© 2018 IJSRST | Volume 4 | Issue 10 | Print ISSN : 2395-6011 | Online ISSN : 2395-602X
Themed Section: Science and Technology

Dijikstra Algorithm In Google Map By Using Fuzzy Graph

M. Bharath Lavanya[1], T. Ramesh[2]

[1]PG Student,Department of Mathematics, Dr. SNS Rajalakshmi College of Arts and Science(Autonomous), Coimbatore, Tamilnadu, India

[2]Assistant professor, Department of Mathematics, Dr. SNS Rajalakshmi College of Arts And Science(Autonomous), Coimbatore, Tamilnadu, India

ABSTRACT

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. The shortest path problem (SPP) is one of the most important combinatorial optimization problems in graph theory due to its various applications. In this paper we introduce the method of finding the shortest path in complete fuzzy graph of uncertainty by using Dijikstra algorithm. To illustrate the effectiveness of the proposed approach we compare these results with Google map and discuss the algorithm by Dijikstra which is most efficient for complete fuzzy graph for uncertainty.

Keywords : Fuzzy Graph, Fuzzy Tree, Fuzzy Network, Fuzzy Sub-Trees.

I. INTRODUCTION

Graphs are a very important model of networks. There are many real-life problems of network of transportation, communication, circuit systems, and so forth, which are modeled into graphs and hence solved. Graph theory has wide varieties of applications in several branches of engineering, science, social science, medical science, economics, and so forth, to list a few only out of many. In this paper we discuss about fuzzy shortest route (path) between two cities by using Dijikstra algorithm.

One of the main real life applications of dijikstra algorithm is in Google Maps. Google map uses many basic algorithms from Graph Theory. For example, to find the shortest path between two nodes in a graph in order to get driving directions. One unique problem is that the graphs used in Google Maps contain millions of nodes, but the algorithms have to run in milliseconds. A technique used to improve performance is graph hierarchies. Dijkstra algorithm is used here and we compare the results with Google map also we show importance of dijikstra algorithm in Google map.

II. BASIC CONCEPTS

2.1. FUZZY GRAPH

A **Fuzzy graph** G= (σ, μ) is a pair of functions σ: V→ [0, 1] and μ: V×V→ [0, 1], where for all u, v∈ V, we have μ (u, v) ≤ σ (u) Λ σ (v). A path P in a fuzzy graph is a sequence of distinct nodes $v_1, v_2, \ . \ . \ . \ , v_n$ such that $\mu(v_i, v_{i+1}) > 0$; $1 \le i \le n$; here n >1 is called the **length of the path P**. The consecutive pairs (v_i, v_{i+1}) are called the **edge of the path.**

2.2. NETWORK:

A Network consists of a set of nodes linked by arcs (or branches). The notation for describing the network is (N, A), where N is a set of nodes and A is the set of arcs.

2.3. PATH:

A Path is a sequence of distinct arcs that join two nodes through other nodes regardless of the direction of flow in each arc.

2.4. FUZZY TREE

A Fuzzy path between the point s (source) to t (sink) of fuzzy graph G is called **fuzzy policy** or **fuzzy tree**.

III. DIJIKSTRA ALGORITHM

Let the node at which we are starting be called the **initial node**. To find the path of minimum distance between the point s(source) to t (sink) of fuzzy graph G can be obtained using the following steps:

1. Identify the decision variables and specify objective function to be optimized for fuzzy networks.
2. Mark all nodes unvisited. Create a set of all the unvisited nodes called the **unvisited set**.
3. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
4. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
5. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
6. Move to the next unvisited node with the smallest tentative distances and repeat the above steps which check neighbors and mark visited.
7. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
8. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3. When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current")

IV. DIJIKSTRA ALGORITHM IN CONTEXT TO THE PROBLEM

Consider a fuzzy network [1] and in this section we discuss applicability of Dijikstra algorithm to find out the shortest path between 21 cities of Tamilnadu. Here we consider each city as vertex/node and the distance between any two cities is represented as edge or arc. In order to develop the above problem in terms of fuzzy graph, the distance between any two cities [1] is consider in the following manner.

Table 1

Distance in km	Membership grades
50-75	0.02
75-100	0.04
100-125	0.06
125-150	0.08
150-175	0.1
175-200	0.12
200-225	0.14
225-250	0.16
250-275	0.18

Here the cities are represented by nodes in an alphabetical order,

A-CHENNAI , B- CHENGALPATTU, C- VELLORE, D- VILLUPURAM, E-THIRUVANAMALAI, F-DHARMAPURI, G- PERAMBALUR, H- NAMAKAL, I- ERODE, J- OOTY, K- THANJAVUR, L- TRICHY, M- COIMBATORE , N- PUDUKOTTAI , O-DINDUKAL, P- KODAIKANAL, Q-RAMANATHAPURAM, R- MADURAI, S-TUTICORIN, T- THIRUNELVELI and U-KANYAKUMARI.

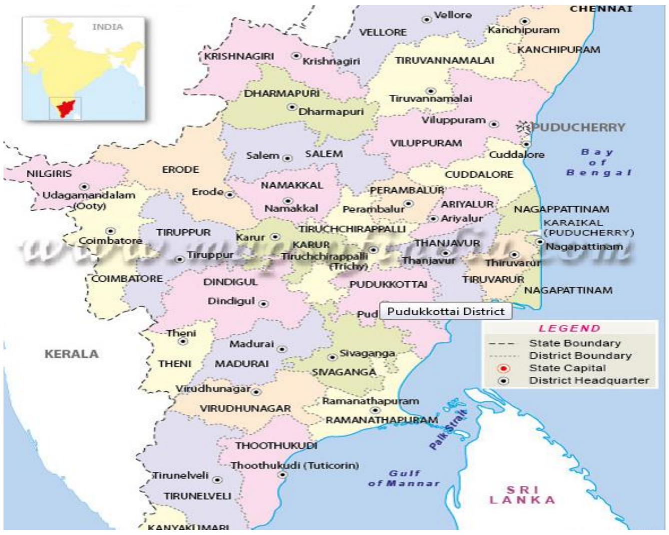

Figure 1. Tamilnadu city map

By using the city map of Tamilnadu we can able to find the distance between two cities (fig 4.1) as mentioned below,

In order to find the shortest path between two cities in terms of fuzzy graph , the distance between any two cities are consider in the following manner:

Table 2

Chennai- Vellore-0.08	Ooty-Coimbatore-0.04
Chennai- Chengalpattu-0.02	Thanjavur-Pudukottai-0.02
Vellore- Villupuram-0.08	Trichi-Pudukottai-0.02
Chengalpattu-Villupuram-0.04	Trichi-Dindukal-0.06
Chengalpattu – Thiruvanamalai-0.08	Coimbatore-Dindukal-0.1
Vellore- Dharmapuri-0.1	Coimbatore-Kodaikanal-0.1
Villupuram-Perambalur-0.06	Pudukottai-Ramanadhapuram-0.08
Villupuram-namakkal-0.16	Pudukottai-Madurai-0.06
Thiruvanamalai-Perambalur-0.08	Dindukal-Madurai-0.02
Thiruvanamalai-Erode-0.14	Kodaikanal-Madurai-0.06
Dharmapuri-Erode-0.06	Ramanathapuram-Tuticurin-0.08
Perambalur-Thanjavur-0.02	Ramanathapuram-Thirunelveli-0.1
Perambalur-Trichi-0.02	Madurai-Tuticorin-0.08
Namakkal-Trichi-0.06	Madurai-Thirunelveli-0.1
Erode-Trichi-0.1	Tutucorin-Kanyakumari-0.08
Erode-Coimbatore-0.04	Thirunelveli-Kanyakumari-0.04

V. FINDING SHORTEST PATH FROM COIMBATORE TO KANYAKUMARI

From this fuzzy graph first let us first find the shortest path from Coimbatore - Kanyakumari (i.e.) node M-U by using dijikstra algorithm. Then the fuzzy graph from Coimbatore – Kanyakumari [1](fig 4.2)is mentioned below ,

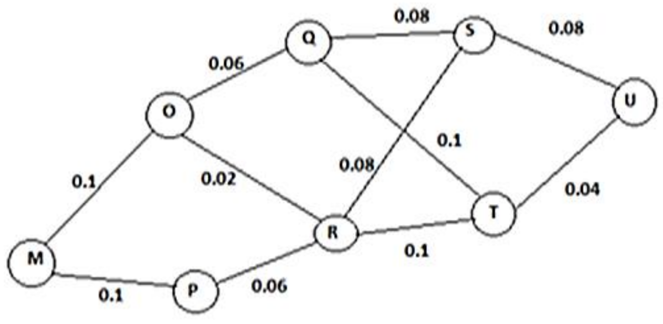

Figure 2. Fuzzy graph from Coimbatore – Kanyakumari with different routes

STEP1: Mark all nodes unvisited. Create a set of all the unvisited nodes called the **unvisited set**.

Table 3

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M		
O		
P		
Q		
R		
S		
T		
U		

Visited-[ ]   Unvisited-[M, O, P, Q, R, S, T, U]

STEP 2: Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.

Table 4

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	∞	
P	∞	
Q	∞	
R	∞	
S	∞	
T	∞	
U	∞	

Visited-[ ]   Unvisited-[M, O, P, Q, R, S, T, U]
Distance to M from M is 0
Distance to all other vertices from M is unknown, therefore infinity.

STEP 3: For the current node, consider all of its unvisited neighbors and calculate their *tentative* distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For the current vertex, examine its unvisited neighbors. We are currently visiting M and its unvisited neighbors are O and P.

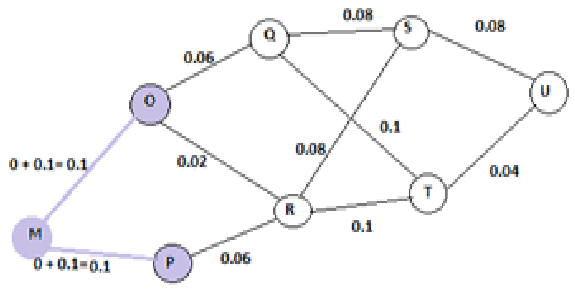

Figure 3

For the current vertex, calculate the distance of each neighbor from the start vertex.

Here M to O = 0.1
M to P = 0.1

If the calculated distance of the vertex is less than the known distance, update the shortest distance. Update the previous vertex for each of the updated distance. In this case we visited O and P via M

Table 5

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	∞	
R	∞	
S	∞	
T	∞	
U	∞	

Now add the current vertex to the list of visited vertices

Visited-[M]   Unvisited-[O, P, Q, R, S, T, U]

STEP 4: Move to the next unvisited node with the smallest tentative distances and repeat the above steps which check neighbors and mark visited.

Therefore by step 4 we visit the unvisited vertex with the smallest known distance from the start vertex, this time is around either O or P because both are having same tentative distance . Let as consider O,

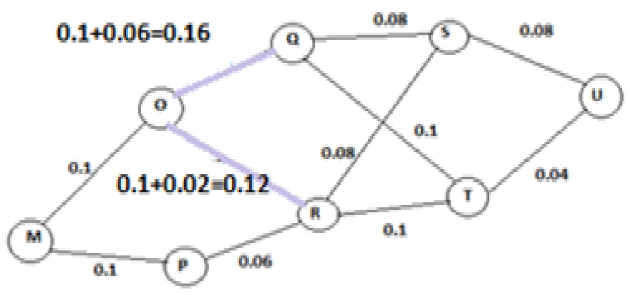

Figure 4

We are currently visiting O and its unvisited neighbors are Q and R.

Here O to Q = 0.1+ 0.06 = 0.16
O to R = 0.1+0.02 = 0.12

If the calculated distance of the vertex is less than the known distance, update the shortest distance. Update the previous vertex for each of the updated distance. In this case we visited Q and R via O

Table 6

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	∞	
T	∞	
U	∞	

Now add the current vertex to the list of visited vertices

Visited-[M, O]   Unvisited-[P, Q, R, S, T, U]

STEP 5: Move to the next unvisited node with the smallest tentative distances.

Here we visit the unvisited vertex with the smallest known distance from the start vertex, this time is around P.

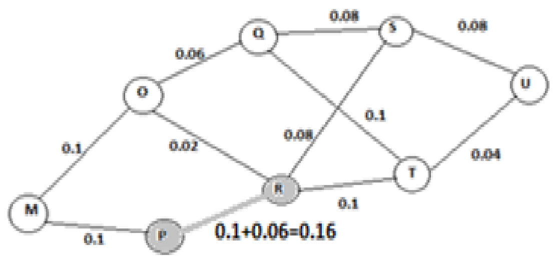

Figure 5

Here P to R = 0.1+0.06=0.16

If the calculated distance of the vertex is less than the known distance, update the shortest distance. But here the calculated distance(0.16) is greater than the known distance(0.12),the known distance remain as it is.

Table 7

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	∞	
T	∞	
U	∞	

Now add the current vertex to the list of visited vertices

Visited-[M, O, P]   Unvisited-[Q, R, S, T, U]

STEP 6: Move to the next unvisited node with the smallest tentative distances.
Here we visit R,

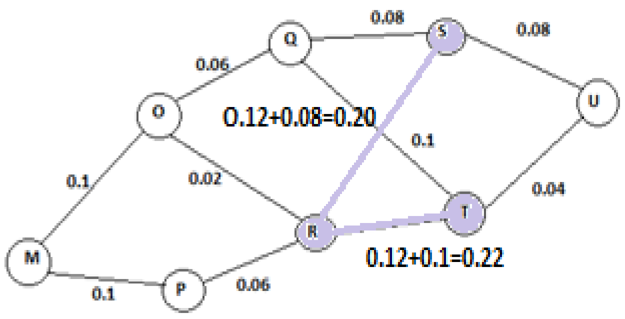

Figure 6

We are currently visiting R and its unvisited neighbors are S and T.
Here R to S = 0.12+ 0.08 = 0.20
R to T = 0.12+0.1 = 0.22

If the calculated distance of the vertex is less than the known distance, update the shortest distance. Update the previous vertex for each of the updated distance. In this case we visited Sand T via R

Table 8

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	0.20	R
T	0.22	R
U	∞	

Now add the current vertex to the list of visited vertices

Visited-[M, O, P, R]   Unvisited-[Q, S, T, U]

STEP 7: Move to the next unvisited node with the smallest tentative distances.
Here we visit Q,

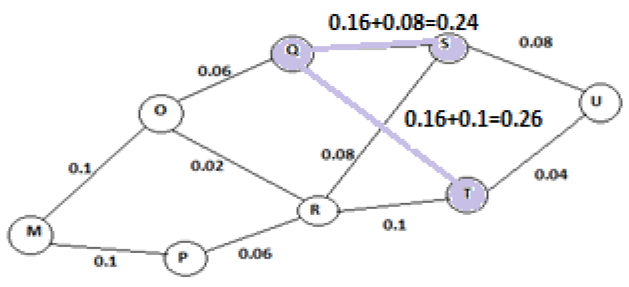

Figure 7

We are currently visiting Q and its unvisited neighbors are S and T.

Here Q to S = 0.16+ 0.08 = 0.2
Q to T = 0.16+0.1 = 0.26

If the calculated distance of the vertex is less than the known distance, update the shortest distance. Here for the vertex S, calculated distance 0.24 is greater than the known distance 0.20. So known distance(0.20) of S remains same and also for the vertex T, calculated distance 0.26 is greater than the known distance 0.22. So known distance(0.22) of T remains same. In this case we visited S and T via Q

Table 9

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	0.20	R
T	0.22	R
U	∞	

Now add the current vertex to the list of visited vertices

Visited-[M, O, P, R, Q] Unvisited-[S, T, U]

STEP 8: Move to the next unvisited node with the smallest tentative distances.
Here we visit S,

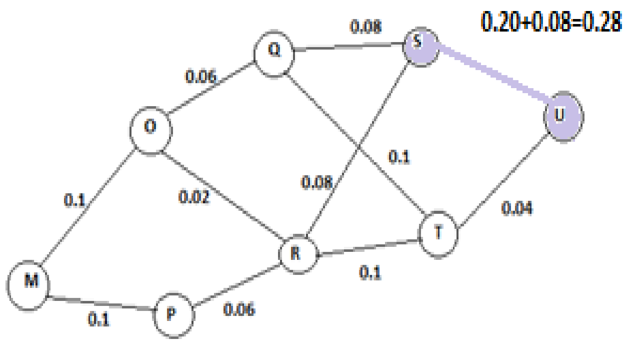

Figure 8

We are currently visiting S and its unvisited neighbors is U.

Here S to U = 0.20+0.08 = 0.28

If the calculated distance of the vertex is less than the known distance, update the shortest distance. In this case we visited U via S

Table 10

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	0.20	R
T	0.22	R
U	0.28	S

Now add the current vertex to the list of visited vertices

Visited-[M, O, P, R, Q, S] Unvisited-[T, U]

STEP 9: Move to the next unvisited node with the smallest tentative distances.

Here we visit T,

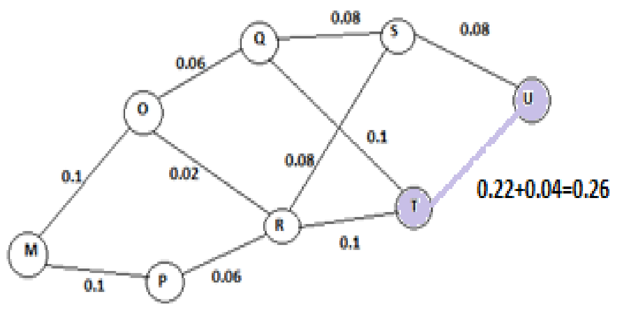

Figure 10

We are currently visiting T and its unvisited neighbor is U.

Here T to U = 0.22+0.04 = 0.26

If the calculated distance of the vertex is less than the known distance, update the shortest distance. Therefore the calculated distance (0.26) is replaced with the known distance (0.28). Update the previous vertex for each of the updated distance. In this case we visited U via T

Table 11

VERTEX	SHORTEST DISTANCE FROM M	PREVIOUS VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	0.20	R
T	0.22	R
U	0.26	T

Now add the current vertex to the list of visited vertices

Visited-[M, O, P, R, Q, S, T] Unvisited-[U]

STEP 10: We are currently visiting U. We can't able to move to the next unvisited node with the smallest tentative distances as there is no unvisited node.

Table 12

VERTEX	SHORTEST DISTANCE FROM M	PREVIOU S VERTEX
M	0	
O	0.1	M
P	0.1	M
Q	0.16	O
R	0.12	O
S	0.20	R
T	0.22	R
U	0.26	T

Now add the current vertex to the list of visited vertices

Visited-[M, O, P, R, Q, S, T, U] Unvisited-[ ]

Hence the algorithm stops until all nodes are visited. Here the vertex U has the previous vertex T, vertex T has the previous vertex R, vertex R has the previous vertex O and vertex O has the previous vertex M. Therefore we get the shortest path from M-U is M-O-R-T-U.

VI. DIJIKSTRA ALGORITHM TABULAR FORM REPRESENTATION FROM COIMBATORE TO KANYAKUMARI

The following tabular column shows the shortest distance between Coimbatore(M) to Kanyakumari(U):

Table 13

V	M	O	P	Q	R	S	T	U
M	$\mathbf{0_M}$	0.1_M	0.1_M	∞	∞	∞	∞	∞
O		$\mathbf{0.1_M}$	0.1_M	0.16_O	0.12_O	∞	∞	∞
P			$\mathbf{0.1_M}$	0.16_O	0.12_O	∞	∞	∞
R				0.16_O	$\mathbf{0.12_O}$	0.20_R	0.22_R	∞
Q				$\mathbf{0.16_O}$		0.20_R	0.22_R	∞
S						$\mathbf{0.20_R}$	0.22_R	0.28_S
T							$\mathbf{0.22_R}$	0.26_T
U								$\mathbf{0.26_T}$

Here $\mathbf{0_M}$, $\mathbf{0.1_M}$, $\mathbf{0.1_M}$, $\mathbf{0.16_O}$, $\mathbf{0.12_O}$, $\mathbf{0.20_R}$, $\mathbf{0.22_R}$, $\mathbf{0.26_T}$ are all represent visited nodes and ∞ represent unvisited nodes . Hence by this we can find the shortest distance from Coimbatore to Kanyakumari (i.e.) M to U. The vertex U has the previous vertex T, vertex T has the previous vertex R, vertex R has the previous vertex O and vertex O has the previous vertex M. Here the shortest distance is M-O-R-T-U = 0.26

Hence the shortest path is Coimbatore– Dindukkal– Madurai– Thirunelveli- Kanyakumari=0.26.

VII. COMPARING THE RESULTS WITH GOOGLE MAP

When we compare the results with Google map we get the same shortest path, it is represented in the figure 11

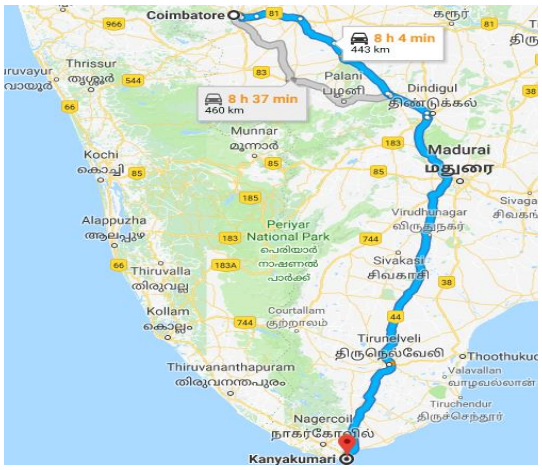

Figure 11. Google map direction from Coimbatore to Kanyakumari

It shows the same path (i.e.)., Coimbatore- Dindukkal- Madurai- Thirunelveli- Kanyakumari. Therefore we come to know that Dijikstra algorithm plays a major role in finding shortest path in Google map.

VIII. IMPORTANCE OF DIJIKSTRA ALGORITHM IN GOOGLE MAP

Dijikstra algorithm is used to find the shortest path. Google Maps does this for us now and we don't even really think about what a complex task it could be. Shortest path problems, a key feature of graph theory with a whole lot of pretty obvious real-world applications, get insanely deep very fast. The result is known (informally) as a combinatorial explosion, which is where the number of different combinations that must be explored in a given problem grows exponentially. The result of such an explosion is that problems, like shortest path problems, grow so quickly as to become practically incomputable, taking a practically infinite amount of time to solve. It only takes a handful of nodes in a given map or graph for the number of possible combinations to push into the billions, requiring vast and unreasonable amounts of time. But here the Google map uses Dijikstra algorithm and within a second it find the shortest path between two cities.

IX. CONCLUSION

In this paper we showed the efficiency of Dijikstra algorithm in Google map and find out the value of uncertain fuzzy shortest route among the cities from Coimbatore to Kanyakumari by Bus is 0.26. The results which are obtained for the given example shows that Dijkstra Algorithm is very effective tool to find the path with lowest cost from node A to node U. Same results have been obtained also for Minimum Spanning Tree by using Kruskal algorithm[1], but this case the procedure is much simpler and easy.

X. REFERENCES

[1]. G. Nirmala & K. Uma – Shortest route algorithm for fuzzy graph, International Journal of Scientific and Research Publications, Volume 3, Issue 11, November 2013 1 ISSN 2250-3153

[2]. Application of Graph Theory to find Optimal Paths for the Transportation Problem- Rame Likaja, Ahmet Shalaa and Mirlind Bruqi, International Journal of Current Engineering and Technology ISSN 2277 – 4106

[3]. An application of Dijkstra's Algorithm to shortest route problem-Ojekudo, Nathaniel Akpofure (PhD) 1& Akpan, Nsikan Paul (PhD), IOSR Journal of Mathematics (IOSR-JM) e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 13, Issue 3 Ver. 1 (May.-June. 2017), PP 20-32