Faster Shortest Path Computation for Traffic Assignment

THE UNIVERSITY OF AUCKLAND
FACULTY OF ENGINEERING

Boshen Chen Department of Engineering Science

Supervisors: Dr. Andrea Raith and Olga Perederieieva

Introduction

- Traffic congestion is currently a major issue for transportation planning
- A transportation forecasting model has been built to predict future traffic and reduce congestion
- The Traffic Assignment (TA) problem is part of the model which deals with selecting the shortest path for travellers in the network to minimise their travel times
- Goal: find a faster algorithm to solve the shortest path problem in the traffic assignment problem

Traffic assignment

- TA is a non-linear problem, where travel times increase dramatically when congestion occurs
- An iterative algorithm called Path Equilibration (PE) is used to solve TA
- PE requires millions of shortest paths to be found
- Solving the shortest path problem faster can speed up TA and benefit transportation modelling greatly

Shortest path algorithms

- A shortest path algorithm finds a path between origins and destinations with the least travel distance or time in a network
- The algorithm searches nodes in the network in some order until the destination is found
- A priority queue is needed to store the searched nodes in some order so the next location to search can be found easily
- Performance of PE is affected by different shortest path algorithms and priority queue implementations

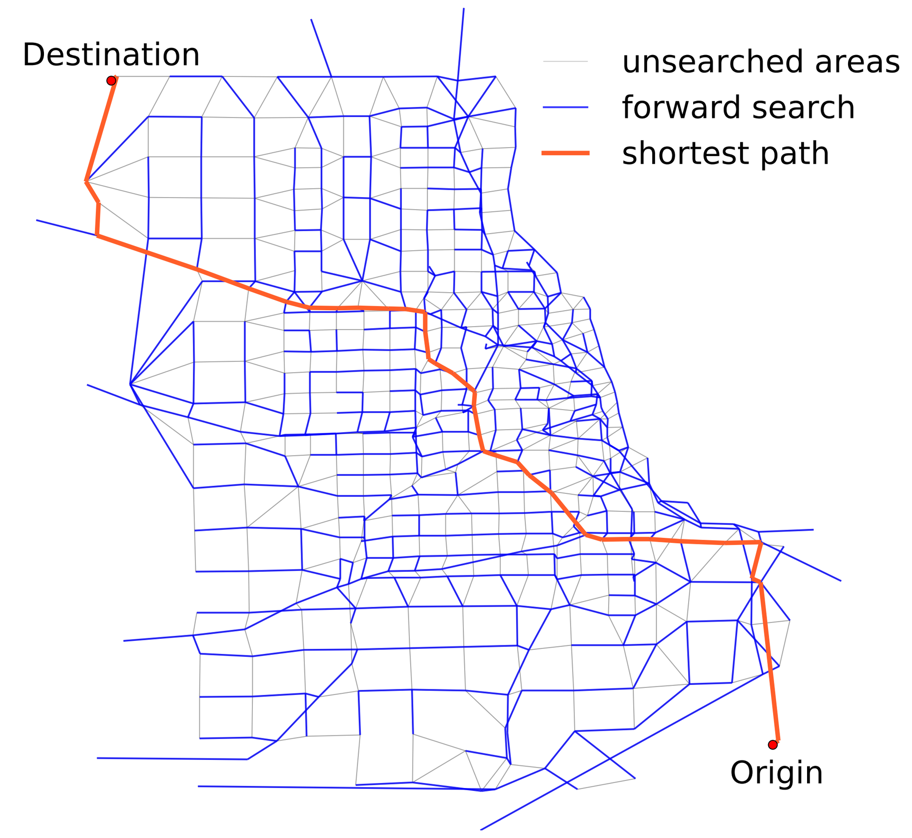
Avoiding shortest paths

- 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
- The first strategy is to avoid the next few iterations if the shortest paths of the previous two iterations are identical
- The second strategy is to randomly avoid the next shortest path calculation in the hope that the shortest path of previous and current iteration are identical

Search areas of shortest path algorithms

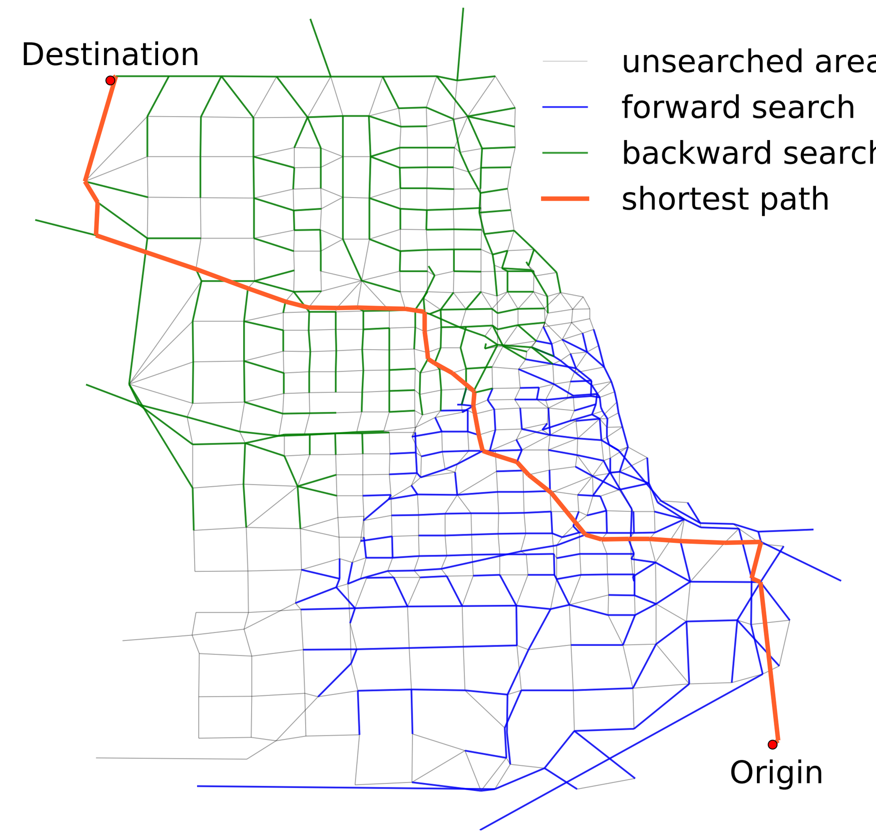
- The performance of shortest path algorithms is heavily dependent on the search areas
- Computational time can be sped up if a smaller area is searched
- The following figures demonstrate search areas of the implemented shortest path algorithms on part of the Chicago regional network, which has 546 nodes and 2,950 arcs

Dijkstra's algorithm



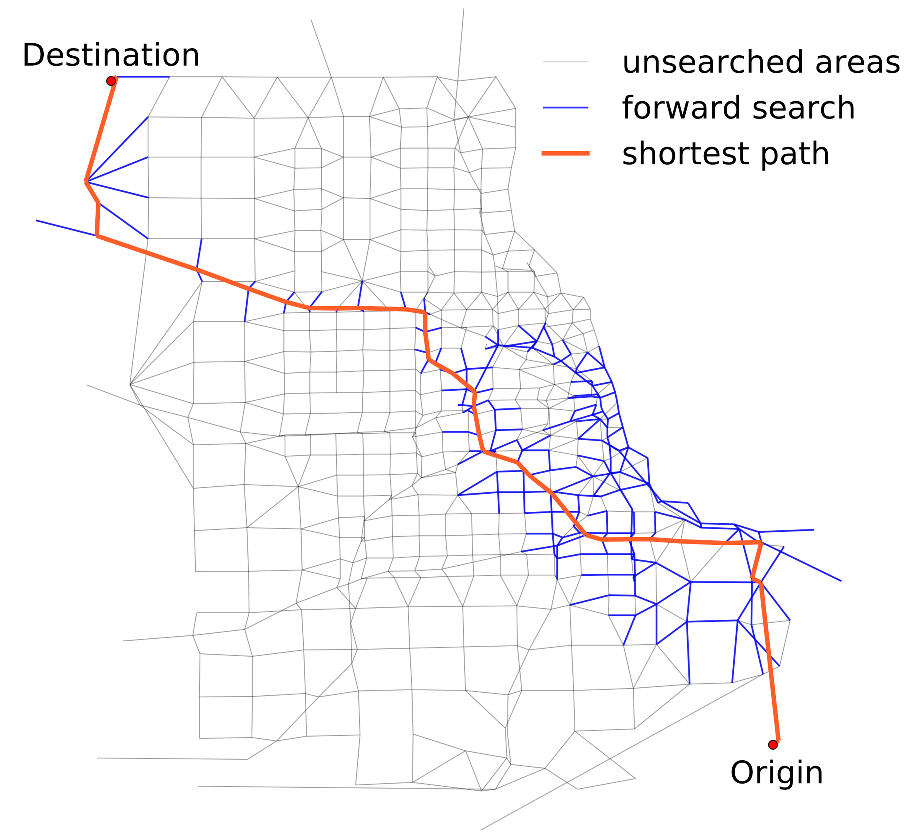
- search the entire network

Bidirectional Dijkstra's algorithm



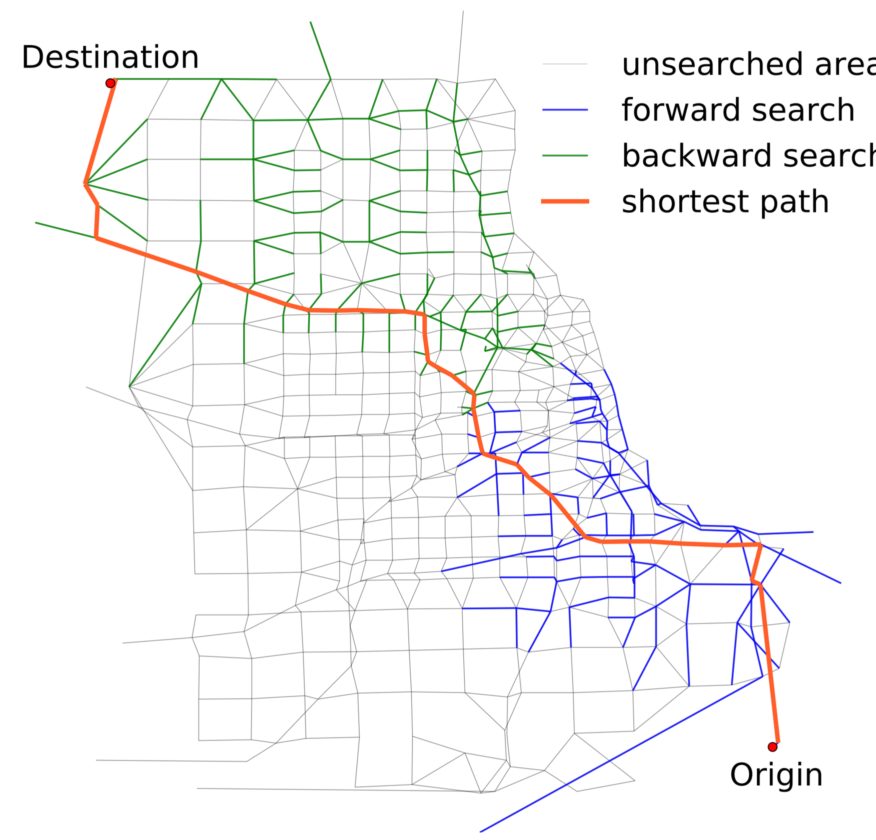
- alternatively search from both ends

A* Search



- search along the expected shortest path

Bidirectional A* Search



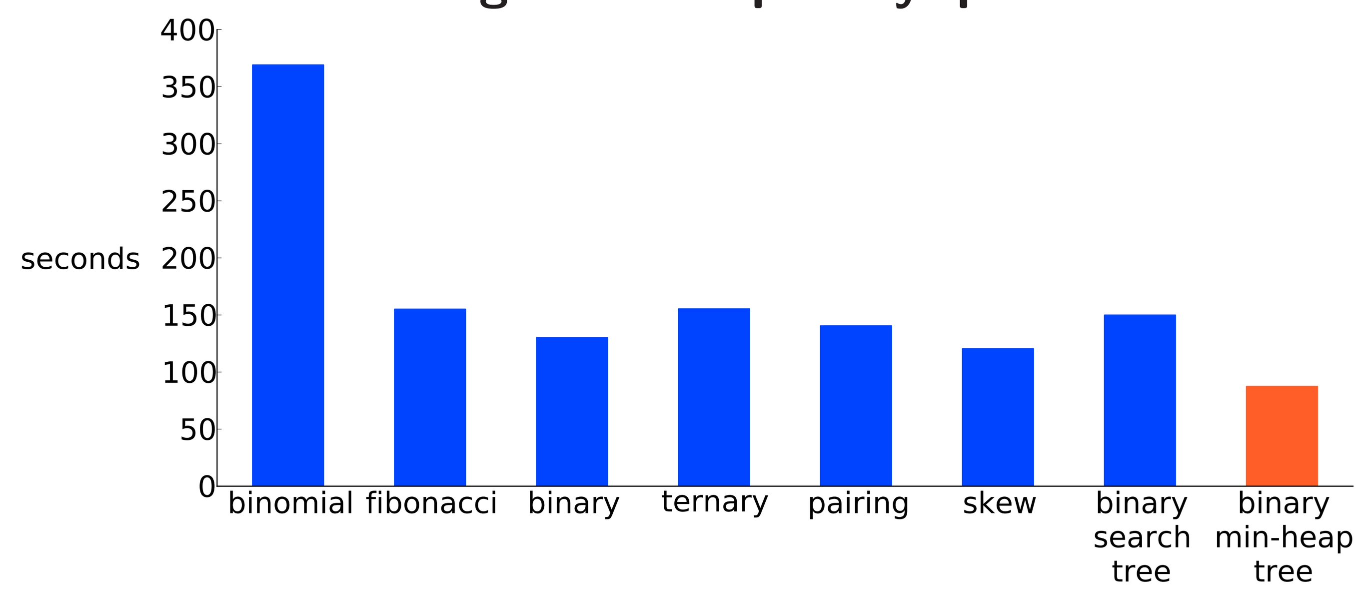
- alternatively search along the expected shortest path from both ends

- Dijkstra's algorithm searches the largest area
- Bidirectional Dijkstra's algorithm performs less searches
- A* search searches the smallest area
- Bidirectional A* search searches more than unidirectional A*

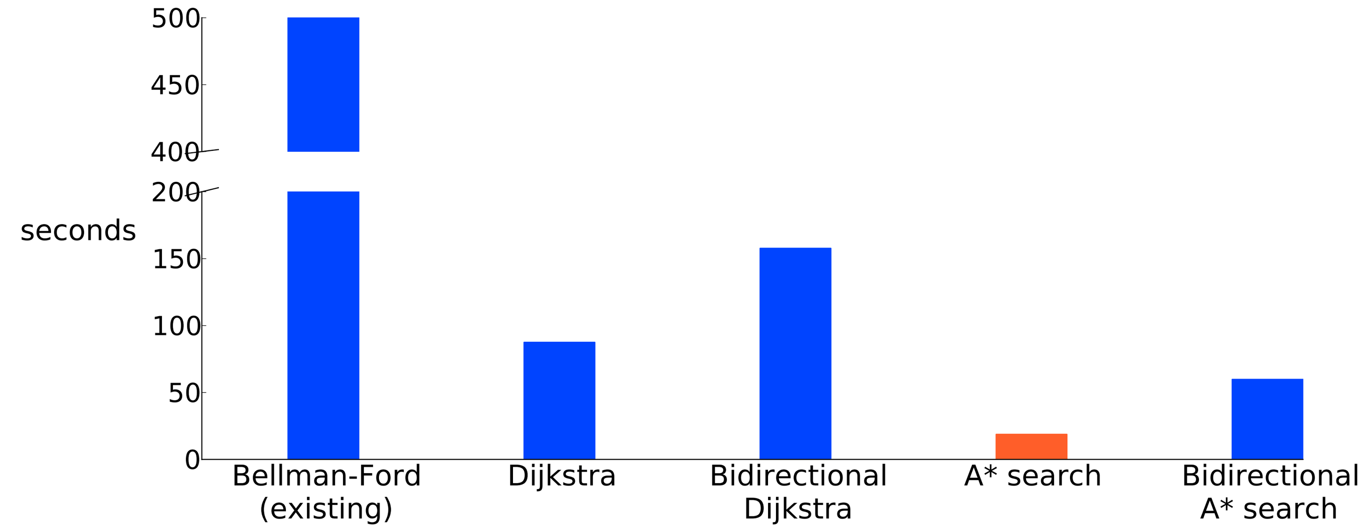
Results

- 8 different priority queues were tested, 4 shortest path algorithms were implemented and 2 strategies for avoiding shortest path calculation in PE were tested on the same part of the Chicago regional network

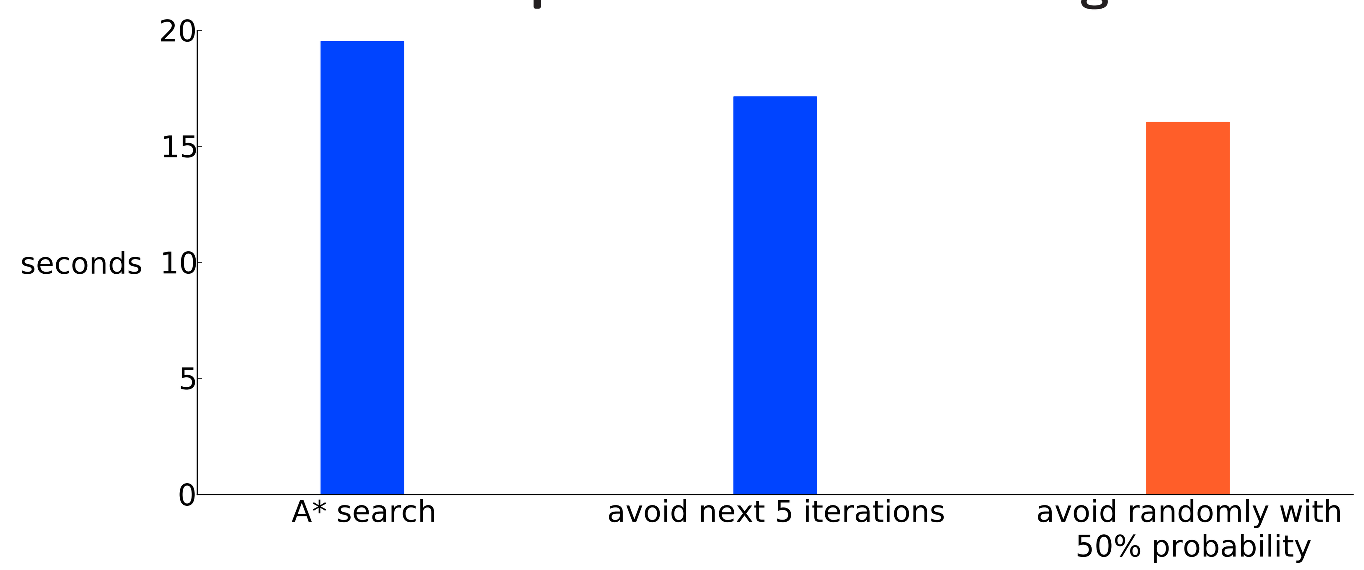
Run times of Dijkstra's algorithm using different priority queues



Run times of shortest path algorithms using binary min-heap tree



Run times of A* search using avoiding shortest path calculation strategies



Conclusions

- A* search algorithm using binary min-heap tree with random avoiding strategy has the best performance
- 30 times faster than the existing implemented shortest path algorithm