Improvement And Experimental Evaluation Bellman-Ford Algorithm

Wei Zhang , Hao Chen , Chong Jiang , Lin Zhu ,
Beijing Institute of Petrochemical Technology
College of information Engineering
Beijing, China，102617
Email:zhangwei@bipt.edu.cn

Abstract

The shortest path problem is the problem of finding a path between two vertices on a graph such that sum of the weights of its constituent edges is minimized.Simulations to compare the efficiencies for Dijkstra's algorithm, SPFA algorithm and improved Bellman-Ford were taken. The result shows that Dijkstra's algorithm, SPFA algorithm have almost same efficiency on the random graphs, the improved algorithm, although the improved algorithm is not desirable, on grid maps the proposed algorithm is very efficient. The proposed algorithm has reduced two-third times processing time than SPFA algorithm.

Keywords: SPFA; Dijkstra; Bellman-Ford Algorithm

1. Introduction

The Shortest Path Faster Algorithm (SPFA) is an improvement of the Bellman–Ford algorithm which computes single-source shortest paths in a weighted directed graph. However, the worst-case complexity of SPFA is the same as that of Bellman–Ford, so for graphs with nonnegative edge weights Dijkstra's algorithm is preferred. The SPFA algorithm was published in 1994 by Fanding Duan[1].

Bellman-Ford algorithm, together with the Dijkstra's algorithm is also solving the shortest path problem. But Dijkstra's algorithm solves only the graphs with non-negative edge weights[2]. Edge weights not only can be express length, but also time, money, loss and any else along path linear accumulative quantity form. In finding a path, if there is a cycle, the cycle is disregarded. The presence of such cycles means that there is no the shortest path, since the total weight becomes lower whenever the cycle is traversed. Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single-source shortest path problemfor a graph with non-negative edge path costs, producing a shortest path tree.Shortest Path Faster Algorithm, Given a weighted directed graph G = (V, E) and a source vertex s, the SPFA algorithm finds the shortest path from s to each vertex v in the graph. The length of the shortest path from s to v is stored in d(v) for each vertex v.

The basic idea of SPFA is the same as Bellman–Ford algorithm in that each vertex is used as a candidate to relax its adjacent vertices[3]. The improvement over the latter is that instead of trying all vertices blindly, SPFA maintains a queue of candidate vertices and adds a vertex to the queue only if that vertex is relaxed. This process repeats until no more vertex can be relaxed[4].

Below is the pseudo-code of the algorithm. Here Q is a first-in, first-out queue of candidate vertices, and w(u, v) is the edge weight of (u, v).

2.Least-Cost Algorithms for Shortest Path Problem

2.1. Shortest path problem

The shortest paths problem is one of the most fundamental network optimization problems[4].In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. This is analogous to the problem of finding the shortest path between two intersections on a road map: the graph's vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of its road segment.

The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

2.2. Algorithms

The most important algorithms for solving this problem are:Dijkstra's algorithm solves the single-source shortest path problems. Bellman–Ford algorithm solves the single-source problem if edge weights may be negative. A* search algorithm solves for single pair shortest path using heuristics to try to speed up the search. Floyd–Warshall algorithm solves all pairs shortest paths. Johnson's algorithm solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs. Additional algorithms and associated evaluations may be found in Cherkassky et al[5].

2.3. Road networks

A road network can be considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment. Using directed edges it is also possible to model one-way streets. Such graphs are special in the sense that some edges are more important than others for long distance travel (e.g. highways). This property has been formalized using the notion of highway dimension. There are a great number of algorithms that exploit this property and are therefore able to compute the shortest path a lot quicker than would be possible on general graphs.

All of these algorithms work in two phases. In the first phase, the graph is preprocessed without knowing the source or target node. This phase may take several days for realistic data and some techniques. The second phase is the query phase. In this phase, source and target node are known. The running time of the second phase is generally less than a second. The idea is that the road network is static, so the preprocessing phase can be done once and used for a large number of queries on the same road network.

2.4. Least-Cost Algorithms

Given a network of nodes connected by bidirectional links, where each link has a cost associated with it in each direction, define the cost of a path between two nodes as the sum of the costs of the links traversed. For each pair of nodes, find a path with the least cost. Most least-cost routing algorithms in use in packet-switching networks are variations of one of two common algorithms, knows as Dijkstra's algorithm and the Bellman-Ford algorithm.

3. Shortest Path Faster Algorithm

Dijkstra's algorithm time complexity V) is faster than Bellman-Ford algorithm time complexity O VE). However, Dijkstra's algorithm can not be used to solve the graphs with negative edge weights. So, a new fast algorithm for shortest path problem had been formulated. This algorithm is shortest path faster algorithm (SPFA) which is an improved Bellman-Ford algorithm. The data structure of SPFA algorithm uses adjacency list and a first input first output (FIFO) queue.

3.1. First Input First Output Queue

First input first output (FIFO) queue is an abstraction related to ways of organizing and manipulation of data relative to time and prioritization. This expression describes the principle of a queue processing technique or servicing conflicting demands by ordering process by "first come first served" (What comes in first is handled first, what comes in next waits until the first is finished).

A FIFO queue is defined as a queue under the following discipline of packet arrival and departure:

During a unit time when the output gate is open, the oldest packet exits. (When no packet resides in the queue, the oldest refers to the entering packet) The input gate is open whenever the buffer is not full. A packet arriving at an open input gate is admitted. SPFA algorithm uses a FIFO queue to store the data. New nodes and their attributes are moved to the footer of the queue and extracted from the top of the queue, then processing proceeds are sequentially in the same order.

3.2. Algorithm

Below is the pseudo-code of the algorithm. Here Q is a first-in, first-out queue of candidate vertices, and w(u, v) is the edge weight of (u, v).

```
procedure Shortest-Path-Faster-Algorithm(G, s)
1   for each vertex v ≠ s in V(G)
2   d(v) := ∞
3   d(s) := 0
4   push s into Q
5   while Q is not empty
6       u := pop Q
7       for each edge (u, v) in E(G)
8           if d(u) + w(u, v) < d(v) then
9               d(v) := d(u) + w(u, v)
10              if v is not in Q then
11                  push v into Q
```

4. Experiments and Results

4.1. The Experiment of Random Maps

To have a view on large amounts of data, an experiment was taken. First, the random maps were used in experiment 1. Dijkstra's algorithm solves the graphs with only non-negative edge weights. So, in the case of negative edge weights, compare SPFA algorithm and the proposed algorithm. In the case of non-negative edge weights, compare SPFA algorithm and the improved algorithm and Dijkstra's algorithm. The first one is SPFA algorithm applied to random maps. This algorithm can be used on the graph with negative edge weights. So there are four case, two maps which have 200000 nodes and 500000 edges, the weight of edges is negative and non-negative. The other maps which have 1000000 nodes and 2000000 edges. the weight of edges is negative and non-negative. The second is proposed algorithm applied to random maps. This algorithm can also be used on the graph with negative edge weights. So there are four case, two maps which have 200000 nodes and 500000 edges, the weight of edges is negative and non-negative. The other maps which have 1000000 nodes and 2000000

edges. the weight of edges is negative and non-negative. The third is Dijkstra's algorithm applied to random maps. This algorithm can not be used on the graph with negative edge weights. So there are two case, one map which have 200000 nodes and 500000 edges, the other map which have 1000000 nodes and 2000000 edges. Three times of experiment was taken, it can be shown in Fig.1.

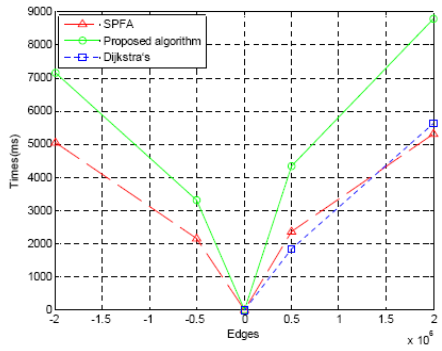

Fig. 1: Comparison of three algorithms (SPFA, the proposed algorithm, Dijkstra's) on random maps

According to comparison of three kinds of algorithms, SPFA algorithm and Dijkstra's algorithm have almost same efficiency. But SPFA algorithm can be used on the graph with negative edge weights. And in the case of negative edge weights, the efficiency of the algorithm is also very high. When the size of the graphs module is randomly generated, the improved algorithm has proven to be inefficient. It takes longer than the other two. So in the next, experiment with a special graph.

4.2. The Experiment of Grid Maps

Keeping iterated is characteristic of improved algorithm, Consequently, it is used on grid maps. Grid maps refer to a number of categories of graphs whose drawing corresponds to some grid, it means that its vertices correspond to the nodes of the mesh and its edges correspond to the ties between the nodes. The depth and the breadth of the maps are equivalent. In the other words, in this two algorithms the iterative times are the same. Grid map is widely used because it is easy to build and maintain without definite geometric parameters.As shown in Fig.2. In order to compare the efficiency of the two algorithms.There are two data: V=700*700 and V=1000*1000.

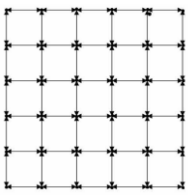

Fig. 2: Example of grid maps

The first one is SPFA algorithm applied to grid maps. The experiment uses the data of known. The second is the proposed algorithm applied to grid maps. The result is as shown in Fig.3.By comparison of the result, the proposed algorithm works better than the original SPFA algorithm.

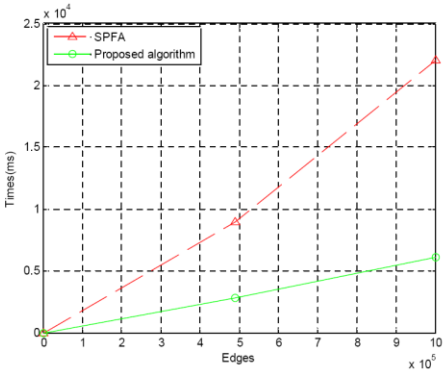

Fig. 3:Comparison of two algorithms (SPFA, the proposed algorithm) on grid maps

4.3. The Experiment Results

As shown in Table. 1,SPFA algorithm and Dijkstra's algorithm have almost same efficiency. But SPFA algorithm can be used on the graph with negative edge weights. And in the case of negative edge weights, the efficiency of the algorithm is also very high.

Table. 1: Random data

Data	SPFA	Proposed algorithm	Dijkstra's algorithm
Negative weights V=200000 E=500000	2135ms	3294ms	-
Negative weights V=1000000 E=2000000	5031ms	7156ms	-
NON-Negative weights V=200000 E=500000	2343ms	4342ms	1830ms
NON-Negative weights V=1000000 E=2000000	5301ms	8779ms	5607ms

As we can see, SPFA algorithm works better than the proposed algorithm, it is more effective. When the size of the graphs module is randomly generated, the improved algorithm is not desirable. It will keep iterated, if a graph that is great depth. It takes too many times to iterative. It will use SPFA algorithm to get a better solution before re-iteration, if the random map contains a certain part of depth maps. For example, as shown in Fig.4. To determin the shortest path from the source node 1 to the destination node 6. First, we use the improved algorithm to solve this problem. After the node 1 is relaxed, we get

path 1-2, the cost is 1. Then the node 2 is relaxed, we get the path 1-2-3, the cost is 6. The process continues until the destination node 6 is relaxed. At that time, the path is 1-2-3-4-6, the cost is 11. Return to the source node, we get the path is 1-5, the cost is 2. Finally, the node 5 is relaxed, we get the path 1-5-6, the cost is 5. The shortest path of this graph is 1-5-6, the cost is 5. This process takes too much time to find the shortest path. However, it will get the shortest path is easy, if we use SPFA algorithm. First, the node 1 is relaxed, we get the path 1-2 and 1-5. Second, the node 2 is relaxed, we get the path 1-2-3, the cost is 6.Then the node 5 is relaxed, we can get the shortest path is 1-5-6, the cost is 5. Because the cost 6 is greater than the cost 5. In other words,the cost of path 1-5-6 lower than path 1-2-3. So the cost of path 1-5-6 certainly be lower than path 1-2-3-4-6. SPFA algorithm is not to be affected by the depth graphs.

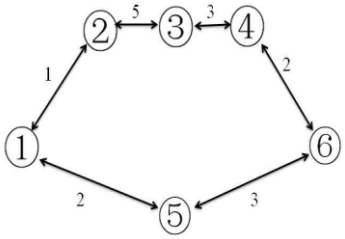

Fig. 4: Example of a random map

Grid maps take advantage of special characteristics of the improved algorithm. The iterative times of two algorithms are the same, when using grid maps. The proposed algorithm will take time to solve the depth problem. Likewise, SPFA algorithm will take time to solve the breadth problem. The depth and the breadth of grid maps are equivalent. On grid maps, the proposed algorithm is more efficient than the original SPFA algorithm. As shown in Table.2.

Table.2: Grid maps data

Data	SPFA	Proposed algorithm
NON-Negative weights V=490000	8973ms	2826ms
NON-Negative weights V=1000000	22075ms	6072ms

As is shown in the table, we can see clearly the relationship between the proposed and SPFA algorithms. In both cases, SPFA algorithm takes 31048 milliseconds, and the proposed algorithm only takes 8898 milliseconds that reduced two-third in time consumption.SPFA algorithm is better than the proposed algorithm in the case of random data. However, the proposed algorithm is more effective on grid maps.

4. 4 . The Application of Grid Map

Grid maps may be arbitrary, or can be based on specific distances. A grid locates a unique square region on the map. The communication may use a simple grid system. For example, this might be used for land navigating for walkers or cyclists etc. The growing availability and decreasing cost of handheld GPS receivers enable determination of accurate grid maps without needing a map. First of all, the city map is divided into a grid of several squares. Each building treat as the node. Thus, we can use the proposed algorithm to compute the shortest distance between the two buildings.

5. Conclusions

The first experimental result shows that SPFA algorithm and Dijkstra's algorithm have almost same efficiency on the random graphs. But, on grid maps the proposed algorithm is very efficient. As shown in Table.2, when CPU is Intel P7350 2.00GHz, windows 7(32bit), SPFA algorithm takes 31048 milliseconds by Matlab, the proposed algorithm only takes 8898 milliseconds that reduced two-third in time consumption. So the proposed algorithm is more effective on grid maps. I can verify that on grid maps the proposed algorithm is very efficient, with the same amount of data within the same grid map that we would save more time. The proposed algorithm than SPFA algorithm reduced two-third in time consumption. This may be broaden great development space for the proposed algorithm in the future.

References

[1] Duan, Fanding, " A Faster Algorithm For Shortest Path-SPFA",*Journal of Southwest Jiaotong university* ,Vol.29,No.6, pp. 207-212, 1994

[2] Tianrui Li, Luole Qi, Da Ruan, "An efficient algorithm for the single-source shortest path problem in graph theory", *International Conference on Intelligent System and Knowledge Engineering*, Vol.1, pp. 152-157, November 2008.

[3] Dijkstra, E.W. "A note on two problems in connexion with graphs". *Numerische Mathematik*, pp,269–271,1959

[4] Kroger, Martin . "Shortest multiple disconnected path for the analysis of entanglements in two- and three-dimensional polymeric systems". *Computer Physics Communications* ,168 (168): pp.209 – 232,2005.

[5] Boris Cherkassky, Andrew V. Goldberg, Tomasz Radzik, "Shortest paths algorithms: Theory and experimental evaluation", *Mathematical Programming*, 129-174, May 1995.