Graph Coloring Problems in Modern Computer Science

Besjana Tosuni

Europian University of Tirana , Tirana, Albania

e-mail.b.tosuni@hotmail.com

Abstract

Graph coloring is one of the most important concepts in graph theory and is used in many real time applications in computer science. The main aim of this paper is to present the importance of graph coloring ideas in various areas of compute applications for researches that they can use graph coloring concepts for the research. Graph coloring used in various research areas of computer science such data mining, image segmentation, clustering, image capturing, networking etc. This papers mainly focused on important applications such as Guarding an Art Gallery, Physical layout segmentation, Round-Robin Sports Scheduling, Aircraft scheduling, Biprocessor tasks, Frequency assignment, Final Exam Timetabling as a Grouping Problem, Map coloring and GSM mobile phone networks, and Student Time Table. In this paper we review several variants of graph colouring, such as precolouring extension, list colouring, multicolouring, minimum sum colouring, and discuss their applications in scheduling. A very important graph parameter is the chromatic number. Presently, graph coloring plays an important role in several real-world applications and still engages exciting research.

Keywords: Graph theory, graph coloring, map coloring , scheduling problems, multicolouring.

1. INTRODUCTION

Nowadays the studies about the behavior of several graph parameters in product graphs have become into an interesting topic of research in graph theory. For instance, is it well known the Hedetniemi’s coloring conjecture [16, 20] for the categorical product (or direct product), which states that the chromatic number of categorial product graphs is equal to the minimum value between the chromatic numbers of its factors. Also, one of the oldest open problems in domination in graphs is related with product graphs. The problem was presented first by Vizing [25] in 1963. After that he pointed out as a conjecture in [26]. The conjecture states that the domination number of Cartesian product graphs is greater than or equal to the product of the domination numbers of its factors.

Graph coloring especially used various in research areas of science such data mining, image segmentation, clustering, image capturing, networking etc., For example a data structure can be designed in the form of tree which in turn utilized vertices and edges.

Similarly modeling of network topologies can be done using graph concepts. In the same way the most important concept of graph coloring is utilized in resource allocation, scheduling. Also, paths, walks and circuits in graph theory are used in tremendous applications say traveling salesman problem, database design concepts, resource networking. This leads to the development of new algorithms and new theorems that can be used in tremendous applications. Graph coloring is one of the most important concepts in graph theory and is used in many real time applications in computer science. Various coloring methods are available and can be used on requirement basis. The proper coloring of a graph is the coloring of the vertices and edges with minimal number of colors such that no two vertices should have the same color. The minimum number of colors is called as the chromatic number and the graph is called properly colored graph .

2. GUARDING AN ART GALLERY

2.1 The Sunflower Art Gallery

Figure 2.1 shows the unusual floor plan of the Sunflower Art Gallery and the locations of four guards. Each guard is stationary but can rotate in place to scan the surroundings in all directions. Guards cannot see through walls or around corners. Every point in the gallery is visible to at least one guard, and theft of the artwork is prevented. Of course, it would be more economical to protect the gallery with fewer guards, if possible.

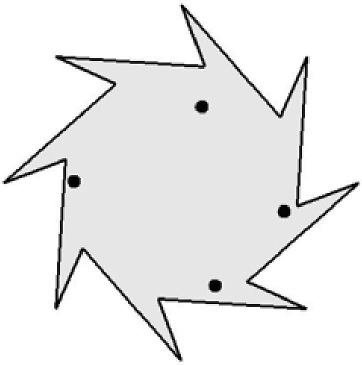

Figure 2.1: The Sunflower Art Gallery

2.2 Art Gallery Problems

Let us define our terms carefully. For our purposes, an art gallery is a polygon in the plane.2 The polygon need not

serve as the floor plan of any real-world art gallery. An art gallery includes the interior region as well as the boundary

segments—the walls. We let G denote an arbitrary art gallery and write Gw for an art gallery with *w* walls.

Let p be any point in an art gallery. The point q is visible to p provided the line segment joining p and q does not exit the gallery. (We also assume that every point is visible to itself.) The segment represents the sight line of a guard. A set of guards protects an art gallery provided every point in the gallery is visible to at least one guard. Note that a guard at a corner protects the two adjacent walls.

Example 1. (a) The four guards in Figure 2.1 protect the Sunflower Art Gallery.

(b) The Sunflower Art Gallery is not protected by guards at the eight outer corners (Figure 2.2). Even though all of the walls are protected, a region in the center of the gallery remains invisible to all the guards.

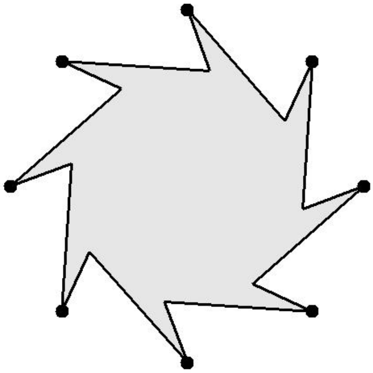

Figure 2.2: The eight guards protect the walls, but not the interior

(c) Each gallery in Figure 2.3 is protected by one or two guards, as shown.

An art gallery is convex provided every point in it is visible to every other point. A convex gallery is easy to guard; a guard can be posted anywhere in the gallery. Every triangle is convex, as are the first two galleries in the top row of Figure 2.3. The other galleries in the figure are nonconvex.

Galleries in Particular

Our desire to post as few guards as possible raises two general problems about art galleries.

The first problem deals with specific galleries, and the second deals with all galle-ries with a fixed number of walls.

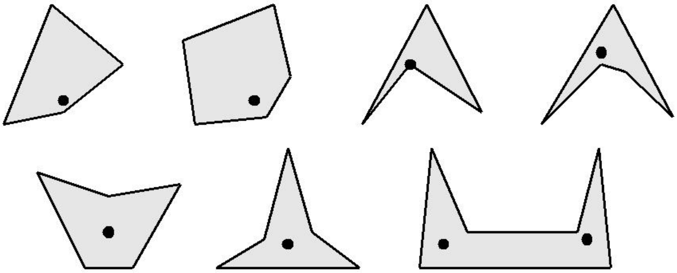

Figure 2.3: The first two galleries in the top row are convex

These are generalizations of the two questions we posed earlier. Let guard(G) = the minimum number of guards needed to protect the art gallery G.

Gallery problem 1. Find the value of guard(G) for every art gallery G. In other words, find the minimum number of guards needed to protect every art gallery.

Example 2. (a) A convex gallery G satisfies guard(G) = 1.

(b) We have seen that the Sunflower Art Gallery G_{16} sat-isfies guard(G_{16}) = 3.

To show that guard(G) = g, we must demonstrate two facts:

- The gallery G can be protected by g guards.
- The gallery G cannot be protected by fewer than g guards.

The first fact implies that guard(G) ≤ g, while the second gives guard(G) ≥ g. The second fact becomes increasingly difficult to demonstrate as the number of walls increases and the shape of the gallery becomes more complicated.

Ideally, we would have an efficient algorithm that takes an arbitrary gallery G as its input and produces the value of guard(G) as its output. Such an algorithm could be carried out by a computer (or a patient, careful person) to determine the minimum number of guards needed to protect any given gallery. Researchers in computational complex-ity, an advanced area of discrete mathematics, have strong evidence that we will never find an efficient algorithm of the desired type. The crux of the matter is that the number of essentially different guard configurations to examine increases exponentially as a function of the number of walls. Any proposed general algorithm becomes effectively worthless, even with the fastest computers available. In this sense, gallery problem 1 remains unsolved.

Galleries in General

Now suppose we know an art gallery has w walls, but we do not know its exact shape. Let g(w) = the maximum number of guards required among all art galleries with w walls. In other words, g(w) is the maximum value of guard(Gw) among all w-walled galleries Gw.

Example 3. (a) Any triangular art gallery can be protected with one guard. Therefore, g(3) = 1.

(b) The Sunflower Art Gallery has 16 walls and requires three guards. Therefore, g(16) 3. We cannot conclude that g(16) = 3 since there could be a 16-walled gallery that requires more than three guards. In fact, we will soon see a

16-walled gallery requiring five guards.

3. Physical Layout Segmentation

Automatic mail sorting machines of most recent systems process about 17 mail pieces per second. That requires a fast

and precise OCR based recognition of the block-address. This recognition is mainly conditioned by a correct address

lines organization . Once the envelope image has been acquired by a linear CCD camera, three principal

modules contribute to the task of the address-block localization: physical layout segmentation of envelope image, feature extraction and address-block identification. Every-day, the postal sorting systems diffuse several tons of mails. It is noted that the principal origin of mail rejection is related to the failure of address-block localization task, particularly, of the physical layout segmentation stage. The bottom-up and top-down segmentation methods bring different knowledge that should not be ignored when we need to increase the robustness. Hybrid methods combine the two strategies in order to take advantages of one strategy to the detriment of other. Starting from these remarks, our proposal makes use of a hybrid segmentation strategy more adapted to the postal mails. The high level stages are based on the hierarchical graphs coloring. Today, no other work in this context has make use of the powerfulness of this tool. The performance evaluation of our approach was tested on a corpus of 10000 envelope images. The processing times and the rejection rate were considerably reduced.

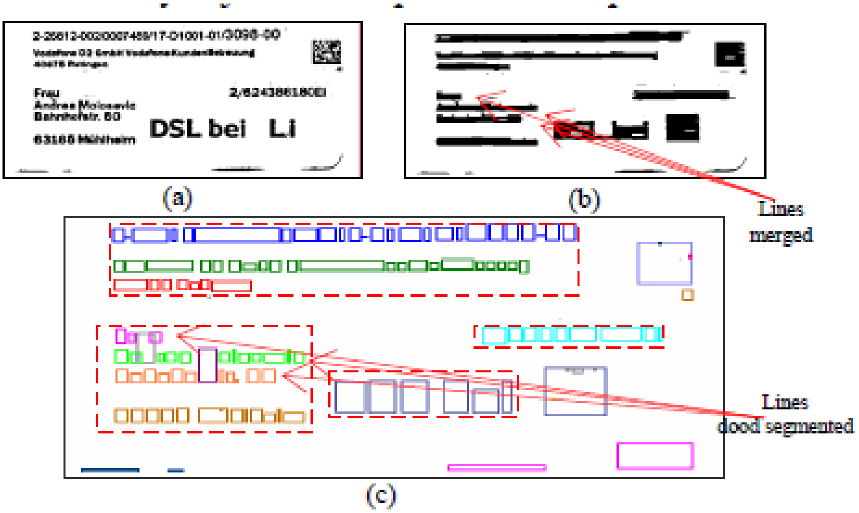

Figure 3.1 Noisy block address (a), emergence of the text lines

by the RLSA method (b), Perception of the text lines by

hierarchical graph coloring (c).

The segmentation technique objective is based on its decision strategy which defines a best block extraction manner in

order to recognize it by the block address recognition module. The segmentation techniques cannot systematically produce uniform and good located blocks in complex environments (difficult envelopes). Consequently, the knowledge delivered by the descriptors of non-homogeneous blocks (containing parasitic elements) is less discriminating.

In order to improve the robustness and exactness of segmentation, it has been necessary to choose an even more advanced tool. The idea is to use a hybrid strategy of segmentation using the richness of pyramidal structure. Our method is mainly based on the powerfulness of graph coloring to regroup correctly the connected components into text lines then the lines into blocks.

4. Time table scheduling

Allocation of classes and subjects to the Teachers is one of the major issues if the constraints are complex. Graph theory plays an important role in this problem. For „t" Teachers with „n" subjects the available number of „p" periods timetable has to be prepared. This is done as follows. A bipartite graph (or bigraph is a graph whose vertices can be divided into two disjoint sets *U* and *V* such that every edge connects a vertex in *U* to one in *V*; that is, *U* and *V* are independent sets) G where the vertices are the number of Faculty say t_1, t_2, t_3, t_4, ……. Tk and n number of subjects say n_1, n_2, n_3, n_4, ……. nm such that the vertices are connected by „pi" edges. It is presumed that at any one period each Teacher can teach at most one subject and that each subject can be taught by maximum one Teacher. Consider the first period. The timetable for this single period corresponds to a matching in the graph and conversely, each matching corresponds to a possible assignment of Teacher to subjects taught during that period. So, the solution for the timetabling problem will be obtained by partitioning the edges of graph G into minimum number of matching. Also the edges have to be colored with minimum number of colors. This problem can also be solved by vertex coloring algorithm. " The line graph L(G) of G has equal number of vertices and edges of G and two vertices in L(G) are connected by an edge iff the corresponding edges of G have a vertex in common. The line graph L(G) is a simple graph and a proper vertex coloring of L(G) gives a proper edge coloring of G by the same number of colors. So, the problem can be solved by finding minimum proper vertex coloring of L(G)." For example, Consider there are 4 Teachers namely t_1, t_2, t_3, t_4,. and 5 subjects say n_1, n_2, n_3, n_4, n_5 to be taught. The teaching requirement matrix p = [pij] is given as.

P	n_1	n_2	n_3	n_4	n_5
t_1	**2**	**0**	**1**	**1**	**0**
t_2	**0**	**1**	**0**	**1**	**0**
t_3	**0**	**1**	**1**	**1**	**0**
t_4	**0**	**0**	**0**	**1**	**1**

Figure 4.1: The teaching requirement matrix for four Teachers and five subjects

The bipartite graph is constructed as follows.

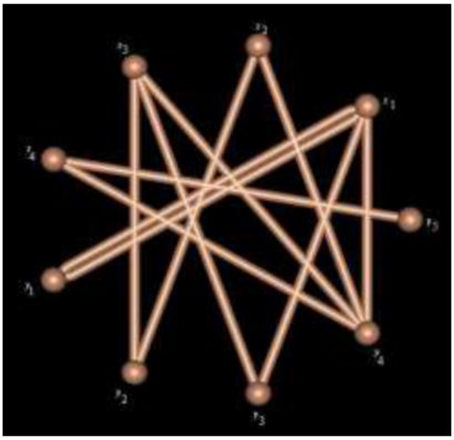

Figure 4.2. The bipartite multigraph G

Finally, the authors found that proper coloring of the above mentioned graph can be done by 4 colors using the vertex coloring algorithm which leads to the edge coloring of the bipartite multigraph G. Four colors are interpreted to four periods.

...	1	2	3	4
t_1	n_1	n_2	n_3	n_4

Figure 4.3: The schedule for the four subjects

5. GSM Mobile Phone Networks

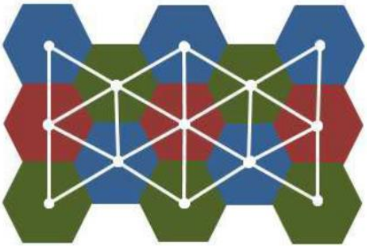

Figure 5 The cells of a GSM mobile phone network

The Groupe Spécial Mobile (GSM) was created in 1982 to provide a standard for a mobile telephone system. The first GSM network was launched in 1991 by Radiolinja in Finland with joint technical infrastructure maintenance from Ericsson. Today, GSM is the most popular standard for mobile phones in the world, used by over 2 billion people across more than 212 countries. GSM is a cellular network with its entire geographical range divided into hexagonal cells. Each cell has a communication tower which connects with mobile phones within the cell. All mobile phones connect to the GSM network by searching for cells in the immediate vicinity. GSM networks operate in only four different frequency ranges. The reason why only four different frequencies suffice is clear: the map of the cellular regions can be properly colored by using only four different colors! So, the vertex coloring algorithm may be used for assigning at most four different frequencies for any GSM mobile phone network, see figure 7 below.

6. Precoloring extension

In certain scheduling problems we do not have full control over the schedule, the assignments of certain jobs are already decided. In this case some of the vertices of the conflict graph has a preassigned color, and we have to solve the *precoloring extension* problem: extended the coloring of these vertices to the whole graph, using the minimum number of colors. Bir´o, Hujter and Tuza [7, 8, 9] started a systematic study of precoloring extension. In [7], the aircraft scheduling problem discussed in Section I. is extended. There is a maintenance period for each aircraft,during which it cannot fly. We can model these maintenance periods by adding a “dummy” flight for the maintenance period of each aircraft, and requiring that the maintenance period of the ith aircraft is assigned to the ith aircraft. Therefore we have to solve the precoloring extension problem on the conflict graph, which is an interval graph. It is shown in [7] that the precoloring extension problem is **NP**-complete for interval graphs, but it can be solved in polynomial time if every color is used only once in the precoloring, that is, if every aircraft has only one maintenance interval (the later result is generalized to chordal graphs in [10]).

7. List coloring

In the list coloring problem each vertex v has a list of available colors, and we have to find a coloring where the color of each vertex is taken from its list of available colors. List coloring can be used to model situations where a job can be processed only in certain time slots, or if it can be processed only by certain machines. Using standard dynamic programming techniques, list coloring can be solved in polynomial time on trees and partial k-trees [11]. By combining dynamic programming with a clever use matching, list coloring can be solved on the edges of trees as well [12]. The multicoloring concept introduced in Section II. can be applied for list colorings as well: each vertex has an integer demand x(v), and vertex v has to receive a set of x(v) colors from its list of colors. The algorithm for list coloring trees and partial k-trees does not generalize for the multicoloring case, as the problem is **NP**-complete already for binary trees [13]. On the other hand, list *edge* multicoloring can be solved in polynomial time on trees: using standard techniques, the good characterization theorem of Marcotte and Seymour [14] can be turned into a polynomial time algorithm. This result is generalized in [15] to a slightly more general class of graphs, that includes odd cycles. Moreover, a randomized algorithm is given for an even more general class of graphs, including even cycles.

8. Minimum sum coloring

Besides minimizing the makespan, another well-studied goal in scheduling theory is to minimize the sum of completion times of the jobs, which is the same as minimizing the average completion time. The corresponding coloring problem is *minimum sum coloring*, introduced in [16]: we are looking for a coloring of the conflict graph such that the sum of the colors assigned to the vertices is minimal. Apart from trees, partial k-trees, and edges of trees, minimum sum coloring is **NP**-hard on most classes of graphs. On the other hand, it turns out that the sum of the coloring is easier to approximate than the makespan (see e.g. [17, 18] for approximation results). The reason for this is that the sum of the coloring and the makespan of the coloring behaves very differently when a small part of the graph is recolored. If we recolor a small part of the graph, then this change has only a small effect on the sum of the coloring, but it can change the makespan significantly. The multicoloring version of the problem can be used to model arbitrary length jobs. Since we want to minimize the sum of the completion times, the objective function of the coloring problem has to be defined as follows. The *finish time* of a vertex is the largest color assigned to it, and the sum of a coloring is the sum of the finish times of the vertices. It is clear that the sum of the finishtimes in a multicoloring is equal to the sum of completion times in the corresponding schedule. This variant of multicoloring was introduced in [19], where approximation algorithms are given for various classes of graphs. The preemptive and non-preemptive versions of the problem can have very different complexity: while the non-preemptive version can be solved in polynomial time for trees [20], the preemptive version is **NP**-hard for binary trees [13], but has a polynomial time approximation scheme [20]. In [21] polynomial time approximation schemes are given for partial k-trees and planar graphs as

well. Unlike minimum sum coloring, the multicoloring version of the problem is **NP**-hard on the edges of trees. However, in this case the problem admits a polynomial time approximation scheme [22].

REFERENCES

a. Bar-Noy, M. Bellare, M. M. Halld´orsson, H. Shachnai, and T. Tamir, “On chromatic sums and distributed resource allocation,” Inform. and Comput., 14 (2):183–202, 1998.

[2] Bar-Noy, M. M. Halld´orsson, G. Kortsarz, R. Salman, and H. Shachnai, “Sum multicoloring of graphs,” J.Algorithms, 37(2):422–450, 2000.

[3] C Viard-Gaudin, D Barba, A multi-resolution approach to extract the address block on flat mail pieces, ICASSP-91, International Conference, vol.4, Pages: 2701 - 2704 .

[4] D. Marx, “List edge multicoloring in graphs with few cycles,” Inform. Process. Lett., 89(2):85–90, 2004.

[5] D. Marx, “Minimum sum multicoloring on the edges of trees,” 2003, To appear in 1st Workshop on Approximation and Online Algorithms, Budapest, 2003.

[6] D. Marx, “Precoloring extension on chordal graphs,” 2003, submitted.

[7] D. Marx, “The complexity of tree multicolorings,” in Mathematical Foundations of Computer Science 2002 (Warsaw-Otwock), pp. 532–542. Springer, Berlin, 2002.

[8] E. Kubicka and A. J. Schwenk, “An introduction to chromatic sums,” in Proceedings of the ACM Computer Science Conf., pp. 15–21. Springer, Berlin, 1989.

[9] K. Giaro and M. Kubale, “Edge-chromatic sum of trees and bounded cyclicity graphs,” Inform. Process. Lett., 75(1-2):65–69, 2000.

[10] K. Giaro, R. Janczewski, M. Kubale, and M. Małafiejski, “A 27=26-approximation algorithm for the chromatic sum coloring of bipartite graphs,” in Proceedings of APPROX 2002, pp. 135–145, 2002.

[11] K. Jansen and P. Scheffler, “Generalized coloring for tree-like graphs,” Discrete Appl. Math., 75(2):135–155, 1997.

[12] L. F., Paquete, and C. M., Fonseca (2001) ”A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms”, Proc. of the 4th Metaheuristics International Conference (MIC 2001), pp. 149-154, Porto. F.T, Leighton, (1979) ”A Graph Coloring Algorithm for large Scheduling Problems”, Journal of Research of the National Bureau Standard, 84 pp 79-100.

[13] M. Bir´o, M. Hujter, and Zs. Tuza, “Precoloring extension. I. Interval graphs,” Discrete Math., 100 (1-3):267–279, 1992.

[14] M. Hujter and Zs. Tuza, “Precoloring extension. II. Graph classes related to bipartite graphs,” Acta Mathematica Universitatis Comenianae, 62(1):1–11, 1993.

[15] M. Hujter and Zs. Tuza, “Precoloring extension. III. Classes of perfect graphs,” Combin. Probab. Comput., 5(1):35–56, 1996.

[16] M. M. Halld´orsson and G. Kortsarz, “Tools for multicoloring with applications to planar graphs and partial k-trees,”J. Algorithms, 42(2):334–366, 2002.

[17] M. M. Halld´orsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai, and J. A. Telle, “Multicoloring trees,”Inform. and Comput., 180(2):113–129, 2003.

[18] Narasingh Deo, “Graph theory with applications to engineering and computer science”, Prentice Hall of India, 1990.

[19] O. Marcotte and P. D. Seymour, “Extending an edge-coloring,” J. Graph Theory, 14(5):565–573, 1990.

[20] Shamim Ahmed, Applications of Graph Coloring in Modern Computer Science.

[21] Wang Ching-Huei, P.W. Palumbo, S.N. Srihari, Object recognition in visually complex environments: an architecture for locating address blocks on mail pieces, Pattern Recognition, 1988., 9th International Conference, IEEE, 1988, vol.1, Pages: 365 – 367.