IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-5, May 2018 E-ISSN: 2347-2693

Exam Time Table Scheduling using Graph Coloring Approach

Rubul Kumar Bania[1*], Pinkey Duarah[2]

[1*] Department of Computer Applications, North-Eastern Hill University, Tura Campus, Meghalaya, India
[2] Department of Computer Applications, North-Eastern Hill University, Tura Campus, Meghalaya, India

[*]*Corresponding Author: rubul.bania@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 18/May/2018, Published: 31/May/2018

Abstract— One of the most common academic scheduling problems which can be perceived in any educational system is the exam time table generation. The presence of vast numbers of students and offered courses makes it difficult to schedule exams in a limited epoch of time. An appropriate schedule can be designed by utilizing different resources like subjects, teachers, students and classrooms in a way to evade conflicts by fulfilling special types of constraints. Graph coloring is one decent approach which can deal with timetable scheduling problem and can satisfy changing requirements. In this work, we have framed a systemic model by applying graph vertex coloring approach for generating exam timetabling with the help of a course matrix generated from given data of an educational institute. From the problem domain, different types of constraints viz., hard and soft are figured out and while solving emphasis is focused on the degree of constraint satisfaction. Workflow of the system is described by using a case study and the output which it has generated is efficient and satisfactory.

Keywords— Time table, Graph coloring, Scheduling, Hard constraints, Soft constraints, Course matrix

I. INTRODUCTION

Exam time table is required in every educational institution. In every semester or year, the universities and colleges are required to generate exam time table for conducting the internal and the final semester exams. The presence of a large number of students and large number of offered courses sometimes makes it difficult to schedule the exam without having any conflict [1]. Moreover, the exam time table could not be reused because the requirements and some restrictions of the problem keep changing. If done manually, generating an exam time table is very time consuming and requires a considerable amount of workforce which sometimes may lead to inefficiency. So it demands an automatic generation of time table with a limited number of user input parameters. The exam time table should be generated in such a way that one subject is scheduled only once. So this can be considered as a good constraint or restriction which must be satisfied while designing an automatic time table generation system. There is no restriction regarding the faculty members. That means any faculty member could be assigned to conduct the examination of any subject. However, if a faculty member conducts the examination of those subjects which he/she teaches then it would be beneficial because sometimes there could be some printing mistakes or some other errors on the question paper, which could be identified and immediately corrected by the teacher. The theory exams could be conducted in any room accordingly to the availability of the room. However, practical exams could be conducted only in the lab. We should also consider those students who have back papers (repeater exams) and thus, need to reappear in some of the papers. The exam time table should avoid such conflicts, like no two or three exams for the identical student should be scheduled at the same period of time. The exam schedule should not leave a big gap between exams for the students. Therefore, there is a much need of an effective and accurate timetable to the performance of any educational institute. Graph coloring [2][10][12] is a method of assigning colors to certain elements of a graph subjected to certain constraints. Thus, optimal solutions to such problem may be found by determining minimal colorings for the corresponding graph. Also, the time table scheduling problem has been classified as a NP-hard problem [1] [3]. That means it is unlikely to find a fast algorithm to solve this problem, and to find a solution to such problem we need to consider all the possible solutions, and then choose the best one.

In this work, we have contributed to formulate the exam timetabling problem followed by a systematic model framed on two methods with some user input parameters. It will automatically generate an exam time table for universities based on some specified hard and soft constraints using the graph coloring approach. We have tested the methods with two case studies of varying size of data and constraints. This system has achieved fairness and tries to minimize the exam time period.

A. *Basic concept of Graph and Graph coloring*

Definition 1: A graph G, is a non-empty finite set V of vertices (nodes) and a non-empty finite set E of edges (arcs) [4] [5] [11]. So, a graph G can be represented as an ordered of (V(G), E(G)) consisting of a non-empty set V of vertices or nodes, where E ⊆VxV. An undirected graph could be pictorially represented as Figure-1, where, V = {A, B, C}, and E = {(A, C), (B, C), (A, B)}. Here e1= (A, C), e2= (B, C) and e3= (A, B).

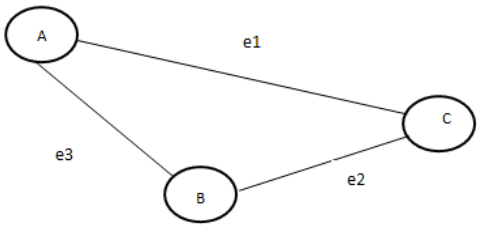

Figure 1. An example of a Graph.

Definition 2: The least number of colors necessary for vertex coloring of a graph G is called the chromatic number of G. It is normally denoted as £(G) [4][5]. It holds two properties:

1. £(G) =1, *iff* G is a null graph.
2. If G is not empty, then £(G) >=2.

Also, if there is a vertex coloring that uses at most *n* colors, then G is said to be n-coverable, it means £(G)<=*n*.

Definition 3: Vertex coloring [5] [7] is the most commonly used graph coloring approach. It is a systematic manner of coloring the vertices of a graph in such a way that no two adjacent vertices have the same color. Two nodes are said to be adjacent if they are connected by an edge. So, vertex coloring of a graph G can be defined as a function Ƒ: V → C where V is a set of vertices of the graph G and C is a set of colors. Proper Ƙ-coloring of G is a coloring function Ƒ which uses exactly Ƙ colors and satisfies the property that Ƒ (x) ≠ Ƒ (y), where (x,y) is adjacent to each other. Edge coloring on the other hand, is a systematic way of assigning colors to the edges of a graph so that no two adjacent edges have the same color. Two edges are said to be adjacent if both of them share a vertex in common. In Figure 2(a)-(b), vertex and edge coloring are shown respectively.

B. *Time Table Scheduling Problem*

The fundamental initiative for designing a optimized conflict-free schedule is to allocate various related resources amongst a number of time-slots fulfilling a range of types of essential and preferential constraints [3][6]. Exam time scheduling is a part of the time table problem. The time table problem could be viewed as a resource allocation problem. Here, the resource could be a faculty member, class room, lab etc.

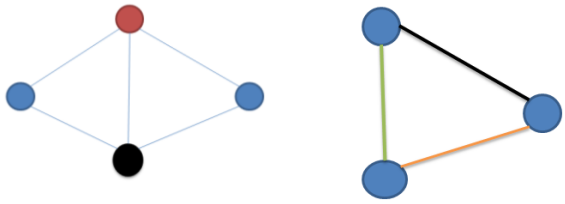

Figure 2. (a) Vertex coloring (b) Edge coloring

The time table problem could be divided into three categories:

1. Exam time scheduling- It involves scheduling the exams for different subjects.
2. Class time table for Universities- It involves creating monthly time table of the university.
3. Class time table for Schools- It involves creating weekly time table of school.

So, after discussing the fundamental and introductory concepts of graph related to coloring and scheduling problem, in the above subsection *A* and *B*, we have presented the literature survey in Section II, which describes the state-of-the-art of several research studies that have been done on scheduling problems using graph coloring method. In Section III formulation of the problem is derived. Methodology with a workflow diagram is discussed in Section IV. In Section V a case study with a worked example is shown. Next Sections VI and VII summarize the conclusion and future work of the present study.

II. RELATED WORK

Solving timetabling problems through the application of modern technology has a long and varied history**.** *Welsh et al.* [1] has illustrated the relationship between timetabling and graph coloring and developed a new general algorithm to give an approximate solution to the minimum coloring problem more efficiently. However, this algorithm needs the vertices to be sorted in descending/ascending order based on their valence (it is the degree of a vertex) in order to have an optimal solution. *Mohammad et al.* [2] have proposed an algorithm to color the schedules of exam time table in two major steps. In the first steps, they have a created a weight matrix and an undirected graph, and then in the second step assign color to different nodes of that undirected graph. By this way, they tried to achieve a fair, accurate, and optimal exam time period by considering different constraints which generally related with university systems. *Burke et al.* [3]

also followed a similar approach like the exam scheduling algorithm using graph coloring. However, in this algorithm, they have addressed only the conflicts without any constraints. Moreover, this algorithm does not eliminate conflicts, and only aims at minimizing conflicts. *Akbulut et al.* [6] proposed a graph coloring algorithm and RFID technology to schedule different exams in same halls simultaneously. This system plans to schedule different exams in same halls at the same time period. The reason is to make use of the hall capacity with 100% performance and decrease cheating attempts. It uses the identity cards of students that have a RFID tag for this purpose. Exam time tabling problem is not only solved by graph coloring approaches but also some researcher has attempted evolutionary approaches [12]. *Jha* [9] has attempted to give a solution to the problem of timetabling, by using the genetic algorithm (GA). Another approach has been carried out by *Verma et al.* [8], where author has proposed a method to schedule timetable and generate it using a combined approach of bacterial foraging and genetic algorithm techniques. But the computational cost of this approach is comparatively high. In a similar work, *Hussain et al.*[7] has presented a method for exam timetabling problem for the centre of foundation studies and extension education, Multimedia University, Malaysia. Here the authors have used graph coloring with the combination of cluster heuristic and sequential heuristic approach for solving the problem. But this approach is not applied for automated or computerized generation of timetable.

III. PROBLEM FORMULATION

A vertex in graph G represents a subject, and an edge between two vertices is drawn only if there is a common student**.** Two vertices could have the same color if and only if they are not adjacent to each other. Since they are not adjacent; the subjects represented by the two vertices could be scheduled at the same time slot. The weight of an edge, if present, represents the number of students who have registered for both the subjects. We need to find the minimum number of colors required to color the graph (i.e. the chromatic number). Thus, we need to find the shortest period within which the exams of all the subjects could be conducted without any conflict. So, by considering a graph G = (V, E). The number of vertex V is equal to the number of subjects and each node represents a subject**.** The Course Matrix of G is a two-dimensional *nxn* array, say course_matrix[n][n], where n is the total number of subjects to be scheduled. Now, if the edge (vi, vj) is present in E(G), then course_matrix[i][j]= 1, and if the edge (vi, vj) is not present in E(G), then course_matrix[i][j] = 0. Also some others terms need to be mentioned:

1. Let H be the set of all periods of a time slot where examination can be conducted. H= $\{h_1, h_2, h_3,\ldots, h_m\}$, where m corresponds to the maximum number of periods of timeslots in a day.
2. Let S be the set of all subjects in a given semester/year, which will be under examination. S= $\{s_1, s_2, s_3,\ldots, s_k\}$, where k is the maximum number of subjects.
3. Let F be the set of faculties available in a given semester examination. F= $\{f_1, f_2, f_3,\ldots, f_k\}$, where k is the number of faculty.
4. Let R be the set of rooms available in a given semester examination. R= $\{r_1, r_2, r_3,\ldots,r_k\}$, where k is the maximum number of rooms available.
5. Let L be the list of combinations of subjects offered to students. L= $\{L_1, L_2, L_3,\ldots, L_i\}$, where L_i ={combination of subjects from set S}.

Also, we need a set of color list C= $\{c_1, c_2, c_3,\ldots, c_n\}$, where *n* is the maximum number of colors. By using these terminologies our primary objective is to minimize the number of colors for solving the problem.

A. Constraints

To designing and solving a scheduling problem, the essential things that are to be considered are the constraints [3]. They are the diverse margins and limitations mixed up in creating a schedule. Based on the fulfilment of constraints, a schedule can be accepted or get rejected. Also in the literature, depending on the degree of strictness, constraints are broadly classified into-two categories namely, hard and soft Constraints [3]. The exam time table should be generated by keeping in mind the number of students, the number of faculty members available to conduct the exams, the availability of rooms, and the capacity of the available rooms.

Hard constraints are compulsory. That means the hard constraints must have to be satisfied for the time table to be feasible. Soft constraints are those constraints, which are not compulsory to be satisfied. However, if satisfied, they make the time table more acceptable.

Some of the hard constraints (also known as essential time tabling conditions) are listed below:

1. Two or more subjects or papers of a student should not be scheduled at the same time.
2. The examination for each subject should be held exactly once.

3. Subjects should be scheduled accordingly to the availability of rooms.
4. Subjects should be scheduled accordingly to the availability of faculty members.

Some of the soft constraints [3][2] (also known as preferential time tabling conditions) are shown below:

1. There should not be any exam on Sunday or holidays.
2. There should be a gap between two exams of a particular stream.
3. Classrooms should be large enough to accommodate the number of students for a particular subject.

IV. METHODOLOGY

In this section, various steps that need to be performed to schedule an exam timetable are discussed. The overall workflow diagram for this work is shown in Figure 3. The system would first take the inputs from the user. So, we need to enter the following information:

1. The list of subjects or papers offered in each semester.
2. The number of students registered in each paper.
3. The number of faculties available to conduct the examinations.
4. The number of rooms available.
5. The capacity of each room should be entered. Each room is identified by a unique name; say an alphabet, followed by the capacity of the room. Example, A50 where A is the unique name and 50 is the capacity of the room.

Then it generates the course matrix using Method-1.To generates the course matrix following steps is being performed:

Method-1:

Step 1: Create an empty *nxn* two dimensional array course_matrix[n][n].

Step 2: Pick a vertex V_i and find the adjacent vertex V_j to make an edge between them.

Step 3: For finding adjacency of V_i , perform a searching in the list L of subjects where it presents.

Step 3.1 If $V_i \in L_i$ then all the subjects in L_i are adjacent to V_i , so assign course_matrix[i][j]= 1

Step 3.2 else course_matrix[i][j] = 0.

Step 4: Repeat the step 2 and step 3 until all the pairs of subjects are not assigned with '0' or '1'.

Based on the course matrix, the system would detect the colliding subjects and assign the colors to the nodes, which represent the subjects. Two courses are said to collide each other if they are adjacent. Adjacent subjects could not be assigned the same color.

For coloring the vertices, following steps are performed and it is shown in Method-2.

Method-2:

Step 1: Create an empty array Arr[n] of size *n*. Each index represents the subjects and initialized the array to '0'. Here, *n* = total no. of subjects.

Step 2: Generate a color array C ={c_1, c_2, c_3….c_k}. Declare an array color_used[k].

Step 3: Assign c1 to Arr[1] . Then Store c1 to color_used[1] .

Step 4: For each Row [1to n] of course_matrix, excluding the first Row [1].

Step 5: For each index i of Arr, excluding the first index. Perform the following steps.

Step 6: Assign new color from C to Arr[i] which is not previously used

Step 6.1 If V_i adjacent to Row [1 to i].

Step 6.2 Then Store color to color_used array.

Step 7: Else Assign color to non-adjacent uncolored vertices from color_used.

Step 8: Repeat *step 4* to *step 7* until all vertices are colored.

Once the color values or colors are assigned, the list of subjects will be sorted based on the color value assigned by using the Merge-sort algorithm. We sort the list to ease the task of generating the exam time schedule and eliminate the need for searching the list again and again. Finally, it would generate the exam time schedule and allocate the suitable room/rooms. The rooms would be allocated depending on the number of students registered in the subjects who are going to be scheduled in the same time slot with available room capacity, and the number of faculties available to conduct the exams.

V. AN EXPERIMENTAL CASE STUDY

In India, generally in Undergraduate/Postgraduate colleges/universities offers a variety of subject combinations to students. In streams like Bachelor of Arts (B.A)/Bachelor of Science (B.Sc) or Master of Arts (M.A) /Master of Science (M.Sc) students can take some combinations of subjects with elective and core papers. Also, some students have some back papers of some previous semesters. So those papers also will be included while offering the subjects. While designing the time table availability of the number of faculties plus numbers of room availability also needed. In this section, we have considered a scenario of an individual department of Computer application, where different subjects are offered by a university in an odd semester of a particular year. The number of papers or subjects offered in different

semesters, number of faculties, numbers of students and number of available rooms are taken into consideration for generating a conflict and constraint free solution to final semester exam timetable.

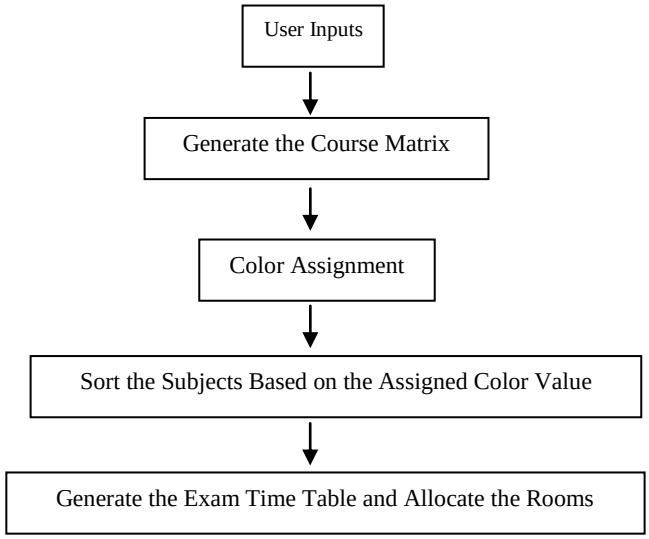

Figure 3. Workflow diagram of the methodology

Suppose we have entered the following information, with core subjects, elective subjects and back-papers.

1. First-Semester subjects: Digital Logic (DL), C Programming (C), Organizational Behaviour (OB), Accounting (AC), Discrete Mathematics (MATH)
2. Third-Semester subjects: Database Management System (DBMS), Data Communications and Network Technologies (DCN), Operating System (OS), Algorithm (ALG), VB.NET (VB), DL, C, MATH.
   NOTE: DL, C, and MATH are the first semester subjects. However, we are adding these subjects in the third semester because we are assuming that some third semester students have got back paper in these three papers and thus need to reappear. Here, Algorithm (ALG) is an elective paper.
3. Fifth-Semester subjects: Compiler Design (CD), System Programming (SP), Data Mining (DM), JAVA, C, ALG.
   NOTE: C, and ALG are first and third semester subjects respectively. However, we are adding these subjects in the Fifth semester because we are assuming that some Fifth semester students have back papers in these subjects and thus need to reappear. Here, Data Mining is an elective paper.
4. Enter the number of students enrolled in each subject.
5. Number of rooms available is 9 (e.g., A30, B30, C30, D40, E40, F30, G40, H60, I40).
6. Number of faculties to conduct the examination is say, seven (7).
7. Starting Date of the exams and Time slots. Starting date : Dec-01, 2017. And Time slots 10 a.m. to1p.m. and 1.30 to 4.30 p.m.

A. Generating the Course Matrix

The next step is to generate the course matrix using Method - 1, based on the data provided by the user. Here, we are generating the course matrix based on the list of subjects offered in each semester including the back papers of the previous semesters. The course matrix generated for the above data is shown in Table-3 and the graph which it will generate is shown in Figure 4.

B. Color Assignment

We assign colors to the subjects based on the course matrix using Method-2. In order to assign a color to subjects we need to find out the colliding subjects. Two courses are said to collide each other if they are adjacent. Adjacent courses could not be assigned the same color. In Table-1, if we observe the first row we see that DL and C could not be assigned the same color since they are adjacent. However, DL and CD could be assigned the same color since they are not adjacent

Now, we take a one-dimensional array of size *n*, *n* = 14, it is shown in Table-1. Each index of the array represents the subjects. Now initialized the array to 0, also a Color array is initialized: {1=Green, 2=Red, 3=Blue, 4=Black, 5=light-orange, 6= light-purple, 7= Orange, 8= light-Blue 9=Pink, 10= Maroon} like that.

Table 1. Color assignment of Digital Logic (DL)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	0	0	0	0	0	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Since, we start with Digital Logic (DL), we assign color 1 to it. It is shown in Table-2.

Table 2. Color assignment of Digital Logic (DL)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	0	0	0	0	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Table 3. Course matrix

Subject	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG	VB	CD	SP	DM	JAVA
DL	-	1	1	1	1	1	1	1	1	1	0	0	0	0
C	1	-	1	1	1	1	1	1	1	1	1	1	1	1
OB	1	1	-	1	1	0	0	0	0	0	0	0	0	0
AC	1	1	1	-	1	0	0	0	0	0	0	0	0	0
MATH	1	1	1	1	-	1	1	1	1	1	0	0	0	0
DBMS	1	1	0	0	1	-	1	1	1	1	0	0	0	0
DCN	1	1	0	0	1	1	-	1	1	1	0	0	0	0
OS	1	1	0	0	1	1	1	-	1	1	0	0	0	0
ALG	1	1	0	0	1	1	1	1	-	1	1	1	1	1
VB	1	1	0	0	1	1	1	1	1	-	0	0	0	0
CD	0	1	0	0	0	0	0	0	1	0	-	1	1	1
SP	0	1	0	0	0	0	0	0	1	0	1	-	1	1
DM	0	1	0	0	0	0	0	0	1	0	1	1	-	1
JAVA	0	1	0	0	0	0	0	0	1	0	1	1	1	-

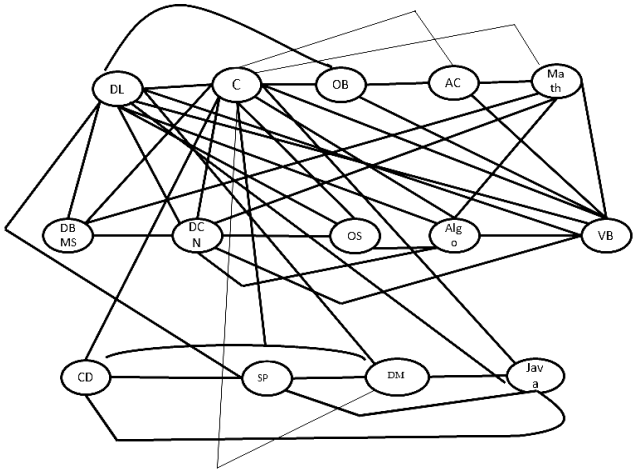

Figure 4. Course matrix graph

Now, we check the second row of the course matrix. Subject C is adjacent to DL. Thus, we need to assign a different color to C and we assign color 2. It is shown in Table-4.

Table 4. Color assignment of C-programming (C)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	0	0	0	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Now, we check the third row of the course matrix where OB is adjacent to both DL and C. So, we assign a new color to OB. It is shown in Table-5.

Table 5. Color assignment for Organizational Behavior (OB)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	0	0	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Now, we check the fourth row of the course matrix where Ac is adjacent to DL, C and OB. So, we assign a new color. It is shown in Table-6.

Table 6. Color assignment of Accounting (AC)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	0	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Now, we check the fifth row of the course matrix, where MATH is adjacent to DL, C, OB and AC. So, we assign a new color to MATH. It is shown in Table-7.

Table 7. Color assignment of Discrete Maths (MATH)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	0	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Now, we check the sixth row of the course matrix. Here, DBMS is adjacent to both DL and C, however, it is not adjacent to OB. So, we could assign DBMS the same color which we have previously assigned to OB. It is shown in Table-8.

Table 8. Color assignment of Data-Base Management system (DBMS)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	0	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Now, we check the seventh row of the course matrix, where DCN is adjacent to both DL and C, however, it is not adjacent to OB. Here, we need to note that DBMS and DCN are in the same semester and thus they are adjacent, and DBMS has already been assigned the color 3 (which is also the color of OB). So, we could not assign DCN the color 3. So, we check the next subject (in the next column) which is Ac. DCN and Ac are not adjacent and also the color assigned to Ac is not yet assigned to any other subjects of the third semester. Thus, we could assign DCN the same color that we have previously assigned to Ac. It is shown in Table-9.

Table 9. Color assignment of Data Communication (DCN)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	0	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Similarly, we check the eighth row of the course matrix, where OS is adjacent to DL and C, but it is not adjacent to OB and Ac. However, color 3 and 4 are already assigned to DBMS-I and DCN-I, which are in the same semester as that of OS. And OS is adjacent to MATH as well. So, we need to assign a new color for OS. It is shown in Table-10.

Table 10. Color assignment of Operating system (OS)

Subjects	DL	C	OB	Ac	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	0

VB	CD	SP	DM	JAVA
0	0	0	0	0

Similarly, for Algorithm in row ninth of the course matrix, we need to assign a new color. It is shown in Table-11.

Table 11. Color assignment of Algorithm (ALG)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
0	0	0	0	0

Similarly, for VB in row tenth of the course matrix, we need to assign a new color. It is shown in Table-12.

Table 12. Color assignment of Visual Basic.Net (VB)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
8	0	0	0	0

Now, we check the eleventh row of the subject matrix. Here, CD is a subject offered in the fifth semester and it is not adjacent to DL. Also none of the subjects offered in the fifth semester has the color 1 assigned to it. Thus, we could assign the color 1, which was previously assigned to DL, to CD. It is shown in Table-13.

Table 13. Color assignment of compiler design (CD)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
8	1	0	0	0

Now, in the twelve row of the course matrix, SP is not adjacent to DL, but color 1 has already been assigned to CD, and CD and SP are offered in the same semester. Thus, they are adjacent and we could not assign the same color to two adjacent subjects. So, we check the next subject which is C. C is adjacent to SP. Then we check the next subject which is OB. OB is not adjacent to SP, and the color 3 is not yet assigned to any of the subjects of the fifth semester. So, assign SP the same color which we have previously assigned to OB. It is shown in Table-14.

Table 14. Color assignment of System Programming (SP)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
8	1	3	0	0

In the same manner, after observing the thirteenth row of the course matrix, DM is being assigned the color which was previously assigned to Ac. It is shown in Table-15.

Table 15. Color assignment of Data Mining (DM)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
8	1	3	4	0

Finally, after observing the fourteenth row of the course matrix, Java is being assigned the color which was previously assigned to MATH. It is shown in Table-16.

Table 16. Color assignment of Java (JAVA)

Subjects	DL	C	OB	AC	MATH	DBMS	DCN	OS	ALG
Color	1	2	3	4	5	3	4	6	7

VB	CD	SP	DM	JAVA
8	1	3	4	5

C. Sorting the Subjects Based on the Color Value

In the next step, we sort the list of courses based on the color values which are assigned automatically. We have used the standard merge sort algorithm [5], as its performance is far better than other sorting algorithms. In worst and best cases the time complexity is Big Theta Θ (*nlogn*). The main objective of this sorting phase is to reduce the computation time for designing the exam time table. Because by performing this phase, the similar types of color assigned subjects resides together. For this small case study, the number of subjects may be less, but in practical scenario sorting the subjects based on color values will enhance the computational performance.

So, by considering the same example, after sorting the list of subjects, the subjects are arranged in the following form which given in Table 17.

Table 17. Sorted list of courses based on colors.

Subjects	DL	CD	C	OB	DBMS	SP	AC	DCN	DM
Color	1	1	2	3	3	3	4	4	4

MATH	JAVA	OS	ALG	VB
5	5	6	7	8

D. Generate the Exam Time Schedule and Allocate Rooms

To generate the final Exam time table, we need the following inputs:

1. **Fix the number of rooms-** First, we need to fix the number of rooms where the exams would be conducted based on the total number of faculties available to conduct the exams. There may be a situation where the number of rooms available where the exams would be conducted is larger than the number of faculties available to conduct the exams. In such a situation, we need to reduce the number of rooms where the exams would be held such that the available number of faculties could conduct the exams nicely.
2. **Set the concurrency value-** Concurrency value determines the number of subjects that could be scheduled for examination in the same time slot. This value is calculated based on the total number of students registered in all the subjects which have the same color value assigned, and the total room capacity (i.e. the capacity of all the rooms taken together). Suppose, if the total number of students registered in the subjects having the same color value is greater than the total room capacity, then the concurrency value would be decreased.

Generating the Exam Time Schedule:

We generate the exam time table based on the color value assigned to the subjects, the concurrency value, and/or the number of students registered on each individual subject.

Allocation of Rooms:

1. We calculate the total number of students registered in all the subjects which are scheduled in the same time slot.
2. We find the room which has the capacity greater than or equal to the number of students, and allots that room for conducting the exams for the scheduled subjects.
3. If the total number of student is greater than any single room capacity, then we allot more than one room for conducting the exam. And the capacity of the allotted rooms should be greater than or equal to the number of students.

Finally, we print the Exam Time Schedule along with the rooms allocated. From the implementation point view, we have designed our program using Java, which runs on a computer with following configuration: Processor: Intel® Corei5 2.5 GHz, Operating System: Microsoft® Windows 7 Ultimate, RAM: 3GB.

Continuing with our previous example, let us consider we have the following amount of data.

Say, the total number of room is 3.

1. A55 (Here 55 is the room capacity)
2. B60
3. C70

Total room capacity = 185. The number of students registered in each subject is given in Table-18. The number of faculties available = 7. Now, continuing with our previous example and the course matrix given in Table-1, along with the above details, the exam time schedule generated by the system would be given below as an Output.

Table 18. Enrolment data of students

Subject	No. of Students
DL	53
C	55
OB	50
AC	50
MATH	53
DBMS	50
DCN	50
OS	50
ALG	53
VB	50
CD	50
SP	50
DM	50
JAVA	50

Output:

Fri 01 Dec 2017.

Morning 10:00am to 01:00pm
Subject: CD Subject: DL Room: A55, B60
Afternoon 01:30pm to 04:30pm
Subject: C Room: A55

Sat 02 Dec 2017.

Morning 10:00am to 01:00pm
Subject: DBMS-1 Subject: SP Subject: OB Room: A55, B60, C70
Afternoon 01:30pm to 04:30pm
Subject: DM Subject: DCN Subject: AC Room: A55, B60, C70

Mon 04 Dec 2017.

Morning 10:00am to 01:00pm
Subject: MATH Subject: JAVA Room: A55, B60
Afternoon 01:30pm to 04:30pm
Subject: OS Room: A55

Tue 05 Dec 2017.

Morning 10:00am to 01:00pm
Subject: ALG Room: A55
Afternoon 01:30pm to 04:30pm
Subject: VB Room: A55

The final colored graph generated for the above data is shown in Figure.5.

VI. DISCUSSION

The output of the timetabling which our model has generated for the case study is that the number of days required to conduct the various exams is four (4) and the chromatic number is eight (8). For instance the Black color is assigned to subject set {DCN, DM, AC}, so according to other constraint satisfactions, these paper will be conducted on same Date and slot. Also, if we observe, particularly a Day say, Fri 01 Dec 2017, in the morning half, Subjects like Compiler Design (CD) and Digital Logic (DL) are conducted on different rooms' viz., Room no: A55, B60. Similarly in the afternoon half Subject like C programming is conducted on room no- A55, without any conflicts. So, similarly on different Dates also various rooms are assigned properly without any conflicts. This is very efficient and less time consuming. Moreover, none of the subjects is overlapped with any other subjects. So it satisfies the hard and soft constraints.

Moreover, for a real time scenario, we have considered another case study of a University system. From different six departments, we have collected the subjects name and papers with their individual code and streams. All total 68 subjects are there and then we have generated the various Lists of subjects by consulting with the concerned Departments. Number of Faculty and room availability also collected and we have tested in our system. In this case study also none of the subjects were overlapped with any other subjects. So it satisfies the hard and soft constraints. The output which we have observed is efficient and satisfactory.

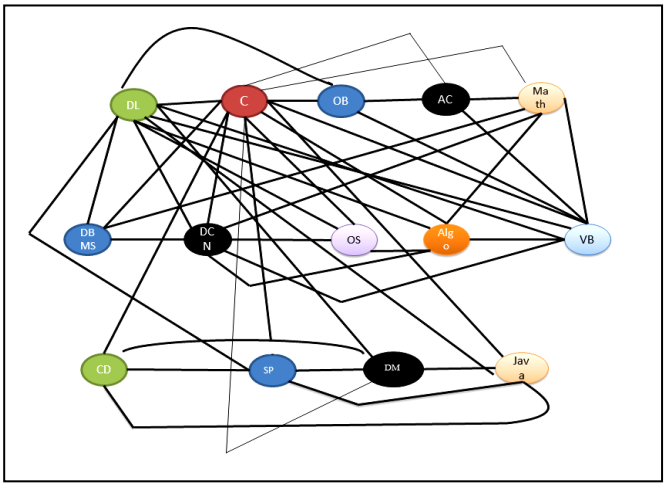

Figure5. Final colored Graph

VII. CONCLUSION & FUTURE WORK

The Time table scheduling problems are directly related to the number of constraints involved. Higher number of constraints might raise more numbers of complexities. There is no fixed algorithm to solve this class of problem. In this work, we

generated the exam time schedule automatically with the help of graph coloring approach by taking the list of subjects, number of faculties available to conduct the exams, number of rooms available for conducting the exams, and the room capacity as inputs. Also, it has successfully satisfied some of the important hard and soft constraints. The system also achieves fairness, accuracy, and optimal exam time period. In future, we are planning to add the class routine generation problem with the help of a teacher-subject conflict matrix by applying edge-coloring with bipartite graph. Also with a large numbers of input data system will be tested.

REFERENCES

[1] Welsh D.J.A., and Powell M.B., "*An Upper Bound for the Chromatic Number of a Graph and it's Application to Timetabling Problems*", The Computer Journal. (1967), Vol.10, No.1, pp. 85-86.

[2] Mohammad M., Mohammad A.H., and Osama A.H., "*A new exam scheduling algorithm using graph coloring*", The International Arab Journal of Information Technology, (2008), Vol. 5, No.1, pp. 80- 86.

[3] Burke K.E., Mccollum B., Meisels A.,and Petrovic S., "*A Graph-Based Hyper-Heuristic for Educational Timetabling Problems*", European Journal of Operational Research (2007), Vol-176, pp. 177-192.

[4] Somasundaram M.R., "*Discrete Mathematical structures*", 2nd edition, PHI, 2010.

[5] Bhasin H., "*Algorithms Design and Analysis*", 1st edition, Oxford University Press, 2015.

[6] Akbulut A., and Yılmaz G., "*University Exam Scheduling System Using Graph Coloring Algorithm and RFID Technology*", International Journal of Innovation, Management and Technology, (2013), Vol. 4, No. 1, pp. 66-72.

[7] Hussain B., Basari A.S.H., and Asmai S.A., "*Exam Timetabling Using Graph Colouring Approach*", In the proceedings of IEEE Conference on Open Systems (ICOS2011), (2011), pp.139-144.

[8] Verma O.P., Garg. R.,and Bisht V.S., "*Optimal Time-Table Generation by Hybridized Bacterial" Foraging and Genetic Algorithm*", In Proceedings of International Conference on Communication Systems and Network Technologies (CSNT'12), (2012), pp. 919-923.

[9] Jha. S.K., "*Exam Timetabling Problem using Genetic algorithm*", International Journal of Research in Engineering and Technology, (2014),Vol.3, No.5, pp. 649-655.

[10] Alon N., "*A Note on Graph Colorings and Graph Polynomials*," Journal of Combinatorial Theory Series B", (1997), Vol. 70, No. 1, pp. 197-201.

[11] Gross J. and Yellen J., Handbook of Graph Theory, Discrete Mathematics and its Applications, CRC Press, Vol. 25, 2003.

[12] Norberciak. M., "*Universal Method for Timetable Construction based on Evolutionary Approach*" World Academy of Science, Engineering and Technology, (2006), pp. 91-96.

Authors Profile

Rubul Kumar Bania, is an Assistant Professor in the Department of Computer Applications, North-Eastern Hill University, Tura campus, Meghalaya, India. He has completed his Master of Computer Application (MCA) in 2008 and Master of Technology in 2012 from Tezpur Central University, Assam, India. He has published research papers in reputed peer reviewed international journals and IEEE conference and it's also available online. He has more than 5 years of teaching experience. His research area of interest is Data mining, Image processing and Artificial intelligence.

Pinkey Duarah is a Post Graduate student and pursuing her Master of Computer Application (MCA) in the Department of Computer Application, North-Eastern Hill University, Tura Campus, Meghalaya, India

