Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Graph Coloring- 'M'colorability Decision Problem

Anoosha Garimella
Student – B.Tech, III year 2nd semester,
Computer Science and Engineering
GITAM University, Visakhapatnam, India

P. S. Chandana
Assistant Professor,
Computer Science and Engineering
GITAM University, Visakhapatnam, India

Abstract- This review paper focuses on the m-colorability decision problem. This problem determines a solution on how to assign the minimum number of colors to a given graph, i.e. the chromatic number (m-colors) given the condition that no two adjacent nodes must have same color. The following paper explains this concept with the help of two examples along with its algorithmic implementation and execution.

Keywords: Graph Coloring, Chromatic number, m-colorability, colors, edges.

I. INTRODUCTION

The 'M-colorability Decision Problem' comes under the concept of graph coloring. A proper coloring of a graph is an assignment of colors to the vertices of the graph so that no two adjacent vertices have the same color. For example, if the vertices of a graph represent traffic signals at an intersection, and two vertices are adjacent if the corresponding signals cannot be green at the same time, a coloring can be used to designate sets of signals that can be green at the same time. This concept is closely related to the concept of an independent set. The chromatic number of a graph is the minimum number of colors required in a proper coloring.

II. METHODOLOGY

Let 'G' be a graph and 'm' be a positive integer. We want to discover whether the nodes of 'G' can be colored in such a way that no two adjacent nodes have the same color. This is known as 'm-colorability decision problem'. The 'm-colorability' optimization asks for the smallest integer 'm' for which graph 'G' can be colored. This integer can be referred to as the "chromatic number" of the graph. For example, consider a graph with 4 nodes as shown below. We apply this technique to find the minimum number of colors which are used for coloring the graph.

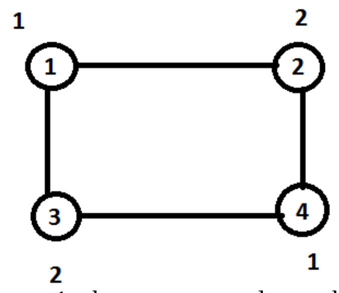


Figure 1: shows an example graph

Here, the maximum number of colors which can be used for coloring a graph are four. Our aim is to find out the minimum number of colors which are used to color this graph such that no two adjacent nodes have the same color. In the above example, consider node1, color1 can be assigned to node1. Node1 is having two adjacent nodes node2, node3 and hence we cannot assign color1 to either node2 or node3. Therefore, assign color1 to node4 as it is not adjacent to node1. Assign color2 to node2. Node2 has two other adjacent nodes node1 and node4 and hence we cannot assign color2 to either of the nodes. Node3 is not adjacent to the second node and hence you assign color2 to node3. All the nodes have been colored using two colors color1 and color2. Thus, two are the minimum number of colors used to color this graph. The value of m=2 according to m-colorability problem.

III. ALGORITHM

```
Algorithm Mcoloring (k)
{
 repeat
 {
   Nextvalue(k);
   if(x[k]=0) then return;
   if(k=n) then write(x[1:n]);
   else Mcoloring(k+1);
  }until(false);
}

Algorithm NextValue (k)
{
 repeat
 {
   x[k]=(x[k]+1)mod(n+1);
   if(x[k]=0) then return;
   for j=1 to n do
  {
     if ((G(k,j)!=0)&&(x[k]=x[j]))
     then break;
  }
  if(j=n+1) then return;
  } until (false);
}
```

IV. EXECUTION OF THE ALGORITHM

The step by step execution of the above algorithm is explained with the help of the below example

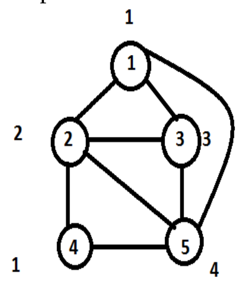
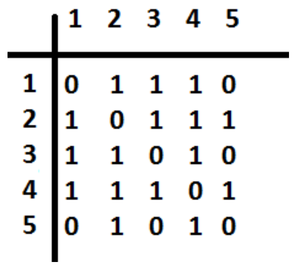


Figure 2: Example Graph

Generally, as explained in the first example, we can assign colors as color1 to node1, color2 to node2, color3 to node3, color1 to node4 and color4 to node5. Thus, the minimal number of colors used for coloring this graph that is the chromatic number of this graph is four.
First draw an adjacency matrix that indicates the relative existence or non-existence of nodes between the nodes. Adjacency matrix is as:



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 4 & 1 & 1 & 1 & 0 & 1 \\ 5 & 0 & 1 & 0 & 1 & 0 \end{array}$$

Figure 3: adjacency matrix

Consider an array of size five that is used to store the color of each node. Initially, all the elements in the array are initialized to zero to prevent any garbage values.
Assigning colors using the execution of the algorithm
The value of k initially is the value of first node, take 1.

- The control starts from Algorithm Mcoloring(1). The first step in the algorithm is nextvalue(k). Now, the control goes to the nextvalue(k) execution.

Nextvalue(k) => Nextvalue(1)
x[1]=(x[1]+1)mod 6
x[1]=1
If(x[1]=0) False
For j=1 to 5
J=1: if(G(1,1)!=0)&&(x[1]=x[1]) => (G(1,1)=0 F and 1=1 T) => F && T= F
J=2 : if(G(1,2)!=0)&&(x[1]=x[2]) => (G(1,2)!=0 T and 1=0 F) => T&&F=F
J=3 : if(G(1,3)!=0)&&(x[1]=x[3]) => (G(1,3)!=0 T and 1=0 F) => T&&F=F
J=4 : if(G(1,4)!=0)&&(x[1]=x[4]) => (G(1,4)!=0 T and 1=0 F) => T&&F=F

J=5 : if(G(1,5)!=0)&&(x[1]=x[5]) => (G(1,5)!=0 F and 1=0 F) => F&&F=F
J=6: loop condition fails and exits the for loop
Since all the above conditions have failed, we can assign '1' that is color1 to node1.
Now, the control passes to the main algorithm and executes the next step after the nextvalue(k).
If(x[1]=0) false as x[1]=1
If(1=5) false
Else Mcoloring (1+1) i.e. Mcoloring (2)

- Mcoloring (2) -> Nextvalue (2) Now the control passes to nextvalue (2) (k=2)

x[2]=(x[2]+1)mod(n+1) => x[2]=(0+1) mod 6 => x[2]=1
x[2]=0 false (as we have x[2]=1)
For i=1 to 5
J=1 : if(G(2,1)!=0) &&(x[k]=x[j]) => (G(2,1)!=0 T and x[2]=x[1] T) => (T&&T=T)
Since the condition has been true break the 'for' loop and repeat the nextvalue (k) algorithm.
X[2]=(1+1)mod 6 = 2mod6 = 2
X[2]=2
If(x[2]=0) False (since x[2]=2)
For j=1 to 5
J=1: if(G(2,1)!=0)&&(x[2]=x[1]) => (G(2,1)=0 T and 2=1 F) => T&& T= F
J=2 : if(G(2,2)!=0)&&(x[2]=x[2]) => (G(2,2)!=0 F and 2=2 T) => F&&T=F
J=3 : if(G(2,3)!=0)&&(x[2]=x[3]) => (G(2,3)!=0 T and 2=0 F) => T&&F=F
J=4 : if(G(2,4)!=0)&&(x[2]=x[4]) => (G(2,4)!=0 T and 2=0 F) => T&&F=F
J=5 : if(G(2,5)!=0)&&(x[2]=x[5]) => (G(2,5)!=0 F and 2=0 F) => T&&F=F
J=6: loop condition fails and exits the for loop
Since all the above conditions have failed, we can assign '2' that is color2 to node2.
Now, the control passes to the main algorithm and executes the next step after the nextvalue (k).
If(x[2]=0) false as x[2]=2
If(2=5) false
Else Mcoloring (2+1) i.e. Mcoloring (3)

- Mcoloring (3) =>Nextvalue (3)

Nextvalue (3)
Now the control goes to Nextvalue (3)
x[3]=(0+1)mod 6=1
If(x[3]=0) false (x[3]=1)
For j=1 to 5
J=1: if(G(3,1)!=0)&&(x[k]=x[j]) (G(3,1)!=0 T and 1=1 T) => T&&T=T
Since the above condition is true, the 'for' loop breaks and the algorithm repeats again.
Now, x[3] becomes
x[3]=(1+1) mod 6= 2 mod 6 =2
If(x[3]=0) 2=0 false
For j=1 to 5
J=1 : if(G(3,1)!=0)&&(x[3]=x[1]) (G(3,1)!=0 T and 2=1 F) => T&&F=F
J=2 : if(G(3,2)!=0)&&(x[3]=x[2]) (G(3,2)!=0 T and 2=2 F) => T&&T=T
Since the above condition is true, the 'for' loop breaks and the algorithm repeats again.
x[3]=(2+1)mod 6=3 mod 6= 3
If(x[3]=0) 3=0 false
For j=1 to 5
J=1: if(G(3,1)!=0)&&(x[3]=x[1]) => (G(3,1)=1 T and 3=1 F) => T&& T= F
J=2 : if(G(3,2)!=0)&&(x[3]=x[2]) => (G(3,2)=1 T and 3=2 F) => T&&F=F
J=3 : if(G(3,3)!=0)&&(x[3]=x[3]) => (G(3,3)!=0 F and 3=3 T) => F&&T=F
J=4 : if(G(3,4)!=0)&&(x[3]=x[4]) => (G(3,4)!=0 T and 3=0 F) => T&&F=F
J=5 : if(G(3,5)!=0)&&(x[3]=x[5]) => (G(3,5)!=0 T and 3=0 F) => T&&F=F
J=6: loop condition fails and exits the for loop
Since all the above conditions have failed, we can assign '3' i.e. x[3]=3, color3 to node3.
The control passes back to the main algorithm.
If(x [3] =0) false as x [3] =3
If (k=n) false as 3=5 is false
The next step is Mcoloring (3+1) i.e. Mcoloring (4)

- Mcoloring (4) => Nextvalue (4)

x [4] = (0+1) mod 5= 1
x [4] =1
If(x [4] =0) false as x [4] =1
For j=1 to 5
J=1: if (G (4, 1)! =0) && (x [4] =x [1]) => (G (4, 1) =1 T and 1=1 T) =>T&&T=T

Since the above condition is true, the ‘for’ loop breaks and the algorithm is repeated again.
x [4] = (1+1) mod 5=2 mod 5= 2
x [4] =2
If(x [4] =0) false as x [4] =2
For j=1 to 5
J=1: if (G (4, 1)! =0) && (x [4] =x [1]) => (G (4, 1) =1 T and 2=1 F) =>T&&F=F
J=2: if (G (4, 1)! =0) && (x [4] =x [1]) => (G (4, 1) =1 T and 2=2 T) =>T&&T=T
Since the above condition is true, the ‘for’ loop breaks and the algorithm is repeated again.
x [4] = (2+1) mod 5=3 mod 5= 3
x [4] =3
If(x [4] =0) false as x [4] =3
For j=1 to 5
J=1: if (G (4, 1)! =0) && (x [4] =x [1]) => (G (4, 1) =1 T and 3=1 F) =>T&&F=F
J=2: if (G (4, 1)! =0) && (x [4] =x [2]) => (G (4, 1) =1 T and 3=2 F) =>T&&F=F
J=3: if (G (4, 1)! =0) && (x [4] =x [3]) => (G (4, 1) =1 T and 3=3 F) =>T&&T=T
Since the above condition is true, the ‘for’ loop breaks and the algorithm is repeated again.
x [4] = (3+1) mod 5=4 mod 5= 4
x [4] =4
If(x [4] =0) false as x [4] =4
For j=1 to 5
J=1: if(G(4,1)!=0)&&(x[4]=x[1]) => (G(4,1)=1 T and 4=1 F) => T&& F= F
J=2 : if(G(4,2)!=0)&&(x[4]=x[2]) => (G(4,2)=1 F and 4=2 F) => F&&F=F
J=3 : if(G(4,3)!=0)&&(x[4]=x[3]) => (G(4,3)=1 T and 4=3 F) => T&&F=F
J=4 : if(G(4,4)!=0)&&(x[4]=x[4]) => (G(4,4)=0 F and 4=4 T) => F&&T=F
J=5 : if(G(4,5)!=0)&&(x[4]=x[5]) => (G(4,5)=1 T and 4=0 F) => T&&F=F
J=6: loop condition fails and exits the for loop
Since, all the above conditions fail, we have x [4] =4 and we can assign 4 i.e. color4 to node4.
The control now goes back to the main algorithm and the steps after nextvalue (k) are executed.
If(x [4] =0) false as x [4] =4
If (4=5) false
Mcoloring (4+1) => Mcoloring (5)
The algorithm for k=5 gets executed,

- Mcoloring (5) => Nextvalue (5)

x [5] = (0+1) mod 5= 1
x [5] =1
If(x [5] =0) false as 1=0 is false
For j=1 to 5
J=1: if(G(5,1)!=0)&&(x[5]=x[1]) => (G(5,1)=0 F and 1=1 T) => F&& T= F
J=2 : if(G(5,2)!=0)&&(x[5]=x[2]) =>(G(5,2)=1 F and 1=2 F) => F&&F=F
J=3 : if(G(5,3)!=0)&&(x[5]=x[3]) => (G(5,3)=0 F and 1=3 F) => F&&F=F
J=4 : if(G(5,4)!=0)&&(x[5]=x[4]) => (G(5,4)=1 F and 1=4 F) => F&&F=F
J=5 : if(G(5,5)!=0)&&(x[5]=x[5]) => (G(5,5)=0 F and 1=1 T) => F&&T=F
J=6: loop condition fails and exits the for loop
Since all the above conditions fail, we have x [5] =1, we can assign 1 i.e. color1 to node5.
The control goes back to the main algorithm,
If (5=5) true
Write (x [1: n]) i.e. x [1] = 1, x[2]=2, x[3]=3, x[4]=4, x[5]=1.
The execution of the algorithm stops as all the nodes have been assigned colors and the conditions fail.
Hence, the execution has been completed.

V. RESULTS

The results for the above example as when calculated are x[1]=1, x[2]=2, x[3]=3, x[4]=4, x[5]=1. Thus, the solution for the m-colorability problem is {1, 2, 3, 4, and 1}.

VI. CONCLUSION

Hence, from the above observed results, we can determine the chromatic number of any given graph. The algorithmic execution provides a platform that can be used in wide number of applications. Graph coloring enjoys many practical as well as theoretical challenges. Some of the applications are Job Scheduling, Register Allocation, Pattern Matching, recreational puzzles like Sudoku, etc. Graph coloring is still a very active field of research.

ACKNOWLEDGEMENT

I hereby thank my guide and co author Assistant Professor P.S. CHANDANA, GITAM UNIVERSITY, Visakhapatnam for her constant guidance and support in completing the paper. I also thank all the authors mentioned in the references.

REFERNCES

[1] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, *Fundamentals of Computer Algorithms*, Galgotia Publications pvt ltd, 2002.
[2] Thomas h. Coremen, Charles E. Leiserson et.al. *Introduction to Algorithms*, 3rd edition, MIT Press.
[3] Aho, Hopecraft, Ullman, *The design and analysis of computer algorithms*.
[4] Michel T. Goodrich and Roberto Tamassia, *Algorithm Design*, *Foundations*, *Analysis and Internet Examples*, John Weily and Sons.
[5] Sara Baase, Allen Van Gelder, *Computer Algorithms: Introduction to design and analysis*, Pearson Education.
[6] John Kleinberg, Eva Tardos, *Algorithm Design*, Pearson.
[7] N. Alon, M. Tarsi, Colorings and Orientation of Graphs, Combinatotica 12(1992).