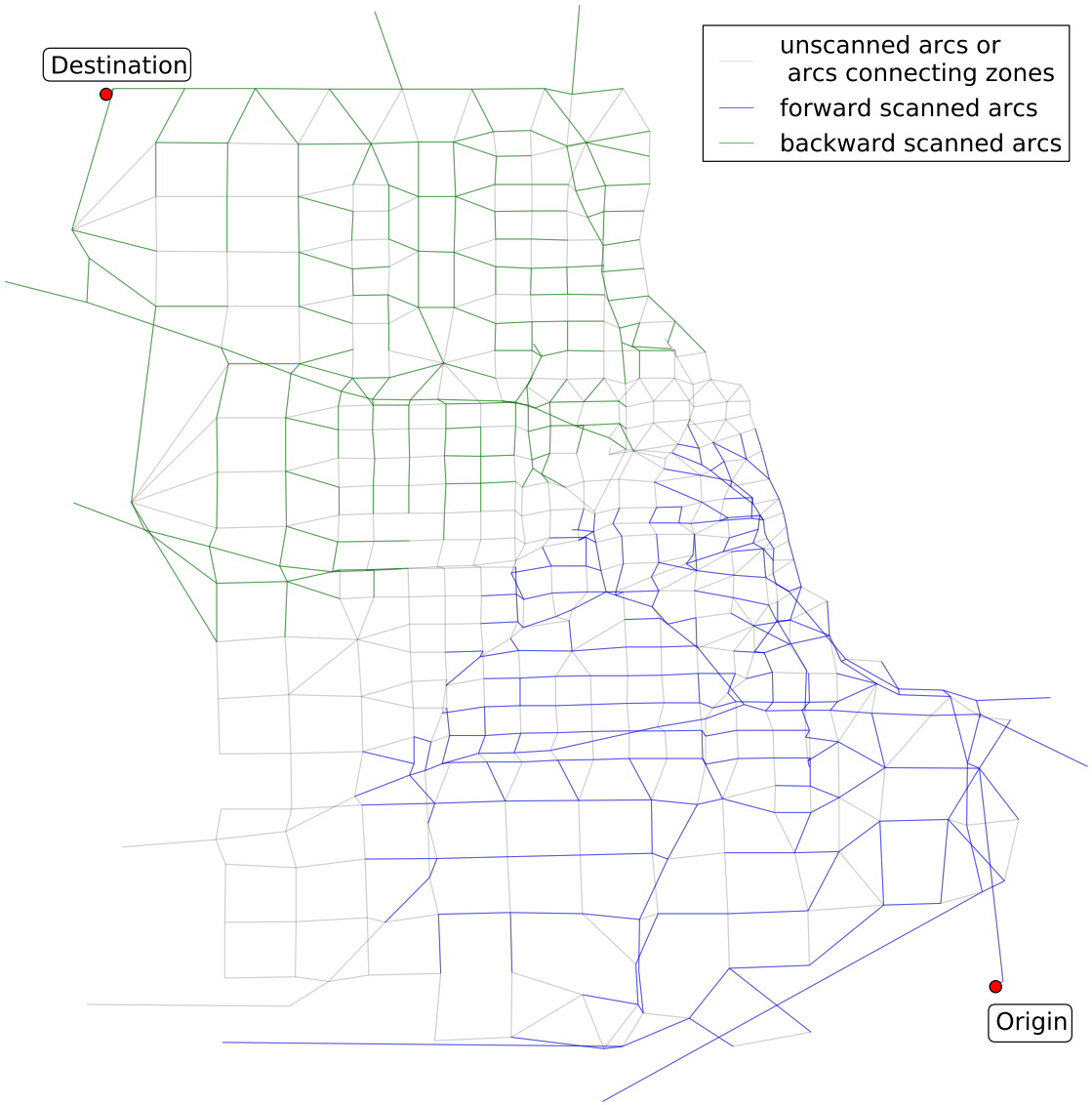
Bidirectional Dijkstra's Algorithm



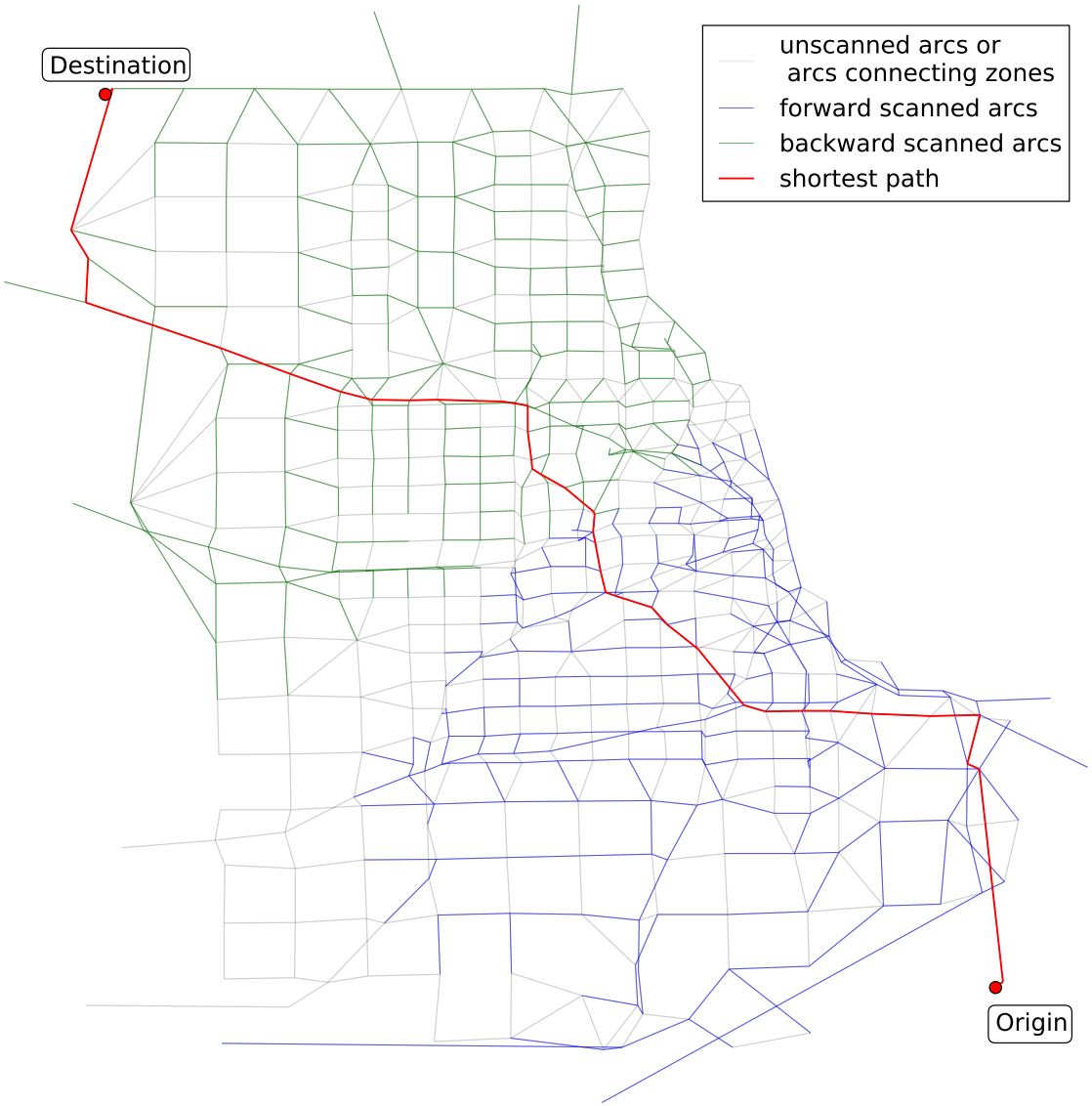
Bidirectional Dijkstra's Algorithm



Bidirectional Dijkstra's Algorithm

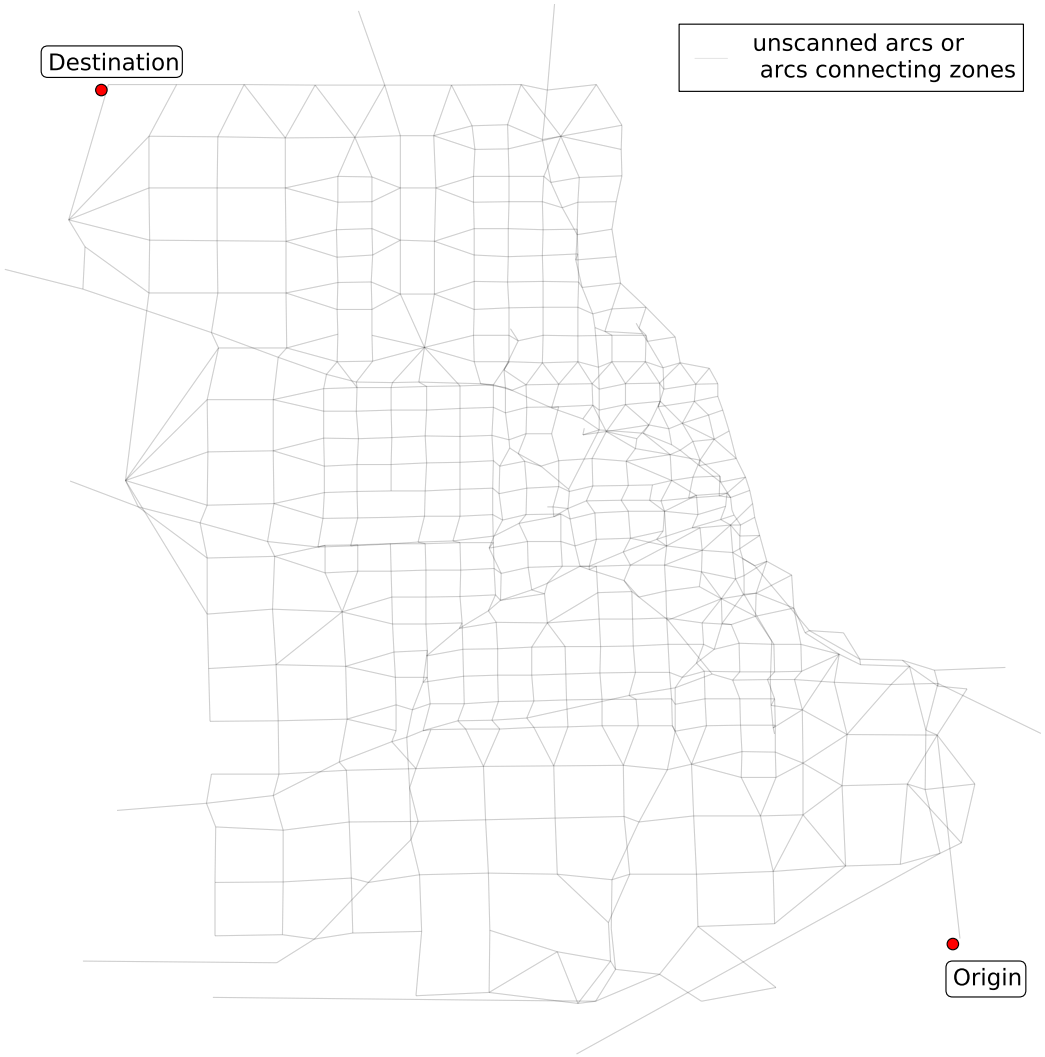


Bidirectional Dijkstra's Algorithm



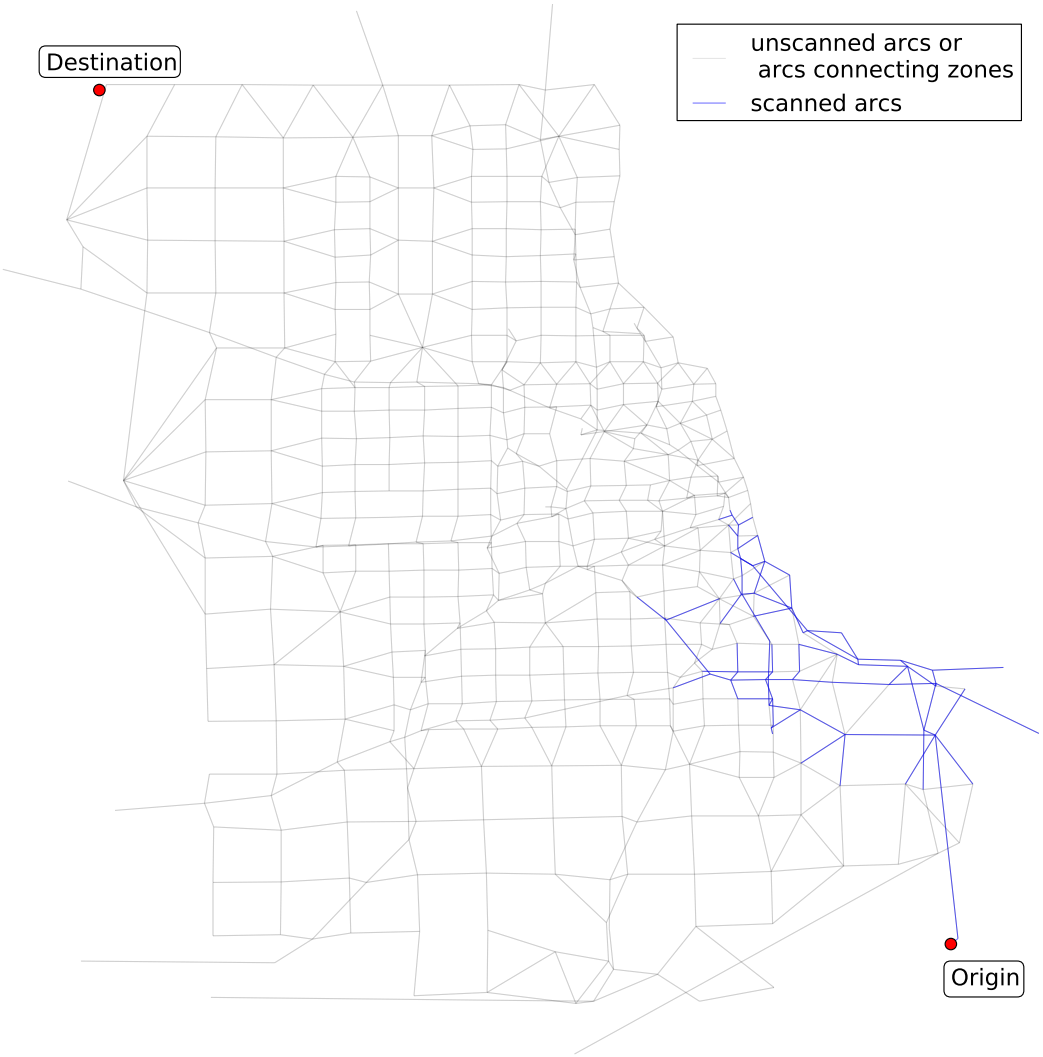
A* Search (Best-First Search)

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



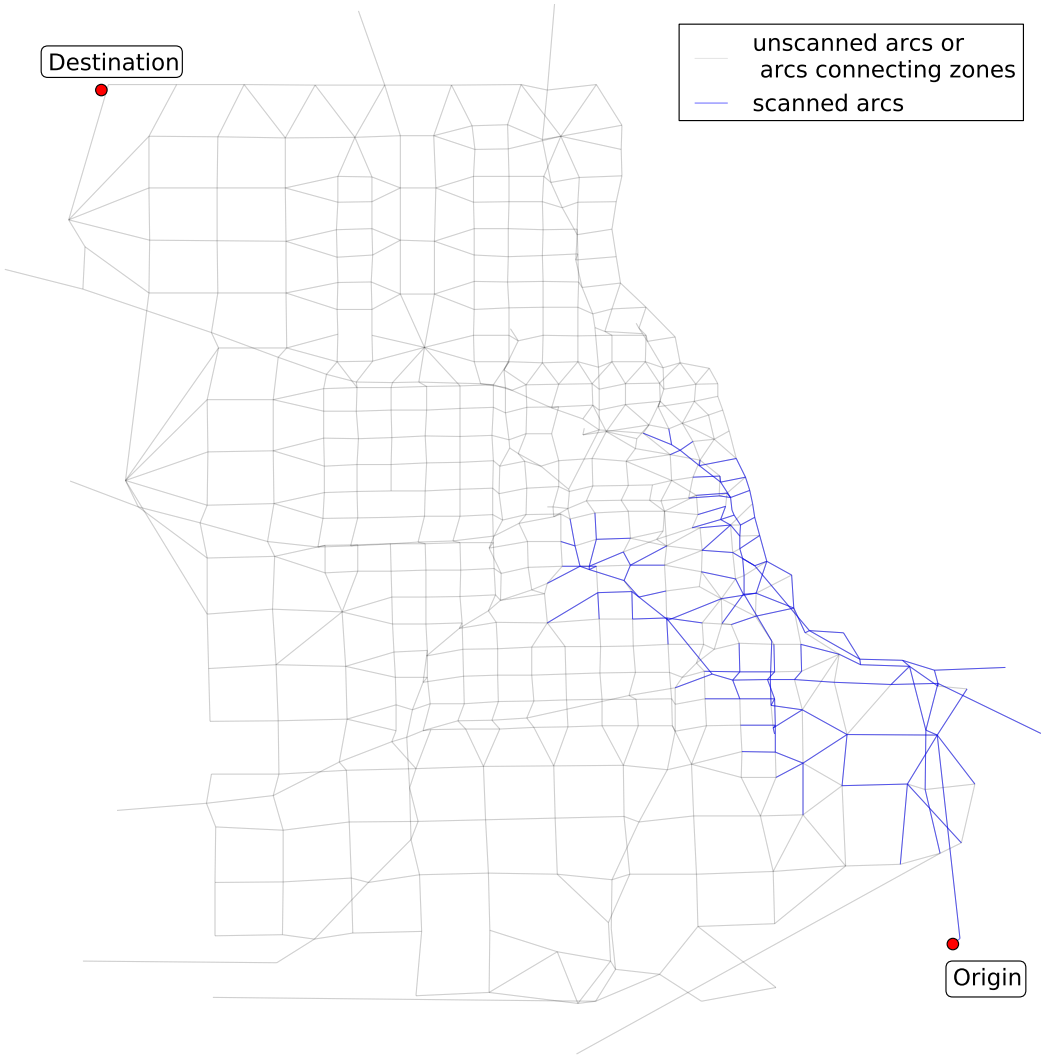
A* Search (Best-First Search)

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



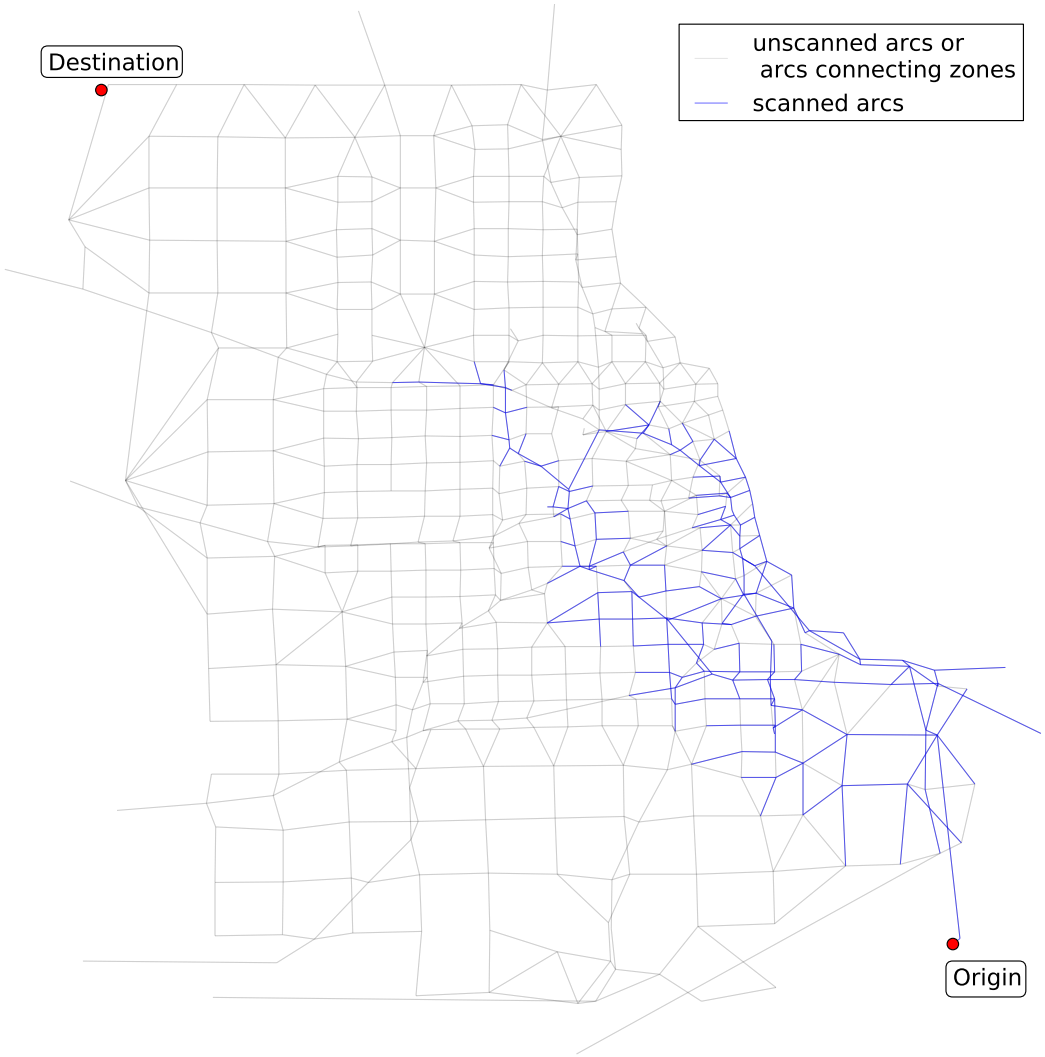
A* Search (Best-First Search)

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



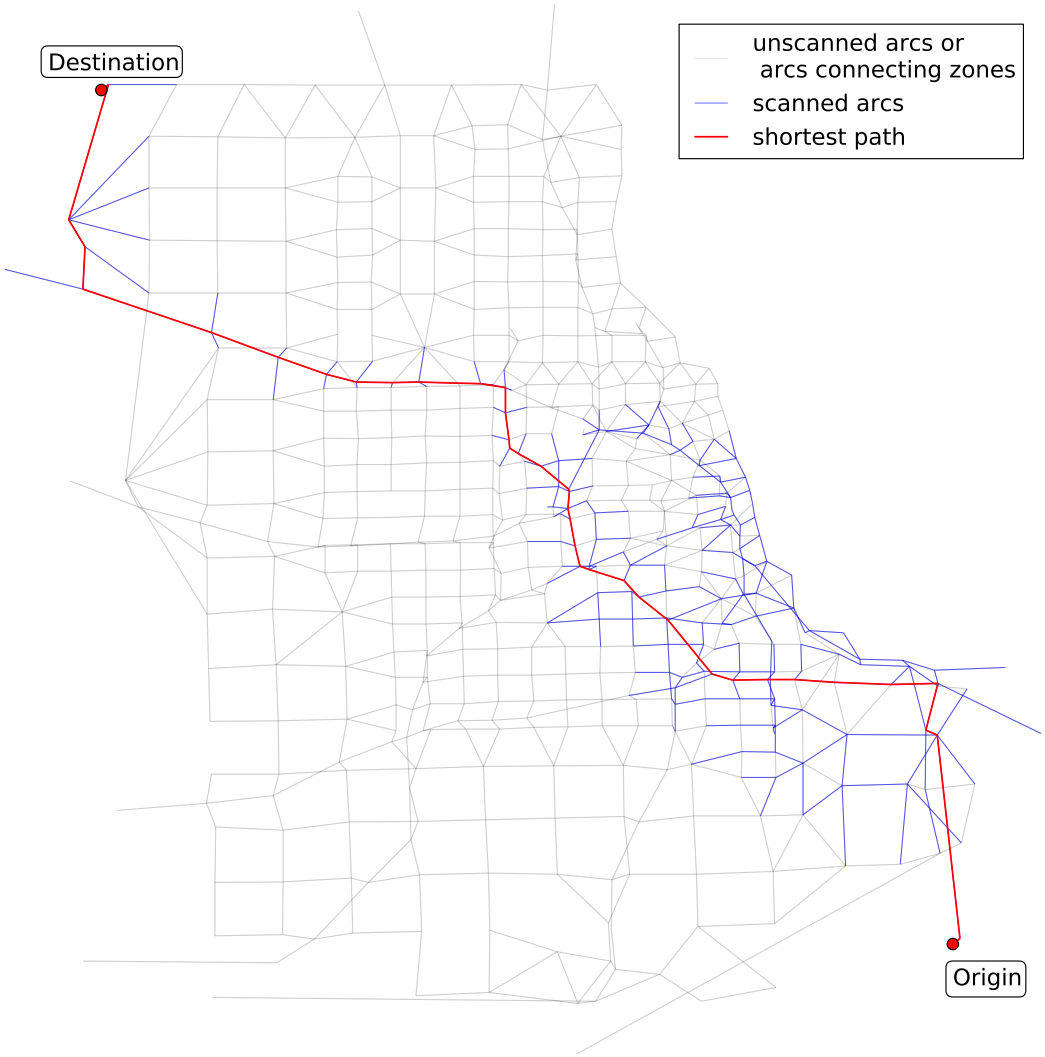
A* Search (Best-First Search)

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

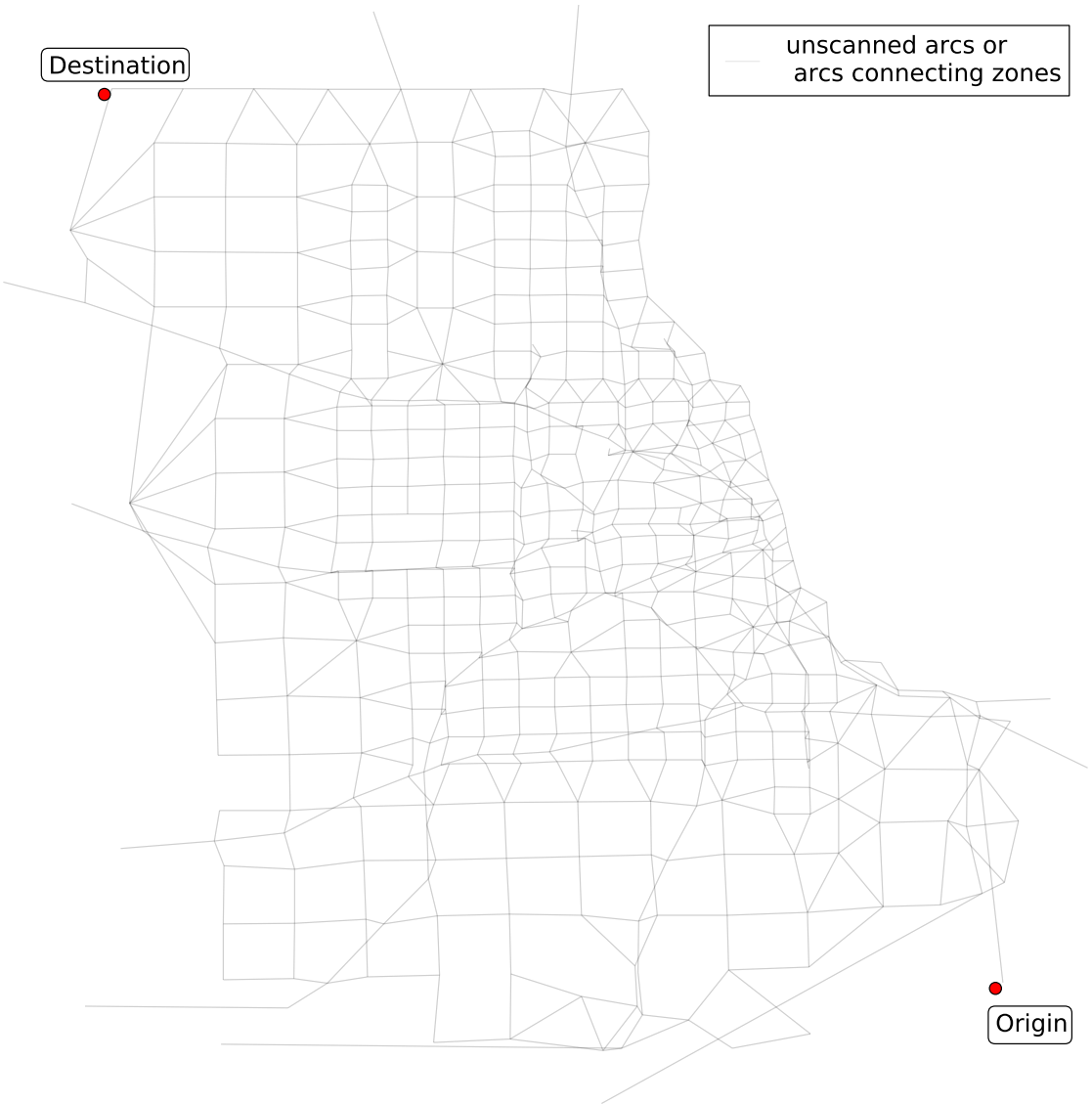


A* Search (Best-First Search)

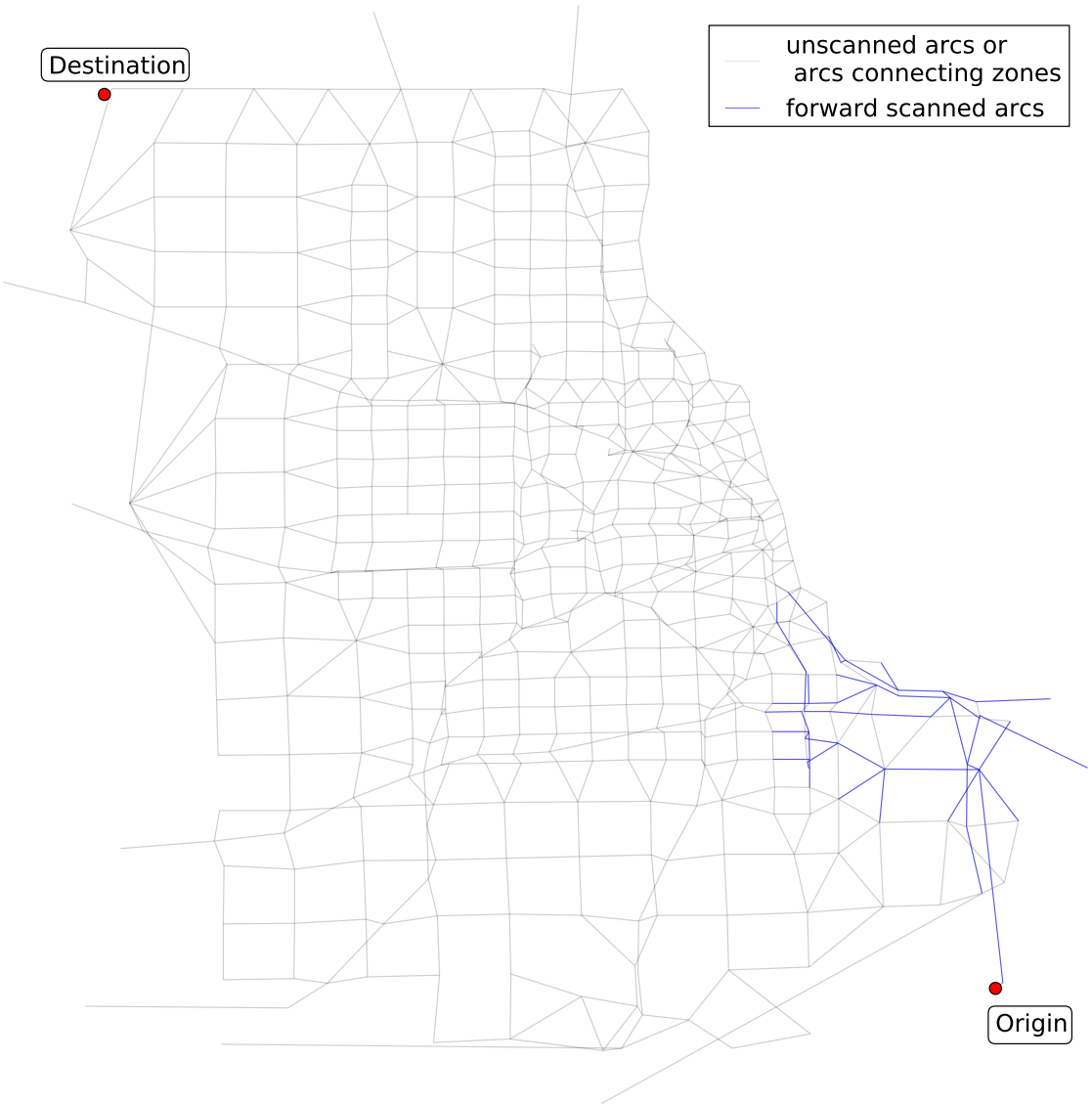
- Visit the next node that is on the expected shortest path.



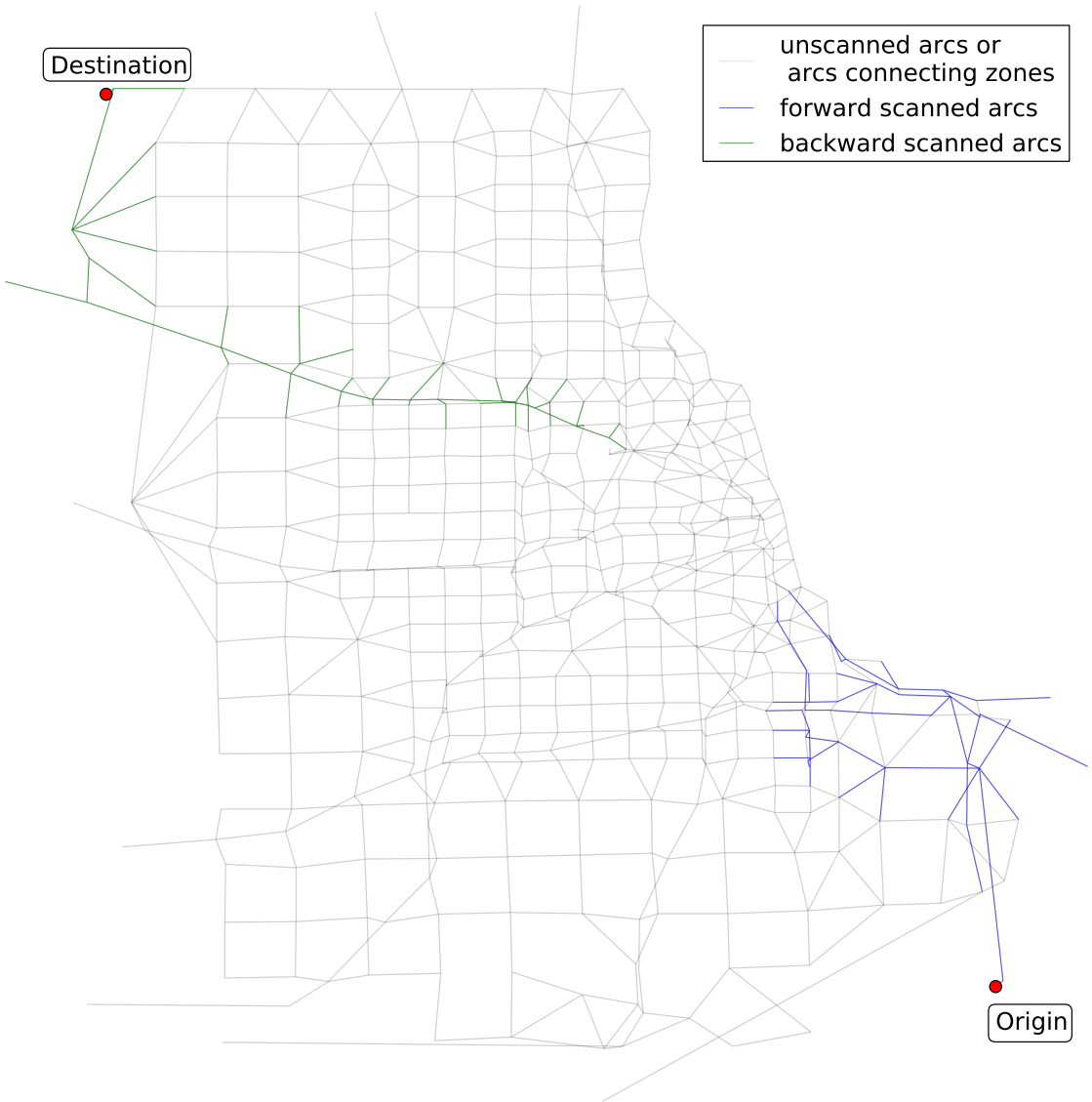
Bidirectional A* search



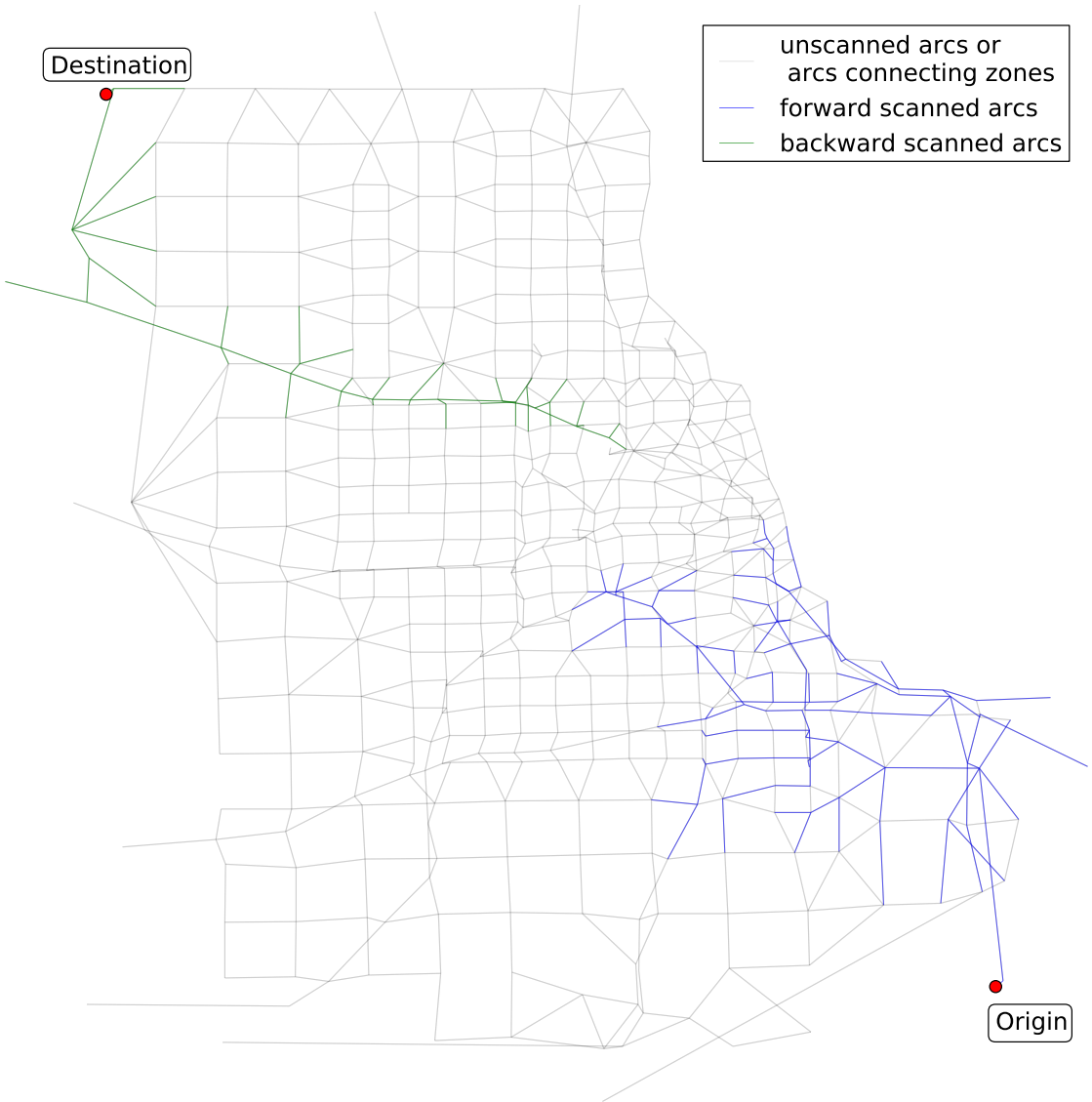
Bidirectional A* search



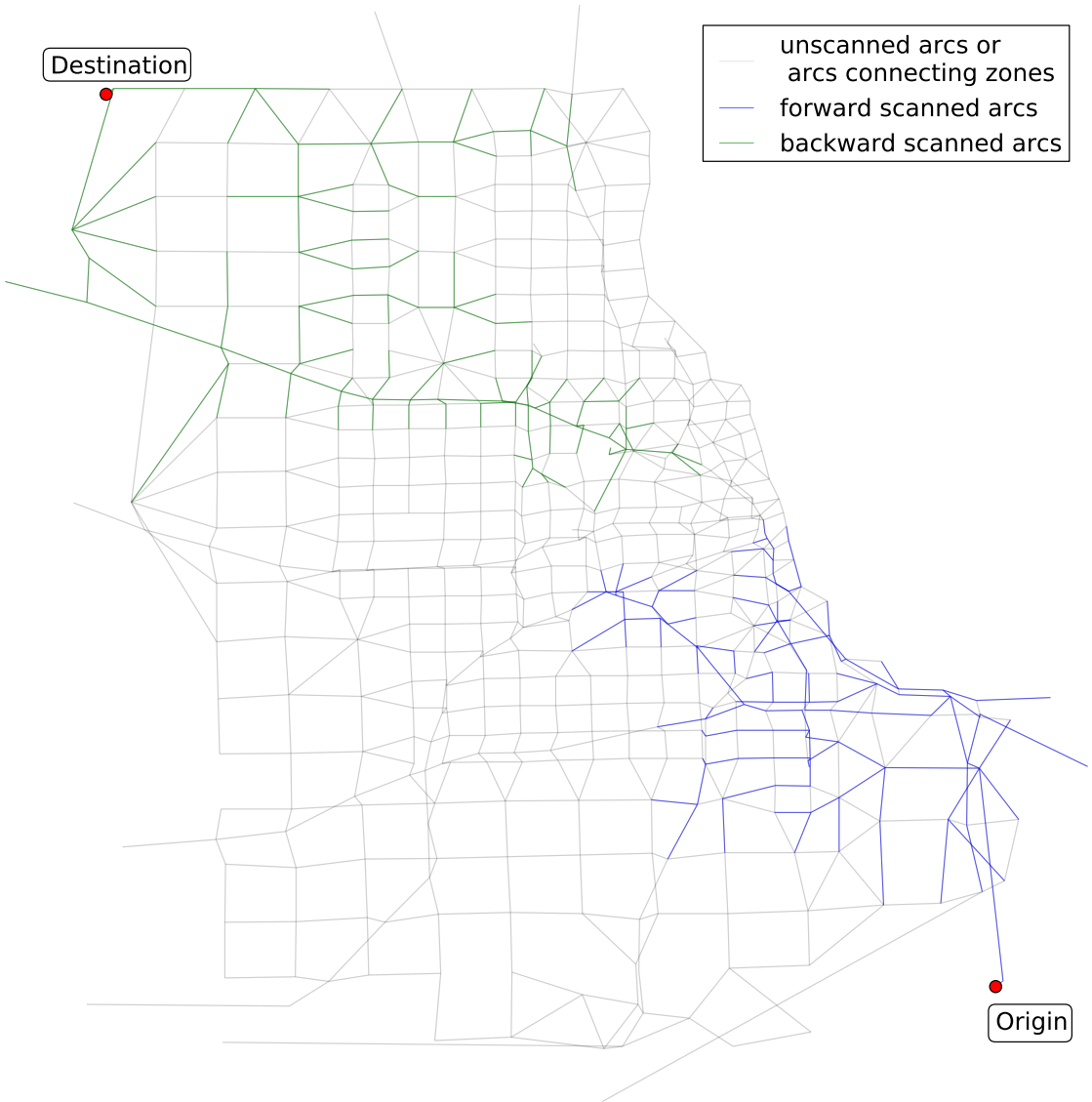
Bidirectional A* search



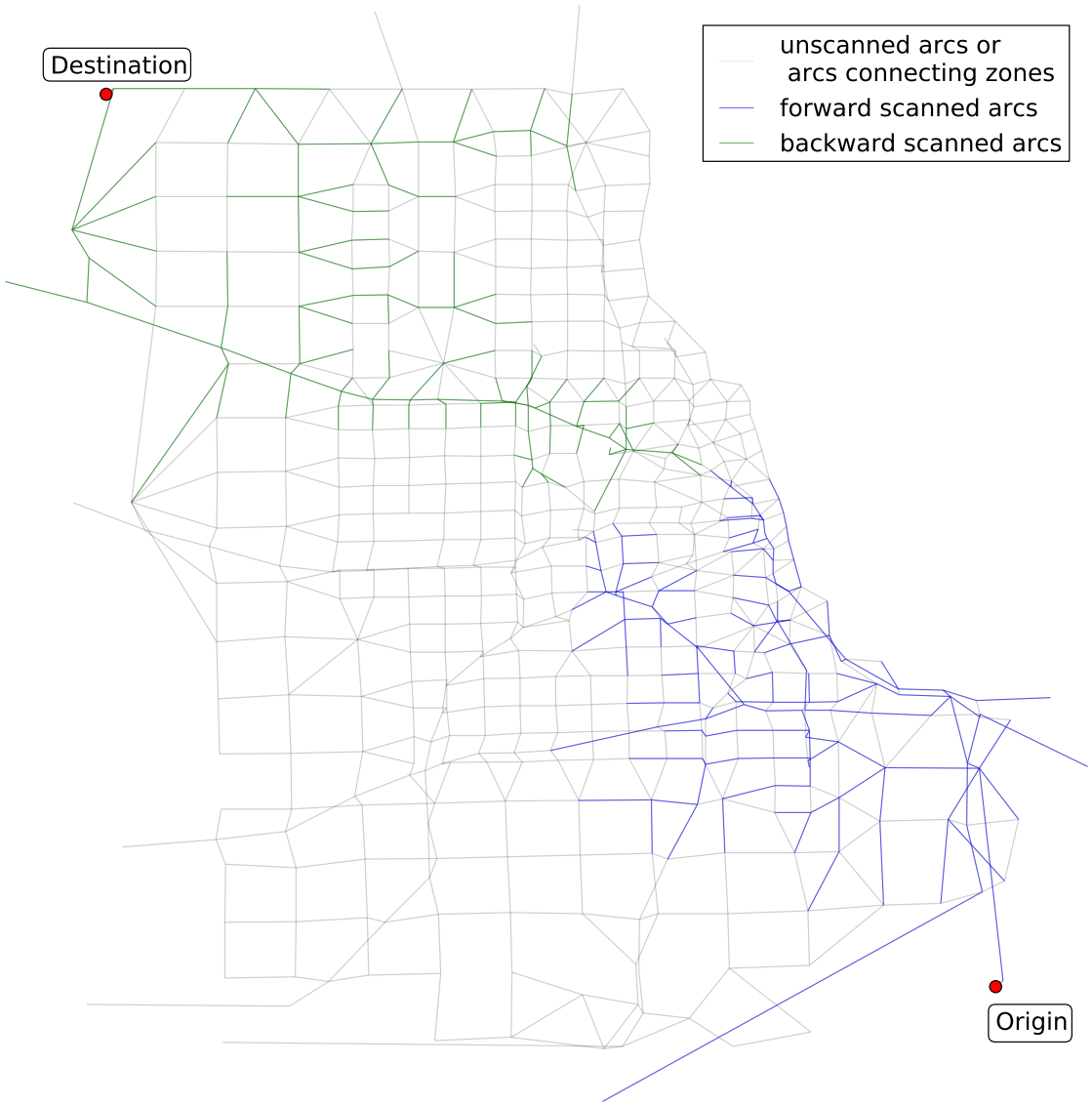
Bidirectional A* search



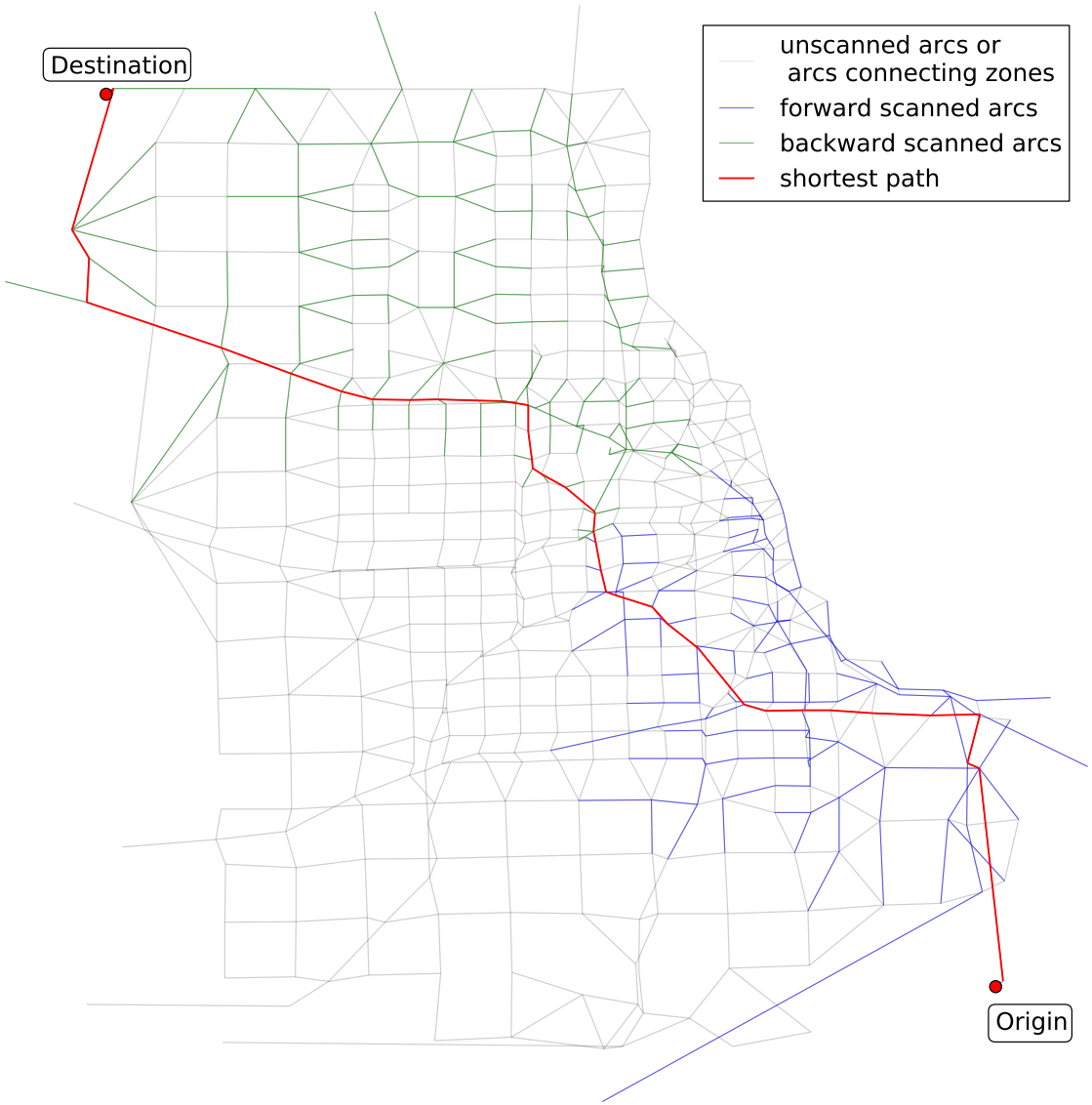
Bidirectional A* search

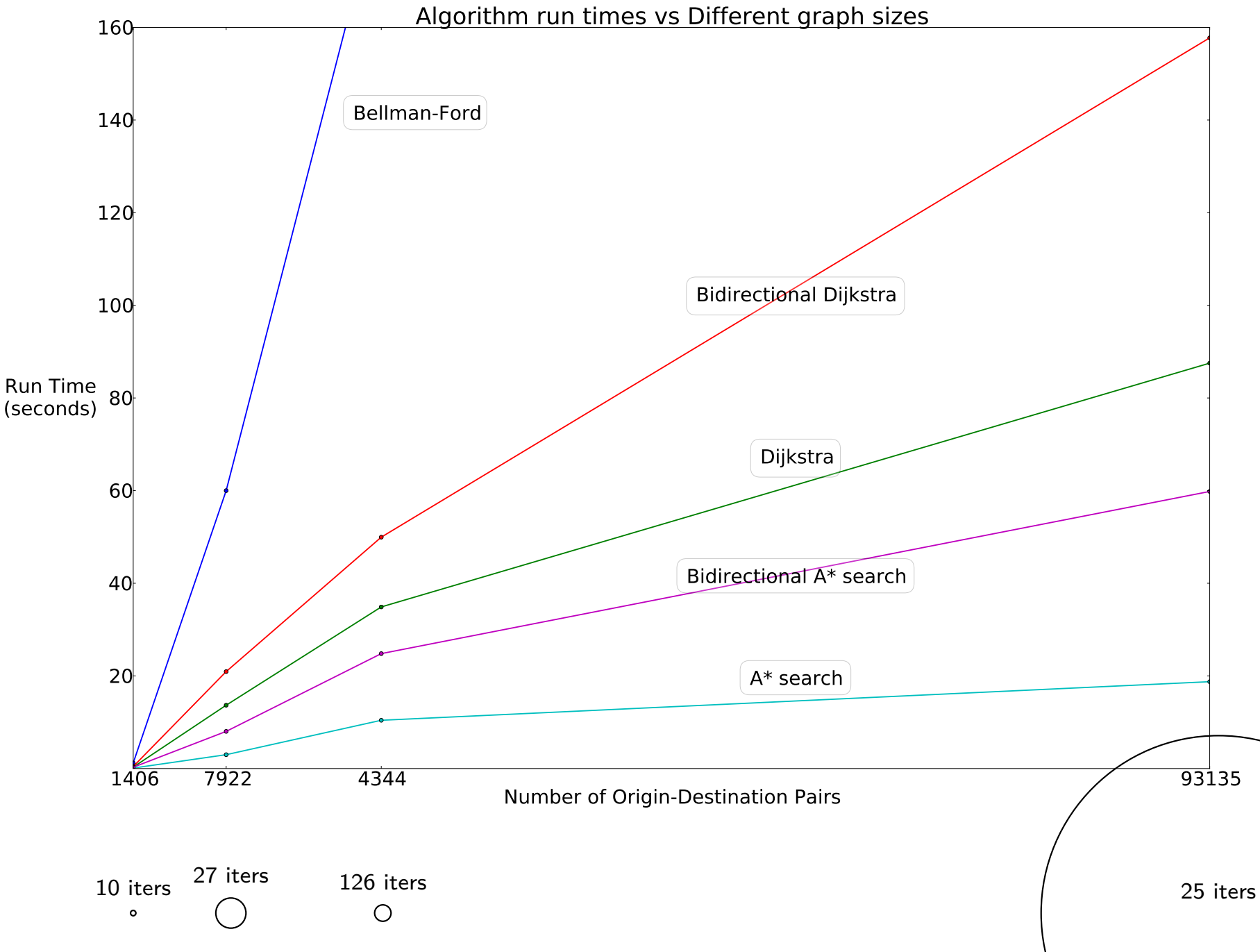


Bidirectional A* search



Bidirectional A* search




Algorithm run times vs Different graph sizes
Run Time (seconds)
160
140
120
100
80
60
40
20
Bellman-Ford
Bidirectional Dijkstra
Dijkstra
Bidirectional A* search
A* search
1406
7922
4344
93135
Number of Origin-Destination Pairs
10 iters
27 iters
126 iters
25 iters

Future Work

- preprocessing

Future Work

- preprocessing
  - A* search combined with landmark distances

Future Work

- preprocessing
  - A* search combined with landmark distances
- use information from previous iteration

Future Work

- preprocessing
  - A* search combined with landmark distances
- use information from previous iteration
  - Incremental heuristic search - Lifelong Planning A*

Future Work

- preprocessing
  - A* search combined with landmark distances
- use information from previous iteration
  - Incremental heuristic search - Lifelong Planning A*
- avoid shortest path calculations if possible

Future Work

- preprocessing
  - A* search combined with landmark distances
- use information from previous iteration
  - Incremental heuristic search - Lifelong Planning A*
- avoid shortest path calculations if possible
  - most of the shortest paths do not change after the first few iterations