Finding the shortest path under polygonal obstacle constraints

Xu Yan[1,a],Dawei Liu[2,b]

[1]Xi'an Jiaotong-Liverpool University, Suzhou, China

[2] Xi'an Jiaotong-Liverpool University, Suzhou, China

[a]xu.yan11@student.xjtlu.edu.cn, [b]dawei.liu@xjtlu.edu.cn

Keywords: Location-based Services, Shortest Path, Polygonal Constraints.

Abstract. This paper discusses the problem of road network navigation under polygonal obstacle constraints from an implementation perspective. It implements the problem provided by the 4th ACM SIGSPATIAL GIS Cup 2015 challenge. In the context, it briefly examines some fundamental graph search algorithms and then primarily demonstrates the notes on the implementation of A* algorithm on the dataset of San Francisco from *OpenStreetMap*.

1. Introduction

Route planner, an important application in Geographic Information System, mobile computing and artificial intelligence has been studied for a long time. Merely returning the shortest path from route planner is often not satisfactory; thus, some route planner provides alternative paths which might be longer than the shortest one but have other desirable properties, e.g., less traffic or more beautiful sights along the path. Another special usage of route planners is along with the rising of emergent situations, e.g., natural disasters or terrorist attacks where a relatively safe evacuation route is preferred to be available [1].

In graph theory, shortest path problem is about finding a path between two vertices so that the total weight of the path is minimum. It has motivated shortest path algorithms answering distance queries on street network of large scale, with instant processing speed and very low space overhead [2].

Polygonal obstacle constraints is an important issue in finding the shortest path in a given road map. For instance, part of the street road is closed during a certain period of time periodically, or due to emergent accidents dynamically. Under such circumstances, it is vital to find a shortest path with obedience to these constraints.

1.1 Preliminaries

Let G be a road network with an edge set E and a vertex set V containing N vertices. Let each edge $e \in E$ be related to a weight $w(e)$, which is assumed to be the length of e.

The shortest path between any given two vertices s, g, is a sequence of (e_1, e_2, e_3,...,e_n) that connects s to g, such that $\sum_{i=1}^{n} w(e_i)$ is minimized. The shortest path based on time is a sequence of edges (e_1, e_2, e_3, ..., e_n) that connects s to g, such that $\sum_{i=1}^{n} \frac{w(e_i)}{spd_i}$ is minimized, where spd_i is the top speed associated with edge e_i.

1.2 Problem Statement

The problem was about road network routing with polygonal obstacle constraints. Concretely, given a source and a target along with a polygonal obstacle, compute the shortest and fastest path which avoids the given polygon [3].

2. Related Work

The A* algorithm is the *de facto* standard used for pathfinding search and it examines the smallest number of nodes necessary to guarantee a shortest path. An evaluation function $f(n)=g(n)+h(n)$, is used to evaluate the nodes which should be examined next [4]. $g(n)$ is the exact cost of an optimal

path from any node *n* to the source *s* and *h(n)* is the estimated cost of an optimal path from *n* to the target *t*. *h(n)* might be the *Euclidean* distance, which is the shortest possible distance between *n* and *t* and it is a lower bound on the actual cost of *n* and *t*. Since road distance is usually much greater than the *Euclidean* distance; thus the heuristic *h* is admissible and will usually, but not always bring about the optimal solution.

IDA* is a space-efficient version of A* and a depth first search with heuristics, which however pays a huge price for the lack of storage that ends up revisiting the same nodes many times. Fringe Search algorithm spans the space/time tradeoff between A* and IDA* [5].

Since path planning is a well-established research area, there are many research that have managed to find the shortest path on regions represented by uniform grid. The uniform grid is a simple indexing data structure, which superimposes a grid over a road map and it has been applied in numerous applications such as map overlapping and segment intersection.

3. Experiments

3.1 Experimental Settings

Graph search algorithms were tested on a computer with a 2.7 GHz Intel Core i7 CPU and 16GB 1600Hz DDR3 RAM. The experiments were performed on the dataset provided by the challenge [3].

3.2 Datasets and Queries

The dataset is available in the format of *text* and *shapefile*, including the sample dataset road map and sample polygonal constraints. The road data consists of 96850 rows of tuples with 6 attributes. The node data contains 42408 rows of tuples with 4 attributes.

3.3 Graph Search and Routing Design

The core of A* is to make use of two sets *open* and *closed*. In each iteration, a non-explored vertex that is closest to the source vertex *s* is added to *open*. A* then uses *f(n)=g(n)+h(n)*, returning the sum of an estimated lower bound for the cost from vertex *n* to target vertex *t* and the exact cost from *n* to *s*. After *n* is added to *open*, the priority of *n* is incremented with respect to its *f* value, which guides the graph search such that nodes with highest priority are evaluated in priority.

The heuristic time function is defined to be the ratio of the *Euclidean* distance between *n* and *t* and the highest speed of the roads on the maps. Thus it returns a lower bound for the cost of path.

```
Data: Weight graph G = (V, E, len), source vertex s and goal vertex g
Result: A shortest path from s to g

while OPEN not empty do
   u ← node with lowest F value from OPEN; remove it from OPEN;
   put u in CLOSED;
   if g ∈ CLOSED then
      break;
   end
   for (u,v) ∈ E do
      if v ∈ CLOSED then
         continue;
      end
      if v ∉ OPEN then
         put v in OPEN;
         make u the parent of v;
         G(v) ← G(u) + dist(u,v);
      end
      if v ∈ OPEN then
         if G(u) + dist(u,v) < G(v) then
            G(v) ← G(u) + dist(u,v);
            make u the parent of v;
         end
      end
   end
end
if g ∈ CLOSED then
   t ← g;
   put t in path;
   chaining back from t to g through predecessors' pointers;
end
else
  no path found;
end
```

Fig. 1. A* Search Algorithm Distance Query

3.4 Roads and Nodes Representation

A detailed description on road data is displayed in Table 1. Road data and node data were stored first by indexed arrays and then by hash tables, to make a comparison between their query efficiency.

Table 1. Summary of roads data.

	road ID	start ID	end ID	length [m]	Speed [m/s]
Min	-967816469	19518076	19518076	2.08	2.778
1st Qu	-23747742	48511855	48511862	43.32	11.11
Median	+23593142	49835536	248041583	79.90	11.11
3st Qu	+23747942	281080216	281081455	127.67	11.11
Max	+967816469	984798521	984798521	3656.85	31.944

3.5 Ruling Out Tentative Nodes Inside the Polygonal Obstacle Constraint

The approach to ruling out nodes inside the polygonal obstacle constraints is by pushing these nodes in the *closed* from the start. Nodes that cannot be examined are labelled as examined so that there is no chance to examine them again. The *Ray Casting*, or *Even-Odd Rul*e is a way of solving *point-in-polygon* by counting the number of times a ray, originating from a point (either outside or inside a polygon), going in any fixed direction, crosses the edges of a polygon [6]. An odd number means it is inside the polygon while an even number indicates it is an outlier.

Therefore, instead of testing every potential edge for intersection, it improves the performance of A* mainly in situations where the shortest paths do not traverse the polygonal obstacle.

3.6 Summary of Experimental Results

Table 2 and Table 3 present the average running time, and several other statistics, of 4 different cases.

Table 2. Summary of Distance Query. *A* and *B* represent using indexed array and hash table. A_1 and B_1 represent road map without polygonal constraints while A_2 and B_2 represent road map under constrainst. Colume *#OPEN* and *#CLOSED* represent, respectively, the number of nodes inside the two sets. Colume *#Vert. visit* and *#Vert. path* represent, respectively, the number of nodes having been explored and the number of nodes along the path.

Test	#OPEN	#CLOSED	#Vert.visit	#Vert. path	Dist [m]	Time [ms]
A_1	94	254	361	134	12251.08	50
A_2	210	1679	2082	251	17392.3	282
B_1	74	248	360	134	12251.08	5
B_2	188	1526	2014	251	17392.3	28

Table 3. Summary of Time Query. Column *Time* and $Time_2$ represents, respectively, the time cost traversing the optimal path in real-world and the time cost of average running time.

Test	#OPEN	#CLOSED	#Vert.visit	#Vert. path	Dist [m]	Time [s]	$Time_2$ [ms]
A1	195	3090	4058	162	12648.61	772.5	638
A2	279	9697	12400	251	18778.86	1203.86	5940
B1	170	3068	4050	162	12648.61	772.5	69
B2	244	9606	12371	251	18778.86	1203.86	622

3.7 GUI

The polygonal obstacle is located on the north-east of the start flag, circled by several red polylines, which can be seen on Fig. 2 (*right*).

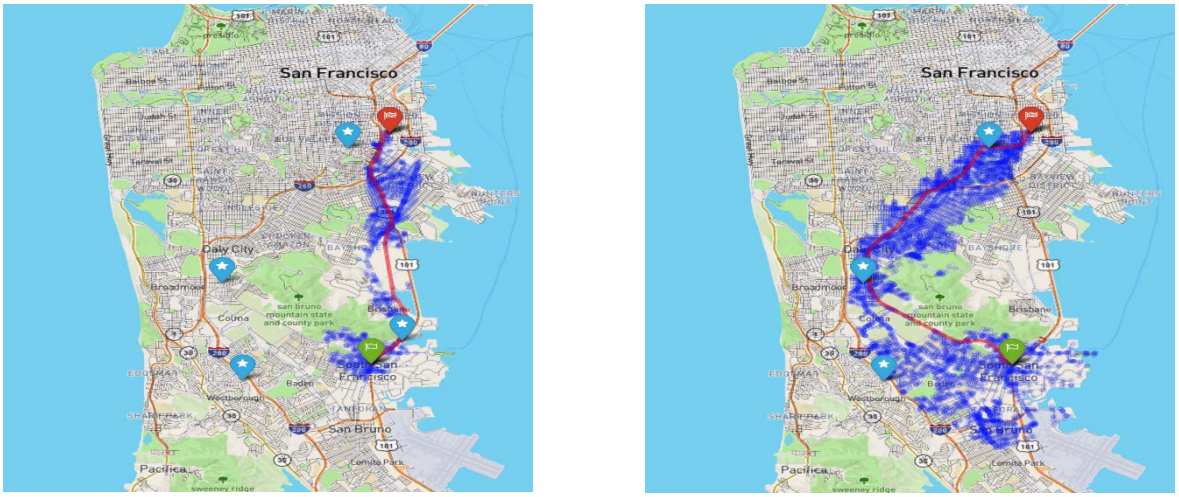

Fig. 2.Optimal paths for distance query (*red curves*) from 48514944 (*green flag*) to 48530319 (*red flag*) with (*right*) or without (*left*) polygonal obstacle constraint; The path on the left consists of 134 nodes with the total distance of 12251.08 *m*; the path on the right contains 251 nodes with the total distance of 17392.3 *m*

4. Conclusions

This paper presents an implementation on ACM SIGSPATIAL GIS Cup 2015 challenge. As for future work, one can possibly do better than A* Search. As this paper aforementioned, there is much to be explored and researched by a well-crafted implementation. I intend to explore some light-weight preprocessing strategies used for shortest and fastest path problem on large-scale road networks. Additionally, the performance of A* Search algorithm under different kinds of heuristics are to be studied.

References

[1] Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, and Uif Leser. Alternative routing: k-shortest paths with limited overlap. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Bellevue, WA, USA, November 2015. ACM.

[2] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *10 Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, Austin, Texas, 2010. Society for Industrial and Applied Mathematics.

[3] ACM SIGSPATIAL. *https://research.csc.ncsu.edu/stac/GISCUP2015/index.php.*

[4] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of System Science and Cybernetics*, 4(2), July 1968.

[5] Dijkstra E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269-271, December 1959.

[6] David Eppstein. Computational geometry. *http://www.ics.uci.edu/~eppstein/161/960307.html*.