**Minimum Distance Path Search**

Bohao Chen

UTD-ID: 2021271333

NET-ID: bxc151130

# Abstract

The paper introduced 4 real world undirected graph, and 7 minimum path search algorithms on these undirected graph, compared performance among the algorithms, find the best solution in specific situation based on experiment result and analyze the result. This paper helps developers choosing most suitable algorithm to handle path search problem.

**Keywords:** Shortest path search algorithm; Network analysis; Geographic Information Systems;

# Introduction

The relevance of a graph search problem is closely related to the ability of building models or representations coming from the real world. [1]Among the most demanded features in graph search problem, we can mention those related to the analysis of routes, some examples are as follows:

* What is the shortest path between places x and y?
* What is the optimal path between places x and y considering a certain criterion?
* What is the lowest cost path between x and y via places x1, x2, …, xn?

Shortest path search has been widely studied. Many applications can be found in various branches of science. The road networks to respond to the above requests are usually large and could have thousands of streets, which is why one should pay particular attention to how such information is processed.

This paper handles the problem of “How to find the minimum path within shortest time”. Before evaluating the performance, roughly, there could be a relationship between result accuracy and time-complexity, that is, result accuracy is inversely proportional to time-complexity.

For solving the problem, I applied 7 minimum search algorithms, they are in two classes: blind search and heuristic search, and compare the performance of these 7 algorithms on 4 real world undirected graphs. The performance includes nodes that expands, total distance of the path, number of node visit, and cost (number of node that expand / number of node in result path). There are several programming languages that can implement search algorithms, such as Java, Python, C++. In my experiments, I choose Java as the programming language.

The following major sections are a related work, graph description, pre-processing techniques, proposed solution and methods, experimental result and analysis, and conclusion. Especially, in the proposed solution and methods, this paper introduced seven algorithms of searching on graph. Accordingly, I got the final solution by experiments.

One of the most used heuristic algorithms is the A\* algorithm, the main goal is to reduce the run time by reducing the search space analyzing only the vertices that have better possibilities to appear in the shortest path. The results obtained by this algorithm depend on the heuristic function used to determine the order in which vertices are visited. If the selected heuristic is optimal the computational complexity is reduced to O(n). That is why the A\* algorithm is widely used for shortest path search.

One approach studied for shortest path search on large graphs is related to the use of some properties of the road networks, mainly to reduce the search space of the shortest path.

In the following paragraphs we will be referring to some relevant researches:

Is there a relationship between result accuracy and time complexity?

Is A star algorithm the best search algorithm among these 7 algorithms?

# Related work

The paper, Algorithm for shortest path search in Geographic Information Systems by using reduced graphs, written by Rafael Rodríguez-Puentecorresponding author and Manuel S Lazo-Cortés, is concerned of several studies about shortest path search show the feasibility of using graphs for this purpose. The paper also introduces one of the most used heuristic algorithms is the A\* algorithm, the main goal is to reduce the run time by reducing the search space. [2]

The paper, Simplicial Dijkstra and A\* Algorithms for Optimal Feedback Planning, written by Dmitry S. Yershov and Steven M. LaValle, is concerned of shortest path algorithms between Dijkstra and A\* algorithm. Computing the shortest path to a given goal is a recurring problem in robotics. In addition to optimal robot navigation and manipulation, the paper focus on finding the shortest path between a given point and a polygonal goal set in an n-dimensional environment with polygonal obstacles. [3]

# Approach

I use Java programming to do the experiment and find

the result. The result that measured can be shown after the

compiler call “MainDriver.java”. The Object-Oriented

Design is shown below.

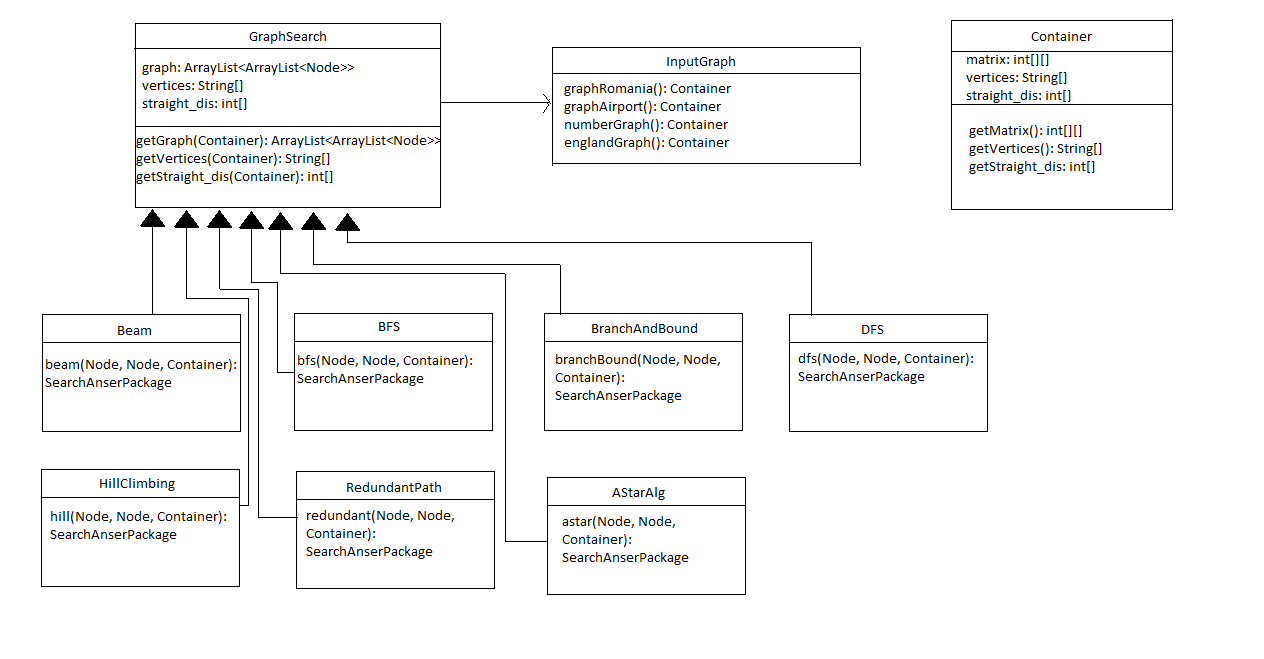
****

Figure 3.1

# Implement

# Graph Description

In this project, I choose 4 graphs for comparing searching performance of each algorithms. These graphs include big size graph (number of nodes is not less than 20), middle size graph (number of nodes is between 10 and 20) and small size graph (number of nodes is less than 10). So that these graphs can simulate the real world problems, and make the result more accurate.

**First Graph**: Romania (big size graph) [4]

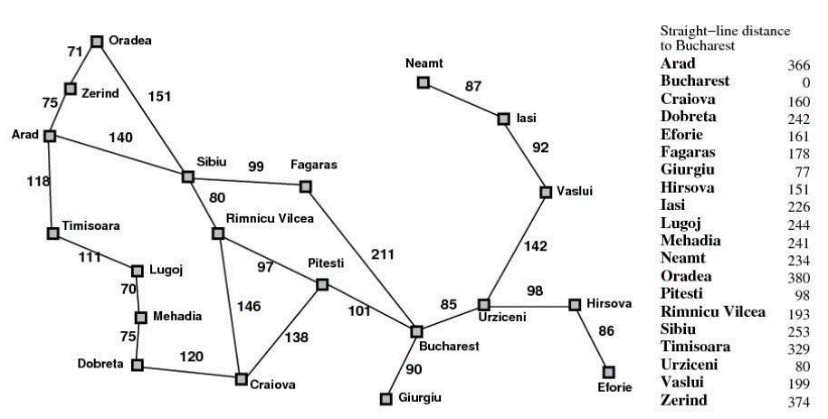


Figure 4.1.1

This graph contains 20 nodes, which is a complex graph relatively. I choose Zerind as the start point, Bucharest as the end point. For reading convenience, I use number id 19 as start point, and number id 1 as end point.

The shortest path from start point 19 to end point 1 is:

19 – 0 – 15 – 14 – 13 – 1

Zerind – Arad – Sibiu – Rimnicu Vilcea – Pitesti - Bucharest

The shortest length is 493.

**Second Graph**: Airport (small size graph) [5]

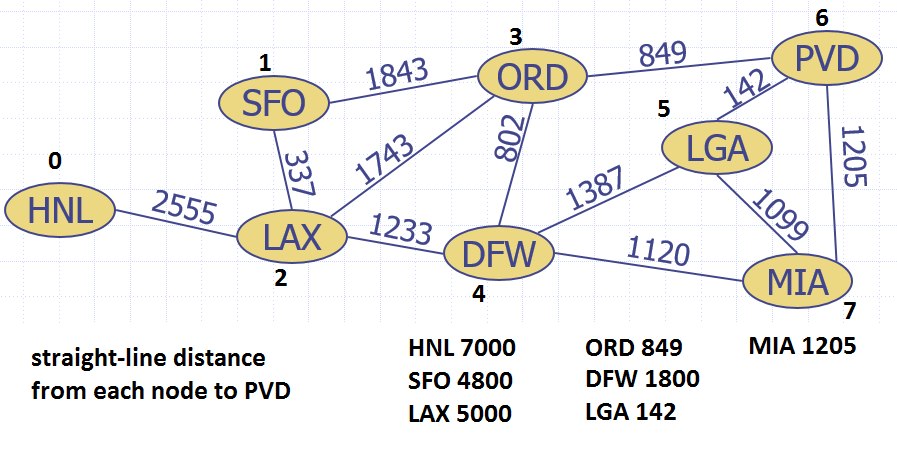


Figure 4.1.2

This graph contains 8 nodes, which is a small and simple graph relatively. I choose HNL (node 0) as start point, MIA (node 7) as end point.

The shortest path from HNL to MIA is:

0 – 2 – 3 – 6

NHL – LAX – ORD - PVD

The shortest path length from HNL to MIA is:

5147

**Third Graph**: number graph (small graph) [6]

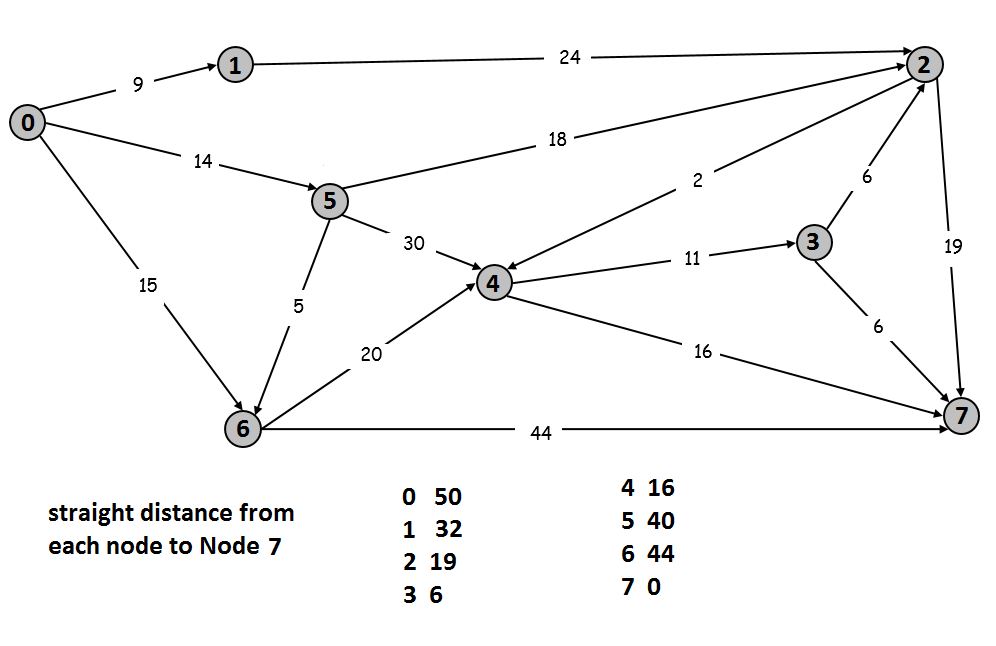


Figure 4.1.3

This graph contains 8 nodes, which is a small and simple graph relatively. I choose node 0 as start point and node 7 as the end point.

The shortest path from node 0 to node 7 is:

0 – 5 – 2 – 3 – 7

The shortest path length from node 0 to node 7 is:

45

**Fourth Graph**: England (middle size) [7]

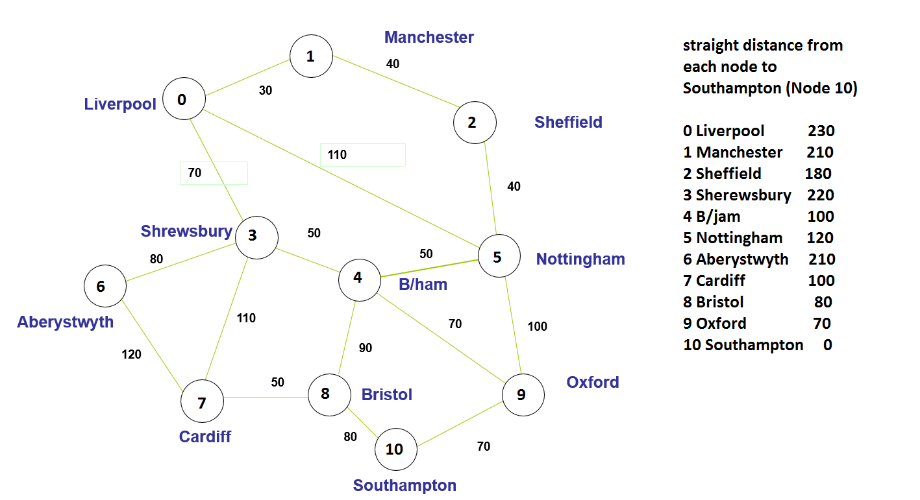


Figure 4.1.4

This graph contains 11 nodes, which is a middle size graph relatively. I choose Liverpool (node 0) as start point and Southampton (node 10) as end point.

The shortest path from Liverpool (node 0) to Southampton (node 10) is:

0 – 3 – 4 – 9 – 10

Liverpool – Sherewsbury – B/jam – Oxford - Southampton

The shortest path length from Liverpool (node 0) to Southampton (node 10) is:

260

# Algorithm Description

The performance includes nodes that expands, total distance of the path, number of node visit, and cost (number of node that expand / number of node in result path).

The cost means: in order to find a node in the result path, the number of nodes that need to be expand. If the value of cost is high, it means this search algorithm has lower performance.

# BFS[8]

Breadth-first search is a blind search that search node from top level to bottom level.

The search result of BFS is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 14 | 5 | 525 | 2.8 | No |
| Graph 2 | 9 | 4 | 5147 | 2.25 | Yes |
| Graph 3 | 11 | 3 | 59 | 3.67 | No |
| Graph 4 | 24 | 4 | 280 | 6 | No |

Form 4.2.1

From the form, we find that the cost of BFS is high. Algorithm needs to pay more than 2 times effort in order to find the path.

Also, BFS only find the shortest path in small graph, and failed to find shortest path in other 3 graphs. The accuracy is low.

# Beam Search[9]

Beam Search is a heuristic search. The search result of Beam Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 9 | 5 | 525 | 1.8 | No |
| Graph 2 | 6 | 4 | 5147 | 1.5 | Yes |
| Graph 3 | 7 | 3 | 59 | 2.33 | No |
| Graph 4 | 11 | 4 | 280 | 2.75 | No |

Form 4.2.2

From the form, we find the cost of Beam Search is lower than BFS. Beam Search needs to pay around 2, sometimes lower than 2 times effort in order to find the path.

However, Beam Search only finds shortest path in second graph, which is a small graph, and failed to find shortest path in other 3 graphs. The accuracy is low.

# DFS[10]

Depth-first search is a blind search. The search result of DFS is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 12 | 5 | 525 | 2.4 | No |
| Graph 2 | 5 | 5 | 6113 | 1 | No |
| Graph 3 | 3 | 3 | 59 | 1 | No |
| Graph 4 | 4 | 4 | 280 | 1 | No |

Form 4.2.3

From the form, we can find the cost of DFS is low. In 3 graphs the cost is 1. It means using DFS on these 4 graphs doesn’t need to pay too much effort in order to find the path.

However, DFS can’t find shortest path for any graph, the accuracy is 0.

# Hill Climbing Search[11]

Hill Climbing Search is a heuristic search algorithm. The search result of Hill Climbing Search is shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 5 | 5 | 525 | 1 | No |
| Graph 2 | 4 | 4 | 5147 | 1 | Yes |
| Graph 3 | 3 | 3 | 52 | 1 | No |
| Graph 4 | 4 | 4 | 280 | 1 | No |

Form 4.2.4

From the form, we can find the cost of Hill Climbing Search are all 1, which means the efficiency of finding path is really high. Every step to expand a node is finding the path.

However, Hill Climbing Search only finds the shortest path once. The accuracy is low.

# Branch and Bound Search[12]

Branch and Bound Search is a heuristic search algorithm. The result of Branch and Bound Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 199 | 6 | 493 | 33.17 | Yes |
| Graph 2 | 76 | 4 | 5147 | 19 | Yes |
| Graph 3 | 44 | 5 | 44 | 8.8 | Yes |
| Graph 4 | 203 | 5 | 260 | 40.6 | Yes |

Form 4.2.5

From the form, we find that the cost of Branch and Bound Search is so high. So many duplicated nodes have been visited again and again. In order to find one step of path, Branch and Bound Search need to pay more than 10 times effort.

However, Branch and Bound Search must return the shortest path.

* + 1. **Redundant Path Search[13]**

Redundant Path Search is a blind search algorithm. The result of Redundant Path Search is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 13 | 6 | 493 | 2.17 | Yes |
| Graph 2 | 7 | 4 | 5147 | 1.75 | Yes |
| Graph 3 | 7 | 5 | 45 | 1.4 | No |
| Graph 4 | 10 | 4 | 280 | 2.5 | No |

Form 4.2.6

From the form, Redundant Path Search has a not very high cost and 50% percentages of accuracy. Performance is in the middle level among these algorithms.

* + 1. **ASTAR Algorithm[14]**

ASTAR algorithm is a heuristic algorithm. It’s a good algorithm to find minimum path. The result form is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nodes that expand | Nodes in result path | Path length | Cost | Is the shortest |
| Graph 1 | 9 | 6 | 493 | 1.5 | Yes |
| Graph 2 | 2 | 2 | 5147 | 1 | Yes |
| Graph 3 | 5 | 5 | 45 | 1 | No |
| Graph 4 | 10 | 4 | 280 | 2.5 | No |

Form 4.2.7

From the form, ASTAR algorithm has a little bit lower cost than Redundant Path Search, the accuracy is 50% which is not very high.

# Experiments Results

Combine the result in part 3. I made the form to compare all algorithm together.

Focus on Cost:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Graph 1 | Graph 2 | Graph 3 | Graph 4 |
| BFS | 2.8 | 2.25 | 3.67 | 6 |
| Beam Search | 1.8 | 1.5 | 2.33 | 2.75 |
| DFS | 2.4 | 1 | 1 | 1 |
| Hill Climbing Search | 1 | 1 | 1 | 1 |
| Branch and Bound Search | 33.17 | 19 | 8.8 | 40.6 |
| Redundant Path Search | 2.17 | 1.75 | 1.4 | 2.5 |
| ASTAR Algorithm | 1.5 | 1 | 1 | 2.5 |

Form 5.1

Figure 5.1

From above, we can find that Branch and Bound Search has the highest cost. And the cost is much higher than any other algorithms. Hill Climbing Search has the lowest cost but is not so much lower than other algorithms.

Now let’s focus on accuracy:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | First Graph | Second Graph | Third Graph | Fourth Graph | Number of Yes | Number of No |
| BFS | No | Yes | No | No | 1 | 3 |
| Beam Search | No | Yes | No | No | 1 | 3 |
| DFS | No | No | No | No | 0 | 4 |
| Hill Climbing Search | No | Yes | No | No | 1 | 3 |
| Branch and Bound Search | Yes | Yes | Yes | Yes | 4 | 0 |
| Redundant Path Search | Yes | Yes | No | No | 2 | 2 |
| ASTAR Algorithm | Yes | Yes | No | No | 2 | 2 |

Figure 5.2

From above, we find Branch and Bound Search has the top accuracy, which is 100%. DFS has the lowest accuracy which is 0. BFS and Beam Search have accuracies that lower than 50%. Redundant Path Search and ASTAR Algorithm have 50% accuracies.

# Conclusion

If we want to find the result that must be the shortest path, we should use Branch and Bound Search algorithm. This algorithm can return the shortest path in any graph. However, it will spend more running time and more space. The situation we choose Branch and Bound Search algorithm is only when user want to find the shortest path no matter how much time cost.

If user want to save time, Redundant Path Search and ASTAR Algorithm would be good to use. Because these algorithms don’t spend too much time and have a 50% accuracy. Hill Climbing Search is not a good choice because the running time is not save too much when we choose it, however, the accuracy is much lower than Redundant Path algorithm and ASTAR Algorithm.

**References**

[1] Rafael Rodríguez-Puentecorresponding and Manuel S Lazo-Cortés, Algorithm for shortest path search in Geographic Information Systems by using reduced graphs, Springerplus. 2013; 2: 291. doi: 10.1186/2193-1801-2-291

[2] Harika Reddy, PATH FINDING - Dijkstra’s and A\* Algorithm’s, December 13, 2013

[3] Dmitry S. Yershov and Steven M. LaValle, Simplicial Dijkstra and A\* Algorithms for Optimal Feedback Planning

[4] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search

[5] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search

[6] Neeraj Gupta, Data Structure and Algorithm CS 5343, MST

[7] Neeraj Gupta, Data Structure and Algorithm CS 5343, Shortest Path

[8] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search, Breadth-first search, https://en.wikipedia.org/wiki/Breadth-first\_search

[9] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search, Beam search, <https://en.wikipedia.org/wiki/Beam_search>

[10] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search, Depth-first search, <https://en.wikipedia.org/wiki/Depth-first_search>

[11] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search, Hill climbing, <https://en.wikipedia.org/wiki/Hill_climbing>

[12] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search, <https://en.wikipedia.org/wiki/Branch_and_bound>

[13] Dan Moldovan, Artificial Intelligence CS 6364, Problem Solving and Search