Journal of Engineering and Applied Sciences 14 (18): 6620-6624, 2019
ISSN: 1816-949X
© Medwell Journals, 2019

Development of the Java-Based Dijkstra Algorithm for Optimal Path Detection

ECH-Chelfi Wiame and El Hammoumi Mohammed
Industrial Laboratory Techniques, FST, Sidi Mohammed Ben Abdellah University (USMBA), Fez, Morocco

Abstract: In a supply chain management, road transport plays an important role in getting goods from a point of origin to a point of destination, this research focuses on the principle of graph theory to find the optimal path between two points, the document proposes a Dijkstra algorithm to determine the k shortest paths from a single source node to several destination nodes. So, a program based on Java language is developed in two stages The first stage is simulated on a simple model of routing of the goods with affection of the costs and in the second stage a general algorithm has developed to answer all the models regardless of the number of nodes and link, the program simulation can automatically detect the shortest path related to the minimum cost.

Key words: SCM, graph theory, Dijkstra, Java, algorithm, optimal path

INTRODUCTION

Finding the shortest time, the shortest distance or the shortest cost of any node-i-source to a node-j (or target) of a given transport network is a significant and fundamental problem for transport modeling, especially in a supply chain that requires more integration and inter-actor coordination.

Many articles study algorithms for the shortest paths (Bako and Kas, 1977; Brander and Sinclair, 1996; BL., 1975; Ahuja *et al*., 1990; Clarkson, 1987; Borgwardt and Kriegel, 2005). Yen cites several additional articles on the subject going back to 1957. It is necessary to distinguish several common variations of the problem. In most of the documents cited above, the paths are limited to being simple, i.e., no vertex can be repeated. Several papers by Katoh *et al.* (1982), Kumar and Ghosh (1994) consider the version of the problem of smaller paths in which repeated vertices are allowed and this is the version we are also studying.

This study illustrates the best way of routing between two points by automatically detecting the least expensive route according to the value assigned to each link. Dijkstra's algorithm is a graph search algorithm that solves the problem of the shortest single-source path with non-negative path costs, producing a shortest path tree. For a given source vertex (node) in the graph, the algorithm finds the search costs from the shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the vertex destination has been determined.

MATERIALS AND METHODS

The supply chain is a network of interdependent partners whose overall goal is lower cost customer satisfaction (Mentzer *et al*., 2001). The supply chain performs several activities: supply of raw materials and components, production of goods and processing of materials, physical distribution of finished products to end customers (Cavellin *et al*., 2005). There are several activities in the distribution process, transport is considered as a link of particular importance because it provides the link between the different levels of the logistics system from supply to distribution (Wiame, 2017). Transport is one of the major elements in the quality of service and logistics cost, since, it is directly related to delays, errors, losses, breakages, theft, damage, etc. (Baglin, 2005).

The company itself can carry out the transport activity by transporting its products either with its own vehicles or by renting a fleet of vehicles with drivers. The company may also have recourse to a professional carrier directly in charge of the carrier or through a forwarding agent/freight forwarder.

In the transport sector, several activities are carried out: the determination of the routes, the planning of the tours, the preparation of the documents of transport, the loading, the unloading, the follow-up of the disputes, etc. Transportation management aims to determine the most effective approach to managing all these activities in order to provide a better level of service to customers at lower costs.

Corresponding Author: ECH-Chelfi Wiame, Industrial Laboratory Techniques, FST, Sidi Mohammed Ben Abdellah University (USMBA), Fez, Morocco

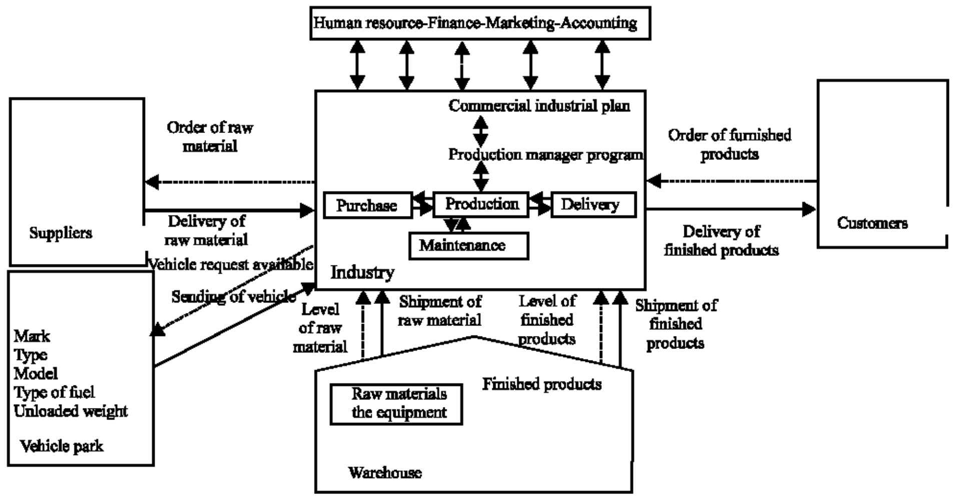
Human resource-Finance-Marketing-Accounting
Commercial industrial plan
Production manager program
Order of raw material
Order of furnished products
Purchase
Production
Delivery
Maintenance
Suppliers
Delivery of raw material
Customers
Delivery of finished products
Vehicle request available
Industry
Sending of vehicle
Level of raw material
Shipment of raw material
Level of finished products
Shipment of finished products
Mark
Type
Model
Type of fuel
Unloaded weight
Vehicle park
Raw materials the equipment
Finished products
Warehouse

Fig. 1: Transport modeling in an SC (Wiame, 2017)

During the routing of the products, unexpected events can occur: appearance of a new request, loss of products, delay, exceeding of the temperature threshold, error of delivery, etc. Not having a real-time visibility on the status of the products transported, the carrier was executing its tours according to the roadmap initially planned. Decisions could not be taken at the right moment, thus, impacting the performance of the transport function.

Faced with this instability of the environment, transport companies must support their management on their ability to adapt to uncertainty and respond quickly to different hazards. For this, the availability of information relating to the state of the resources (location of the vehicle, arrival times, etc.) and the products transported (conformity, integrity, etc.) becomes more and more necessary for a decision. Reactive decision-making, especially with the environmental awareness of companies and sustainable development. Thus, new reflections are put in place consisting of developing an optimal choice of path system based on the dynamic information of carriers and adapting in real time to different path disturbances. The next study will introduce the dijkstra algorithm, that we will develop it using Java language to have speed, flexibility and assistance to the supply chain manager to make the right decision adapted to the requirements of cost, delay, safety, vehicle capacity and trip status (Fig. 1).

Dijkstra algorithm with Java language: Dijkstra has developed one of the most efficient algorithms for finding the shortest paths from one node to all other nodes in the network. This algorithm assumes that all the lengths d (i, j) of all the links of the network G = (N, A) are non-negative.

We denote by-a-the node for which we must discover the shortest paths to all the other nodes of the network. During the process of discovering these shortest paths each node can be in one of two possible states: in an open state if the node is designated by a temporary tag or in a closed state if it is marked by a permanent label. In the case of the permanent label, we are not sure that the open path is the shortest path.

Dijkstra's algorithm progressively changes temporary labels into permanent labels. The initial distances between any two nodes of the network are defined as follows. The distance between the node-a-at the node-a-is zero. The distance between two nodes is equal to ∞ if there is no link between these two nodes. If there is a link that connects two nodes, the distance between these nodes is equal to the length of the link that connects them. If there are some links that connect two nodes, the distance between these nodes is equal to the length of the shortest link that connects them.

Dijkstra is asymptotically the best known single-source short-path algorithm for arbitrary-oriented graphs with unbounded non-negative weights. These are the motivations that led us to use the Dijkstra algorithm.

RESULTS AND DISCUSSION

The calculation steps of the Dijkstra algorithm discussed in this research are as follows:

Step 1: Position the different nodes of the coordinates (x_1, y_1), (x_2, y_2), ..., (x_n, y_n).

Step 2: Make a link between the different nodes.

Table 1: Summary table of trip data

Steps	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2, u	5, u	1, u	8	8
1	ux	2, u	4, x		2, x	8
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					

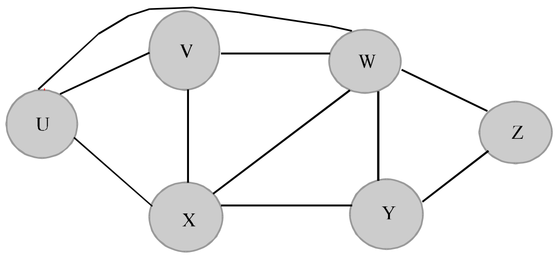

Fig. 2: The sample network

Step 3: Assign a weight to each stop that can designate the cost, delay or distance in our case it was estimated that the weight of the stop from an original node to a destination node is the cost or a formula that takes into consideration the three parameters or a formula based on a variable number of parameters

Step 4: Run the program on a simple model.

Step 5: Develop a general path processing language.

Step 1; Position the different nodes: We took an example, Table 1 to test the right approach of our development principle, Table 1 mentions 6 nodes whose U is the original node and (V-Z) can be destination or passage nodes, the table specifies at each destination the minimal cost and the preceding-p-for example, in step 0 from node U-Z it is ∞ because there is not a direct path between U and Z. To view the best road transportation path, a graph is drawn. Figure 2 to analyze the different routes available.

Step 2; Make a link between the different nodes: To creat a link between the diffierent nodes show the sample network in Fig. 2.

Step 3; Assign an estimated cost: A graph without assigning the parameters is useless, the affected cost can be in relation with the expenses of the route, the time consumed, the degree of security. This research affects estimated costs to find the minimum cost of travel and in the end it is up to the decision-maker to validate the choice of trip according to his fixed priorities.

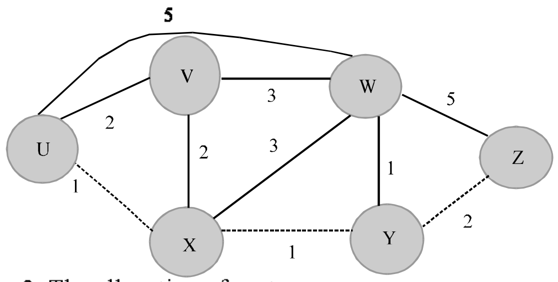

Fig. 3: The allocation of costs

Figure 3 presents the different costs associated with each link, the UXYZ path is the optimal path determined manually according to the principle of the Dijkstra algorithm and thus, step 4 using a developed program must validate what is theoretically determined in step 3.

Step 4: Run the program on a practical example: Java is considered among the most well-known languages, it knows today a great craze in particular thanks to the internet web. Algorithm 1 a developed program capture contains the declaration of differnts nodes as well the cost affection for each liason.

Algorithm 1; Statement nodes and the addition of costs:

```
private void exp3 () {
        Stats s1=new Stats ("u", 1)
        Stats s2=new Stats ("v", 2)
        Stats s3=new Stats ("w", 3)
        Stats s4=new Stats ("x", 4)
        Stats s5=new Stats ("y", 5)
        Stats s6=new Stats ("z", 6)
        MapProject mp=new MapProject (s1)
        mp.addRoadCost (s1, s2, s2)
        mp.addRoadCost (s1, s4, s1)
        mp.addRoadCost (s2, s3, s3)
        mp.addRoadCost (s2, s4, s2)
        mp.addRoadCost (s3, s4, s3)
        mp.addRoadCost (s3, s5, s1)
        mp.addRoadCost (s3, s6, s5)
        mp.addRoadCost (s1, s3, s5)
        mp.addRoadCost (s4, s5, s1)
        mp.addRoadCost (s5, s6, s2)
TransportRoadMap tr=new TransportRoadMap
            tr.calcLigne()
      tr.affiche ()
}
public static void main (Staring[] args)  {
   // T0D0 Auto-generated method strub
new buildMap Test ()
        }
}
```

The first example, result display is done on a console linked to the Java interface algorithm 2.

Algorithm 2; Java interface algorithm:

```
u:v:coast = 2
u:x:coast = 1
u:w:coast = 5
u:y:coast = -1
u:z:coast = -1
targetJocProv:Target [target=[x-, y-], statsSource = u, stats Destination = y, Etat=sherch]
targetJocProv:Target [target=[x-, y-, w-], statsSource = u, stats Destination = w, Etat=sherch]
targetJocProv:Target [target=[x-, y-, z-], statsSource = u, stats Destination =z, Etat=sherch]
targetJocProv:Target [target= [v-], statsSource = u, stats Destination = v, Etat=sherch]
Target jok=Target [target=[v-], statsSource=u, statsDestination=v, Etat=end]
null
Target [target=[x-], statsSource=u, statsDestination=x, Etat=end]
Target [target=[x-, y-, w-], statsSource=u, statsDestination=w, Etat=end]
Target [target=[x-, y-], statsSource=u, statsDestination=y, Etat=end]
```

The simulation of this development confirmed the result obtained manually to go from U-Z the console displays as optimal path UXYZ path whose U is the source node, X and Y are passing nodes and Z the destination node.

The next step is an extension and generalization of the program to trace the n nodes, assign the costs and detect automatically and visually the optimal path to each terminal node.

Step 5: make the operation automatic and general: Algorithm 3 presents the routing code of the goods in a multi-node graph with a projection of information on a table, the program content helps to declare the nodes and assign a cost in the middle of each link and a piece of code is reserved for displaying an interface to run the simulation.

Algorithm 3; Extract Java language developed:

```
package Map
Import Java.util.Hashtable
public class Transport RoadMap  {
        private  MapProject  mp
        private  Stats statsSource
        private  Hashtable<Stats,Target> targets
        private  Vector<Stats> stats
        private  Target targetJoc
        public  TransportRoadMap (MapProject mp, Stats statsSo1
              this.mp=mp
              this.statsSource=statsSource
              targets=new Hashtable<Stats,Targets> ()
              stats=mp.getStats ()
              statsSource=mp.getStatsSource ()
        }
public void EndTarget () {
for (int i=0;i<stats.size (); i++) {
                Stats st=stats.get (i)
                Target tr=targets.get (st)
                If (tr !=null)   {
                If tr.equals(targetJoc) ) {
                      Targets.get (st) . end ()
                      return
```

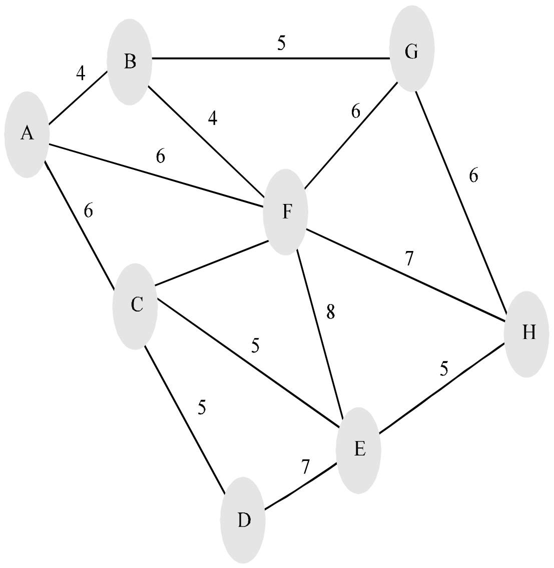

Fig. 4: Manual tracing of nodes and allocation of costs in the developed application

Table 2: Display of results obtained

Sources	Destination	Target	Coast
A	B	[B-]	4
A	C	[C-]	6
A	G	[B-, G-]	9
A	F	[F-]	6
A	D	[C-, D-]	11
A	E	[F-, E-]	14
A	H	[F-, H-]	13

```
public Vector<Vector<String>> affiche () {
        System.out.printIn("Target jok="+targetJoc)
        System.out.printIn("Target.get (statsSource) )
        Vector<Vector<String>> vv=new Vector<Vector<String>> ()
        for(int i=1;i<stats.size (); i++) {
            Vector<String> v1=new Vector<String> ()
            Stats st=stats.get (i)
        v1.add (targets.get (st).getStatsSource() .getName () )
             v1.add (targets.get (st).getStatsDestination().getName() )
             v1.add (targets.get (st).getTarget()+"")
             v1.add (targets.get (st).getCoastTarget ()+"")
                vv.add(v1)
               System.out.printIn(targets.get(st) )
    }
return vv
}
Override
public String toString() {
    return "TransportRoadMap [mp="+mp+", statsSource="+statsSc
        +",targets="+targets+", stats="+stats+"]"
}
```

After the program simulation to detect the optimal path from a source to different locations, displaying a routing graph drawing interface (Fig. 4) and a summary table of results, Table 2 is necessary at the end to make the work accessible and understandable.

CONCLUSION

The study proposes a computerized Dijkstra algorithm to determine the k shortest paths from a single source node to multiple destination nodes. The algorithm uses the connection matrix of a given network and links to obtain the shortest paths. The proposed Dijkstra algorithm has been applied to two successive examples of network topology The first example is a test model to check the program execution and its adequacy with the theoretical and manual resolution, so, the second developed program is general and adaptable to a number of nodes and links unlimited, the results obtained at the end of study are satisfactory produced by a minimum number of operations.

The proposed program is considered the first algorithm that uses Dijkstra algorithms to obtain the k shortest paths of a single source node with multiple destination nodes developed on the Java programming language 6.

REFERENCES

Ahuja, R.K., K. Mehlhorn, J. Orlin and R.E. Tarjan, 1990. Faster algorithms for the shortest path problem. J. ACM., 37: 213-223.

BL, F., 1975. K-th shortest paths and applications to the probabilistic networks. ORSA/TIMS. Joint National Meeting, 23: 263-263.

Baglin, G., 2005. Industrial and Logistics Management: Designing and Managing the Supply Chain. 5th Edn., Edition Economica, Paris, ISBN:9782717853421, Pages: 746 (In French).

Bako, A. and P. Kas, 1977. Determining the K-th shortest path by matrix method. Szigma, 10: 61-66.

Borgwardt, K.M. and H.P. Kriegel, 2005. Shortest-path kernels on graphs. Proceedings of the 5th IEEE International Conference on Data Mining, November 27-30, 2005, Houston, TX., USA., pp: 74-81.

Brander, A.W. and M.C. Sinclair, 1996. A Comparative Study of K-Shortest Path Algorithms. In: Performance Engineering of Computer and Telecommunications Systems, Merabti, M., M. Carew and F. Ball (Eds.). Springer, London, UK., ISBN:978-3-540-76008-5, pp: 370-379.

Cavellin, J., M.C. Debourg and O. Perrier, 2005. Marketing Practice. 2nd Edn., BERTI Editions, Dely Ibrahim, Algeria, Pages: 372.

Clarkson, K., 1987. Approximation algorithms for shortest path motion planning. Proceedings of the 19th annual ACM Symposium on Theory of Computing (STOC'87), May 25-27, 1987, ACM, New York, USA., ISBN:0-89791-221-7, pp: 56-65.

Katoh, N., T. Ibaraki and H. Mine, 1982. An efficient algorithm for K shortest simple paths. Networks, 12: 411-427.

Kumar, N. and R.K. Ghosh, 1994. Parallel algorithm for finding first K shortest paths. Comput. Sci. Inf., 24: 21-28.

Mentzer, J.T., W. DeWitt, J.S. Keebler, S. Min, N.W. Nix, C.D. Smith and Z.G. Zacharia, 2001. Defining supply chain management. J. Bus. Logist., 22: 1-25.

Wiame, E.H.E.C., 2017. The organizational modeling of a supply chain management. Intl. J. Innovations Eng. Technol., 8: 76-81.