UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, DEHRADUN

Improvement of Dijkstra's Algorithm for Wireless Networks

Project Report of the project (Minor-1) in Semester V

Submitted by Team No: 13

S.No.	Student Name	Roll No.	Sap ID
1	PHAGUN JAIN	R171219022	500076430
2	AKSHAT GOYAL	R171219003	500076394
3	MUDIT ARYA	R171219016	500076553

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE ENGINEERING

With specialization in DevOps

Under the guidance of

Dr. Dhirendra Kumar Sharma

School of Computer Science (SOCS)

Department of Cybernetics, UPES

Bidholi Campus, Energy Acres, Dehradun – 248007

INDEX

1.0 Title of the Project

Improvement of Dijkstra's Algorithm for Wireless Networks using Fuzzy Logic.

2.0 Introduction

2.1 Background Information

In current trends, people are connected by wireless technology. This way we are continuously disseminating data, which shows its significance and importance in our day-to-day life. Effective data transfer is one of the most prominent tasks because in wireless communication data loss is a major issue. In another way, we can see data loss, over inefficient path also cause delay, which can be optimized by creating a better path. In wireless network path, evolution can be understood by using Dijkstra's algorithm. Dijkstra algorithm has been evolved to represent communicating device as a graph node where these nodes are connected through edges. Here a graph cost is estimated with weighted edge values between two end nodes. These end nodes are designated as source and destination, which can be mapped concerning the real-time scenario. Further, the Dijkstra algorithm solves several shortest path problems (SPP) with other approaches. It finds the shortest distance between two nodes keeping the source node fixed, that is solves the problem of single source shortest path problem, however while finding the distance between the two nodes the hopping cost between two nodes is the constant parameter for example in case of road network it may be the distance in kilometres, although it is completely right implementation but the scope of the hopping cost can be expanded way further, here it is proposed that instead of single parameter use a set of values again considering the road network values then values can be weather conditions, road quality, traffic conditions, in addition to the previous distance parameter, similarly in case of the data communication while transferring the data packet from one node to another various factors are required to be taken under consideration to pass the data packet like latency on the next router, traffic on the router, transmission speed on the router after all

these considerations the decision is made on to which router/node of available options should the data packet should be sent so that the transmission takes as less time as possible with as less loss as it can be. Dijkstra is further extended by using the fuzzy set as a representative of the several factors involved in edge length considered while hopping from one node to the next node.

This improvisation of Dijkstra's algorithm falls under the soft computing domain, soft computing addresses the certitude that the human psyche can store and deal with data that is loose and needs sureness. Real-world problems are taken care of with soft computing as it is a randomly defined process that can be analyzed statistically but not with accuracy. Since usable solutions are created with soft, computing based on the model of the human mind it is also referred to as computational intelligence. It holds the tolerance of semi-truth and estimation that the traditional computing methods do not have.

2.2 Motivation

One of the uses of Dijkstra's algorithm is one where we see a navigation system. Nevertheless, it is also used in networking where connections are considered as edges and intermediary

devices are considered as nodes. The network layer is focused on getting data packets from the source to the destination. Getting to the destination might require making numerous

jumps at intermediate switches in route.

Dijkstra's algorithm is the most commonly used method to solve the shortest path problem(Deng *et al.*, 2012), on account of the crisp number to demonstrate curve lengths, the Dijkstra calculation can be effective to execute, but because reason that numerous enhancement techniques for crisp numbers can't be applied straightforwardly to fuzzy numbers, a few changes are required before utilizing the classical technique. In the traditional Dijkstra algorithm, the edge length is the single crisp number, which can be used to address only a single parameter at a time. This can be extended to use multiple parameters as

discussed earlier above in the example of road transport networks like distance, traffic, road quality or weather, etc. further since this is done to expand the scope to real-world scenarios the parameters can be utilized as fuzzy sets indicating the uncertainty to mimic the real-world situations.

2.3 Contribution

In this project a generic Dijkstra algorithm is presented to handle the shortest path problem (SPP) in case of uncertain conditions (or more precisely real-world scenarios) mimicked by the use of fuzzy numbers in arc lengths depicting the lack of sureness, two key issues need to be addressed in SPP with fuzzy parameters. One is how to decide the addition of two edges. The other is the way to look at the distance between two distinct ways with their edge lengths addressed by fuzzy numbers [1]. One very simple approach to handle the issue is by transforming the fuzzy number into the crisp number. A common work changes the trapezoidal number into a crisp number by the defuzzification function, also called Yager's ranking record [2] By and large talking, two fundamental issues should be settled while applying the Dijkstra calculation in a fuzzy climate. One is the adding activity of fuzzy numbers; the other is the positioning and examination of fuzzy numbers, which is still an open issue in fuzzy set theory research fields.

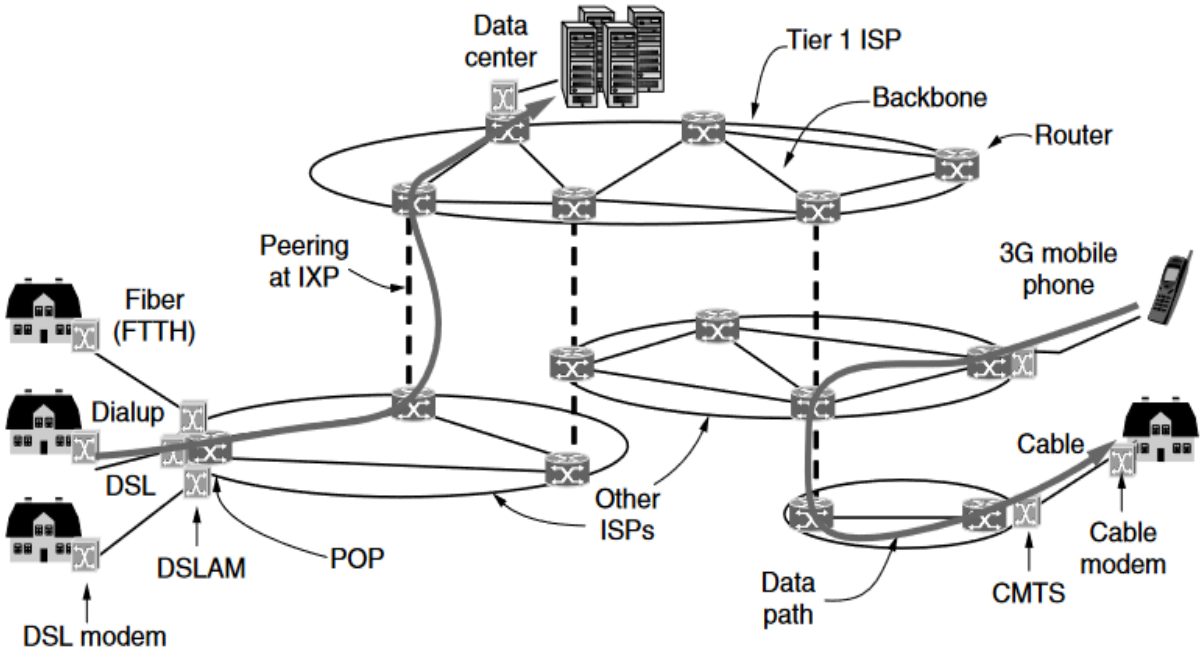


Fig.1 Illustration of a typical internet architecture source: ("Prentice Hall - Computer Networks Tanenbaum 4ed," no date)

3.0 Related Work

3.1 Works have been done until now

3.1.2 An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers

(Akram, Habib and Alcantud, 2021)The Dijkstra calculation is an achievement in the examination of the SPP with fresh data. At the point when curve lengths are crisp numbers, it very well may be effortlessly perceived and applied. Be that as it may, in organizations of equivocal nature, the advancement techniques

Intended for crisp lengths may not be promptly employed when information is fuzzy numbers. The design of the issue produces objective qualities that are fuzzy numbers as well. For this explanation, Klein clarified that 'fuzzy most limited ways are hampered by the likelihood that the fuzzy most limited length may not compare to a genuine way in the organization,' An issue previously noted by the spearheading Dubois and Prade. Hence, a few variations are needed to resort to the traditional methodology. We list yet a couple of arrangements in the writing. Liu and Kao utilize a defuzzification work known as Yager's positioning record. With the assistance of this device, they change an issue where trapezoidal fuzzy numbers into a fresh detailing weight the curves. Klein states new models, which depend on fuzzy briefest ways also various destinations. Peyer et al. sum up the Dijkstra's SP calculation and apply it to VLSI directing. Shu-Xi recommends a further developed Dijkstra's SP calculation with applications. Then again, other examination papers explore the fuzzy Dijkstra calculation for SPP, with the goal that it tends to be applied under a few questionable conditions this paper extends the traditional Dijkstra algorithm so that we can find the minimum cost of the picture fuzzy SPP (PFSPP). The PFSPP aims at providing decision-makers with the PFSP length and the shortest path in a network with picture fuzzy arc lengths. For traveling each arc, the cost parameters are assumed TPFNs. Pseudocode for this problem is proposed based on Dijkstra's methodology. Some operational laws and expected values of TPFNs and the comparison between two TPFNs by score and accuracy functions are explained.

3.1.3 A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights

(di Caprio *et al.*, 2022) This paper presents a fuzzy-based Ant Colony Optimization calculation for tackling the most limited way issues with various kinds of fuzzy loads. The outcomes affirm that the fuzzy-based improved ACO calculation could combine in around half less time than the option metaheuristic calculations. fostered a fuzzy developmental PSO calculation to decide advanced steering ways and upgrade the functional utilization of the organization. Dudeja proposed a fuzzy-based adjusted PSO calculation as a system to beat the briefest way issue with dubious edges and lessen cost and time utilization.

Gupta and Srivastava tackled the distance enhancement issue utilizing both PSO and ACO, and led a quantitative examination between their exhibitions with the recreated outcomes showing the predominance of the last advancement calculation. proposed a further developed ACO calculation for time-set off streams in time-delicate organizations. utilized a further developed non-ruled arranging hereditary calculation to plan a vigorous vehicle directing issue conspire for an independent robot route technique ready to enhance both harvest yield and quality subject to a base expense. Propose an answer technique where a hereditary calculation is combined with a variable area search.

They observe the briefest way involving the route limit of insects through the chart as dictated by the standards of the ACO calculation inside a fuzzy setting. The outcomes acquired from running reproductions of the proposed ACO calculation on a little, medium and huge size diagrams have been contrasted and those got while applying GA, PSO, and ABC calculations. The mathematical outcomes show the predominant presentation of the proposed ACO calculation as far as several emphases and combination time. In Section 4, we depict the ACO calculation proposed to take care of the most limited way issue.

3.1.4 Solving fuzzy multi-objective shortest path problem based on data envelopment analysis approach

(Bagheri *et al.*, 2021) The briefest way issue (SPP) is a unique organization organized direct programming issue that shows up in a wide scope of applications. Old style SPPs consider just a single target in the organizations while some or all of the various, clashing, and disproportionate destinations like the advancement of cost, benefit, time, distance, hazard, and nature of administration might emerge together in certifiable applications. These kinds of SPPs are known as the multi-objective briefest way issue (MOSPP) and can be addressed with the

current different methodologies. This paper fosters a Data Envelopment Analysis (DEA)-based way to deal with addressing the MOSPP with fuzzy boundaries (FMOSPP) to represent genuine circumstances where input–yield information incorporates vulnerability of three-sided enrollment structure. This way to deal with making an association between the MOSPP and DEA is more adaptable to bargain with genuinely viable applications. To this end, each curve in an FMOSPP is considered as a dynamic unit with different fuzzy data sources and results. Then, at that point, two fuzzy proficiency scores are acquired compared to each bend. These fuzzy productivity scores are consolidated to characterize an interesting fuzzy relative productivity. Consequently, the FMOSPP is changed over into a solitary goal Fuzzy Shortest Path Problem (FSPP) that can be settled utilizing existing FSPP calculations.

The principal commitments of this study are summed up as follows: (1) To the best of our

information, this review is the main endeavors for settling FMOSPP utilizing FDEA approach, (2) The utilization of the FDEA approach permits the synchronous utilization of expansion and minimization capacities in displaying the FMOSPP, (3) The proposed FDEA approach gives a proficient arrangement in changing over FMOSPP into a fresh single objective SPP, (4) rather than the existing multi-objective improvement arrangement draws near, for example, objective programming, the proposed arrangement procedure saves the design of SPP for simplicity of arrangement and implementation, (5) The FDEA approach proposed in this paper provided a proficient arrangement for the FMOSPP without unequivocal the information on the chief's fuzzy utility capacity.

3.1.5 FPGA based robotic computing

(Lin, Wu and Ma, 2021; Wan *et al.*, 2021) In this paper, we give an outline of past work on FPGA-based mechanical gas pedals covering various phases of the automated framework pipeline. robotic systems, like manipulators, legged robots, unmanned aerial vehicles, self-driving cars have been designed for search and rescue, exploration, package delivery, entertainment, and more applications and scenarios. s. Notwithstanding, conventional CPU

and GPUs generally burn through 10 W to 100 W of force, which are significant degrees higher than whatever is accessible on the asset restricted automated framework.

Other than CPU and GPU, FPGAs are standing out and turning into a stage contender to accomplish energy-effective advanced mechanics errands handling. FPGAs require little power and are regularly incorporated into little frameworks with less memory. FPGAs have been effectively used in business independent vehicles. In this paper, we survey how PerceptIn fostered its figuring framework by depending vigorously on FPGAs, which perform heterogeneous sensor synchronizations, yet in addition the speed increase of programming parts on the basic way.

Furthermore, FPGAs are utilized intensely in space automated applications, for FPGAs offered remarkable adaptability and significantly decreased the design cycle and development cost. In this paper, we additionally dive into space-grade FPGAs for robotic applications

3.1.6 Weighted Spatio-temporal taxi trajectory big data mining for regional traffic estimation

The assessment of traffic conditions in urban areas is becoming fundamental to set up a supportable transportation framework and to assist with dealing with the board specialists planning the traffic of urban communities. As of late, taxi direction enormous datasets are being gathered during cabbies are steering around the urban areas. Taxi direction datasets give conduct data about the city occupants, metropolitan progressions of the taxi travelers, and framework for traffic condition assessment. This review plans to appraise the provincial traffic speed of New York City utilizing the New York taxi direction dataset. Another strategy is suggested that utilizations weighted spatial-fleeting direction huge information mining approach and scores every area of the urban communities as far as traffic speed. Another calculation is proposed, in particular Regional Traffic Velocity Estimation calculation, which uses the proposed local spatial-transient speed assessment technique and is tentatively assessed utilizing the New York taxi direction dataset. Trial results show that every district in New York has diverse speed and utilization qualities as far as hourly and day-by-day examinations. What's more, precinct-level investigations are played out that uncover information about the

districts of New York. The assessed provincial traffic speed of urban areas dependent on taxi direction datasets would give a choice emotionally supportive network to leaders as far as local hourly and day-by-day assessment of urban areas with sans cost and boundless city traffic dataset.

Main Highlights:

- Taxi direction large dataset digging is performed for traffic assessment.
- Another strategy is suggested that finds regional spatial-worldly speed.
- Proposed technique utilizes weighted Spatio-temporal trajectory data mining.
- Another calculation is suggested that utilizations proposed strategy.
- The proposed technique is experimentally assessed on a genuine taxi direction dataset.

3.1.7 An effective heuristic clustering algorithm for mining multiple critical nodes in complex networks

Impact expansion is of incredible importance in complex organizations, and numerous techniques have been proposed to settle it. To move past the blemishes, a powerful heuristic grouping calculation is proposed in this paper: hubs that have been allocated to bunches are barred from the organization design to ensure they don't take an interest in resulting bunching. The K-shell and Neighbourhood Coreness worth of hubs in the excess organization are recalculated, which guarantees the hub impact can be changed during the grouping system. A center hub and a steering hub are chosen for each bunch to together decide the underlying spreader, which adjusts the nearby and worldwide impact. A progression of tests depending on the Susceptible–Infected–Recovered stochastic model affirm that the proposed strategy has

positive execution under various introductory imperatives against known techniques, including Vote Rank, HC, GCC, HGD, and DLS-AHC.

Highlights

• A novel heuristic clustering algorithm for influence maximization is proposed.

• The node influence will be updated during the clustering process.

• Each cluster's initial spreader is determined by the hub and routing node jointly.

• The proposed algorithm is verified on six real networks.

• Experiments show the proposed algorithm achieves the SOTA results.

4.0 Problem Statement

4.1 Problem in Dijkstra

Dijkstra's algorithm can be stated as: Find the shortest paths from a given source node to all other nodes by developing the paths in order of increasing path length. The algorithm proceeds in stages. By the kth stage, the shortest paths to the k nodes closest to the source node have been determined; these nodes are in a set T. At a stage, the node not in T that has the shortest path from the source node is added to T. As each node is added to T, its path from the source is defined,

Brief working of the Dijkstra's Algorithm:

(1) The set of nodes so far incorporated (T) consists of only the source node (s), and the initial path costs (w) to neighboring nodes are simply the link costs (L): T = {s}

L (node) = w (s, node) for node / = s. (1)

(2) The neighboring node not in T that has the least-cost path from node s is found and then that node is incorporated into T. Also incorporated is the edge that is incident on that node and a node in T that contributes to the path

Find $x \notin T$ such that $L(x) = \min_{j \notin T} L(j)$. (2)

Add x to T along with the edge that is incident on x and that contributes the least-cost component to $L(x)$.

(3) A comparison is made between the path cost from node s to any node in the network and the summation of path costs from node s to node x (x is another node in the network) and the link cost between node x and the node considered, and the minimum is chosen:

$L(\text{node}) = \min [L(\text{node}), L(x) + w(x, \text{node})] \forall \text{ node} \notin T$. (3)

(4) Steps (2) and (3) are repeated until final paths have been assigned to all nodes in the network. The algorithm ends when all nodes have been taken. The running time of this algorithm is $O(N2)$ where N is the number of nodes in the network.

4.2 Example Illustration

Let us discuss Dijkstra's algorithm. Dijkstra requires the input graph to have only one source node. It also requires that all the edges in the graph have non-negative weights. Suppose we are given a map such as the one below and wish to travel from B to other nodes. In a simple graph like this, we can see that there are several paths to travel from B to other nodes.

STEP 1

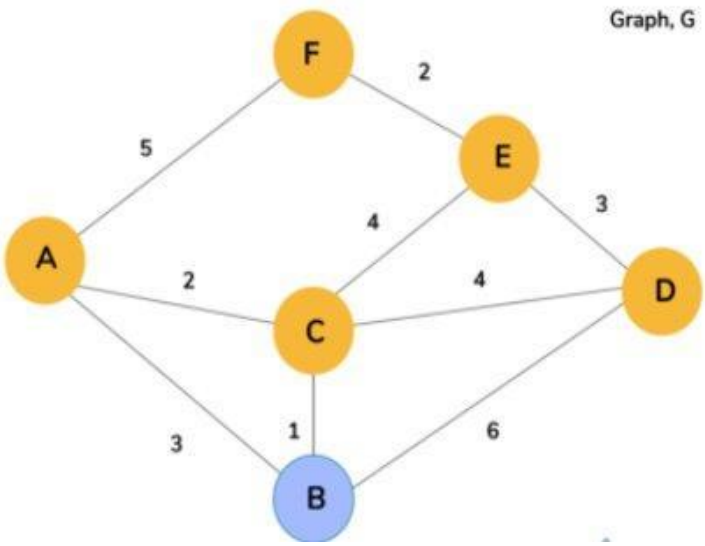


Fig.2 Network of cities

· Source node = 0 (cost of visiting self is zero)

B	A	C	D	E	F
0	∞	∞	∞	∞	∞

Step 2

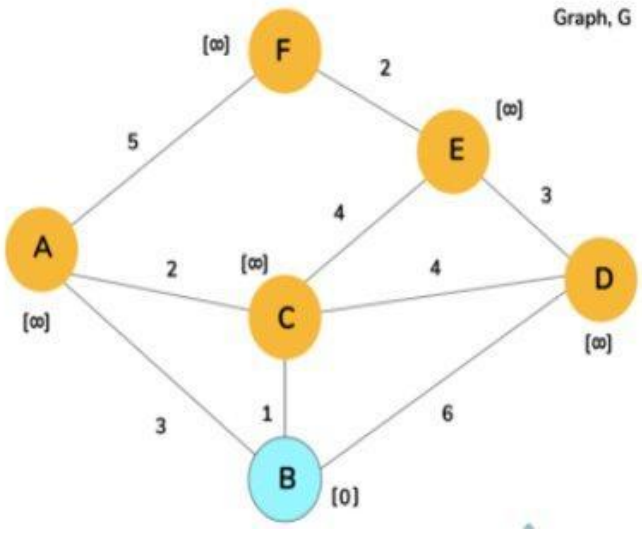


Fig.3 Dijkstra operating on the graph

B	C	A	D	E	F
0	∞	∞	∞	∞	∞
0 [VISITED]	1	3	6	∞	∞

- For neighbor A: cost = Minimum (∞, 0+3) = 3
- For neighbor C: cost = Minimum (∞, 0+1) = 1
- For neighbor D: cost = Minimum (∞, 0+6) = 6

Step 3

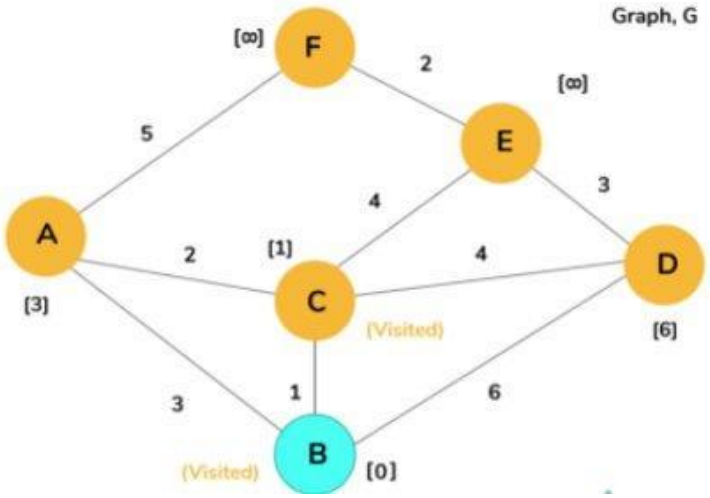


Fig.4 Dijkstra marking processed nodes as visited

- For neighbor A: cost = Minimum (3, 2+1) = 3
- For neighbor E: cost = Minimum (∞, 1+4) = 5
- For neighbor D: cost = Minimum (6, 1+4) = 5

B	C	A	D	E	F
0	∞	∞	∞	∞	∞
0 [VISITED]	1	3	6	∞	∞
0 [VISITED]	1 [VISITED]	3	(5<6) => 5	5	∞

Step 4

- For neighbor F: cost = Minimum (∞, 3+5) = 8

B	C	A	D	E	F
0	∞	∞	∞	∞	∞
0 [VISITED]	1	3	6	∞	∞
0 [VISITED]	1 [VISITED]	3	5	5	∞
0 [VISITED]	1 [VISITED]	3[VISITED]	5	5	8

Step 5

- For neighbor E: cost = Minimum (5, 5+3) = 5

B	C	A	D	E	F
0	∞	∞	∞	∞	∞
0 [VISITED]	1	3	6	∞	∞
0 [VISITED]	1 [VISITED]	3	5	5	∞
0 [VISITED]	1 [VISITED]	3 [VISITED]	5	5	8
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5	8

Step 6

- For neighbor F: cost = Minimum (8, 5+2) = 7

B	C	A	D	E	F
0	∞	∞	∞	∞	∞

0 [VISITED]	1	3	6	∞	∞
0 [VISITED]	1 [VISITED]	3	5	5	∞
0 [VISITED]	1 [VISITED]	3 [VISITED]	5	5	8
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5	8
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5 [VISITED]	7

Step 7(Last step)

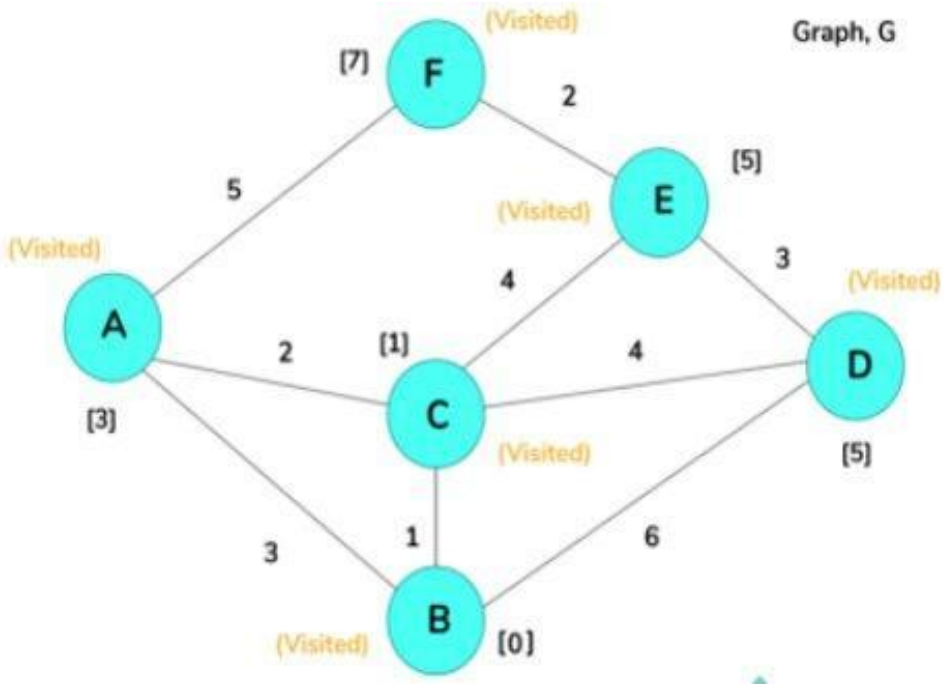


Fig.5 Final graph after Dijkstra finished processing

- Visiting the final node

B	C	A	D	E	F
0	∞	∞	∞	∞	∞
0 [VISITED]	1	3	6	∞	∞
0 [VISITED]	1 [VISITED]	3	5	5	∞
0 [VISITED]	1 [VISITED]	3 [VISITED]	5	5	8
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5	8
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5 [VISITED]	7
0 [VISITED]	1 [VISITED]	3 [VISITED]	5 [VISITED]	5 [VISITED]	7 [VISITED]

5.0 Proposed method

Yong Deng, Yuxin Chen, Yajuan Zhang, and Sankaran Mahadevan inspire this project mostly from the work in (Deng *et al.*, 2012) where they explained the fuzzy shortest path problem in a detailed manner, also in (Deng *et al.*, 2012) the old-style Dijkstra calculation is summed up given the canonical representation of procedure on triangular fuzzy numbers to deal with the fuzzy shortest path problem. The standard representation of procedure on triangular fuzzy numbers that depend on the graded mean integration representation technique prompts the outcome multiplication and the addition of two fuzzy numbers can be addressed as a crisp number.

fuzzy rationale is a heuristic methodology that takes into account further developed choice tree handling and better coordination with rules-based programming. Fuzzy rationale is a speculation from standard rationale, wherein all assertions have a reality worth of one or zero. In fuzzy rationale, explanations can have a worth of fractional truth, for example, 0.9 or 0.5. Hypothetically, this offers the methodology more chance to mirror genuine conditions, where proclamations of essential fact of the matter or lie are uncommon.

Let us begin by first briefly introducing the fuzzy set theory (Porchelvi and Banumathy, no date; Clement Joe Anand, Bharatraj and Nadu, 2017; Mishra, 2019). The key terms that need to be understood are fuzzy sets, fuzzy numbers, membership functions, triangular fuzzy numbers, and the graded mean integration representation of the triangular fuzzy numbers.

"**Fuzzy set**": Let X be a universe of discourse. Where $\tilde{A}$ is a fuzzy subset of X; and for all $x \in X$, there is a number $\mu_{\tilde{A}}(x) \in [0, 1]$, which is assigned to represent the membership degree of x in $\tilde{A}$, and is called the membership function of $\tilde{A}$.

"**Fuzzy number**": A fuzzy number $\tilde{A}$ is a normal and convex fuzzy subset of X.

Here, "normality" implies that:

$\exists x \in R, \vee x \mu_{\tilde{A}}(x) = 1$ (1) and

"Convex" means that:

$\forall x1 \in X, x2 \in X, \forall \lambda \in [0, 1]$,

$\mu_{\tilde{A}}(\alpha x1 + (1 - \alpha) x2) \geq \min(\mu_{\tilde{A}}(x1), \mu_{\tilde{A}}(x2))$

A **triangular fuzzy** number $\tilde{A}$ can be defined by a triplet (a, b, c), where the membership can be determined as follows.

General Equation for membership function of triangular fuzzy numbers

$$\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & x > c \end{cases}$$

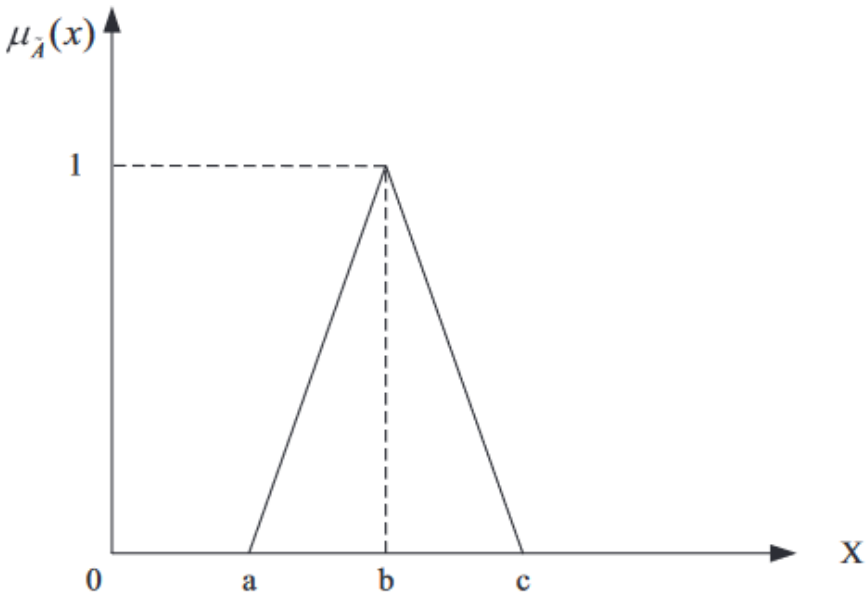


Fig.6 Representation of a triangular fuzzy number

Given a triangular fuzzy number ˜A = (a1, a2, a3), **the graded mean integration representation** of triangular fuzzy number˜A is defined as:

$$P(\tilde{A}) = 1/6\,(a1 + 4 \times a2 + a3) \quad (1)$$

Let $\tilde{A} = (a1, a2, a3)$ and $\tilde{B} = (b1, b2, b3)$ be two triangular fuzzy numbers. By applying Eq. (1), the graded mean integration representation of triangular fuzzy numbers $\tilde{A}$ and $\tilde{B}$ can be obtained, respectively, as follows:

$$P(\tilde{A}) = 1/6\,(a1 + 4 \times a2 + a3)$$

$$P(\tilde{B}) = 1/6\,(b1 + 4 \times b2 + b3)$$

The representation of the addition operation $\oplus$ on triangular fuzzy numbers $\tilde{A}$ and $\tilde{B}$ can be defined as:

$$P(\tilde{A} \oplus \tilde{B}) = P(\tilde{A}) + P(\tilde{B}) = 1/6\,(a1 + 4 \times a2 + a3) + 1/6\,(b1 + 4 \times b2 + b3) \quad (2)$$

The canonical representation of the multiplication operation on triangular fuzzy numbers $\tilde{A}$ and $\tilde{B}$ is defined as:

$$P(\tilde{A} \otimes \tilde{B}) = P(\tilde{A}) \times P(\tilde{B}) = 1/6\,(a1 + 4 \times a2 + a3) \times 1/6\,(b1 + 4 \times b2 + b3)$$

fuzzy logic is widely regarded as it offers a mechanism that is used a lot, which is inferencing i.e. how can one use existing data to infer new knowledge (*Machine Intelligence - Lecture 17 (Fuzzy Logic, Fuzzy Inference) - YouTube*, no date) explains that the fuzzy logic was supposed to be computing with the words, although computers today are not there yet and those cannot process linguistic variables.

Linguistic variables, the variables, whose values are not numbers instead sentences or words in an artificial or natural language as said by L.A Zadeh.

the linguistic variable is characterized by a quintuple: (L>, *T*(*L*), *U*, *G*, *M*)

in which L is the name of the variable;

T(*L*) is the *term-set* of L, that is, the collection of its linguistic values;

U is a universe of discourse;

G is a *syntactic rule* which generates the terms in *T*(*L*);

and *M* is a *semantic rule* which associates with each linguistic value *X* its *meaning*,

M(X), where *M(X)* denotes a fuzzy subset of *U*.

Usually working with variables is like having a variable name that has a value but linguistic variables are more complicated, that is so because it mimics the way the human mind works as humans convey a lot of information in addition to the properties such as ambiguous and compressed nature, for example, “John is not that old” there is a significant amount of vague information in the sentence what does “not that” suppose to mean, it is not accessible to computer because the computer cannot understand this ambiguity (Zadeh, 1975).

5.1 What exactly was done?

In the proposed method the edge weight is fuzzified before the computation to the perception of the crisp number corresponding to the real world (Malczewski, 2002) shows a good work on fuzzy screening and describes the linguistic variable in good detail inspiring from the work, in the method the edge weight distance is considered as a linguistic variable takes the values “short”, “medium”, and “long”. A linguistic value is characterized by a label and a semantic value). The label is a word or sentence belonging to a linguistic term set and the meaning is a fuzzy subset is defined in a relevant interval, which is described by a membership function. (Asadzadeh *et al.*, 2017) The idea of linguistic variable is based on operationalizing evaluation

criteria and inclinations, additionally for the collection methodology in the fuzzy screening techniques.

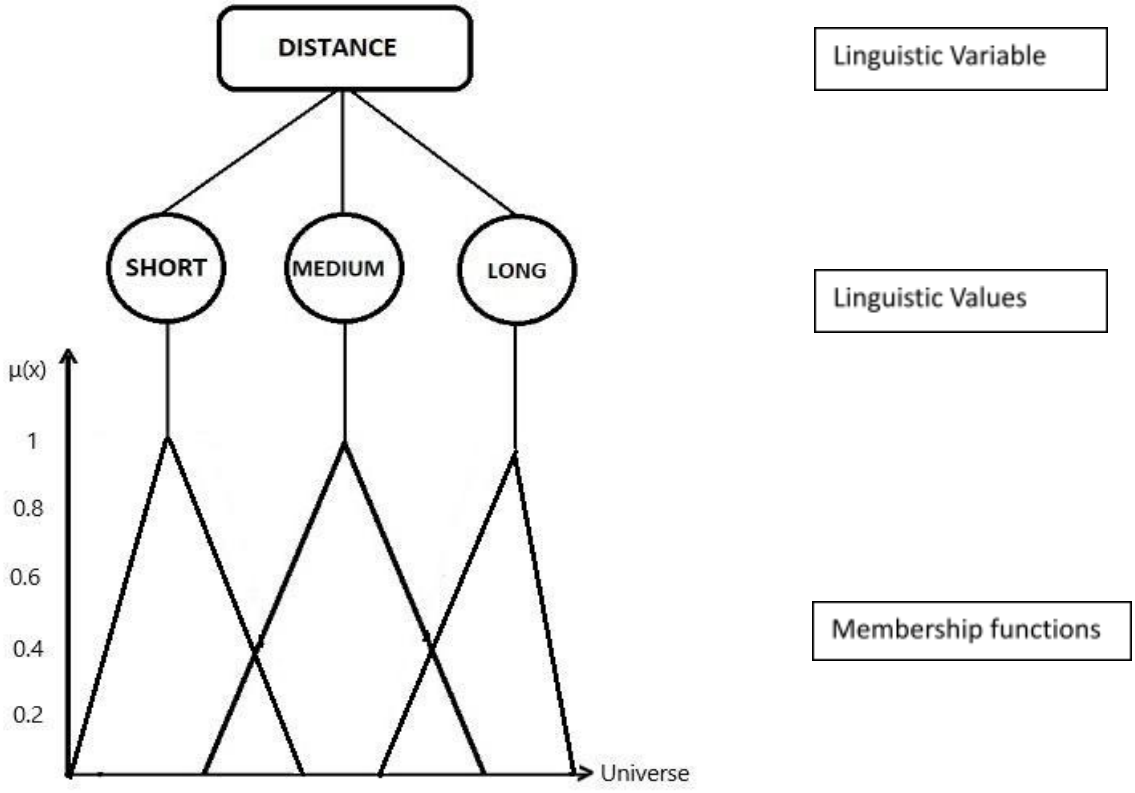
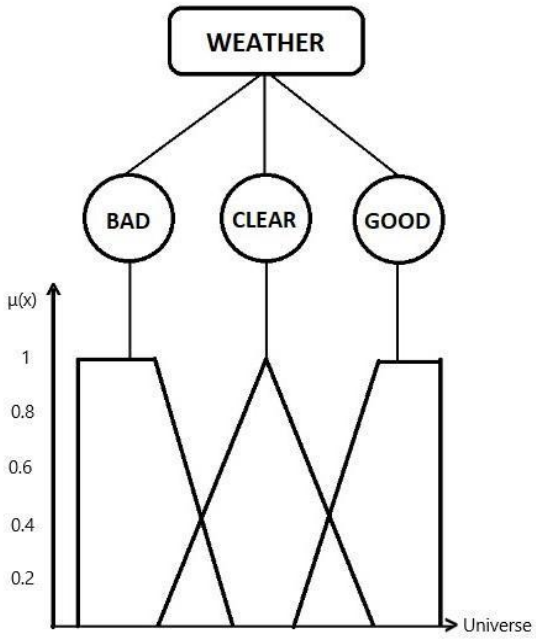


Fig.7 Illustration of linguistic variable "distance"



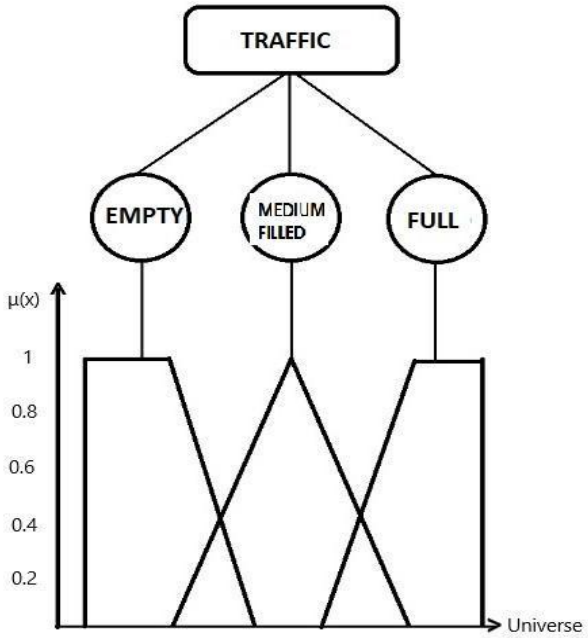
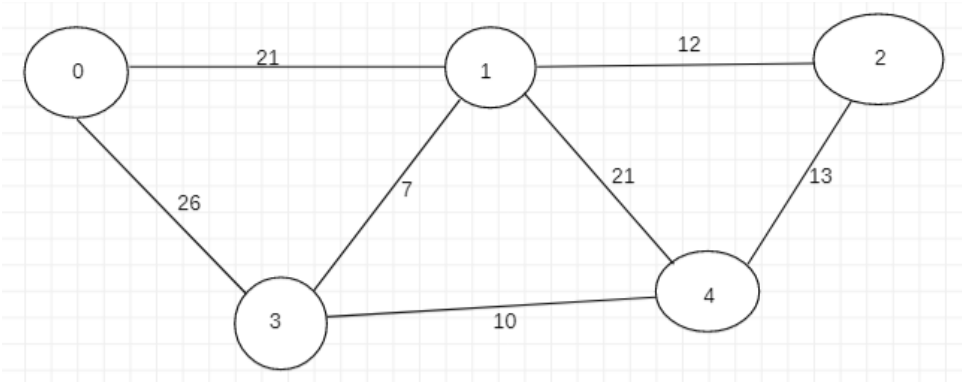


Fig.8 Illustration of linguistic variable "weather"

Fig.9 Illustration of linguistic variable "traffic"

Figure 7, 8, 9 illustrates the linguistic variables with their respective linguistic values that are given meaning by the membership functions in a defined interval. Figure 7 illustrates three linguistic values short, medium, and long in the universe of discourse; also, all the membership functions are triangular functions hence the interval defined tries to broadly mimic the human perception of long-short or medium distances.

5.2 Example Illustration



$$\mu_{\tilde{A}}(x) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & x > c \end{cases}$$

For this member function and this graph, we have 3 categories short, medium, and long

- For short a=1, b=5, c=10
- For medium a=11, b=15, c=20
- For long a=21, b=25, c=30

Adjency matrix for this graph will be

0,0,0	21,0,3	0,0,0	26,0.8,3	0,0,0
21,0,3	0,0,0	12,0.25,2	7,0.6,1	22,0.25,2
0,0,0	12,0.5,2	0,0,0	0,0,0	13,0.5,2
26,0.8,3	7,0.6,1	0,0,0	0,0,0	10,0,1
0,0,0	22,0.25,2	13,0.5,2	10,0,1	0,0,0

5.3 Fuzzy based Solution

Three main values we need are distance value for each edge, degree of belongingness i.e., membership value, and linguistic variable. So, we merge all these values and save them in 3D array ARRAY[From which node][To which node][To navigate between the above 3 values]. Initially, we make an adjacency matrix with the distance value saved in them and initialize the degree and linguistic variable as 0. Multiple functions like membership fun () and labeling () to assign the values of degree and linguistic variable. By calling the function dijkstrapro() we start the main computation. In dijkstrapro() we assume 2 different arrays for bool visited[all elements are initialized as false] and int distance[maximum values to all elements]. So as we move forward we keep marking nodes visited or not using visited [] array and saving the distance in distance [] array. We decide the next node based on linguistic variable and degree using the function next_node_selection () and in the last, as we have visited all the nodes in last we print the node and the distance to it.

6.0 Evaluation

6.1 Explanation and working w.r.t fuzzy

```
52  float membershipfunc(int y)
53  {
54      float a,b,c,x=y;
55      if(x<1 || x>30)
56          return 0;
57
58      else if(x>=1 && x<=10){                 //short
59          a=1,b=5,c=10;
60          if(x>=a && x<=b)
61              return (x-a)/(b-a);
62          else if(x>b && x<=c)
63              return (c-x)/(c-b);
64      }
65
66      else if(x>=11 && x<=20){                //medium
67          a=11,b=15,c=20;
68          if(x>=a && x<=b)
69              return (x-a)/(b-a);
70          else if(x>b && x<=c)
71              return (c-x)/(c-b);
72      }
73
74
75      else if(x>=21 && x<=30){                //far
76          a=21,b=25,c=30;
77          if(x>=a && x<=b)
78              return (x-a)/(b-a);
79          else if(x>b && x<=c)
80              return (c-x)/(c-b);
81      }
82  }
```

```
83
84  int labelfunc(int x){
85      if(x>=1 && x<=10)
86          return 1;
87      else if(x>=11 && x<=20)
88          return 2;
89      else if(x>=21 && x<=30)
90          return 3;
91      else
92          return 0;
93  }
94
95  void printdistance(float graph[][6][3],int distance[]){
96      cout<<"\nNode\tDistance from source Node\n";
97      for(int i=0;i<6;i++){
98          cout<<i<<"\t\t"<<distance[i]<<endl;
99      }
100 }
101
102 int next_node_selection(float graph[6][6][3],int i,bool visited[],int distance[]){
103         int minlabel = 3,maxmembervalue = 0,nextnodeindex=0,j=0;
104         for(j=0;j<6;j++){
105             if(!visited[j] && graph[i][j][0]!=0 && distance[j]>=(distance[i] + graph[i][j][0])){
106                 distance[j]=distance[i] + graph[i][j][0];
107                 if(minlabel >= graph[i][j][2] && maxmembervalue<graph[i][j][1]){
108                     nextnodeindex = j;                                          //to which node
109                     maxmembervalue  = graph[i][j][1];                           //membership value
110                     minlabel = graph[i][j][2];                                  //linguestic variable
111                 }
112             }
113         }
114         visited[i]=true;
115         if(nextnodeindex == 0){
116             for(int k=0;k<6;k++){
117                 if(!visited[k])
118                     nextnodeindex=k;
119             }
120         }
121     cout<<endl<<i<<","<<nextnodeindex<<endl;
122     return nextnodeindex;
123 }
124
```

```
124
125 void dijkstrapro(float graph[6][6][3],int initial_node){
126     int i = initial_node,distance[6]={INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX,INT_MAX};
127     bool visited[6] = {false,false,false,false,false,false},done=false;
128     distance[i]=0;
129     while(!done){
130         for(int k=0;k<6;k++){
131             if(!visited[k])
132                 break;
133             else if(k==5)
134                 done=true;
135         }
136         i=next_node_selection(graph,i,visited,distance);
137     }
138     printdistance(graph,distance);
139 }
140
```

```
141  int main()
142  {
143      float graph[6][6][3]={
144          { {0,0,0},{27,0,0},{14,0,0}, {0,0,0}, {0,0,0}, {0,0,0}},
145          {{27,0,0}, {0,0,0}, {0,0,0},{20,0,0}, {7,0,0}, {0,0,0}},
146          {{14,0,0}, {0,0,0}, {0,0,0},{15,0,0},{10,0,0}, {0,0,0}},
147          { {0,0,0},{20,0,0},{15,0,0}, {0,0,0},{28,0,0},{22,0,0}},
148          { {0,0,0}, {7,0,0},{10,0,0},{28,0,0}, {0,0,0}, {1,0,0}},
149          { {0,0,0}, {0,0,0}, {0,0,0},{22,0,0}, {1,0,0}, {0,0,0}}};
150      for(int i=0;i<6;i++){
151          for(int j=0;j<6;j++){
152              graph[i][j][1] = membershipfunc(graph[i][j][0]) ;
153              graph[i][j][2] = labelfunc(graph[i][j][0]);
154          }
155      }
156      for(int l=0;l<6;l++){
157          for(int m=0;m<6;m++){
158              for(int n=0;n<3;n++){
159                  cout<<graph[l][m][n]<<",";
160              }
161              cout<<"  ";
162          }
163          cout<<endl;
164      }
165      dijkstrapro(graph,0);
166
167      return 0;
168  }
169
170
171
```

```
C:\Users\omgill\Desktop\temp.exe
14,0.75,2,  0,0,0,  0,0,0,  15,1,2,  10,0,1,  0,0,0,
0,0,0,  20,0,2,  15,1,2,  0,0,0,  28,0.4,3,  22,0.25,3,
0,0,0,  7,0.6,1,  10,0,1,  28,0.4,3,  0,0,0,  1,0,1,
0,0,0,  0,0,0,  0,0,0,  22,0.25,3,  1,0,1,  0,0,0,

0,2

2,3

3,5

5,4

4,1

1,0

0,0

Node    Distance from source Node
0               0
1               27
2               14
3               29
4               24
5               51

--------------------------------
Process exited after 0.1659 seconds with return value 0
Press any key to continue . . .
```

7.0 Conclusion

In this project the single linguistic variable i.e., distance is utilized for the computation of the next most suitable node using the fuzzy logic by comparing linguistic values and their corresponding degree of belongingness, which is defined by the value given by the membership function. It ought to be called attention to that the uncertainty in the shortest path the issue isn't restricted to the mathematical distance. For instance, due to the climate and other startling elements, the movement time from one city to one more city might be addressed as a fuzzy variable, even on the off chance that the mathematical distance is fixed. Thus, further efforts are required to include those variables in the node computation process and hence this project shall be extended in the future to develop a suitable technique that can appropriately address those variables and help induce real-world perception even more in the process.

8.0 References

[1] Akram, M., Habib, A. and Alcantud, J.C.R. (2021) "An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers," *Neural Computing and Applications*, 33(4), pp. 1329–1342. doi:10.1007/s00521-020-05034-y.

[2] Asadzadeh, A. *et al.* (2017) "Operationalizing a concept: The systematic review of composite indicator building for measuring community disaster resilience," *International Journal of Disaster Risk Reduction*, 25, pp. 147–162. doi:10.1016/J.IJDRR.2017.09.015.

[3] Bagheri, M. *et al.* (2021) "Solving fuzzy multi-objective shortest path problem based on data envelopment analysis approach," *Complex & Intelligent Systems*, 7(2), pp. 725–740. doi:10.1007/s40747-020-00234-4.

[4] di Caprio, D. *et al.* (2022) “A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights,” *Alexandria Engineering Journal*, 61(5), pp. 3403–3415. doi:10.1016/j.aej.2021.08.058.

[5] Clement Joe Anand, M., Bharatraj, J. and Nadu, T. (2017) *Theory of Triangular Fuzzy Number*. Available at: https://www.researchgate.net/publication/318946539.

[6] Deng, Y. *et al.* (2012) “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment,” *Applied Soft Computing Journal*, 12(3), pp. 1231–1237. doi:10.1016/j.asoc.2011.11.011.

[7] Lin, L., Wu, C. and Ma, L. (2021) “A genetic algorithm for the fuzzy shortest path problem in a fuzzy network,” *Complex & Intelligent Systems*, 7(1), pp. 225–234. doi:10.1007/s40747-020-00195-8.

[8] Machine Intelligence - Lecture 17 (Fuzzy Logic, Fuzzy Inference) - YouTube (no date). Available at: https://www.youtube.com/watch?v=TReelsVxWxg (Accessed: January 7, 2022).

[9] Malczewski, J. (2002) “Fuzzy screening for land suitability analysis,” *Geographical and Environmental Modelling*, 6(1), pp. 27–39. doi:10.1080/13615930220127279.

[10] Mishra, K.K. (2019) “Dijkstra’s algorithm for solving fuzzy number shortest path problems,” *Malaya Journal of Matematik*, S(1), pp. 714–719. doi:10.26637/MJM0S01/0127.

[11] Porchelvi, R.S., and Banumathy, & A. (no date) *GENERALIZED DIJKSTRA’S ALGORITHM FOR SHORTEST PATH PROBLEM IN INTUITIONISTIC FUZZY ENVIRONMENT*.

[12] “Prentice Hall - Computer Networks Tanenbaum 4ed” (no date).

[13] Wan, Z. *et al.* (2021) “A Survey of FPGA-Based Robotic Computing,” *IEEE Circuits and Systems Magazine*, 21(2), pp. 48–74. doi:10.1109/MCAS.2021.3071609.

[14] Zadeh, L.A. (1975) “The concept of a linguistic variable and its application to approximate reasoning—I,” *Information Sciences*, 8(3), pp. 199–249. doi:10.1016/0020-0255(75)90036-5.