Fuzzy defining number of the Edge Fuzzy Graph

CSE2009

(Soft Computing)

Submitted by:

Pabbisetty Revanth Venkata sai – 20BCI7009

Shaik Mohammad Waseem Akram -20BCD7141

Padamata Sai Satyanarayana -20BCE7135

Submitted to:

Dr.Arindam Dey

Department of computer science and engineering

Table of Contents

Abstract

In this paper, neighbourly irregular fuzzy graphs, neighbourly total irregular fuzzy graphs, highly irregular fuzzy graphs and highly total irregular fuzzy graphs are introduced. A necessary and sufficient condition under which neighbourly irregular and highly irregular fuzzy graphs are equivalent is provided. We define d2 degree of a vertex in fuzzy graphs and total d2 -degree of a vertex in fuzzy graphs and (2, k)-regular fuzzy graphs, totally (2, k)- regular fuzzy graphs are introduced. (2, k)- regular fuzzy graphs and totally (2, k)-regular fuzzy graphs are compared through various examples.

Given a graph G=(V,E), a coloring function C assigns an integer value C(i) to each node i∈V in such a way that the extremes of any edge {i,j}∈E cannot share the same color, i.e., C(i)≠C(j). Two different approaches to the graph coloring problem of a fuzzy graph are introduced in this paper. The classical concept of the (crisp) chromatic number of a graph is generalized for these approaches. The first approach is based on the successive coloring functions Cα of the crisp graphs Gα=(V,Eα), the α-cuts of G̃; the traffic lights problem is analyzed following this approach. The second approach is based on an extension of the concept of coloring function by means of a distance defined between colors; a timetabling problem is analyzed within this approach

Key words:- Fuzzy graph, defining number, chromatic number

1.Introduction

Graph coloring is one of the most studied problems of combinatorial optimization. Many problems of practical interest can be modeled as coloring problems. The general form of this application involves forming a graph with nodes representing items of interest. The basic graph coloring problem is to group items in as few groups as possible, subject to the constraint that no incompatible items end up in the same group.

Formally, given a graph G = (V; E), a coloring function is a mapping C : V → N

identifying C(i) as the color of node i ∈ V, in such a way that two adjacent nodes cannot share the same color, i.e., C(i) □= C(j) if {i; j} ∈ E. These nodes i and j will be denoted as incompatible and, in this context, graph G will be denoted the incompatibility graph.

A k-coloring Ck is a coloring function with no more than k di3erent colors Ck : V → {1;:::;k}:

A graph is k-colored if it admits a k-coloring. The minimum value k such that G is k-colored is the chromatic number of G and it is denoted as (G). The graph coloring problem (for short, the coloring problem) consists of determining the chromatic number of a graph and an associated coloring function. This problem is known to be NP-hard. An important area of application of the coloring problem is management science. Classical applications include wiring of printed circuits loading problems resource allocation frequency assignment problem a wide variety of scheduling problemsand computer register allocation .

2. Preliminary Notes

We will use the classical definition of fuzzy set A defined on a non empty set X as the family A { (x, {1A I a; e X}, where : X + I is the membership function and (x) reflects the ambiguity of the assertion x belongs to A.

In classical fuzzy-set theory the set I is usually defined as the interval [0 1] in such a way that "A (x) — 0 indicates that x does not belong to A, "A (x) $= 1$indicates that a; strictly belongs to A, and any intermediate value represents the degree in which a; could belong to A. However, the set I could be a discrete set of the forml {0, 1, . k}, where"A (x) [IA indicates that the degree of membership function a; to A is lower than the degree of membership function x'. In general, the set I can be any ordered set, not necessarily numerical, for instance, I {null, low, medium, high, total}. Let G = (V, É) be a fuzzy graph, where V is the vertex set, the fuzzy edge set E is characterized by the matrix p $= (\mu_{ij})_{i,j\in V}$; /lij = $\mu_E(\{i,j\})$ Vi,j e V such that i j and

μ_E: V x V + I is the membership function. In this sense, a fuzzy graph can also be denoted as G = (V,$\mu)$.

Graph coloring is one of the most studied problems of combinatorial optimization. Many problems of practical interest can be modeled as coloring problems. The general form of this application involves forming a graph with vertices representing items of interest. The basic graph coloring problem is to group items in as few groups as possible, subject to the constraint that no incompatible items end up in the same group. Formally, given a graph G = (V, E), a coloring function is a mapping C : V + N identifying C(i) as the color of vertex i e V, in such a way that two adjacent vertices cannot share the same color, i.e., C(i) C(j) if {i, j} e E. These vertices i and j will be denoted as incompatible and, in this context, graph G will be denoted the incompatibility graph.

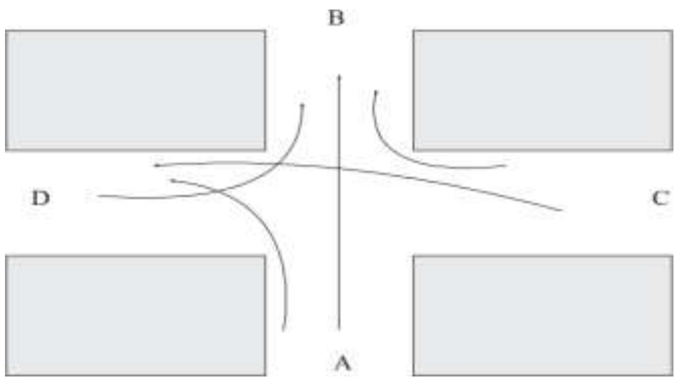
3. Main Result

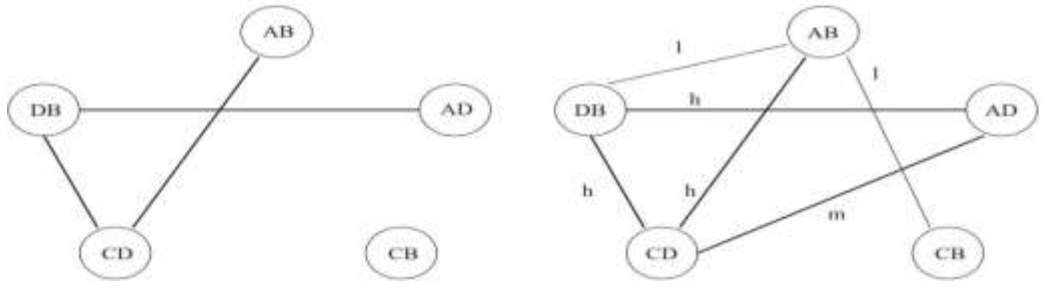
3.1 The fuzzy coloring problem

In order to introduce the concepts of the coloring function of a fuzzy graph and its associated chromatic number, a scheduling problem is presented.

The tra6c lights problem consists of controlling a tra6c lights system in such a way that certain level of security will be attained. This problem has been studied as an intersection graph . Other authors have modeled it as an assignment set problem. The tra6c lights problem can also be modeled as a graph coloring problem. The following example illustrates this approach.

Example 3.1. The tra6c Now at the corner of two streets is depicted in Fig. 1. Certain lanes are compatible with one,





another, such as AD and CB, while others are incompatible, such as AB and CD. In order to avoid collisions, we wish to install a tra6c light system to control the Now of vehicles. This problem can be modeled by means of an incompatibility graph G = (V; E) whose nodes are given by the lanes, and a pair of lanes de4nes an edge if they are incompatible, i.e., they can cause a collision. In this case, V = {AB; AD; CB; CD; DB}; E = {{AB; CD}; {AD; DB}; {CD; DB}}.

Any k-coloring Ck of the graph G identi4es a control policy of the lights system. The entire cycle of the lights system is divided into k time periods or slots (with any time-length). For any slot c ∈ {1;:::;k} circulation movements i such that Ck (i) = c are the only ones allowed. Therefore, the chromatic number
(G) gives the minimum number of time periods required to control the system. The chromatic number of G is
(G) = 2 and a 2-coloring is C2 (AB)=1; C2 (AD)=2; C2 (CB)=1; C2 (CD)=2; C2 (DB)=1: Obviously, the control policy of the lights depends on the incompatibility of the lanes. The concept of incompatibility could be fuzzy and it could be graduated. This graduation, which does not need to be numerical, is associated to the desired security level for the tra6c Now at the corner. The maximum security level is attained when all lanes are considered incompatible and the graph is complete; in this case, the chromatic number is the number of lanes and the

control policy of the lights assure that only one movement is allowed in any slot of the cycle. On the other hand, the minimum security level is attained when the incompatibility edge set is empty; in this case, the chromatic number is 1 and all movements are allowed at any instant.

In Example 3.1, for instance, lanes CD and DB are more incompatible than lanes AB and DB. Let I = {n; l; m; h; t}, where n, l, m, h and t denote the incompatibility degrees null, low, medium, high and total, respectively. The problem stated in Example 3.1 could be modeled by means of the fuzzy graph G˜ = (V; □), where V = {AB; AD; CB; CD; DB};

$$\mu = \begin{pmatrix} - & n & l & h & l \\ n & - & n & m & h \\ l & n & - & n & n \\ h & m & n & - & h \\ l & h & n & h & - \end{pmatrix}.$$

Definition 3.1. Given a fuzzy graph G˜ = (V; □), its chromatic number is the fuzzy number (G˜) = {(x; $(x))=x ∈ X }; where X = {1;:::; |V|}, $(x) = sup{□ ∈ I= x ∈ A□} ∀x ∈ X and A□ = {1;:::; □} ∀□ ∈ I. The chromatic number of a fuzzy graph is a normalized fuzzy number whose modal value is associated with the empty edge-set graph. Its meaning depends on the sense of index □, and it can be interpreted in the following way: for lower values of □ there are many incompatible links between nodes and, consequently, more colors are needed in order to consider these incompatibilities; on the other hand, for higher values of □ there are fewer incompatible links between nodes and less colors are needed. The chromatic number sums upall this information in order to manage the fuzzy problem.

Definition 3.2. Given a fuzzy graph G˜, a dissimilarity measure d and a scale function f, the minimum value k for which a (d; f)-extended k-coloring of G˜ exists is the (d; f)-chromatic number of G˜ and it is denoted by d;f(G˜). The (d; f)-coloring problem consists of determining the (d; f)-chromatic number of a fuzzy graph and an associated (d; f)-extended coloring function.

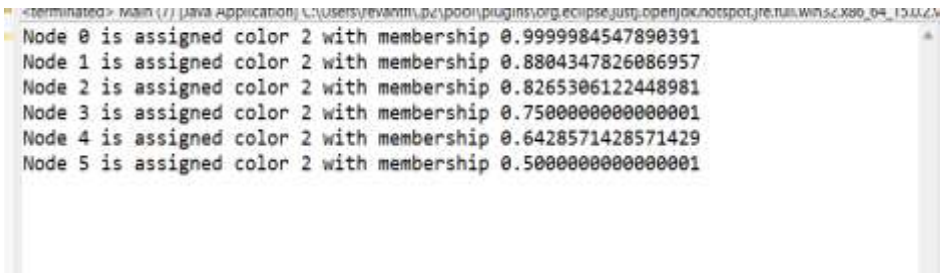
4. Appendix

4.1 Code for fuzzy membership values

```
package softcomputing;
import java.util.*;

public class fuzzygraphcoloring1 {
    public static void main(String[] args) {
        // Define the graph
        int[][] graph = {{0, 1, 1, 0}, {1, 0, 1, 1}, {1, 1, 0, 1},
{0, 1, 1, 0}};
        int n = graph.length;

        // Initialize the fuzzy color matrix
        double[][] fuzzyColors = new double[n][3];

        // Set the initial fuzzy color values for each vertex
        for (int i = 0; i < n; i++) {
            fuzzyColors[i][0] = 1.0; // Red color
            fuzzyColors[i][1] = 0.0; // Green color
            fuzzyColors[i][2] = 0.0; // Blue color
        }

        // Loop until convergence
        boolean converged = false;
        while (!converged) {
            converged = true;

            // Update the fuzzy color values for each vertex
            for (int i = 0; i < n; i++) {
                // Compute the average color of the neighboring
vertices
                double[] avgColor = new double[3];
                int numNeighbors = 0;
                for (int j = 0; j < n; j++) {
                    if (graph[i][j] == 1) {
                        avgColor[0] += fuzzyColors[j][0];
                        avgColor[1] += fuzzyColors[j][1];
                        avgColor[2] += fuzzyColors[j][2];
                        numNeighbors++;
                    }
                }
                avgColor[0] /= numNeighbors;
                avgColor[1] /= numNeighbors;
                avgColor[2] /= numNeighbors;

                // Compute the Euclidean distance between the
current and average color
                double dist = Math.sqrt(Math.pow(fuzzyColors[i][0] -
avgColor[0], 2) +
                                        Math.pow(fuzzyColors[i][1] -
avgColor[1], 2) +
                                        Math.pow(fuzzyColors[i][2] -
avgColor[2], 2));

                // Update the fuzzy color value for the vertex
```

```java
                    if (dist > 0.1) { // convergence threshold
                        converged = false;
                        fuzzyColors[i][0] = (1 - dist) *
fuzzyColors[i][0] + dist * avgColor[0];
                        fuzzyColors[i][1] = (1 - dist) *
fuzzyColors[i][1] + dist * avgColor[1];
                        fuzzyColors[i][2] = (1 - dist) *
fuzzyColors[i][2] + dist * avgColor[2];
                    }
                }
            }

            // Print the final fuzzy color values for each vertex
            System.out.println("Fuzzy colors:");
            for (int i = 0; i < n; i++) {
                System.out.println("Vertex " + i + ": (" +
fuzzyColors[i][0] + ", " + fuzzyColors[i][1] + ", " +
fuzzyColors[i][2] + ")");
            }
        }
}
```

OUTPUT:

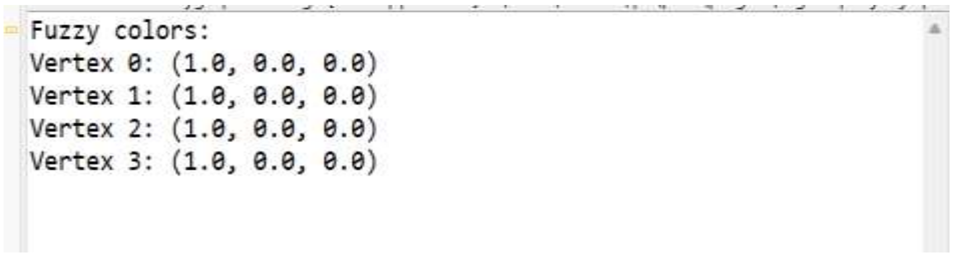


```
<terminated> Main (7) [Java Application] C:\Users\revanth\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v
Node 0 is assigned color 2 with membership 0.9999984547890391
Node 1 is assigned color 2 with membership 0.8804347826086957
Node 2 is assigned color 2 with membership 0.8265306122448981
Node 3 is assigned color 2 with membership 0.7500000000000001
Node 4 is assigned color 2 with membership 0.6428571428571429
Node 5 is assigned color 2 with membership 0.5000000000000001
```

4.2 Code for fuzzy colors for vertices

```java
package softcomputing;
import java.util.*;

public class fuzzygraphcoloring1 {
    public static void main(String[] args) {
        // Define the graph
        int[][] graph = {{0, 1, 1, 0}, {1, 0, 1, 1}, {1, 1, 0, 1},
{0, 1, 1, 0}};
        int n = graph.length;

        // Initialize the fuzzy color matrix
        double[][] fuzzyColors = new double[n][3];

        // Set the initial fuzzy color values for each vertex
        for (int i = 0; i < n; i++) {
            fuzzyColors[i][0] = 1.0; // Red color
            fuzzyColors[i][1] = 0.0; // Green color
            fuzzyColors[i][2] = 0.0; // Blue color
        }

        // Loop until convergence
        boolean converged = false;
        while (!converged) {
            converged = true;

            // Update the fuzzy color values for each vertex
            for (int i = 0; i < n; i++) {
                // Compute the average color of the neighboring 
vertices
                double[] avgColor = new double[3];
                int numNeighbors = 0;
                for (int j = 0; j < n; j++) {
                    if (graph[i][j] == 1) {
                        avgColor[0] += fuzzyColors[j][0];
                        avgColor[1] += fuzzyColors[j][1];
                        avgColor[2] += fuzzyColors[j][2];
                        numNeighbors++;
                    }
                }
                avgColor[0] /= numNeighbors;
                avgColor[1] /= numNeighbors;
                avgColor[2] /= numNeighbors;

                // Compute the Euclidean distance between the 
current and average color
                double dist = Math.sqrt(Math.pow(fuzzyColors[i][0] - 
avgColor[0], 2) +
                                        Math.pow(fuzzyColors[i][1] - 
avgColor[1], 2) +
                                        Math.pow(fuzzyColors[i][2] - 
avgColor[2], 2));

                // Update the fuzzy color value for the vertex
                if (dist > 0.1) { // convergence threshold
                    converged = false;
                    fuzzyColors[i][0] = (1 - dist) * 
fuzzyColors[i][0] + dist * avgColor[0];
                    fuzzyColors[i][1] = (1 - dist) * 
fuzzyColors[i][1] + dist * avgColor[1];
                    fuzzyColors[i][2] = (1 - dist) * 
fuzzyColors[i][2] + dist * avgColor[2];
                }
            }
        }
        // Print the final fuzzy color values for each vertex
        System.out.println("Fuzzy colors:");
        for (int i = 0; i < n; i++) {
            System.out.println("Vertex " + i + ": (" +
fuzzyColors[i][0] + ", " + fuzzyColors[i][1] + ", " +
fuzzyColors[i][2] + ")");
        }
    }
}
```

OUTPUT:



```
Fuzzy colors:
Vertex 0: (1.0, 0.0, 0.0)
Vertex 1: (1.0, 0.0, 0.0)
Vertex 2: (1.0, 0.0, 0.0)
Vertex 3: (1.0, 0.0, 0.0)
```

5. Conclusion

In this paper, the concept of crisp incompatibility among the nodes of a graph is extended to include vagueness and graduation. To handle this new situation, two approaches to the coloring problem of fuzzy graphs with crisp nodes and fuzzy edges have been introduced. The 4rst one is based on the natural fuzzi4cation of the classical coloring problem on graphs; an example—the tra6c lights problem—has been fuzzi4ed and solved with this approach. The associated chromatic number can be de4ned and interpreted according to the meaning of the fuzzy edges of the fuzzy graph.

Fuzzy-set theory proposes to be a tool for handling vagueness and degrees of certainty, and for giving a consistent representation of linguistically formulated knowledge which allows the use of precise properties and algorithms. In this paper new concepts about fuzzy graphs and two new fuzzy optimization problems have been introduced, which can be very useful in order to model and solve real-life problems without undue simplications.

6. Reference

[1] Garey MR, Johnson DS. Computers and intractability. A guide to the theory of NP—completeness. New York: W.H. Freeman and Company; 1979.

[2] Cutello V, Montero J, Y)a ˜nez J. Structure functions with fuzzy states. Fuzzy Sets and Systems 1996;83(2):189–202.

[3] https://www.researchgate.net/publication/258049993_Coloring_fuzzy_graphs

[4][4] https://www.google.com/search?q=abstract+on+fuzzy+graph&sxsrf=APwXEdfRw5oKGMH3gSkl34gBz3CoiKvi9Q%3A1682145728829&source=hp&ei=wIFDZLS5MLuz2roPg6uamAg&iflsig=AOEireoAAAAAZEOP0DAleNkNZirUgnloGzT8H_Uf9XhK&oq=abstract+on+fuzzy+gra&gs_lcp=Cgdnd3Mtd2l6EAEYADIFCCEQoAEyCgghEBYQHhAPEB0yCAghEBYQHhAdMggIIRAWEB4QHTIKCCEQFhAeEA8QHToECCMQJzoLCC4QgAQQsQMQgwE6EQguEIAEELEDEIMBEMcBENEDOggIABCABBCxAzoLCAAQigUQsQMQgwE6EQguEIAEELEDEMcBENEDENQCOgsILhCABBDHARCvAToFCAAQgAQ6CwgAEIAEELEDEIMBOgYIABAWEB46CAgAEBYQHhAPOggIABCKBRCGAzoHCCEQoAEQClAAWIE0YNFLaABwAHgBgAGuAogBjBSSAQgxLjE5LjAuMZgBAKABAQ&sclient=gws-wiz#imgrc=xt3Bl-iasg1c3M

[5] https://www.sciencedirect.com/science/article/abs/pii/S0305048304000660

[6] Rosenfeld A. Fuzzy graphs. In: Zadeh LA, Fu KS, Shimura M, editors. Fuzzy sets and their applications to cognitive and decision processes. New York: Academic Press; 1975. p. 77–95.

[7] Bullnheimer B. An examination scheduling model to maximize student's study time. In: Burke E, Ross P, editors. Practice and theory of automated timetabling II. Berlin: Springer; 1998. p. 78–91.

[8] https://www.mdpi.com/2075-1680/11/12/697

[9] https://scholar.google.co.in/scholar?q=fuzzy+graph+coloring&hl=en&as_sdt=0&as_vis=1&oi=scholart

[10] Herrmann F, Hertz A. Finding the chromatic number by means of critical graphs. Journal of Experimental Algorithmics 2002; 7(Article 10):1–9.

[11] Kubale M, Jackowski B. A generalized implicit enumeration algorithm for graph coloring. Communications of the ACM 1985;28(4):412–8.

[12] Peem Yoller J. A correction to Br)elaz's modi4cation of Brown's coloring algorithm. Communications of the ACM 1983;26(8):595–7.