Chapter 1

1. Introduction

Generally, plane coloring is the process of assigning different colors to different points of the plane. It is becoming popular day by day, as it is used in many fields, such as map coloring (e.g. ATLAS), art gallery problem using cameras, allocations of frequencies in cell phone networks, etc. Coloring theory was started with the coloring the map of different countries in such a way that no two countries that have the common border receive the same coloring, different colors are used to color the neighboring states or countries or regions [10]. If we denote the countries by points in the plane, and connect each pair of points that correspond to the countries with the same border by a curve, we obtain a planar graph [1, 8]. Similarly, in the case of art gallery problem, cameras are used instead of guards for the security purpose, so model for art gallery problem can be created by assigning different colors to different regions, visible from different cameras. It is also used in the distribution of different radio frequencies, and frequency assignments in cellular networks [7] by coloring geometric regions. The regions of a plane are considered as nodes of a graph, and coloring the nodes is equivalent to coloring of the regions of a plane.

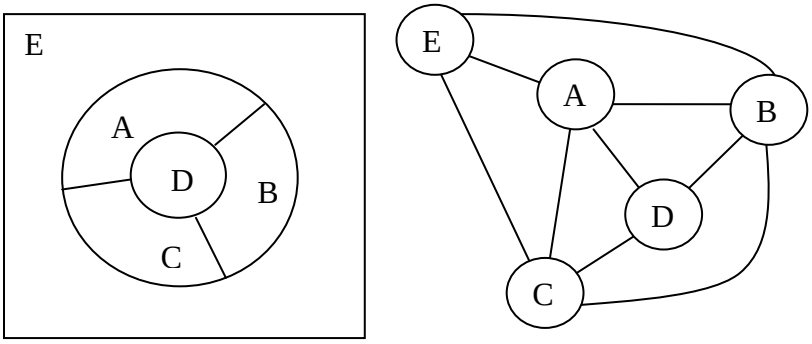


Figure 1.1: Plane coloring is equivalent to Graph coloring.

Figure 1.1 shows plane can be represented using graphs and can be colored subsequently using graph coloring. Here, assigning color to region A is equivalent to assigning color to

node A, and similarly, other planes B, C, D and E can be colored with the same color corresponding to the nodes of a graph B, C, D and E.

Basically, unit distance graph can be drawn on a plane by taking any two points as adjacent vertices, such that the distance between these two points must be a unit. The unit distance graph can be colored likely as simple graph, and the adjacent nodes which are unit apart, are colored by distinct colors. The concept of unit distance graphs can be used in plane coloring in such a way that the points on a plane are taken as nodes of unit distance graph. Since, the technology is developing rapidly in the field of networks. Such as, various types of wireless networks, including cellular networks in which the nodes represent broadcast stations with a uniform broadcast range. A network that divides a geographic area into cells such that the same frequency can be reused in two cells that are a certain distance apart is known as cellular networks. The networks that can be modeled using unit distance graphs as unit distance wireless networks (UDW) [7], in which every station for these networks are equidistance from each other, and the frequency is distributed uniformly, such that neighboring cells have different frequencies. The distribution of a frequency is modeled by coloring the regions such that no two adjacent regions are monochromatic. A model of cellular networks can be drawn using unit distance graph, assuming stations as nodes. Coloring the nodes is equivalent to coloring the frequency region. Conventionally, hexagonal regions are used as a model for the frequency regions [7]. One broadcast station is kept in one hexagonal region, and the frequency is broadcasted within the hexagonal region. For example, in the case of mobile networks, the frequency stations are of the same network, and it shows the name of particular network station if a cell phone comes under the coverage of its frequency. The frequency allocation problem in wireless networks is equivalent to the problem of multicoloring the nodes of a graph. In particular, given the interference graph of a wireless cellular network, if we imagine frequencies as colors, then allocating frequencies to users of the network is equivalent to multicoloring the nodes of the graph. And, seven different colors used in the model of hexagonal cells [1] indicate the number of different frequency stations. Colors indicate the frequencies, so minimum number of colors used in the plane coloring for cellular networks implies number of frequencies used in different cells. Frequency reuse is the main idea behind cellular networks; similar colored hexagons show the use of same frequency but some distance apart, to avoid the use of same frequency in adjacent regions.

As we have mentioned that the plane coloring can be equivalent to graph coloring, hence, finding the chromatic number of any graph can be the chromatic number of the plane for which the graph is configured. There are many heuristic based algorithms, which are used in the field of graph coloring. Since, heuristic algorithms are used to find the chromatic number of any graph, so they can be used in the field of graph coloring and unit distance graph coloring. Graph coloring is an assignment problem [2, 4] i.e. the assignments of colors to the vertices. The simple heuristic for the graph coloring is the selection of vertex and assignment of color to particular vertex. For example, heuristic might be the ordering of vertices in certain order before the assignment of the colors in graph coloring process. Heuristic gives the good and fast coloring of a graph. Such as, in register allocation, heuristic algorithms are used to find the minimum number of registers used in the program execution.

1.1 Thesis Organization:

After the brief discussion on introduction and applications of plane coloring problem, and different graph coloring algorithms, we organize the thesis through following chapters;

In Chapter 2, we give a brief description of the chromatic number of a plane, and various heuristic based graph coloring algorithms. Further it consists of the background used in finding the result.

In Chapter 3, we describe some of the researches which have been done in the field of plane coloring, and various graph coloring algorithms. The chapter covers the detail literature review.

In Chapter 4, we describe the implementation model of this research work. All the algorithms used in this study are implemented in MATLAB. In this chapter, we describe the creation of unit distance graphs, specific unit distance graphs as test cases, and coloring them using the graph coloring algorithms.

In Chapter 5, we analyze different coloring algorithms on the basis of elapsed time and number of colors, by showing the results through tables and line charts.

In Chapter 6, we summarize the thesis and briefly describe the future scope of this research work in the field of cellular networks and register allocation problem.

Chapter 2

2. Problem Formulation and Background

2.1 Problem Definition

The chromatic number of the plane is the minimum number of colors that are needed to paint all points in the plane, so that no two points adjacent to each other are colored alike. This number is called the chromatic number of the plane and is denoted by χ. To color the plane means to assign one color to every point of the plane. In general, even though there has been lots of researches done in chromatic number of planes, but still finding the optimal chromatic number is an open problem. Besides this, the time in the coloring of a plane has its significance in many applications. In this context, this study will include implementation and analysis of coloring different unit distance graphs using different heuristic based algorithms viz. Contraction based Recursive Largest First [8], Sequential (DSATUR) algorithms [9], and Indexed Degree Ordering (IDO) based algorithm [2, 4], so as to result the optimal coloring unit distance graphs. And the comparison will be done between these heuristic algorithms, and they will be analyzed on the basis of elapsed time, and the number of colors required, in the coloring of unit distance graphs. Coloring the unit distance graph is equivalent to hexagonal plane coloring used in cellular networks, and different coloring indicates distribution of different frequencies over hexagonal regions. So, fast coloring of a hexagonal planes, determines the fast assignment of frequency in hexagonal cells. Similarly, minimum use of colors and fast coloring has the equal importance in the register allocation problem. Minimum colors show less use of registers, and which improves the performance of a program. Thus, coloring algorithms are compared in terms of chromatic numbers and elapsed time. Hence, this study is also dealt with the time required to color the plane.

2.2 Background of the study

For this study, it requires some basic terminologies. The Euclidean plane is parameterized by coordinates; so that each point is located on a plane is based on its position with respect to two perpendicular lines, called coordinate axes. The points located on a Euclidean plane are in the pair of x and y co-ordinates (i.e. (x, y)) and the positions of particular points in a plane are indicated by xy co-ordinates. Let $\mathbb{E}^2$ be an Euclidean plane and the minimum number of

colors used in coloring of an Euclidean plane, is called the chromatic number of a plane and is denoted by $\chi(\mathbb{E}^2)$. Let G = (V, E) be the graph generated from the plane by taking the points of the plane as vertices V, and E be the edges between these vertices. Two vertices are said to be adjacent, if they are connected by an edge $e \in E$.

A graph is a pair (V, E) of a set of vertices V and set of edges E. The elements of V are called vertices and the elements of E are called edges. Each edge is identified with a pair of vertices. Our discussions, in this thesis are concerned only with undirected graphs. We use the symbols $v_1, v_2, v_3, \ldots$ to represent the vertices and the symbols $e_1, e_2, e_3, \ldots$ to represent the edges of a graph. The vertices v_i and v_j associated with an edge e_l are called the end vertices of e_l. The edge e_l is then denoted as $e_l = v_i v_j$. A graph is called a simple graph if it has no parallel edges or self-loops. In this thesis, we are working with the simple graphs only. A graph G is planar if there exists a drawing of G in the plane in which no two edges intersect in a point other than a vertex of G. A degree of a vertex v is the number of vertices adjacent to v, and is denoted by deg(v), saturation degree of a particular vertex is defined as the number of differently colored vertices in the neighborhood, and the incidence degree of any vertex is the number of colored vertices in the neighborhood.

A graph G is a unit distance graph if the length of each edge is one unit (i.e. the adjacent nodes must be unit distance apart). Let G be a unit distance graph, and $\chi(G)$ be the minimum number of chromatic number used in the coloring of the G, such that adjacent vertices are unit distance apart, bearing distinct color. The unit distance is defined as the Euclidean distance between two points (x_1, y_1) and (x_2, y_2) must be a unit, in co-ordinate system.

The coloring of a unit distance graph G = (V, E) is a mapping C: $v \rightarrow s$, where "s" is a finite set of colors, such that if $vw \in E$ then $C(v) \neq C(w)$. In other words, adjacent vertices unit distance apart, are not assigned the same color. The problem that arises is the coloring of a graph provided that no adjacent vertices have the same color. The chromatic number $\chi(G)$ is the minimum number of colors needed for a coloring of G. A graph G is k-chromatic, if $\chi(G) = k$, and G is k-colorable, if $\chi(G) \leq k$ [4, 7].

Basically, graph coloring is the process of coloring the nodes of a certain graph such that, adjacent nodes must be of different colors, it is also known as vertex coloring. Applications of graph coloring are as follows;

- It has been used in the frequency assignments in cellular networks and radio stations [2].

- In register allocation process, the variables are the nodes of a graph and the registers used are the number of colors used in graph coloring. Two variables needed at the same time cannot be kept in the same register, and these two variables are adjacent, which are colored differently. So, finding the minimum number of colors helps the use of minimum number of registers in any computation of a program. [5]

- Many applications such as map coloring (assuming countries as nodes in a map and assigning color to each node such that the adjacent nodes or countries are colored uniquely), art gallery problem where the cameras are assumed as nodes and coloring them such that the adjacent regions or cameras are colored differently.

- Many other applications such as; distribution of items: e.g. set of animals which can and cannot live together can be assumed as vertices of a graph which are colored by different colors. Similarly, set of plants that can and cannot kept together, set of food items which can and cannot consumed together, set of people who can and cannot stay together, are distinguished using different colors.

Chapter 3

3. Literature Review

Lots of researches have been done in the field of plane coloring. There have been many researches done for finding the chromatic number of planes using unit distance graphs. And, so many heuristic algorithms based on the ordering of vertices were developed in order to color the graph.

3.1 Simple Unit Distance Graphs

Lots of researches has been done in unit distance graphs for plane coloring among which some of the specific and simple unit distance graphs are; Grid graph, Cycle graph, Wheel graph, Star graph, which are determined in [1, 6]. The chromatic number for a wheel graph (W_n) containing odd number of nodes is determined to be three, and four for even number of nodes. Similarly, a cycle graph contains a cycle through all nodes where the chromatic number is determined to be 3 for odd nodes and 2 for even nodes. For star graph the chromatic number is determined to be two, no matters whether n are odd or even. Similarly, the chromatic number of a grid graph is found to be two.

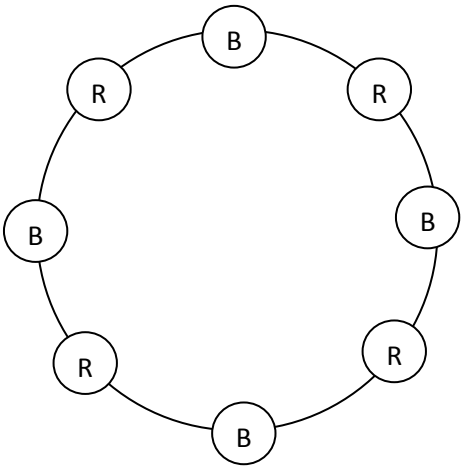


Figure 3.1: Even nodes Cycle graph.

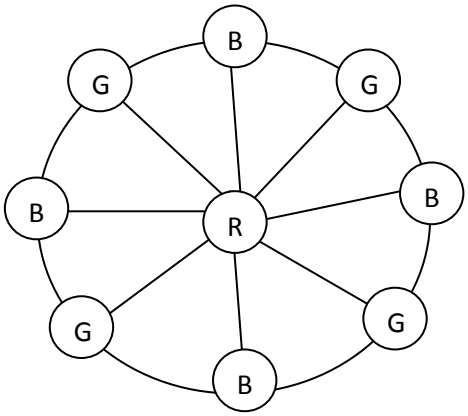


Figure 3.2: Odd nodes Wheel graph.

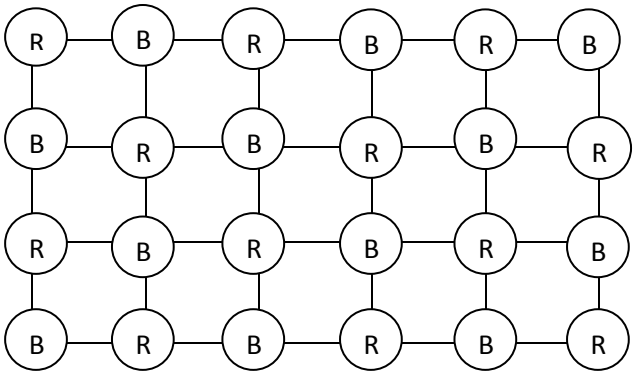


Figure 3.3: Grid graph.

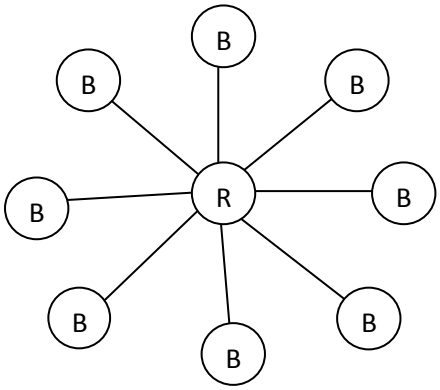


Figure 3.4: Star graph.

3.2 Four coloring of a plane using Moser graph

The basic coloring of a plane is originated from the two coloring the nodes as end points of line segment of length one unit, but it was not possible to color the plane with just two colors [6]. Then, three coloring of a plane is found with the help of equilateral triangle of each side length one unit, shown in figures below;

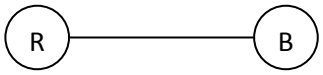


Figure 3.5: Two coloring of a line.

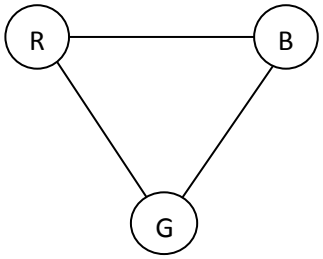


Figure 3.6: Three coloring of a plane using equilateral triangle.

In figure 3.5, the endpoints R and B of a line segment are colored by two different colors Red R and Blue B. Similarly, in figure 3.6, the vertices R, B and G formed an equilateral triangle,

so all three vertices have different colors Red R, blue B and green G, respectively as they are unit distance apart. The base for the four coloring problem is developed from the three coloring of a plane i.e. equilateral triangle.

The authors in [1] have shown that plane can be four colored. The four coloring for the plane coloring is given by Moser graph. Leo Moser and William Moser [1, 6] found a graph called Moser graph, which is a unit distance graph which strictly requires four colors.

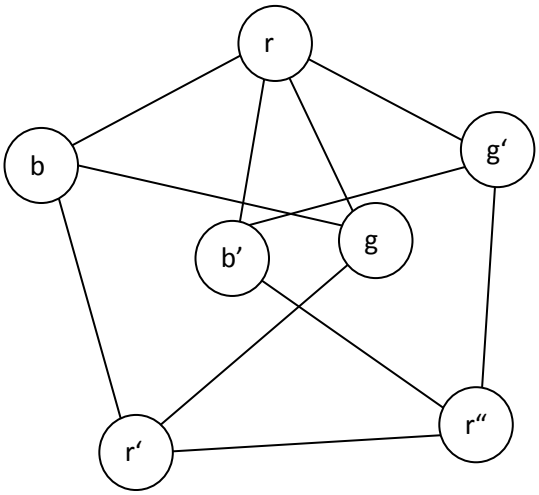


Figure 3.7: Moser graph.

Hugo Hadwiger [1] in 1961, proposed that Moser graph can be made up of four equilateral triangles or two rhombuses with each side of length one. Each equilateral triangle needs three colors; let these colors be red r, blue b, and green g. Let r, b, r' and g be one rhombus. If the rhombus is rotated around the point r through some angle, a newly formed rhombus be r, b', g' and r", and the point r is rotated in such a way that r' is unit distance away from r" so that there must be an edge of length one unit. It needs three colors as it is formed by equilateral triangle but at point r" all three colors red, blue and green are not possible so there must be another color say yellow.

3.3 Nine coloring of a plane

The authors of [1] had shown that the plane can be colored by 9 different colors. It is proved by tiling the plane with unit squares and coloring one square by one color and its eight neighbors in colors 2, 3… 8, 9. Obviously, from figure 3.7, every adjacent unit squares have a

distinct color. It is realized that the unit distance between any two points in the plane in figure 3.8, must be in the range $\sqrt{2} < d < 2$ to be colored uniquely. The length of diagonal of each unit square is $\sqrt{2}$ unit, which can be derived with the help of Pythagoras Theorem, and same colored unit squares are 2 units apart.

8	9	7	8	9	7	8	9
1	2	6	1	2	6	1	2
4	3	5	4	3	5	4	3
8	9	7	8	9	7	8	9
1	2	6	1	2	6	1	2
4	3	5	4	3	5	4	3
8	9	7	8	9	7	8	9
1	2	6	1	2	6	1	2
4	3	5	4	3	5	4	3

Figure 3.8: Unit square-based 9-coloring of a plane.

3.4 Seven coloring of a plane

Hadwiger [1] improved nine coloring of a plane by using hexagon tiling rather than unit square tiling. The solution for this problem is given by tiling the hexagons of side 1 unit. Hexagon first colored with red color and six neighbors colored with blue, green, yellow, brown, pink, and violet colors, so that no two adjacent hexagons had same color. In the case of hexagonal plane coloring, it is realized that the distance between any two adjacent points in the plane formed by tiling the hexagonal planes, is in the range $2 < d < \sqrt{7}$, to be colored distinctly, where 2 unit is the length of a largest diagonal of a hexagon and $\sqrt{7}$ units is the distance between two similar colored hexagons. Obviously, the length of largest diagonal of a hexagon is 2, so the distance between any two adjacent points must be greater than 2, to be colored distinctly. And, in the case $d < \sqrt{7}$, we can find it by assuming the isosceles triangle as shown in the figure 3.7, and eventually, by using Pythagoras Theorem we find the minimum distance between two same colored hexagonal planes is $\sqrt{7}$. If we assume d=2.1unit, then two points distance d unit apart are definitely adjacent points and colored uniquely. This type of plane coloring restricts the same colors between neighbors. In cellular networks, hexagonal

planes are used as a unit distance wireless networks (UDW) [7]. In UDW, each hexagon is taken as a region of particular broadcast station, and different frequencies are distributed among different hexagons. The group of frequencies can be reused in other cells, provided that the same frequencies are not reused in adjacent neighboring.

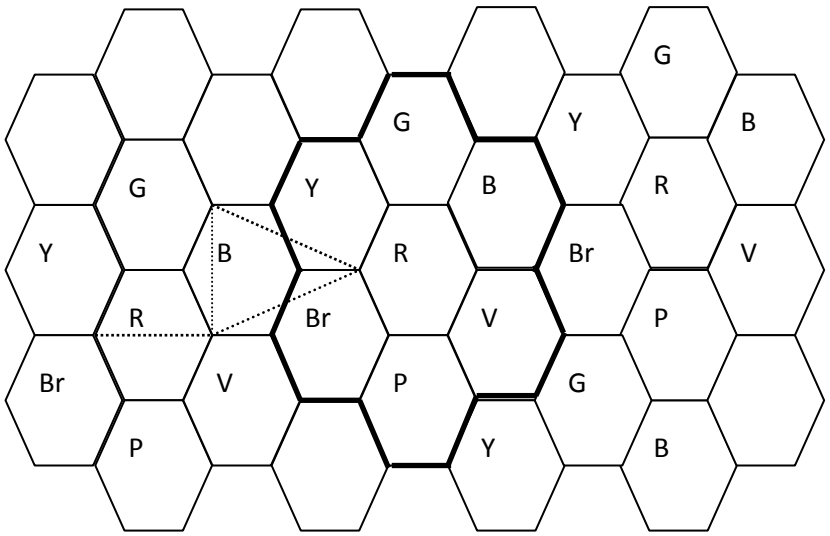


Figure 3.9: Hexagon-based seven coloring of the plane.

3.5 Three coloring of a plane via Petersen graph

Erdos, Hararay and Tutte [3], had shown that all generalized Petersen graphs are planar unit distance graphs (i.e. the length of each edge is a unit), which are three colorable. It is also shown that the Petersen graph of Figure 3.10 (a) can be drawn in the Euclidean plane in such a way that the vertices are mapped to distinct points in the plane and edges to line segments of length one. They proposed unit-distance representation of the Petersen graph with rotational symmetry in the Euclidean plane that can be seen in Figure 3.10 (b). The special kind of unit distance graph was proposed which requires three colors, which shows that the plane can be three colorable, since the plane can be denoted by the graphs and colored via graph. The drawing can be obtained from the standard drawing of the Petersen graph by suitably scaling the inner pentagram and rotating it against the outer pentagon, in such a way that the edges connecting the pentagram with the pentagon become of length one. This procedure is known as twist. So, we can represent the plane as Petersen graph resulting three colors for the plane.

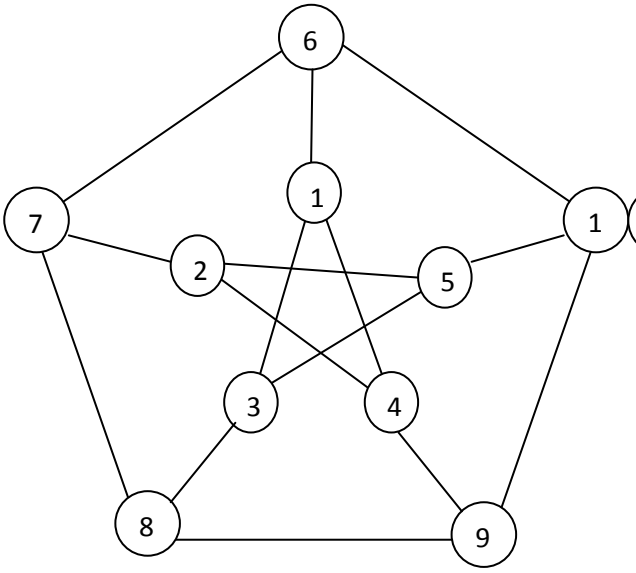


Figure 3.10(a): Petersen Graph.

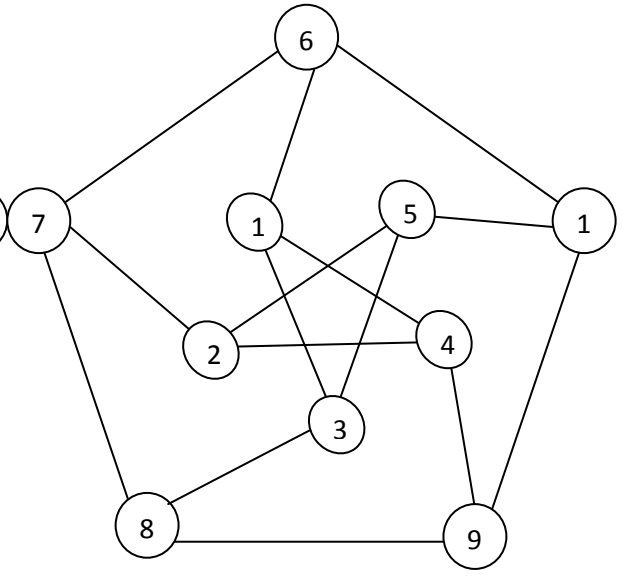


Figure 3.10 (b): Symmetrical Petersen graph using rotational symmetry.

3.6 Graph Coloring Algorithms

Number of algorithms on graph coloring can be found in [2, 4, 8]. Author of [8] has given the idea of contraction for graph coloring in heuristic greedy approach. And, also in [2, 4], there are some heuristics used in order to color the graph.

3.6.1 Contraction based RLF graph coloring algorithm

In graph theory, contraction is defined as the process of merging one vertex into another and all the links to the merged vertex are forwarded to that vertex where it is merged. Formal definition of vertex contraction is; for any non-adjacent vertices x and y we denote contraction by *G/x,y* i.e. graph after contracting y into x, y is deleted, and the neighbor set of x say $N_G(x)$ becomes $N_G(x) \cup N_G(y)$, where $N_G(y)$ is the neighbor set of y. The widely used contraction based algorithm is the Recursive Largest First (RLF) [8]. RLF is a heuristic based greedy algorithm; heuristic preference is the degree of vertex. Ordering of vertices in non-increasing order makes it heuristic and selection of maximal degree vertex at every step makes it greedy. It colors the graph by coloring as many vertices with one color as possible, then moving on to the next color. The steps for RLF algorithm are as follows;

Given a graph G = (V, E), with vertex list V and edge list E.

- *Determine a vertex x of maximal degree in G.*
- *Color x with the first available color.*
- *Find the non-neighbor vertices of x.*
- *Contract the non-neighbor vertices into vertex x, and the contracted non-neighbors are given the same color as x.*
- *If any two non-neighbor vertices are adjacent, then any one of them is contracted.*

  Contraction removes the contracted vertex and all the edges associated with it, but the neighbors of the contracted vertex become the neighbor of x.
- *Remove the vertex x from G.*
- *Vertex x and all vertices contracted into x constitute a color class.*

Continue the process until all the vertices of G are colored.

3.6.2 Ordering based heuristic graph coloring algorithms

The heuristic is one of the methods of problem solving, which selects one solution among many alternative solutions, it is used to boost the performance of any algorithm, and find a good solution quickly. Simply, heuristic is the change in any algorithm for better performance. And, the greedy heuristic makes the choice that seems best at the moment. Greedy algorithms are free of backtracking, e.g. in graph coloring, greedy means once the vertex is assigned a color; it never changed. Some of the heuristics which can be used in graph coloring algorithms in [2, 4] are as follows;

Largest Degree Ordering (LDO): In this heuristic, the vertices are ordered in non-increasing order. Let vertices $v_1, v_2 \ldots v_{i-1}$ have been chosen and colored, vertex v_i is chosen to be the next vertex with maximum degree among the set of uncolored vertices. Maximum degree means, the vertex having highest number of neighbors. E.g. if the vertex *v* has four neighbor vertices, then the degree of *v* is 4, and if it is the highest degree among other vertices then *v* is the largest degree vertex.

Saturation Degree Ordering (SDO): The famous heuristic which can be used in the graph coloring algorithm is the SDO. Let the vertices of a graph be $v_1, v_2 \ldots v_{i-1}$ have been chosen and colored. Then at step *i*, vertex v_i with the maximum saturated degree is selected. The

saturation degree of a vertex is defined as the number of colors adjacent to the vertex, in the neighboring vertices. E.G. if a vertex *v* has degree equal to four where one of its neighbors is uncolored, two of them are colored with color equal to 1, while the last one is colored with color equal to 3 then *v* has saturation degree equal to two.

Incidence Degree Ordering (IDO): Another heuristic is the IDO. Let v_1, v_2... v_{i-1} are the vertices have been chosen and colored. Vertex v_i with the maximum incidence degree is chosen at step *i*. The incidence degree of a vertex is defined as the number of its adjacent colored vertices rather than the number of differently colored vertices. For example, if a vertex *v* has degree equal to 6, where one of its neighbor is uncolored, two of its neighbors are colored with color 1, and three are colored with color 2 then *v* has an incidence degree equal to five.

3.6.2.1 DSATUR sequential graph coloring algorithm

Sequential Coloring (SC) [8] performs the coloring of a graph, according to the degree of the vertices. Let a_0, a_1..., a_{n-1} are the vertices of a graph, and then the ordering of these vertices should be in non-increasing order;

- $d(a_0) > d(a_1) > \ldots > d(a_{n-1})$, where *d* denotes the degree of a vertex.

One of the widely used SC algorithms is the DSATUR (Degree of saturation) algorithm [8, 9]. Since, it uses a heuristic SDO; heuristic preference is the degree of saturation of a vertex, which changes the ordering of vertices and then uses the greedy method to color these vertices. The saturation degree of any vertex *x*, $deg_s(x)$, is the number of different colors at vertices adjacent to *x*. DSATUR starts by assigning color 1 to a vertex with the maximal degree. The vertex to be colored next in the sequential coloring procedure of DSATUR is a vertex with maximal $deg_s(x)$. The steps for DSATUR algorithm, when the set of vertices of a graph G are given, are as follows;

- *Arrange the vertices by non-increasing order of degrees and keep the colors in the array.*

  $d(a_0) > d(a_1) > d(a_2) > \ldots\ldots\ldots > d(a_n)$,
- *Color a vertex of maximal degree with color 1.*
- *Find the set of uncolored vertices, which includes;*

- *Choose a vertex with maximal saturation degree. This means that we have to choose the vertex that has most number of unique neighboring colors.*
- *If there is equality in saturation degree, choose a vertex of maximal degree in the uncolored sub-graph.*

- *Color the chosen vertex with the least possible color from the array of colors.*

If all the vertices are colored, stop. Otherwise, return to 3rd step.

3.6.2.2 Incidence Degree Ordering heuristic algorithm

The authors of [2, 4] have given the heuristic incidence degree ordering (IDO) graph coloring algorithm working with Largest Degree Ordering (LDO); heuristic preference is the degree of incidence of a vertex. There are two criteria for choosing the vertex to be colored:

- The number of vertices connected to the vertex LDO, and
- The number of colored vertices connected to the vertex IDO.

The steps which are used in this algorithm are given as;

- *Initialization of NoOfColoredNodes = 0.*
- *While NoOfColoredNodes < NoOfNodes*
- *Initialize max = -1.*
- *For every vertex x in the vertex set;*
  - *If the vertex x is not colored.*
    - *Find the degree, deg(x).*
    - *If deg(x) is greater than max;*

      Max = deg(x) and index is the index(x).
    - *If deg(x) = max;*

      If incidence degree of i^{th} vertex in vertex set is greater than incidence degree of x, then index is the index of i^{th} vertex.
- *Color the vertex which is on the 'index' position of vertex set.*
- *NoOfColoredNodes is increased by one.*

Repeat the steps until all the nodes are colored.

Chapter 4

4. Implementation

4.1 Implementation model

The graph coloring algorithms are implemented so as to color Euclidean planes (plane coloring-graph coloring). The algorithms are simulated for various flavors of unit distance graphs such as Petersen graph, wheel graph, cycle graph, grid graph and star graph. The contraction based recursive largest first (RLF) heuristic algorithm, heuristic based sequential (DSATUR), and incidence degree ordering (IDO) heuristic algorithms are implemented and tested in Matlab R2011b. The configuration of the test machine is on Intel® core™ i5-2410M CPU @ 2.30 GHz, with 2Gb RAM in Windows 7 Professional 64 bit Operating System.

4.2 MATLAB

Among many programming languages, MATLAB is one, which is a powerful language for technical computing. It is developed by Mathworks (Multi-national Corporation that specializes in mathematical computing software). The name MATLAB stands for MATrix LABoratory, because its basic data element is a matrix (array). Matlab can be used for math computations, modeling and algorithm development. It can also display information graphically. It can allow interfacing with programs written in other languages, including C, C++, JAVA, and FORTRAN. The standard MATLAB program has functions or tools that can be used to solve problems. In addition, it has optional toolboxes that are the collections of specialized programs designed to solve the specific types of problems. In this study, we have used MATGRAPH toolbox.

4.2.2 Matgraph

Matgraph is a toolbox for working with simple graphs in MATLAB. To create unit distance graphs, Matgraph provides some important ideas such as, joining vertices by edges, checking whether two vertices has any edge or not, deletion or addition of vertices or edges etc. All graphs handled by Matgraph are simple and undirected. And the vertex set of all graphs in Matgraph is always in the form *{1, 2, 3… n}*, where, $n \geq 0$ is the number of vertices.

4.3 Implementation model

The flowchart of the implementation model is given as;

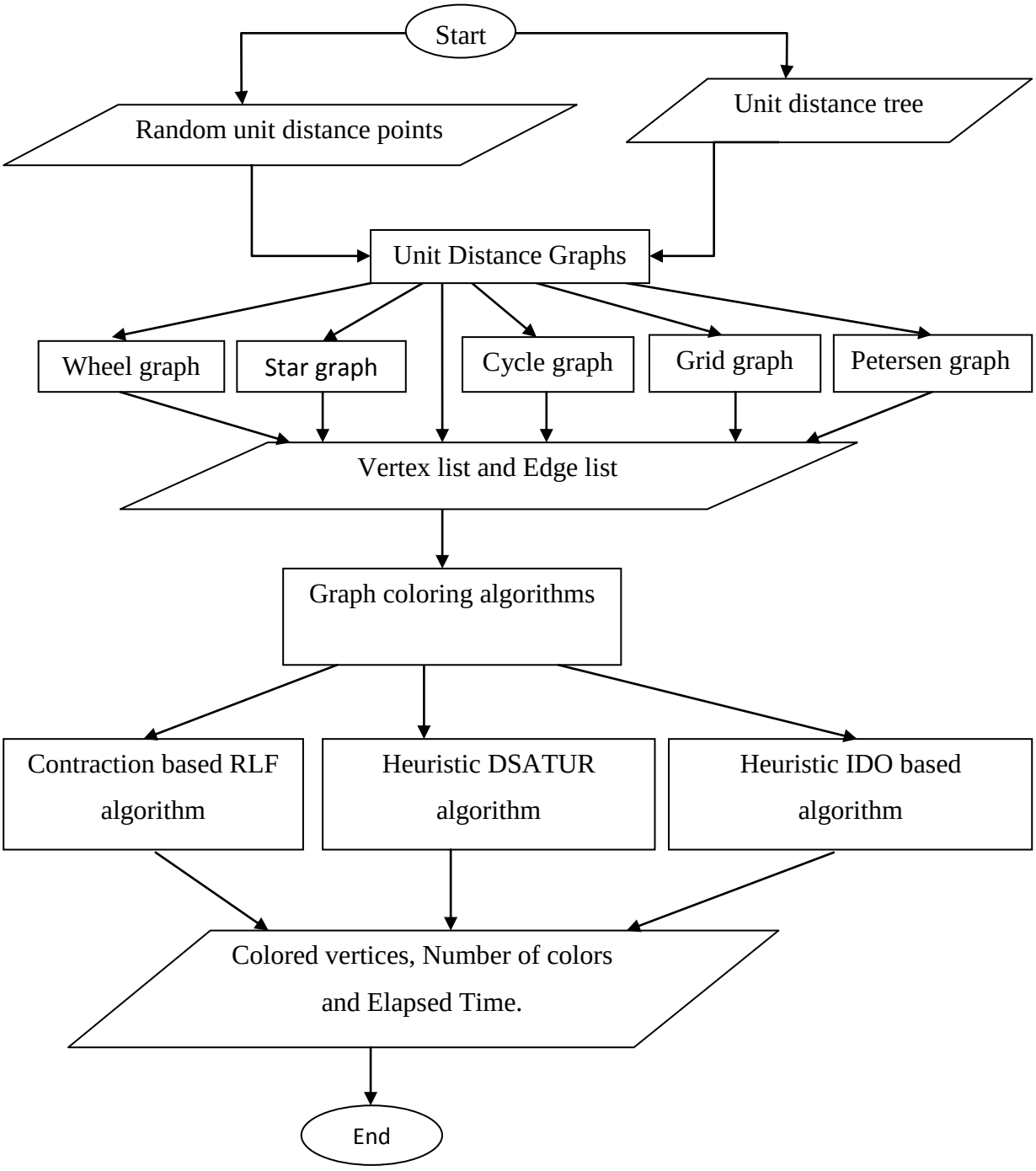


Figure 4.1: Implementation model.

4.4 Implementation of Unit Distance Graph

Unit distance graph is created by taking the points on a plane in such a way that the distance between adjacent vertices must be one unit. In general, it is not easy to draw these types of graphs with the random points. So, there are some ideas which give the way of drawing unit distance graphs. The procedures which have been used in the creation of UDG are as follows;

- By choosing the unit distance points from the random points with the help of distance formula, i.e. $\sqrt{(x_1- x_2)^2 + (y_1-y_2)^2} = 1$. Code sample for selecting the random unit distance points is mentioned below;

Code sample 1:

```
Function [xy] = Unit_Distance_Points (100)
 XY=rand(100, 2);
   for i=1:100
     for j=i+1:100
        X1=XY(i,:);
        X1=XY(j,:);
        Distance = Sqrt(((X2(1,1) – X1(1,1))^2 + X2(1,2) – X1(1,2))^2);
        if Distance == 1
        xy = [xy; XY(i,:) ; XY(j,:)];
      End
    End
End
```

Listing 4.1: Selection of unit distance points.

- By using the unit distance tree, which is created by choosing some random points on the circumference of a unit circle (a circle of unit radius), and then taking these points as centers of new circles. New circles are drawn, and same process is repeated for these new circles. Then, the centers and intersection points of corresponding circles are joined in order to get a unit distance tree.

Code sample 2:

```
Function [t] = unit_distance_tree(g, v_list, e_list)
    vertex = v_list(end,1);
    g=graph(vertex);
    embed(g, vertex);
     for i=1:size(e_list,1)
      add(g, e_list(i,1), e_list(i,2));
    End
  ndraw(g);
End (function)
```

Listing 4.2: Creation of unit distance tree.

Now, unit distance graph can be created, by using random points, making each vertex as a center of a circle, drawing the circles with radii of one unit. After that, the intersection points of all circles are determined and lines are drawn from the centers of intersecting circles to the corresponding intersecting points. Similarly, UDG can be drawn by using Circle Intersection method to draw a unit distance graph from the unit distance tree, by taking each tree node as a center of circle.

Code sample 3:

```
Function [g] = Unit_Distance_Graph(vertex)
V_coords = xy;
No_of_circles = size(v_coords,1);
Centers = v_coords;
For i = 1: no_of_circles
   For j = i+1: no_of_circles
   Center1=centers(i,:);
   Radius1=1;
   Center2=centers(i,:);
   Radius2=2;
   Line(Center1, Center2);
   [intersect_1, intersect_2] = my_circle_intersection(center1, radius1, center2, radius2);
   Line(Center1,intersect1); Line(Center2,intersect1);
   Line(Center1,intersect2); Line(Center2,intersect2);
    End (for j)
 End (for i)
 v_list={Center1, Center2, intersect1, intersect2};
 g=graph(size(v_list));       % creating graph of size v_list .
 End (function)
```

Listing 4.3: Creation of unit distance graph.

4.5 Implementation of Simple Unit Distance Graphs

Simple unit distance graphs include Petersen graph, Wheel graph, Cycle graph, Grid graph, and Star graph. We have taken these unit distance graphs as test cases in this study, and the modules of these test cases in MATLAB are given below.

- Petersen graph: The Petersen graph is a unit distance graph of 10 vertices and 15 edges. Following is the code sample for the creation of Petersen graph;

Code sample 4:

```
Function [P] = Petersen_graph(e_list)
    g = graph(10);
    e_list = [1 6; 2 7; 3 8; 4 9; 5 10;1 2; 2 3; 3 4; 4 5; 5 1; 6 8; 7 9; 8 10; 9 6; 10 7];
    add(g,e_list);
    ndraw(g);
End
```

Listing 4.4: Creation of Petersen graph.

- Cycle Graph: A cycle graph or circular unit distance graph is a graph that consists of some number of vertices connected in a closed chain. Code sample for creating cycle graph is given as;

Code sample 5:

```
Function [C ] = cycle_graph(no_of_vertices)
    g = cycle_graph(11);
    ndraw(g);
```

End

Listing 4.5: Creation of cycle graph.

- Wheel Graph: Wheel graph is a graph with n vertices, formed by connecting a single vertex to all vertices of (n-1) cycles. The code sample for the creation of a wheel graph is;

Code sample 6:

Function [W] = wheel_graph(no_of_vertices)

g = wheel_graph(20);

ndraw(g) ;

End

Listing 4.6: Creation of wheel graph

.

- Grid Graph: A grid graph is a unit distance graph whose vertices correspond to the points in the plane with integer coordinates, X-coordinates being in the range 1, …., n, y-coordinates being in the range 1, …, m, and two vertices are connected by an edge whenever the corresponding points are at distance 1. Module for grid graph is as follows;

Code sample 7:

Function [G] = grid_graph(no_of_vertices)

g = grid_graph(5, 5);

ndraw(g);

End

Listing 4.7: Creation of grid graph.

- Star Graph: It is a graph of n vertices formed by connecting a single vertex to all other vertices, and distance between single vertex and others is a unit. The module for star graph is shown as;

Code sample 8:

```
Function [ S] = star_graph(no_of_vertices)
    g = star_graph(26);
   ndraw(g);
End
```

Listing 4.8: Creation of star graph.

4.6 Implementation of Graph Coloring Algorithms

During this study, we have implemented four coloring algorithms in order to color different unit distance graphs. The purpose for the implementation of these heuristic based coloring algorithms is to color different unit distance graphs with minimum chromatic number.

4.5.1 RLF Graph Coloring Algorithm

Since, contraction is based on the merging of vertex into another vertex, and is used in coloring the graph and finding the chromatic number of any graph. Here, we have used the Recursive Largest First algorithm (RLF), which is contraction based heuristic algorithm for the graph coloring purpose. The code sample used in RLF algorithm is as follows;

Code sample 9:

```
Function [coloring ] = my_rlf(v_list, e_list)
  n = numel(v_list)
  colors = 0;
```

```
Degree = calculate_degree(v_list, e_list);
While n>0
    temp = max(degree(v_list));
    colors = colors + 1;
    coloring (temp) = 1;
     nn = non-neighbors(temp) ;
   nb = numel(nn);
    While(nb>0)
      maxcn = -1;
      y_degree = -1;
       for i = 1:numel(nn)
           cn = numel(common neighbor(temp, nn(i) ));
            if cn>maxcn || (cn=maxcn && deg(nn(i)>y_degree)
               y = nn(i);
               y_degree = deg(nn(i));
                maxcn = cn;
             End
        End (for)
        If maxcn == 0
           y = max(deg(nn));
        End
        coloring(y) = colors;
```

```
        contract(v_list, E_list, temp, y);
    End (while nn)
    v_list = setdiff(v_list, temp);
  End (while n)
End (function)
```

Listing 4.9: The contraction based RLF algorithm.

The contraction based RLF algorithm is used to color the vertices of unit distance graph such that the adjacent vertices must be colored differently. This is a code for coloring the vertices by choosing non-neighbors of a particular vertex and it assigns the same color to the vertex and its non-neighbors under some condition. This piece of code works on contraction i.e. let v vertex is contracted into u, then, the edges incident on v becomes the incident edges of u and the vertex v is deleted.

4.5.2 DSATUR sequential graph coloring algorithm

This is a heuristic based greedy graph coloring algorithms, which manage the vertices in decreasing order using LDO heuristic, and choosing the next vertex on the basis of SDO heuristic. The code sample for DSATUR algorithm is given as;

Code sample 10:

```
Function [ coloring ] = DSATUR(v_list, e_list)
  available_colors = 1;
 for i = 1:numel(v_list)
     Degree(i, 1) = size([E(find(E(:,1)==v),2 ; E(find(E(:,2)==v,1)],1);
  End
  for i=1:numel(v_list)
```

```
        if i==1

        [ value index ] = max(Degree);

        v = index(1);

        coloring(v) = 1;

        else

          Uncolored = find(coloring==0);

          Temp=   find(sat_degree(Uncolored)=max(sat_degree(Uncolored));
           [ v index1 ] = max(Degree(Uncolored));
           Neighbors = neighbor(v);
        for j = 1: colors
           If coloring(neighbors)==j,1)==0
              coloring(v) = j;
           End
        End
    End

End (Function)
```

Listing 4.10: DSATUR algorithm.

This code colors the first vertex on the basis of maximal degree and colors the next vertex of a graph on the basis of maximal saturation degree, and in the case of equal saturation degree of two or more vertices in uncolored set of vertices, the code go for the largest degree vertex from the set and colors it the next available color.

4.5.3 IDO based algorithm

The incidence degree ordering (IDO) heuristic is the ordering of vertices according to the degree of incidence. Incidence degree of any vertex is the total number of colored vertices adjacent to it. The code sample used in the implementation of this algorithm;

Code sample 11:

```
Function [ ] my_ido(v_list, e_list)
    Coloring(:, 2) = 0;
    NoOfColoredNodes = 0;
    While(NoOfColoredNodes<numel(v_list))
      Max = -1;
      For i = 1:numel(e_list)
        If coloring(v_list(:, 2) = 0)
          D = degree(V(i), 2);
          If D>Max
            Max = D; index = i;
        End
        If D = Max
            If my_id(v_list, e_list, coloring(v_list(i)) > my_id(v_list, e_list,
              coloring(v_list(index));
            index = i;
          End
      End (if)
    End (for)
    Coloring = my_color(e_list, coloring, v_list(index));
  End (while)
End (function)
```

Listing 4.11: IDO based algorithm.

This code is used to color the vertices of a graph on the basis of incidence degree. Incidence degree can be obtained from the following sample of code;

Code sample 12:

```
Function [id] = my_id(v_list, e_list, v)
        Nbr =  neighbor(e_list, v);
        Id = numel(find(coloring(nbr)));
End
```

Listing 4.12: Index degree finding function.

This piece of a code is very useful as it finds the incidence degree (number of colored nodes in the neighbor) of a particular vertex.

Chapter 5

5. Analysis

In this chapter, we will analyze different unit distance graphs on the basis of number of colors and the time elapsed in coloring procedure using different graph coloring algorithms. After the implementation of coloring algorithms, the observed data are the number of colors and the elapsed time in milliseconds according to the increment in the number of vertices. The corresponding values of number of vertices for coloring different unit distance graphs using RLF, IDO based and DSATUR algorithms are obtained and are analyzed using tables and line charts.

Following are the analysis tables for different unit distance graphs colored by RLF, IDO based and DSATUR coloring algorithms. The table below shows the details of elapsed time and number of colors for different coloring algorithms which are used to color the unit distance graphs formed from random unit distance points.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
172	3	462	3	690	3	15
499	3	2133	3	5231	3	24
957	3	7281	4	17208	3	56
1357	3	12154	4	34044	3	76
1554	3	20276	4	53264	3	104

Table 5.1: Coloring time and number of colors used by UDG formed from random unit distance points.

From the table 5.1, DSATUR is found to be good as for the parameters; chromatic number and the elapsed time, than the other two, RLF and IDO based algorithms. In the comparison of RLF and IDO based heuristic algorithms, RLF seemed good than the IDO based algorithm in terms of time as well as the number of colors. IDO based algorithm gives more number of

colors in comparison to RLF and DSATUR algorithms. Both DSATUR and RLF give the same number of colors, but in the case of time, DSATUR is very fast than RLF. Here, the number of colors obtained, are almost same for all the coloring algorithms, so the chart shown below is on the basis of elapsed time.

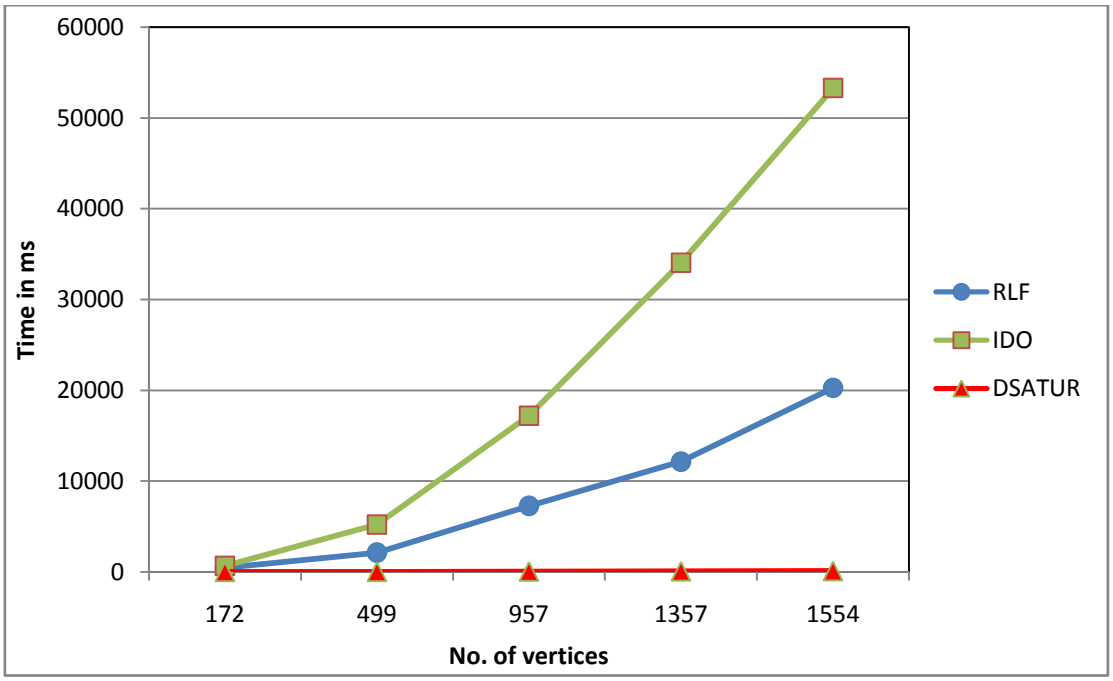


Figure 5.1: A line chart for time elapsed while coloring UDG formed by random unit distance points.

The table 5.2 shows the comparison of different coloring algorithms viz. RLF, IDO based and DSATUR, on the basis of number of colors and elapsed time, which are used to color the unit distance graphs formed by unit distance tree.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
100	3	220	3	528	3	3
400	3	2618	3	5326	3	27
580	3	4956	3	8404	3	31
900	3	10183	3	24850	3	43
1600	3	38519	3	66545	3	113

Table 5.2: Coloring time and number of colors used by UDG formed by unit distance tree.

Similarly, coloring of unit distance graph created by unit distance tree, the chromatic number is found same for all the algorithms but according to the elapsed time RLF is better over IDO based and DSATUR is better over RLF. Since there is no variation found in number of colors, the following chart shows the variation in time elapsed to color the graph.

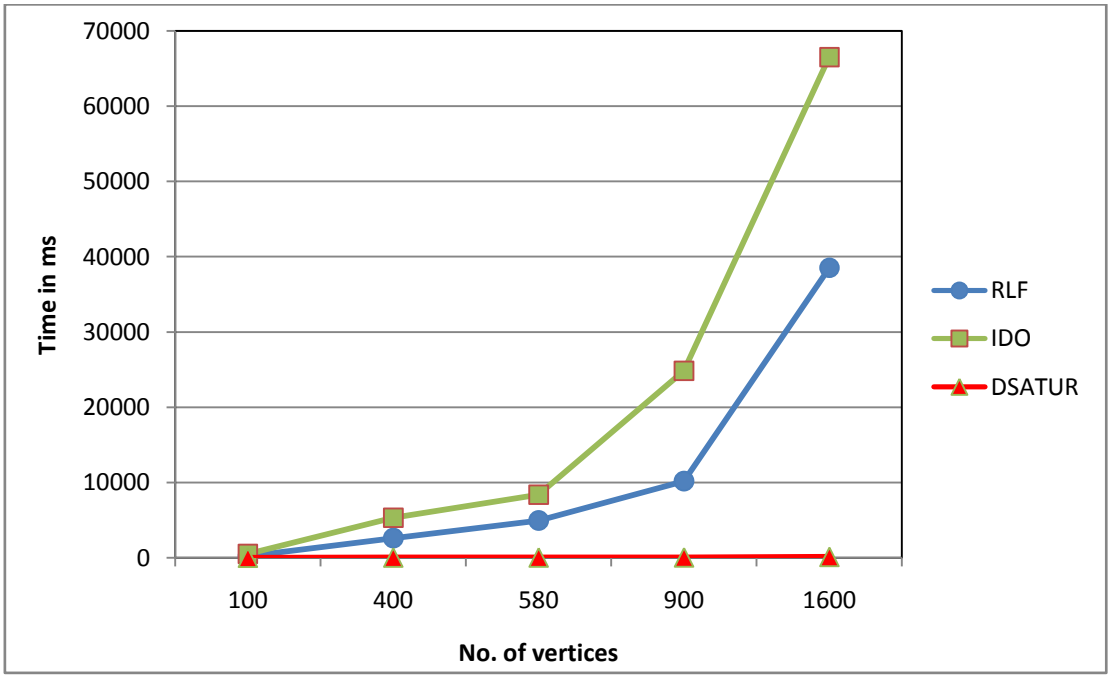


Figure 5.2: Line chart for time elapsed while coloring UDG formed by unit distance tree.

We have discussed about some test cases including simple unit distance graphs such as wheel, cycle, star, grid and Petersen graphs. Followings are the analysis tables for these test cases;

In table 5.3, the different coloring algorithms are compared on the basis of number of colors and the elapsed time in the coloring of a wheel graph. Since, the chromatic number for a wheel graph is 3 for odd number of vertices and 4 for even number of vertices, and in the analysis of wheel graph we found that the chromatic number of a wheel graph is 3 for any odd number of vertices and the chromatic number of a wheel graph is 4 for any even number of vertices, which shows that the graph coloring algorithms have worked properly.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
10	4	4	4	16	4	0.33
25	3	20	3	38	3	0.82
50	4	66	4	147	4	1
100	4	233	4	594	4	3
201	3	800	3	1785	3	7

Table 5.3: Time factor and number of colors obtained while coloring wheel graph.

In the following figure 5.3, the analysis for the wheel graph is given by using line chart on the basis of elapsed time as we can see in the above table 5.3, that the number of colors obtained by coloring the wheel graph using RLF, DSATUR and IDO based algorithms are same. And it is found that DSATUR takes less time than RLF and IDO, and in the comparison of RLF and IDO based algorithms, the RLF takes less time to color the wheel graph.

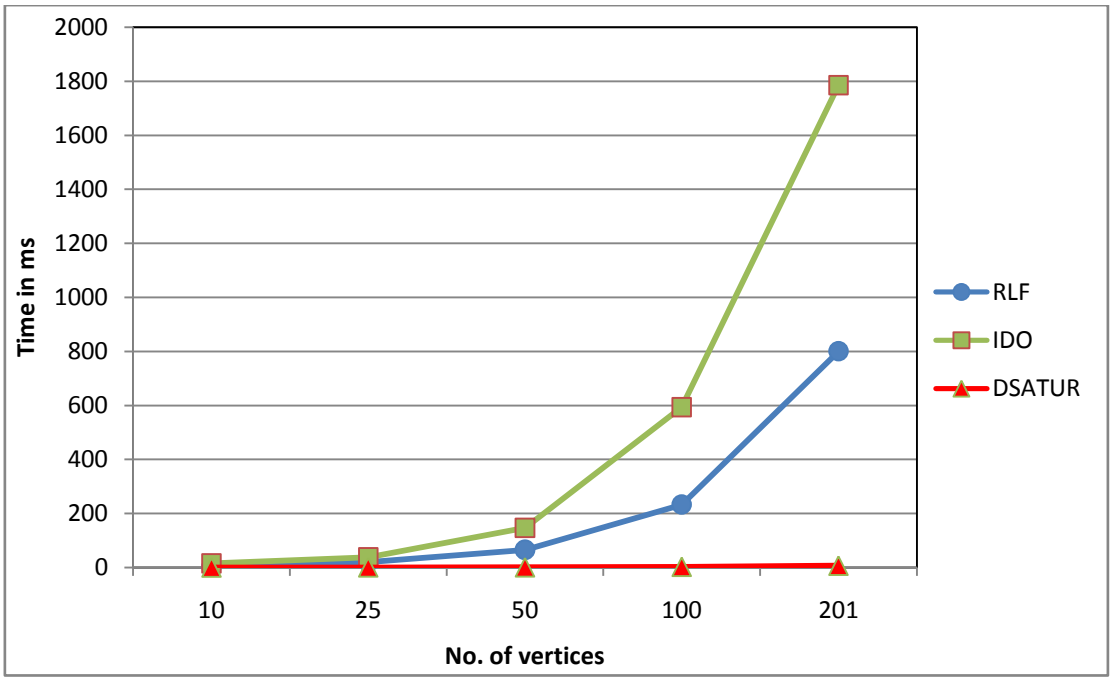


Figure 5.3: Line chart showing detail of time elapsed while coloring wheel graph.

In table 5.4, the resulting parameter for coloring algorithms RLF, IDO based and DSATUR are shown on the basis of chromatic number and elapsed time while coloring star graph. From the following analysis, the chromatic number of star graph is found to be 2.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
10	2	4	2	9	2	0.5
25	2	26	2	45	2	1
50	2	93	2	113	2	2
100	2	399	2	580	2	4
201	2	1137	2	1976	2	8

Table 5.4: Time factor and number of colors obtained while coloring star graph.

In the following figure 5.4, the analysis for star graph is shown on the basis of elapsed time in coloring by using coloring algorithms. After the analysis, the RLF is found superior to IDO based algorithms, and DSATUR is found superior to both RLF and IDO based algorithms.

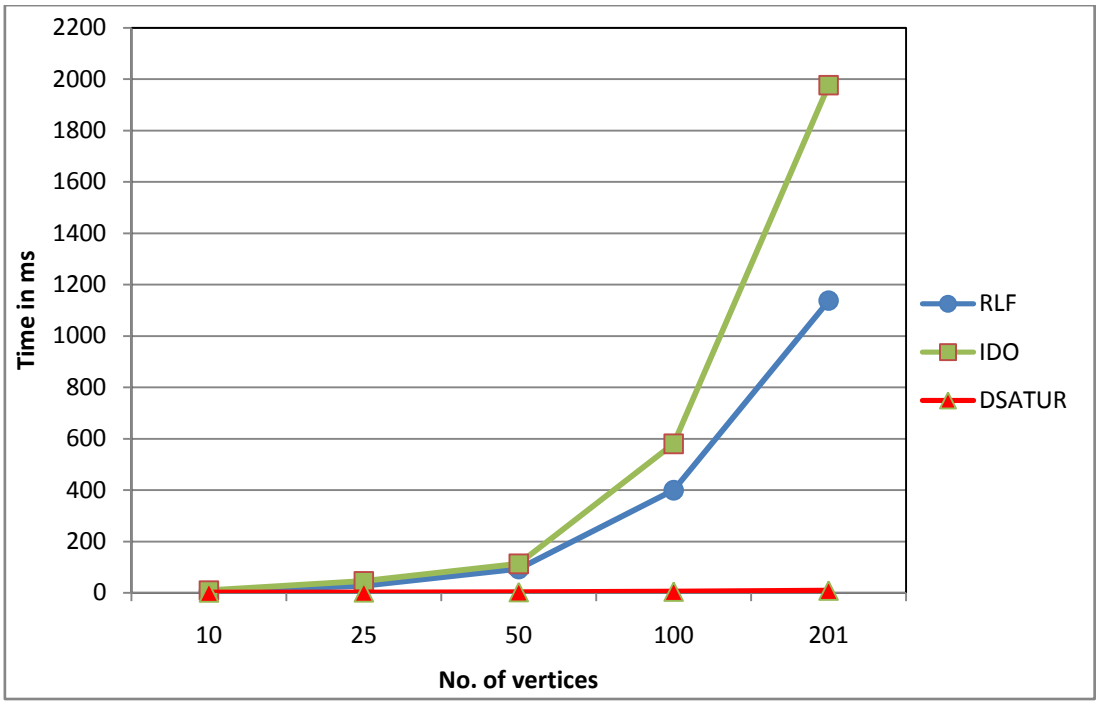


Figure 5.4: Line chart showing detail of time elapsed while coloring star graph.

In table 5.5, cycle graph is analyzed on the basis of chromatic number and the elapsed time to color the cycle graph using RLF, IDO based and DSATUR algorithms. It is found that the chromatic number of a cycle graph is 2 for the even number of vertices and 3 for the odd number of vertices.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
10	2	5	2	6	2	0.3
25	3	21	3	37	3	0.5
50	2	84	2	155	2	1
100	2	306	2	496	2	2
201	3	1066	3	1830	3	5

Table 5.5: Time factor and number of colors obtained while coloring cycle graph.

In the following figure, it is found, while coloring the cycle graph, DSATUR algorithm works better than RLF and IDO based algorithms. Since, number of colors obtained while coloring cycle graph are same using different coloring algorithms, so the following chart analysis is only based on the elapsed time.

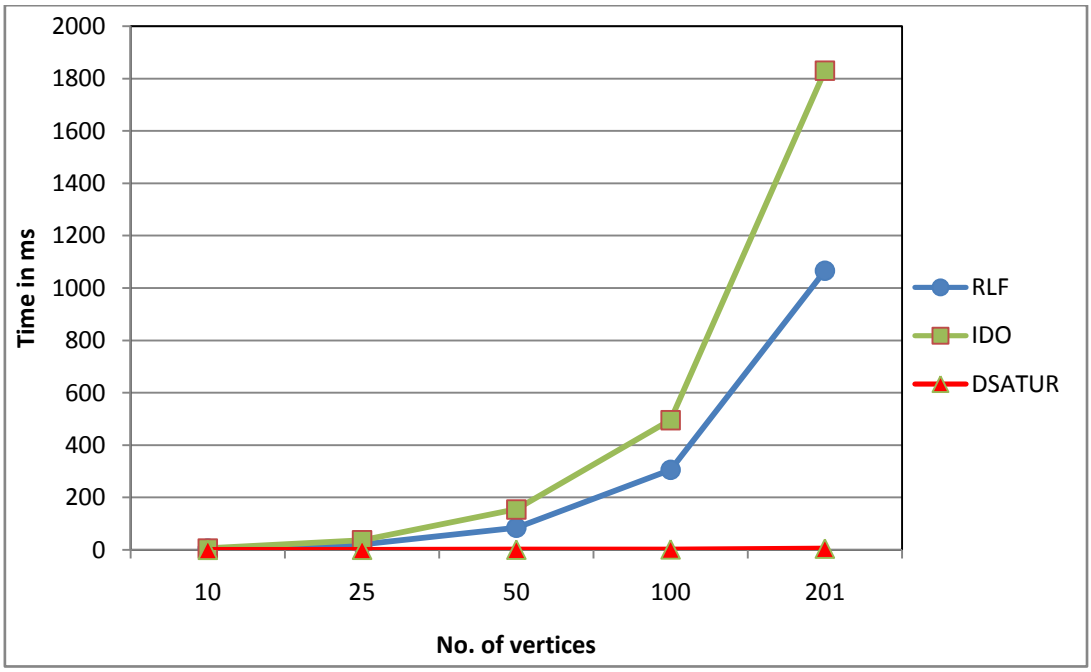


Figure 5.5: Line chart showing detail of time elapsed while coloring cycle graph.

In table 5.6, different graph coloring algorithms are compared on the basis of chromatic number and the elapsed time while coloring grid graph using RLF, DSATUR and IDO based algorithms. And the minimum number of colors found to color the grid graph is 2, which is given by RLF and DSATUR but IDO based algorithm colors the grid graph using 3 or 4 colors. So, RLF and DSATUR algorithms are better than IDO based in the case chromatic number.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
9	2	4	3	34	2	0.3
25	2	30	4	26	2	0.8
64	2	53	4	46	2	2
100	2	332	4	463	2	4
225	2	1110	4	1931	2	10

Table 5.6: Time factor and number of colors obtained while coloring grid graph.

In the following figure 5.6, the comparison is based on the time elapsed while coloring the grid graph. DSATUR again found to be superior to IDO based and RLF algorithms.

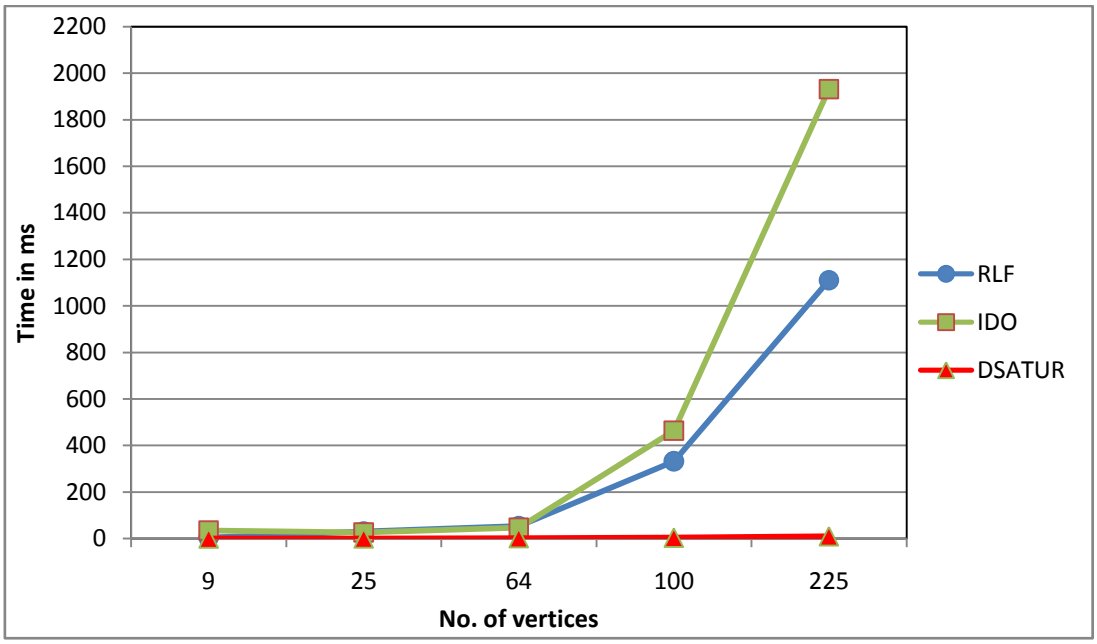


Figure 5.6: Line chart showing detail of time elapsed while coloring grid graph.

In table 5.7, the various coloring algorithms are compared on the basis of number of colors and the elapsed time while coloring Petersen graph. So, it has been found that the Petersen graph is three colorable. In the case of chromatic number, all the algorithms have given the same number of colors.

Number of vertices	RLF		IDO based		DSATUR	
	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms	No. of colors	Elapsed time in ms
10	3	2	3	4	3	0.2

Table 5.7: Time factor and number of colors obtained while coloring Petersen graph.

Since, Petersen graph is a unit distance graph of ten vertices and 15 edges, so we cannot analyze it by taking more or less than 10 vertices. In the following figure 5.7, the analysis is done on the basis of elapsed time using line chart. Here, again, DSATUR is seemed to be superior to RLF and IDO based algorithms.

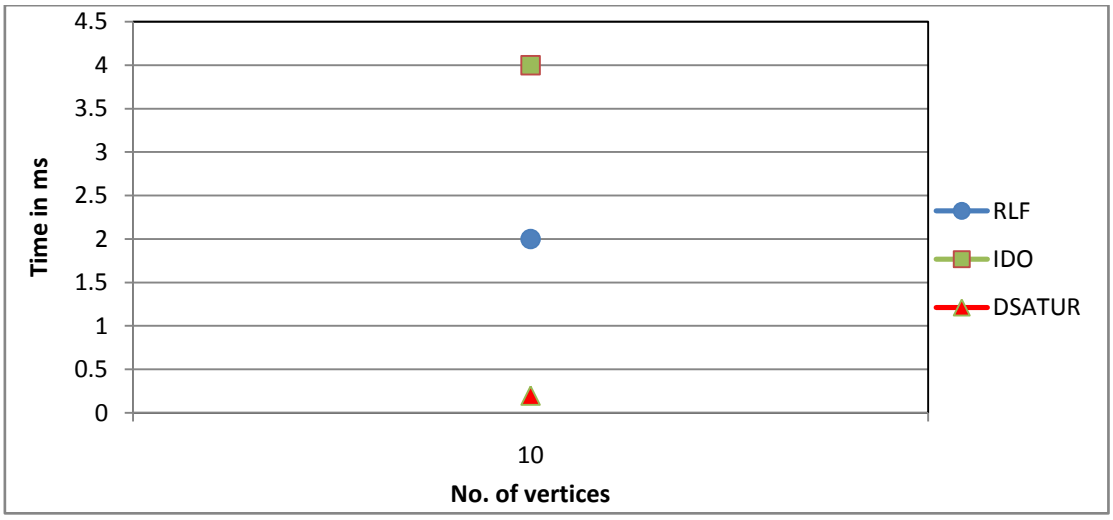


Figure 5.7: Line chart showing detail of time elapsed while coloring Petersen graph.

Result:

After the analysis of different unit distance graphs colored using RLF, IDO based and DSATUR graph coloring algorithms, it is found that the heuristic based DSATUR algorithm is the better graph coloring algorithm than RLF and IDO based algorithms, on the basis of coloring time (i.e. elapsed time) and number of colors required. In this analysis, it is found that not all the heuristic based algorithms give the same results, some heuristic based algorithms perform better and some do not. In this analysis, we found that among three graph coloring algorithms RLF is superior to IDO based and DSATUR is superior to RLF.

Chapter 6

6. Conclusion and Future Work

6.1 Summary and Conclusion

As we have mentioned that the plane coloring can be done via unit distance graph coloring. So in this study we have created different unit distance graphs, and colored them using three heuristic based graph coloring algorithms. Heuristics include LDO, SDO and IDO on the basis of ordering of vertices. The contraction based RLF algorithm uses the LDO heuristic to select the maximal degree vertex every time when next color is to be used. DSATUR algorithm uses SDO heuristic, which select the next vertex on the basis of maximal saturation degree. Similarly, IDO based selects the next vertex on the basis of incidence degree.

After the comparison of different heuristic graph coloring algorithms used to color different unit distance graphs on the basis of elapsed time and the number of colors, it is found that not every heuristic based algorithm give good results. RLF and DSATUR algorithms have given the same chromatic number, but the elapsed time is different. It is found that the elapsed time for DSATUR is very small comparing with RLF and IDO based algorithms. As analyzed in the tables and the line charts, the IDO based algorithm has taken the maximum time to execute and given extra colors. So, in terms of chromatic number and time, RLF seemed better than IDO based, and DSATUR seemed better than RLF and IDO based graph coloring algorithms.

And using RLF, IDO based and DSATUR heuristic algorithms, we have colored the unit distance graphs created from unit distance points and unit distance tree, and found the optimal chromatic number.

6.2 Further Recommendations

This study is limited to different flavors of unit distance graphs. There are various types of graphs, such as bipartite graph, complete graph, etc. and the study may be continued on the applications of different graphs rather than unit distance graphs.

We can extend this work in various problems like register allocation problem via graph coloring [6] and unit distance frequency distribution (UDW) [7], which are not discussed vastly in this study.

Appendix

Some of the figures obtained by implementing the unit distance graphs and coloring them by using RLF, DSATUR and IDO based algorithms in MATLAB, are shown in figures below.

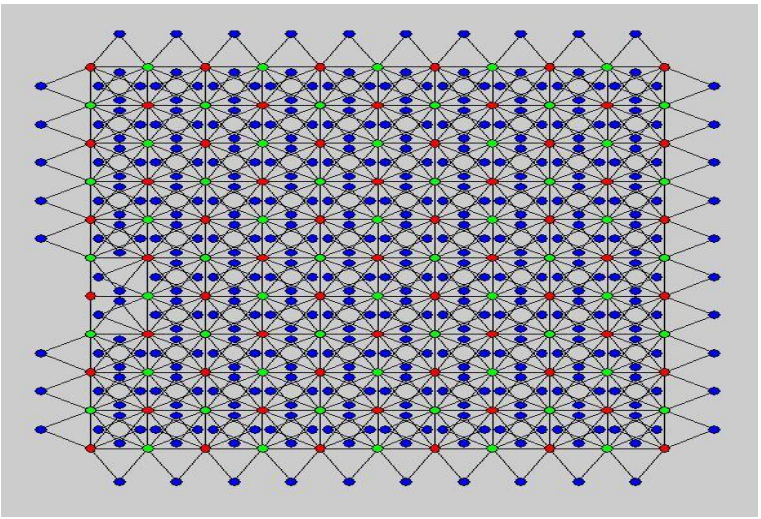


Figure A.1: Unit distance graph using random unit distance points colored by DSATUR and RLF algorithm (three colors are needed).

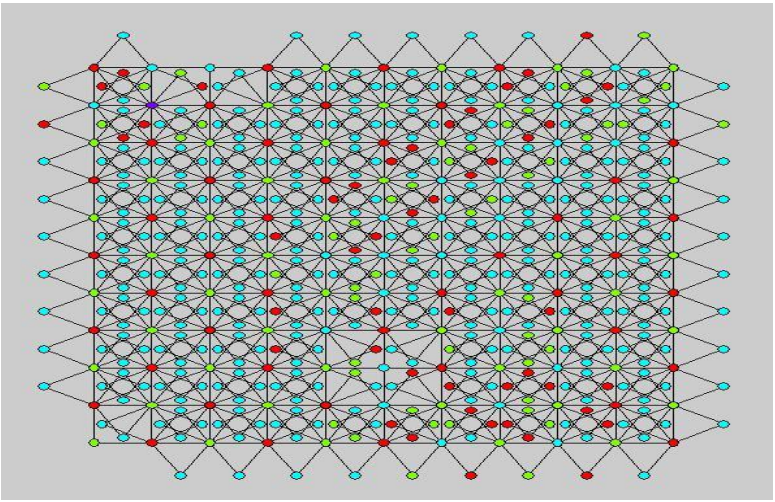


Figure A.2: Unit distance graph using random unit distance points colored by IDO based algorithm (four colors needed).

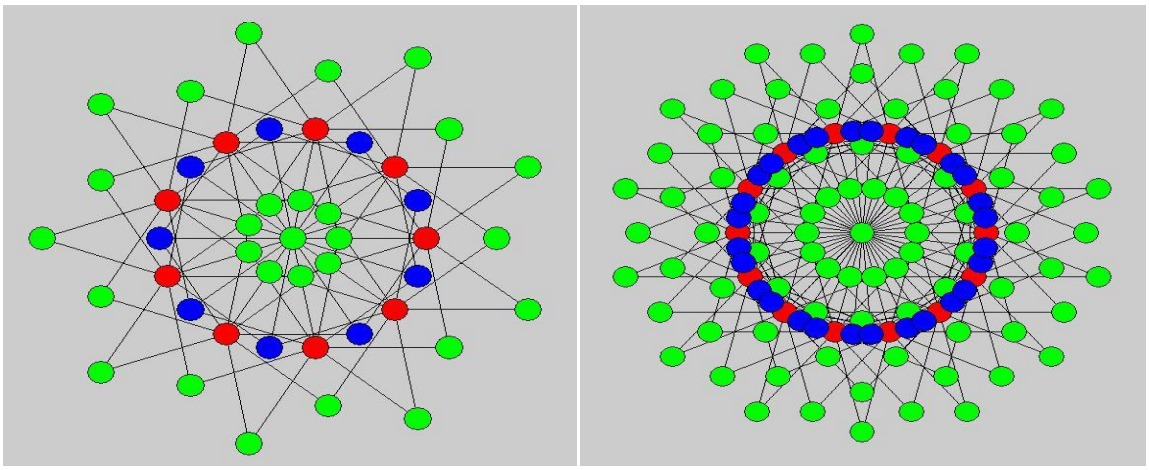


Figure A.3: Unit distance graphs using unit distance tree (three colors needed).

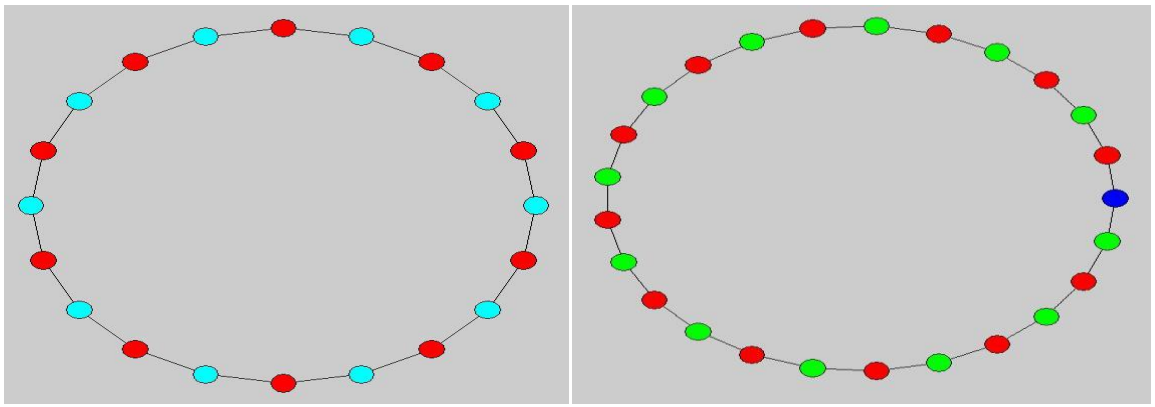


Figure A.4: Cycle graph for even nodes. Figure A.5: Cycle graph for odd nodes.

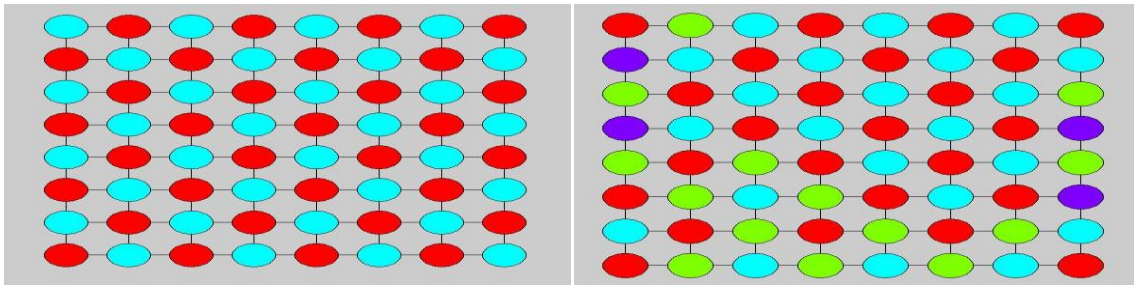
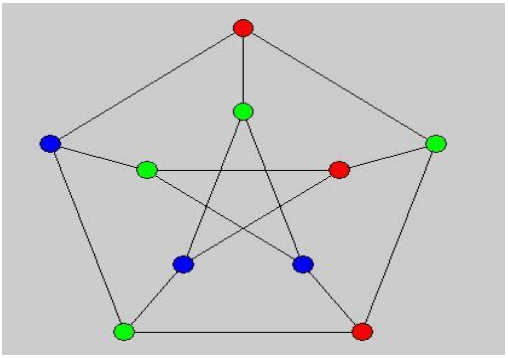


Figure A.6: Grid graph by DSATUR and RLF. Figure A.7: Grid graph by IDO based.



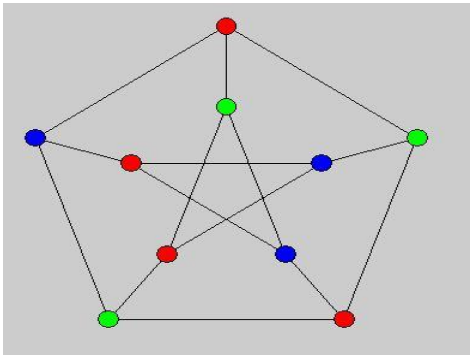
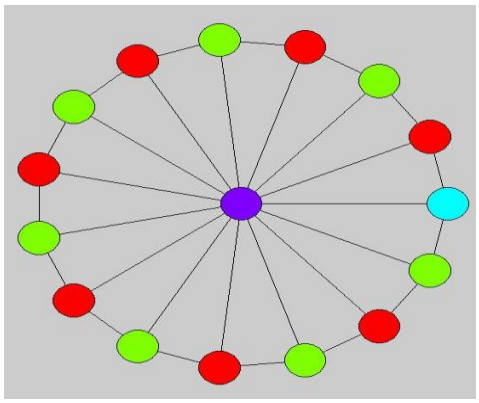


Figure A.8: Petersen graph by DSATUR & RLF. Figure A.9: Petersen graph by IDO based.



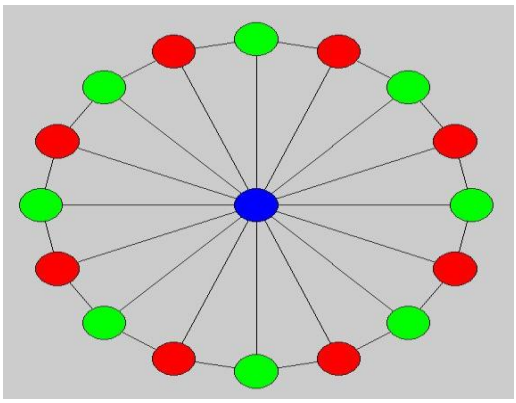


Figure A.10: Wheel graph for even nodes. Figure A.11: Wheel graph for odd nodes.

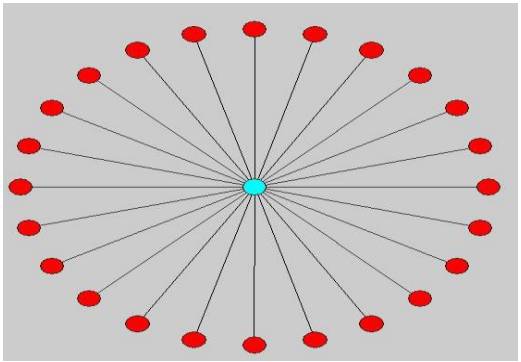


Figure A.12: Star graph.

References

1. Alexander Soifer, *The mathematical coloring book*, Mathematics of coloring and the colorful life of its creators, Springer, New York, 2009.

2. Akhlak Mansuri, Vijay Gupta and R. S. Chandel, *Coloring Programs in Graph Theory*, Int. Journal of Math. Analysis, Vol. 4, 2010, 50, 2473 – 2479.

3. Arjana Zitnik, Boris Horvat, and Tomaz Pisanski,, *ALL GENERALIZED PRTERSEN GRAPHS ARE UNIT- DISTANCE GRAPHS*, IMFM, University of Ljubljana, University of Primorska Slovenia, December 28, 2009.

4. Dr. Hussein Al-Omari and Khair Eddin Sabri, *New graph coloring algorithms*, American Journal of Mathematics and Statistics 2 (4): 739-741, 2006.

5. John Cavazos, *Register allocation via graph* coloring, University of Delaware, Computer and Information Sciences Department.

6. Joseph O'Rourke, Computational Geometry column 46, *Internat. J. Comput. Geom. Appl.*, 14(6):475-478, 2004, *SIGACT News*, **35**(3):42-45 (2004), Issue 132.

7. J. Urrutia, *Local Solutions for Global Problems in Wireless Network,* Supported by CONACYT of Mexico, May 1, 2006.

8. Klotz Walter, *Graph Coloring Algorithms,* Mathematik-Bericht 5 (2002), 1-9, TU Clausthal.

9. Padal Nihar, *Chromatic Numbers*, Indiana State University Terra Haute IN, USA. December 7, 2011.

10. Vitaly I. Voloshin, *Graph Coloring: History, results and open problems*, Alabama Journal of Mathematics Spring/Fall 2009.

Bibliography

1. Joseph O'Rourke, Computational Geometry in C, second edition, CAMBRIDGE University Press.
2. Ivan Graham, MATLAB MANUAL AND INTRODUCTORY TUTORIALS, Mathematical Sciences, University of Bath, February 9, 2005.
3. Assefaw Hadish Gebremedhin, Parallel Graph Coloring, Department of Informatics, University of Bergen, Norway, Spring 1999.