

Probing a Fuzzy Shortest Path in a Network

A PROJECT REPORT

Submitted in partial fulfillment for the award of the degree of

B.TECH

in

Information Technology

by

**ADITYA RAJ (11BIT0155)
ARUN RAVINDRAN (11BIT0035)
AFZAL AHMAD (11BIT0038)**

**Under the Guidance of
Prof. MENAKA S.**



School of Information Technology & Engineering

NOVEMBER 2014

DECLARATION BY THE CANDIDATE

I here by declare that the project report entitled "**Probing a Fuzzy Shortest Path in a Network**" submitted by me to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech.(Information Technology)** is a record of bonafide project work carried out by me under the guidance of **Prof. Menaka S..** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Signature of the Candidate

Date: 07-Nov-2014

Aditya Raj (11BIT0155)



School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the project report entitled **Probing a Fuzzy Shortest Path in a Network** submitted by **Aditya Raj (11BIT0155)** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the degree of **B.Tech.(Information Technology)** is a record of bonafide work carried out by him under my guidance. The project fulfills the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Prof. Menaka s.

SUPERVISOR

Assistant Professor(Senior), SITE

Prof.R.K.Nadesh

Programme Chair

B.Tech (IT)

Internal Examiner –

External Examiner –

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I am grateful to Chancellor –Dr. G.Viswanathan, Vice Chancellor Dr V.Raju, Pro-Vice Chancellor Dr. S. Narayanan, Dean of SITE Dr. K. Ganeshan, Program Chair Prof. R.K. Nadesh, Year Coordinator Prof. Varalakshmi, Internal guide Prof. Menaka s. for the guidance, inspiration and constructive suggestions that helps me in the preparation of this project entitled- **"Probing a Fuzzy Shortest Path in a Network"**.

Submitted By :

Aditya Raj

11BIT0155

LIST OF FIGURES

<u>Figure No.</u>	<u>Title</u>	<u>Page No.</u>
Fig. 6.1	Work Breakdown Structure	13
Fig. 4.1.1	High Level Design	15
Fig. 4.2.1	Low Level Design	16
Fig. 6.1	Result/Output	23-24
Fig. 6.2	Results analysis	25

CHAPTER 1

INTRODUCTION

1.1 Background

The shortest path problem is a quandary of finding the shortest path or route from a starting point to the destinations connected to it with the objective of minimizing the cost of the route. Hence the shortest path problem can be formulated as min-cost flow problem. This formulation of shortest path problem is unbounded linear programs if negative cycles are present which is NP-hard. Various techniques such as Dijkstra's algorithm, Bellman-Ford algorithm, Floyd Warshall's Algorithm and Genetic Algorithm were proposed and proved to solve this type of problem in the current literature.

1.2 Problem Statement

This project proposes an algorithm to discover the shortest path connoted on fuzzy logic. Our aim is to design an efficient algorithm that addresses the problem of routing packets given the nodes in a network and the source and destination. So we basically try to address issues like performance, space complexity, time complexity and maintainability.

1.3 Importance

Various techniques such as Dijkstra's algorithm, Bellman-Ford algorithm, Floyd Warshall's Algorithm and Genetic Algorithm were proposed and proved to solve this type of problem in the current literature. Yet, we opted for fuzzy logic because of the advantages mentioned below. Fuzzy logic is a powerful, yet straightforward, problem-solving technique with widespread applicability, especially in the areas of control and decision making. In general, it is most useful in handling problems not easily definable by practical mathematical models. Fuzzy derives much of its power from its ability to draw conclusions and generate responses based on vague, ambiguous, qualitative, incomplete, or imprecise information. In this respect, fuzzy-based systems have a reasoning ability similar to that of humans. In fact, the behaviour of a fuzzy system is represented in a very simple and natural way. This allows quick construction of

understandable, maintainable, and robust systems. In addition, a fuzzy approach generally requires much less memory and computing power than conventional methods, thereby permitting smaller and less expensive systems.

1.4 Organization Of report

The report contains tables, figures and descriptive content arranged in well-ordered fashion. Each chapter has its own significance. Chapter 1 presents the background, problem statement and importance of this application. Chapter 2 gives the overview of the application and challenges faced during the development. Also it gives the details of hardware and software Requirements. Chapter 3 presents the literature survey for this Project. Chapter 4 gives the high-level and low-level design of the application. Chapter 5 contains code and details of unit and integration testing. Chapter 6 analyzes the results and discussion about the results. Chapter 7 is conclusion and future work for this Application. Lastly, Chapter 8 contains references to the helpful articles.

CHAPTER 2

OVERVIEW AND PLANNING

2.1 Proposed System Overview

Proposed system is the shortest path predictor based on the given nodes in a network and the source and destination. These parameters along with congestion parameters when taken as input from routing table (or say matrix) are enough to compute the shortest path out of the available paths. The user can enter the parameters like node existence in terms of binary, membership functions as positive integers and source to destination parameters as positive whole numbers. These parameters along with congestion parameters (positive whole numbers) when taken as input from routing table (or say matrix) are enough to compute the best path out of the available paths.

2.2 Challenges

The first challenge is to find the paths available between the given source and destination. Then we need to calculate parameters to feed in mathematical expressions. There may be several approaches to do the job but we aimed to reduce the run time and compile time complexity by avoiding unnecessary comparisons and over-looping.

2.3 Assumptions

Routing of packets while we go online is one of the basic criterion to be dealt efficiently and effectively. Such intensive task and interaction can be viewed as a manipulation of matrix drawn in terms of 0 and 1. So we hold the assumption that retrieving information like membership function parameters can be pre-defined (currently based on users input). These parameters can vary dynamically and thus the algorithm has to be modified a bit when implementing in real world.

2.4 System Architecture

Understanding Fuzzy logic to compute in C: Fuzzy sets are an extension of crisp (two valued) sets to handle the concept of partial truth, which enables the modelling of uncertainties of natural language. Different to classical sets, elements of a fuzzy set have membership degrees to that set. The degree of membership to a fuzzy set indicates the certainty (or uncertainty). With universe set X , Fuzzy set S is a subset of X that:

$$S = \{ (x, \mu_S(x)) \mid x \in X \wedge \mu_S \in [0,1] \}$$

Consider that we need to evaluate a fuzzy condition (the membership of x in a fuzzy set F). Initially, the result is the empty fuzzy set 0 , i.e., $\mu_S(x) \equiv 0$. Then, based on the membership, ξ , of x in F , we union R_1 with the and with ξ ; and based on the negation of that membership, we union R_2 with the and with $\sim\xi$:

```
fuzzy_set S = 0; // S is the 0 (empty) fuzzy set
if (μF(x)) {
    S = S ∪ (μF(x) ∩ R1);
}
else {
    S = S ∪ (∼μF(x) ∩ R2);
}
```

Here:

- \sim is the fuzzy not operation (usually $\sim\xi = 1 - \xi$),
- Any fuzzy value ξ may be interpreted as the fuzzy set with a constant member function $\mu_S(x) \equiv \xi$,
- The fuzzy union of two fuzzy sets S and T , $S \cup T$, is the fuzzy set with member function $\mu_{S \cup T}(x) = \mu_S(x) \vee \mu_T(x)$ (usually $= \max(\mu_S(x), \mu_T(x))$), and
- The fuzzy intersection of two fuzzy sets S and T , $S \cap T$, is the fuzzy set with member function $\mu_{S \cap T}(x) = \mu_S(x) \wedge \mu_T(x)$ (usually $= \min(\mu_S(x), \mu_T(x))$).

- In this case, it would appear obvious that both bodies must be executed. This information must be accessible within the statements within the executed body. To solve this, we will introduce a variable this fuzzy.

Terms involved:

1. Triangular Fuzzy Number:

It is a fuzzy number represented with three points as follows: $A = (a_1, a_2, a_3)$. This representation is interpreted as membership functions and holds the following conditions:

- (i) a_1 to a_2 is increasing function
- (ii) a_2 to a_3 is decreasing function
- (iii) $a_1 \leq a_2 \leq a_3$

$$\mu_s(x) = \begin{cases} 0 & \text{for } x < a_1 \\ (x-a_1)/(a_2-a_1) & \text{for } a_1 \leq x \leq a_2 \\ (a_3-x)/(a_3-a_2) & \text{for } a_2 \leq x \leq a_3 \\ 0 & \text{for } x > a_3 \end{cases}$$

2. Similarity degree:

If A and B are two points in the universe X then there exists a suitable non decreasing function g such that, $\delta(A, B) = g[d(A, B)]$. Similarly in case of set theoretic similarity measures it is observed that crisp transitivity is a much stronger condition to be put upon similarity measure.

$$S(\bar{L}_i, \tilde{L}_\mu^{\min}) = \begin{cases} 0 & , \bar{L}_i \cap \tilde{L}_\mu^{\min} = \varnothing \\ \frac{100(c - a_i)^2}{2(c_i - a_i)[(c - b) + (b_i - a_i)]} & , \bar{L}_i \cap \tilde{L}_\mu^{\min} \neq \varnothing \end{cases}$$

$$S(\bar{L}_i, \tilde{L}_\gamma^{\max}) = \begin{cases} 0 & , \bar{L}_i \cap \tilde{L}_\gamma^{\max} = \varnothing \\ \frac{100(c - a_i)^2}{2(c_i - a_i)[(c - b) + (b_i - a_i)]} & , \bar{L}_i \cap \tilde{L}_\gamma^{\max} \neq \varnothing \end{cases}$$

Triangular intuitionistic fuzzy number

For two intuitionistic fuzzy path length:

$$L1 = [(a1, b1, c1)(l1, m1, n1)]$$

$$L2 = [(a2, b2, c2)(l2, m2, n2)]$$

a) For membership function,

$$\diamond L_{\min} = (a, b, c)$$

$$\diamond a = \min(a1, a2)$$

$$\diamond b = \begin{cases} \min(b1, b2) & \text{if } (\min(b1, b2) \leq \max(a1, a2)) \\ [(b1 * b2) - (a1 * a2)] / [(b1 + b2) - (a1 + a2)] & \text{if } (\min(b1, b2) > \max(a1, a2)) \end{cases}$$

$$\diamond c = \min[(\min(c1, c2), \max(b1, b2))]$$

b) For non-membership function,

$$\diamond L_{\max} = (l, m, n)$$

$$\diamond l = \min(l1, l2)$$

$$\diamond m \left[\begin{array}{ll} \min(m_1, m_2) & \text{if}(\max(m_1, m_2) > \min(l_1, l_2)) \\ [(m_1 * m_2) - (l_1 * l_2)] / [(m_1 + m_2) - (l_1 + l_2)] & \text{if}(\max(m_1, m_2) \leq \min(l_1, l_2)) \end{array} \right]$$

$$\diamond n = \max[(\max(n_1, n_2), \min(m_1, m_2))]$$

An ALGORITHM to find out the shortest path:

1. Find out all the possible paths from Source node S to Destination node D and compute the corresponding path lengths L_i , $i = 1, 2, \dots, n$.
2. Compute L_{min} by using fuzzy shortest path length procedure.
 - a) Compute all the possible paths Lengths L_i from $i = 1, 2 \dots n$ where $L_i = (a_1', b_1', c_1')$.
 - b) Initialize $L_{min} = (a, b, c) = L_1 = (a_1', b_1', c_1')$.
 - c) Compute using the membership function mentioned above.
 - d) Set $L_{min} = (a, b, c)$ as calculated in step 4.
 - e) $++i$.
 - f) If $i < n+1$, goto 4
3. Find the Euclidean distance d_i for $i=1, 2 \dots n$ between all the possible path and L_{min} .
4. Decide the shortest path with the path having lowest Euclidean distance.

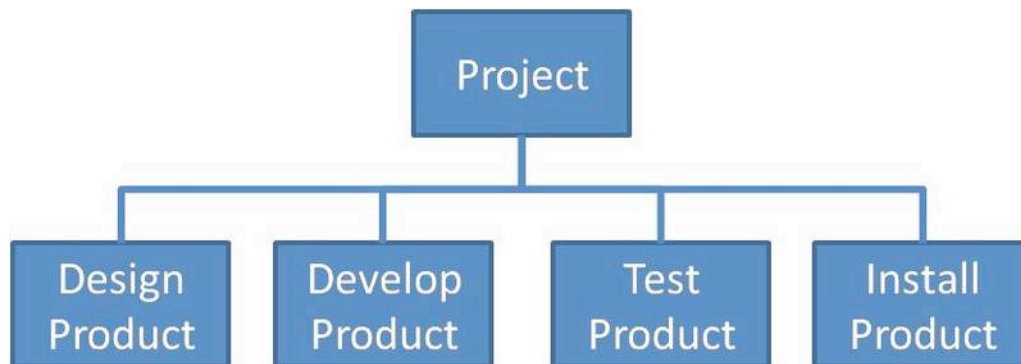
2.5 Hardware Requirements

- Processor- Intel i3 or above
- Operating System- Windows 7 or above
- Processor- Intel i3 or above
- RAM- 1GB
- Disk Space-4GB

2.6 Software Requirements

- Xcode
- DevCpp
- Eclipse

2.7 Work Breakdown Structure



CHAPTER 3

LITERATURE SURVEY AND REVIEW

3.1 Literature Survey

The project to find the shortest path or route from source to destination has been a quite interesting challenge for researchers. Several possible ways to find the shortest path have been proposed. But algorithms are selected to suit the pre-defined conditions. Variables known may vary from one problem to another.

All prior studies on path finding and path planning behaviour assumed that all required spatial information was available but in real life navigators deal with incomplete or imprecise spatial knowledge resulting in spatial uncertainties. Wiener et al. in 2008 presented experiments studying path planning under spatial uncertainties. In those researches a hierarchical planning scheme was used to navigate the shortest possible path to find an object hidden in one of four places and to bring it to the final destination. Mao in 2008 designed and compared different path finding algorithms for a graph with weighted edges. Nikolova et al. in 2006 presented new complexity results and efficient algorithms for optimal route planning in the presence of uncertainty. They employed a decision theoretic framework for defining the optimal route and identified a family of appropriate cost models and travel time distributions.

Cornelis et al. in 2004 showed which criteria must be met for path finding algorithm correctness and explained an efficient method, based on de-fuzzification of fuzzy weights, for finding optimal paths. How involving in this project helps learners? Firstly, it acquaints the learners with all the basic concepts involved in coding a real world problem. Secondly, we need to design the algorithm based on the pre- defined constraints. As the traffic condition in a network vary from time to time and there are usually a huge amount of requests occur at any moment, it needs to quickly find the best path. The algorithm takes into account the overall level of services and service schedule on a route to determine the shortest path and transfer points.

There are several methods for path-finding:

1. In Dijkstra's algorithm the input of the algorithm consists of a weighted directed graph G and a source vertex in Graph. Let's denote the set of all vertices in the graph G as V . For a given pair of vertices s and t (say) in V , the algorithm finds the path from s to t with lowest cost (i.e. the shortest path).
2. Best First Search (BFS) algorithm has some estimate (called a heuristic) of how far from the goal any vertex is, instead of selecting the vertex closest to the starting point, it selects the vertex closest to the goal. BFS is not guaranteed to find a shortest path and runs much quicker than Dijkstra's algorithm because it uses the heuristic function.
3. A* was developed in 1968 to combine heuristic approaches like BFS and formal approaches like Dijkstra's algorithm and can guarantee a shortest path. The A* algorithm integrates a heuristic into a search procedure. A* is the most popular choice for path-finding, because it's fairly flexible and can be used in a wide range of contexts.

CHAPTER 4

SYSTEM DESIGN

4.1 High level Design

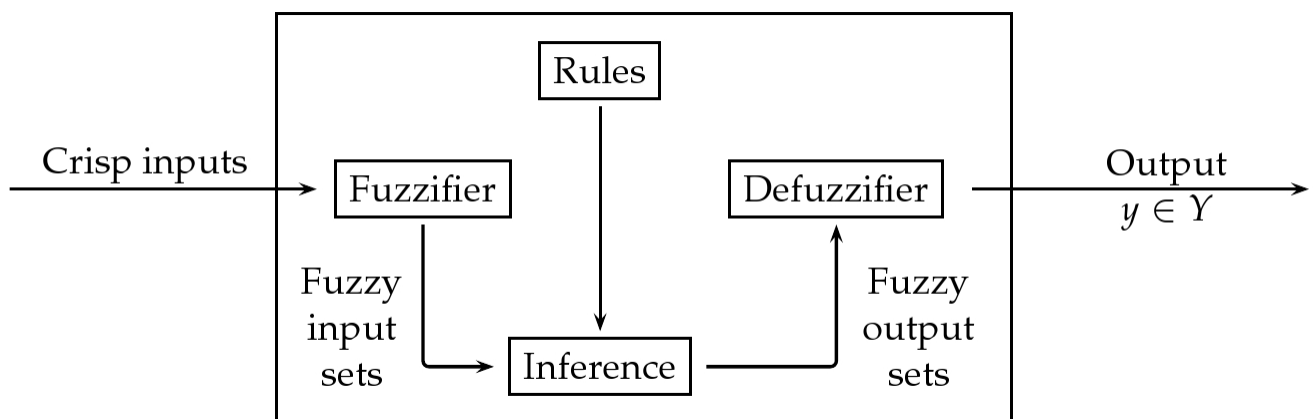


Fig. 4.1.1 High-level Design

High-level design also called as software architecture is the first design step after analyzing all requirements for software. The goal is to define a software structure which is able to fulfill requirements. Also the non-functional requirements such as scalability, portability are analyzed in high-level design.

4.2 Low Level Design

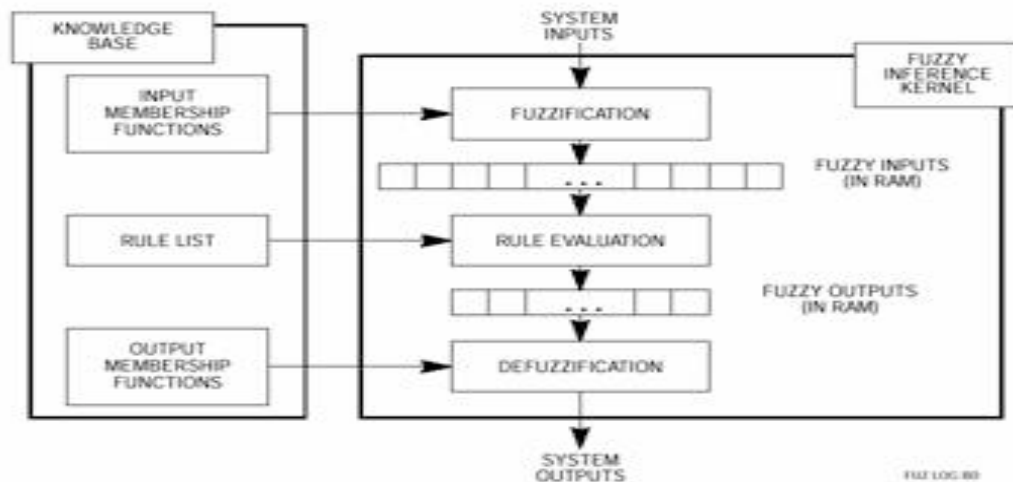


Fig. 4.2.1 Low-Level Design

4.3 Test Cases Generation

With a rapid increase in burden on networking devices, it is the need of the hour to modify or rewrite the existing algorithm. Therefore, to obtain a sufficient source of relevant data, we chose to enter data manually as user input. We are also interested in seeing how overall index can be predicted therefore a chart view is displayed on output screen. Furthermore for comparison purpose we also choose sorting algorithm to sort the results based on similarity degree and thereby displaying the optimum path.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Code

```
#include<stdio.h>
#include<iostream>
#include "math.h"
int reach[20];
int visited[10]={0},stack[10],top=-1, m=0;

using namespace std;

// Defining structures
struct data
{
    float x=0;
    float y=0;
    float z=0;
    int flag=0;
}a[10][10];

data path[100];
struct nu
{
    float val;
    int degree;
};

//Defining Functions to calculate minimum of the array upto desired
length value
data calc_min(data a[], int n)
{
    data small;
    small.x=999;small.y=999;small.z=999;
    for(int i=0;i<n;i++)
    {
        //cout<<"\n\n"<<i+1<<"\t\t"<<a[i].x<<"\t"<<a[i].y<<"\t"<<a[i].z<<"\n";
        if (a[i].x)
        {
            if(a[i].x<small.x)
                small.x=a[i].x;
            if(a[i].y<small.y)
                small.y=a[i].y;
            if(a[i].z<small.z)
                small.z=a[i].z;
        }
    }
}
```

```

    }
    }
    return small;
}

// Calculate Minimum of two numbers
float cal_min(float a, float b)
{
    if(a<=b)return a;
    else return b;
}

// Calculate Maximum of two numbers
float cal_max(float a, float b)
{
    if(a>=b)return a;
    else return b;
}

//Computation for calculating similarity degree
data compute(data a, data b)
{
    data c;
    c.x=cal_min(a.x, b.x);
    c.z=cal_min(cal_min(a.z, b.z), cal_max(a.y, b.y));
    if(cal_min(a.y, b.y)<=cal_max(a.x, b.x))
        c.y=cal_min(a.y, b.y);
    else
        c.y=((a.y*b.y)-(a.x*b.x))/((a.y+b.y)-(a.x+b.x));
    return c;
}

//DFS Search
void dfs(int v)
{
    int i;
    reach[v]=1;
    for(i=1;i<=6;i++)
        if(a[v][i].flag && !reach[i])
        {
            printf("\n %d->%d",v,i);
            dfs(i);
        }
}

int find(int a1,int b1,int size)
{
    if(size>1)
    {
        cout<<endl<<"Paths Available are: "<<endl;
        int i=a1,j;
    }
}

```

```

visited[i]=1;
stack[++top]=a1;
for(j=1;j<=size+1;++j)
{
    if(a[i][j].flag==1 && visited[j]==0)
    {
        visited[j]=1;
        stack[++top]=j;
        i=j;
        j=1;
    }
    if(stack[top]==b1)
    {
        cout<<"Path "<<m+1<<": ";
        for(int x=0;x<=top;++x)
        {
            cout<<stack[x]<<" ";
            if (x<top) {
                path[m].x+=a[stack[x]][stack[x+1]].x;
                path[m].y+=a[stack[x]][stack[x+1]].y;
                path[m].z+=a[stack[x]][stack[x+1]].z;
            }
        }
        cout<<endl;
        ++m;
        j=stack[top];
        visited[j]=0;
        i=stack[--top];
    }
    if(j>=size)
    {
        j=stack[top];
        visited[j]=0;
        i=stack[--top];
    }
    if(top==--1)
        break;
}
return 1;
}
else
return 0;
}

int main()
{
    int node_count, total_count=0;
    cout<<"Enter Number of nodes for the graph ";
    do{
        cin>>node_count;
        if(node_count<2)

```

```

        cout<<"\nCount of nodes should be atleast 2\nEnter number
of nodes again ";
    }
    while (node_count<2);
    cout<<"Enter node to node arc values(a, b,c) in the form of
adjacency matrix: "<<endl;
    for (int i=1; i<=node_count; ++i)
        for (int j=i+1; j<=node_count; ++j) {
            int flag;
            if(i!=j)
            {
                cout<<endl<<"Current Node is "<<i<<" to "<<j;
                cout<<endl<<"Enter '1' if node Exists else enter '0' ";
                cin>>flag;

                if(flag){
                    total_count++;
                    a[i][j].flag=a[j][i].flag=1;
                    cin>>a[i][j].x;
                    cin>>a[i][j].y;
                    cin>>a[i][j].z;
                    a[j][i].x=a[i][j].x;
                    a[j][i].y=a[i][j].y;
                    a[j][i].z=a[i][j].z;
                    continue;
                }
                else
                    a[i][j].flag=0;
                continue;
            }
        }
    int start_node, dest_node;

    do
    {
        cout<<endl<<"Enter starting node ";
        cin>>start_node;

        cout<<"Enter destination node ";
        cin>>dest_node;

        if(start_node<1 || dest_node>node_count)
            cout<<"\nOut Of bounds Check Start and end node\n ";
    }while (start_node<1 || dest_node>node_count);

    find(start_node, dest_node, node_count);
    data min[2];
    min[0]=calc_min(path,m);

    cout<<"\n";
    cout<<"Path Cost: ";

```

```

        for(int i=0;i<m;++i)
            cout<<"\n"<<"PATH " <<i+1<<":\t"<<path[i].x<<" "<<path[i].y<<"
"<<path[i].z;

        cout<<"\n\nCalculating Lmin (a, b, c)";
        cout<<"\na"<<"\tb"<<"\tc";
        cout<<"\n\n"<<min[0].x<<"\t"<<min[0].y<<"\t"<<min[0].z;

        cout<<"\n\nComputing (a,b,c)";
        cout<<"\nIterations"<<"\ta"<<"\tb"<<"\tc";

        data cz[2];
        int i=1;
        cz[0]=path[0];

        while(i<m){
            cz[1]=compute(cz[0],path[i]);
            //cout<<"\n\n"<<i-
1<<"\t\t"<<cz[1].x<<"\t"<<cz[1].y<<"\t"<<cz[1].z<<"\n";
            if(i==1)
                cz[0]=cz[1];
            cz[0]=calc_min(cz, 2);
            i++;
            cout<<"\n"<<i-
1<<"\t\t\t"<<cz[0].x<<"\t"<<cz[0].y<<"\t"<<cz[0].z;

        }

        cout<<"\n\nAccepted
Val"<<"\n"<<cz[0].x<<"\t\t"<<cz[0].y<<"\t"<<cz[0].z<<"\n";
        data Lmin=cz[0];

        cout<<"\n\nCalculating Similarity degree By finding Euclidean
distance \n";
        nu s[m];

        for(int i=0;i<m;++i)
        {
            s[i].val=pow((pow((path[i].x-Lmin.x), 2)+pow((path[i].y-
Lmin.y), 2)+pow((path[i].z-Lmin.z), 2)), 0.5);
            s[i].degree=i+1;
            cout<<"\nPATH " <<s[i].degree<<"\t"<<s[i].val;
        }

        cout<<"\n\nRank\t"<<"Optimum Path"<<"\t"<<"Similarity Degree\n";

        for(int i=0;i<m;i++)
            for(int j=i+1;j<m;j++)
                if(s[i].val>s[j].val)

```

```

        {
            nu temp=s[i];
            s[i]=s[j];
            s[j]=temp;
        }

    for(int i=0;i<m;++i)
        cout<<i+1<<"\t\tPATH
"<<s[i].degree<<"\t\t\t\t"<<s[i].val<<"\n";
}

```

5.2 Unit Testing

- 1) Module Interface test: In module interface test, we checked whether the information is properly flowing in to the program unit (or module) and properly happen out of it or not. Ex: Return values from the functions.
- 2) Local data structures: We tested if the local data within the module is stored properly or not. Ex: Structures and arrays defined in the program.
- 3) Boundary conditions: It is observed that much software often fails at boundary related conditions. That's why boundary related conditions were tested to make safe that the program is properly working at its boundary conditions. Ex: Nodes input greater than 2, Input values must be compatible, Range for input of membership function parameters.
- 4) Independent paths: All independent paths are tested to see that they are properly executing their task and terminating at the end of the program.
- 5) Error handling paths: These were tested to review if errors are handled properly by them or not.

5.3 Integration Testing

We tested the working of multiple modules collectively. It is one of the extensive exercises of the software testing in which particular software modules are merged and

tested as a group. After unit testing, functions were tested as a whole for accurate output. The return values of the functions, successfully gave accurate result as expected

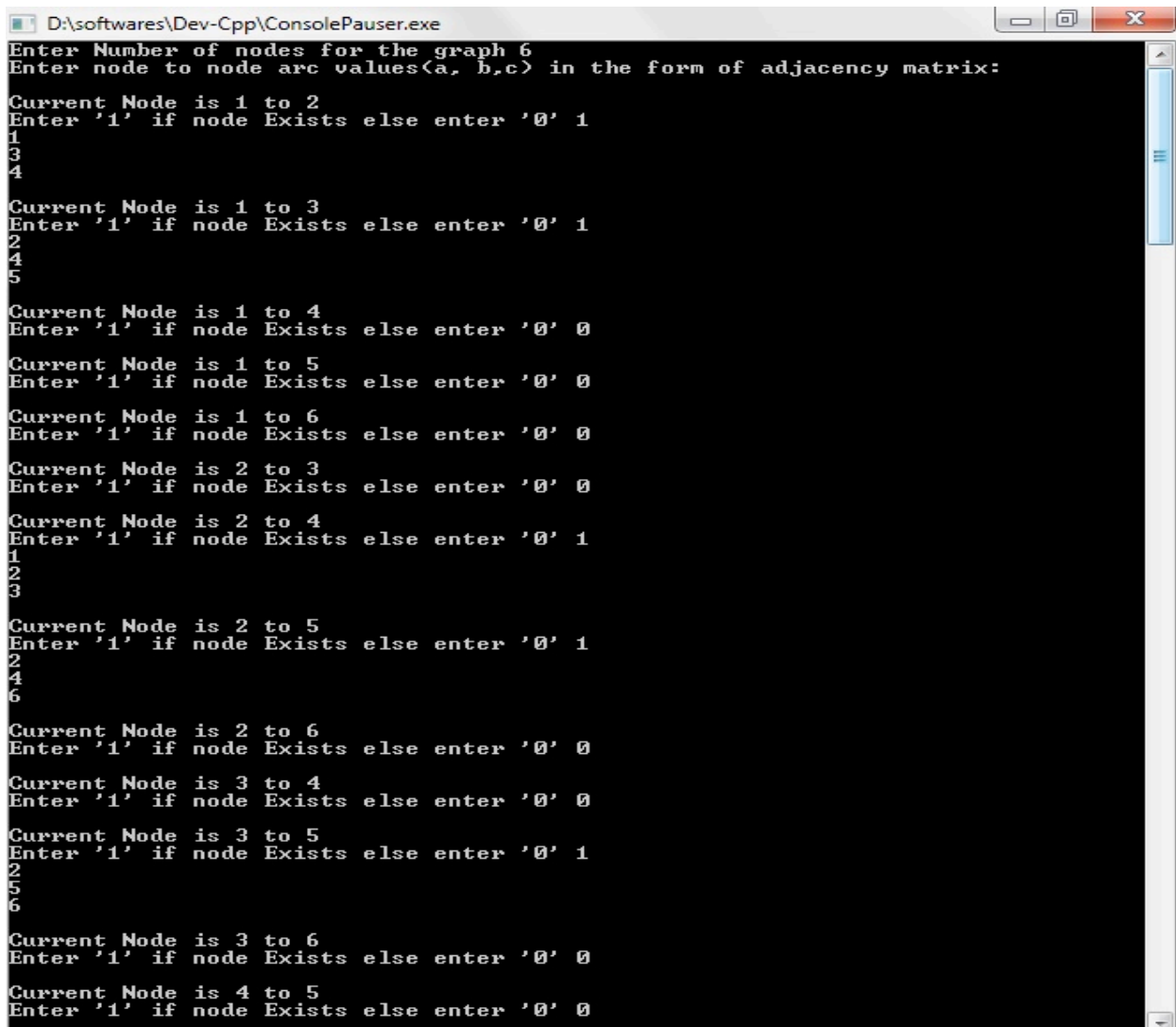
5.4 Test Results

Expected result successfully matched the theoretically calculated result set. And even with more precision at programming end, it can be successfully concluded that efficient routing can be done given the congestion parameters in terms of membership functions.

CHAPTER 6

Results and Discussion

6.1 Output/Results



```
D:\softwares\Dev-Cpp\ConsolePauser.exe
Enter Number of nodes for the graph 6
Enter node to node arc values(a, b,c) in the form of adjacency matrix:

Current Node is 1 to 2
Enter '1' if node Exists else enter '0' 1
1
3
4

Current Node is 1 to 3
Enter '1' if node Exists else enter '0' 1
2
4
5

Current Node is 1 to 4
Enter '1' if node Exists else enter '0' 0

Current Node is 1 to 5
Enter '1' if node Exists else enter '0' 0

Current Node is 1 to 6
Enter '1' if node Exists else enter '0' 0

Current Node is 2 to 3
Enter '1' if node Exists else enter '0' 0

Current Node is 2 to 4
Enter '1' if node Exists else enter '0' 1
1
2
3

Current Node is 2 to 5
Enter '1' if node Exists else enter '0' 1
2
4
6

Current Node is 2 to 6
Enter '1' if node Exists else enter '0' 0

Current Node is 3 to 4
Enter '1' if node Exists else enter '0' 0

Current Node is 3 to 5
Enter '1' if node Exists else enter '0' 1
2
5
6

Current Node is 3 to 6
Enter '1' if node Exists else enter '0' 0

Current Node is 4 to 5
Enter '1' if node Exists else enter '0' 0
```

```
D:\softwares\Dev-Cpp\ConsolePauser.exe
Current Node is 4 to 5
Enter '1' if node Exists else enter '0' 0

Current Node is 4 to 6
Enter '1' if node Exists else enter '0' 1
2
3
8

Current Node is 5 to 6
Enter '1' if node Exists else enter '0' 1
3
5
7

Enter starting node 1
Enter destination node 6

Paths Available are:
Path 1: 1 2 4 6
Path 2: 1 2 5 6
Path 3: 1 3 5 2 4 6
Path 4: 1 3 5 6

Path Cost:
PATH 1: 4 8 15
PATH 2: 6 12 17
PATH 3: 9 18 28
PATH 4: 7 14 18

Calculating Lmin (a, b, c)
a      b      c
4      8      15

Computing (a,b,c)
Iterations  a      b      c
1           4      7.2    12
2           4      7.2    12
3           4      7.13725 12

Accepted Val
4           7.13725 12

Calculating Similarity degree By finding Euclidean distance

PATH 1  3.12159
PATH 2  7.25578
PATH 3  19.975
PATH 4  9.59673

Rank    Optimum Path    Similarity Degree
1       PATH 1        3.12159
2       PATH 2        7.25578
3       PATH 4        9.59673
4       PATH 3        19.975
```

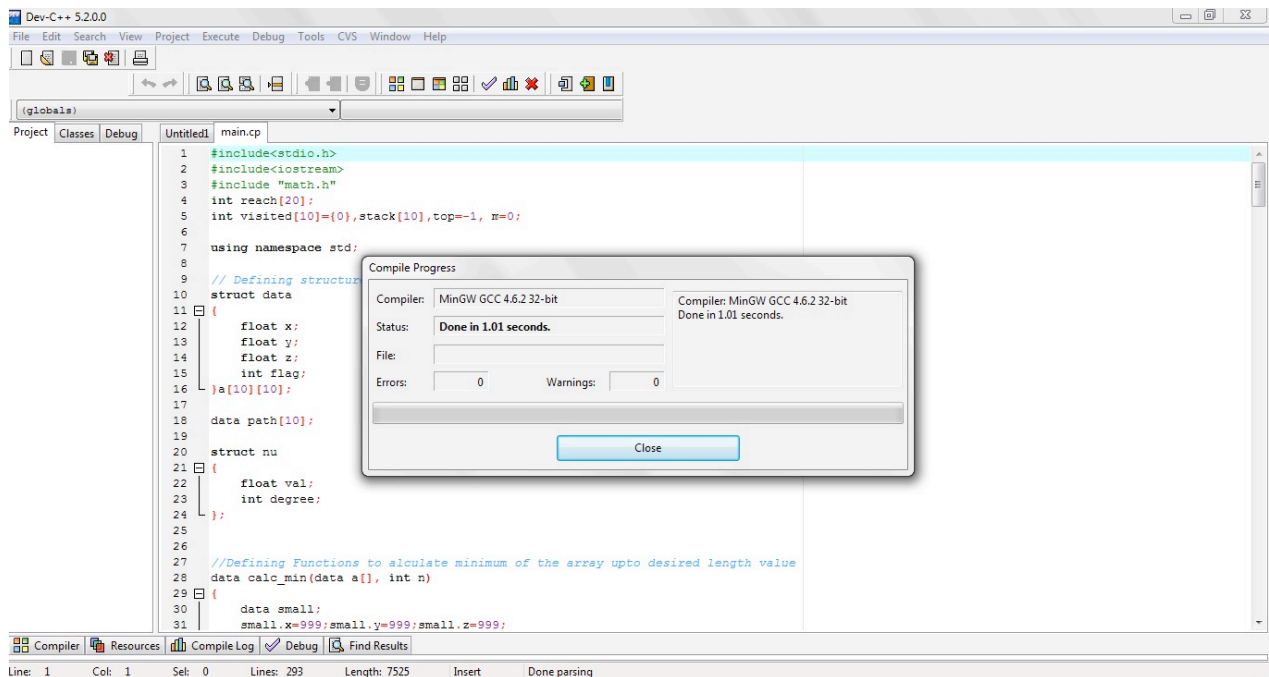

6.2 Results Analysis

Results based on the input routing parameters matched the expected theoretical calculations. This proves that we can implement this to real world and get the benefits in terms of accuracy, efficiency and performance.

Program file size: 8 kB

Exe file size: 1.37 mB

Compile time: 1.01 seconds



7. Conclusion and Future Work

In this paper Similarity degree has been defined for triangular fuzzy numbers. Based on the highest similarity degree shortest path has been identified in a network problems. The ranking given to the paths are based on the highest similarity degree, which helps the decision maker to identify the preferable path alternatives. We implemented an algorithm for solving shortest path problem on a network with fuzzy arc lengths. The algorithm can be helpful to decision makers as they make decision. We have explained the method by an example with the help of a hypothetical data.

8. References

- [1] D.Dubois and H.Prade," Fuzzy Sets and Systems", Academic Press, New York,1980.
- [2] K.Atanassov, Intuitionistic Fuzzy Sets and Systems, volume 20, No.187- 96 1986.
- [3] Kung J.Y.,Chuang T.N, and C.T.Lin ,Decision Making On Network Problem With Fuzzy arc lengths, IMACS Multiconference on Computational Engineering in Systems Applications(CESA578- 580)2006.
- [4] Odada S., Gen M., "Fuzzy Shortest Path Problems". Computer and Industrial Engineering, 27 ,465-468,1994.
- [5] Chuang T.N., Kung J.Y.,"A new Algorithm for the Discrete FuzzyShortest Path Problem in a Network ".Applied Mathematics and Computation. 174.660-668.2006
- [6] Hyung LK, Song YS, Lee KM., Similarity Measure between Fuzzy sets and between Elements .Fuzzy Sets and Systems, 62291 -3 1994.
- [7] Amit kumar, Manoj kaur," A new Algorithm for Solving Network Flow Problems with Fuzzy arc Lengths" TJFS: Turkish Journal of Fuzzy Systems -volume.2 No.1.pp.2011.
- [8] A.Nagoor gani, M.Mohammed Jabbamlla "On Searching Intuitionistic Fuzzy Shortest Path in a Network ". Applied Mathematical Sciences Vo14,no, 69,3447-3454, 2010.
- [9] Iraj Mahadavi,Ali Tajdin,Rahele Nourifar "Using Fuzzy Optimization Approach for Shortest Path Problem in Network flow".

- [10] Lin K.C., Chern M.S., The Fuzzy Shortest Path Problem and its Most Liu Vital arcs., Fuzzy Sets and Systems 58, 343- 353, 1993. Vital arcs., Fuzzy Sets and Systems 58, 343- 353, 1993.
- [11] X. Entropy, Length Measure and Similarity Measure of Fuzzy Sets and their Relations, Fuzzy sets and systems, 52:305 -18, [13] 1992.
- [12] L. Sujatha and S. Elizabeth "Fuzzy Shortest Path Problem based on Similarity Degree"., Applied Mathematical Sciences, Vol., 5, 200, no. 66, 3263 -3276.
- [13] Klein, C.M 1991. Fuzzy Shortest Paths, Fuzzy Sets and Systems 39, 27-41, 2011 Using Fuzzy Logic"., International Journal of Computational Wang Cognition vol 8 no.1. March 2010.