Annals of Optimization Theory and Practice

Volume 3, Number 2, 65-70
November 2020

DOI: 10.22121/aotp.2020.243077.1038

On the shortest path calculation time in the large-scale dynamic post-disaster environment

Seyed Mahdi Shavarani✉ . Béla Vizvári

Department of Industrial Engineering, Eastern Mediterranean University, Famagusta, Turkey

✉ mahdi.shavarani@cc.emu.edu.tr

(Received: August 9, 2020 / Accepted: October 8, 2020)

Abstract There are many metropolitan cities where serious disasters are expected. The disaster, especially if it is an earthquake, damages the roads structure. Thus, the shortest path between two points can be changed. Emergency vehicles, including ambulance, fire-engine, police car, and technical aid, must get the temporary shortest path in real time to work effectively. The shortest path algorithm available in MATLAB, Dijkstra, is tested on two metropolitan cities of San Francisco and Tehran. The road systems of both cities are represented by high number of nodes and connecting arcs. The conclusion is that the algorithm is suitable for finding shortest path for emergency vehicles. However, it is too slow for being a subroutine of a solver for vehicle routing problem.

Keywords Shortest path; Dijkstra algorithm; Bellman equation; Post-disaster period; Emergency vehicle

1. Introduction

In the past, several huge cities suffered from disaster, like Istanbul (1766), Jerusalem (1927), Lisbon (1755), San Francisco (1906), Tehran (1830), and Tokyo (1923), just to mention a few earthquakes. The major earthquake is considered cyclic in these cities with exception of Lisbon. Therefore, it is reasonable if the city authorities establish, organize, and train a relief organization which can save lives in the case of a disaster (Vizvári, 2017; Nedjati *et al.* 2016).

The central agent of the relief organization is the Disaster Command Center (DCC) which also serves as the information and communication center. Reports and appeals on injured persons, fires, and other side effects of the disaster arrive to DCC from various and numerous sources which also include reports on the current state of the road system. Thus, DCC has an updated description of the roads at every moment. DCC sends the emergency vehicles to the spot where help is needed. The vehicle is not at DCC; however, it is at the end of the previous mission. When a mission is completed in any way, e.g. the injured person is transported to a hospital or died, then the DCC must determine the next mission immediately to save time and use the vehicles efficiently. DCC must inform the vehicle or its driver on the current shortest path from its current position to the location of the next mission. Thus, the system must be able to find the shortest path between any two points of the city. This information is needed in real-time as the vehicle or its driver must know into which direction they should start moving.

Thus, the practical problem is that which shortest path method can satisfy the requirements, i.e. the path between any two points of the city is determined in real-time mode in a rapidly changing environment. The method does not need to work in any graph. It is enough if it works in graphs of large cities. This kind of shortest path problem is a technical element of the response part of the disaster management in the post-disaster period. The aim is providing with a good technical solution.

The research on obtaining shortest path has a long history. The origin of the method which is called Dijkstra algorithm goes back to the 1950's to several papers (Dijkstra, 1959; Leyzorek, 1957; Bellman, 1958). It is a simple method. Dijkstra defined the algorithm originally in a way that it finds the shortest path from one node to another node. However, mostly authors interpret it as a method to find the shortest path from one node to all other nodes of the network. The method is based on dynamic programming and Bellman Principle (BP). BP uses dual type variables which is also called potential values (Bellman, 1958; Bellman, 1957). There are many different variants of the original algorithm and all of them are practically based on BP (Ahuja, 1993).

The shortest path procedure of MATLAB is a Dijkstra algorithm as well. This code was selected as the basis of the analysis as MATLAB is available almost everywhere. There were two main alternatives for the outcome of this research at the beginning. Either the MATLAB code satisfies the practical requirements, or a theory must be developed for using effectively the dynamically arriving information about changes in the state of the road system.

This paper is organized as follows: In the next session the BP is explained in summary. Section 3 presents the case studies and the results are discussed in section 4. Conclusions are drawn in section 5.

2. On the Bellman Principle

The Bellman Principle in the case of the shortest path declares that any part of a shortest path is the shortest path between the two end points of the part. Assume that the problem is to find the shortest paths from one node to all other nodes in a directed graph. Assume the graph is $G(N, A)$. The length of arc $(u, v) \in A$ is denoted by l_{uv}. Let a be the starting node. Finally, the potential value of node $u \in N$ is $p(u)$. The potential values must satisfy the so-called Bellman equations. The meaning of the first equation is that the starting point can be reached by zero distance. The second equation says that the shortest path arrives to every other node via the neighbor such that the total length of reaching the neighbor and going from it to the point is minimal:

$$p(a) = 0 \quad (1)$$

$$\forall v \in N, v \neq a: p(v) = \min\{p(u) + l_{uv} |\ (u.v) \in A\} \quad (2)$$

If the graph contains a negative directed circuit, i.e. the total length of the circuit is a negative value, and a path contains a node from the negative circuit, then there is no minimal path as by repeating the negative circuit several times, the passed length becomes arbitrarily small. If a circuit has zero length, then it is possible to decrease the potential values along the circuit such that the values still satisfy the Bellman equations as the following example shows. Let $N = \{a, b, c, d\}$, $A = \{(a, b), (b, c), (c, d), (d, b)\}$ with $l_{ab} = 1, l_{bc} = -3, l_{cd} = 2,$ and $l_{db} = 1.$ The lengths of minimal paths and potential values at the same time $p(a) = 0, p(b) = 1, p(c) = -2,$ and $p(d) = 0.$ However, the values $\bar{p}(a) = 0, \bar{p}(b) = 0, \bar{p}(c) = -3,$ and $\bar{p}(d) = -1$ also satisfy the Bellman equations. If these two cases do not occur, then the Bellman equations are enough to determine the lengths of the shortest paths (Vizvári, 2006; Theorem 5.2.1).

Theorem. Assume that $G(N, A)$ is a directed graph which does not contain negative and zero directed circuit. Assume further on that for every $u \in N$ the value $p(u)$ is finite if there is directed path from node $a \in N$ to node u and otherwise $p(u)$ is plus infinite. For all $u \in N$, the value $p(u)$ is equal to the length of the shortest directed path from a to u if and only if the values $p(u)$ satisfy the Bellman equations.

The theorem discusses the case when the problem is to determine shortest paths from a fixed node (node a in the theorem) to any other node. There are several algorithms to solve this problem. Practically, all algorithms use the Bellman equations directly or indirectly. The complexity of the methods in the case of general graphs is $O(n^2)$ even for the best method (Thorup, 2004). Dijkstra's original method (Dijkstra, 1959) does less. It determines the shortest path to the target point and to points which are closer to the starting point than the target point. It is an important property. We get back to this point in the discussion of the numerical results.

3. Case Studies

3.1 General Notes

To perform the case study of this paper the transportation network of San Francisco and Tehran was acquired by ArcGIS software. Each test was consisted of random selection of two points, finding the route and length of the shortest path between the selected nodes. For each of the selected cities the test is performed 100,000 times and the CPU time of each test was recorded. The tests were performed on core i5 laptop with 6 gigabytes RAM running a windows 10 operating system. It is obvious that the recorded times would significantly dropped if the tests are performed on a laptop with state of the art capabilities. The tests were deployed by MATLAB R2016b.

3.2 San Francisco

The transportation network of San Francisco is consisted of 68,609 arcs and 45,453 nodes representing the streets and intersections spread over the 121 square kilometers area of this city.

Table 1. Summarizes the results of the experiment.

Summarizes the results of the experiment	
Number of nodes	45453
Number of edges	68609
Total number of runs	100,000
Max CPU time (seconds)	0.1268
Min CPU time (seconds)	0.0442
Mean CPU time (seconds)	0.0594
Maximum number of nodes in a route	277
Mean number of nodes in a route	22.93
Maximum length of the routes (Km)	51.67
Mean length of the routes (Km)	18.07

3.3 Tehran

Tehran’s network is comprised of 90378 nodes and 124171 arcs and covers the area of this city which is approximately 750 square kilometers. The results of the experiment consisting of 100,000 tests is illustrated in Table 2.

Table 2. The results of the experiment for the city of Tehran.

The results of the experiment for the city of Tehran	
Number of nodes	90378
Number of edges	124171
Total number of runs	100,000
Max CPU time (seconds)	10.2616
Min CPU time (seconds)	0.0760
Mean CPU time (seconds)	0.1342
number of runs with CPU time less than one second	99999
Maximum number of nodes in a route	446
Mean number of nodes in a route	145.95
Maximum length of the routes (Km)	44.158
Mean length of the routes (Km)	12.60

4. Discussion of the Numerical Results

The results of the tests indicate that finding a shortest path, even in mega cities such as San Francisco and Tehran with tens or hundreds of thousands of nodes and edges, takes only a fraction of a second. Furthermore, the geographical topography and orientation of the two cities are completely different, one having an oblong shape with a very narrow width and one having a square format. In both cases the calculations take less than a tenth second, however the mean time for San Francisco is considerably less than that of Tehran which may be explained by fairly higher number of nodes in the routes for Tehran.

There is only one test in the whole experiment with a CPU time greater than one second (10.26) which could be attributed to the performance of the computer or generally could be considered as noise and deleted from the results.

The underlying mathematical explanation of the surprisingly good results is as follows. The complexity of the shortest path algorithm is $O(n^2)$for graphs in general. However, the road system of a city has many special properties. First of all, it is a planar graph. More importantly, the degrees of the nodes are restricted. Most of the degrees are 4. There are few degrees equal to 3 and even fewer greater than 4. Another important property is that all lengths are positive. The last property makes it possible to apply Dijkstra's original method (Dijkstra, 1959). The limited values of the degrees make the complexity of the method linear in the number of nodes. Moreover, Dijkstra's method uses the nodes in an order of increasing potential values. It implies that the method may stop if the potential value of the next node is greater than or equal to the potential value of the target node. It is exactly what (Dijkstra, 1959) does. Thus, the complexity and the CPU time is reduced further on. Our findings are supported by (Johannes, 2020). It suggests Dijkstra's method in the case of undirected circuits and arbitrary length of edges. However, it does not mention any implementation and running time. (Zhu, 2018) also compares theoretically three algorithms. There is no better method in the case of the graphs of the cities.

5. Conclusion

The question which was investigated in this study is as follows: Assume that there is a large-scale disaster caused for example by an earthquake in a metropolitan city. There are many persons who need urgent medical help at the same time. The road system is also damaged. The damages are uncovered step by step from reports of different sources. The emergency vehicles need information on the temporary shortest path of the assigned mission. Is Dijkstra's method and its MATLAB program suitable for this task?

The findings of the research are as follows:

The Dijkstra's method was tested on two large cities. They are San Francisco and Tehran. Both cities were represented by tens of thousands of nodes and edges. The maps were obtained by ArcGIS software.The shortest path was determined between 100,000 pairs of randomly selected nodes in both cities. Out of the 200,000 experiments, there was only one such that the CPU time exceeded 1 second.

The underlying main factors of the fast running time are as follows:

The graph of a city is a planar graph. Hence, it has relatively small number of edges.
The degrees of the nodes are small. Thus, the forward iteration of Dijkstra's method is very fast.
There is no need for a complete spanning tree of minimal paths. The algorithm can stop much earlier.

In the real practice, further reduction of the CPU time can be expected. One reason is that the average distances are expected less than 18.07 and 12.6 kilometers, respectively, found in San Francisco and Tehran experiment. Another reason is that in the implementation other computer language than MATHLAB can be used which produces faster program codes.

In this research, it is supposed that the schedule of the missions is determined by another subsystem of the disaster management. It is reasonable to carry out research in the future such that the schedule of the missions is optimized in a short time window. This type of calculation may request even faster methods for the determination of the shortest path.

References

1. B. Vizvári, (2017). On the Economic Aspects of the Design of a Relief System in a Metropolitan City -- A Top-Down Approach, *EURO HOpe mini-conference (European Working Group in Humanitarian Operations, IFORS).*
2. Nedjati, A., Vizvari, B., & Izbirak, G. (2016). Post-earthquake response by small UAV helicopters. *Natural Hazards*, *80*(3), 1669-1688.
3. Dijkstra, E.W, (1959). A note on two problems in connexion with graphs. *Numer*. https://doi.org/10.1007/BF01386390
4. Leyzorek, M.; Gray, R. S.; Johnson, A. A.; Ladew, W. C.; Meaker, Jr., S. R.; Petry, R. M.; Seitz, R. N. (1957).— A Study of Model Techniques for Communication Systems. Cleveland, Ohio: Case Institute of Technology. *Investigation of Model Techniques — First Annual Report — 6 June 1956 — 1 July 1957.*
5. Bellman, R. (1958). On a routing problem. Quarterly of applied mathematics, 16(1), 87-90.
6. Bellman, R.E. (1957). Dynamic Programming. *Princeton University Press, Princeton, NJ.* Republished 2003: Dover, ISBN 0-486-42809-5.
7. Ahuja, R. K., Magnanti, T. L., & Flows, J. O. N. (1993). Theory, algorithms, and applications. In *Network flows*.
8. B. Vizvári, (2006). Integer programming, (in Hungarian), *Typotex, , Budapest*
9. Thorup, M. (2004). Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, *69*(3), 330-353.
10. Johannes Baum, (2020). 5 Ways to Find the Shortest Path in a Graph, Medium.com, Better Programming, https://medium.com/better-programming/5-ways-to-find-the-shortest-path-in-a-graph-88cfefd0030f.
11. Zhu Wang, X. (2018). The Comparison of Three Algorithms in Shortest Path Issue. *JPhCS*, *1087*(2), 022011.