# EXAM TIME TABLE SCHEDULER

**Sonali Rasal**
Department of Computer Engineering Thakur college of Engineering & Technology
Mumbai, India
rasalsonali24@gmail.com

**Pranjal Singh**
Department of Computer Engineering Thakur college of Engineering & Technology
Mumbai, India
pranjal366@gmail.com

**Aroma Sinha**
Department of Computer Engineering Thakur college of Engineering & Technology
Mumbai, India
askurkure123@gmail.com

**Ms. Nikki Modi**
Department of Computer Engineering, Thakur college of Engineering & Technology Mumbai, India
nikki.modi@thakureducation.org

***Abstract*—*In any educational institution, the two most common academic scheduling problems are course timetabling and exam timetabling. A schedule is desirable which combines resources like teachers, subjects, students in a way to avoid conflicts satisfying various essential and preferential constraints.Hence a heuristic approach is preferred to find a nearest optimal solution within reasonable running time. Graph coloring is one such heuristic algorithm that can deal timetable scheduling satisfying changing requirements, evolving subject demands and their combinations. It emphasizes on degree of constraint satisfaction, even distribution of courses, test for uniqueness of solution and optimal outcome. It aims at properly coloring the course conflict graph and transforming this coloring into conflict-free timeslots of courses. exam Conflict graph is constructed with courses as nodes and edges drawn between conflicting courses i.e. having common students. Exam scheduling is a challenging task for educational institutions. We evaluate our algorithm on real-world datasets and compare it with existing approaches. Our results show that the proposed algorithm is effective in generating conflict-free exam schedules in a reasonable amount of time.***

***Keywords*—*Graphs, Backtracking***

## I. INTRODUCTION

In the year 1736, graph theory originated from the Konigsberg bridge problem pointed out by mathematician Euler which later led to the concept of Eulerian graph [4]. In the same decade, Gustav Kirchhoff established the concept of a tree, a connected graph without cycles which was used in the calculation of currents in electrical networks or circuits and later to enumerate chemical molecules. In 1840, A.F Mobius came up with the idea of complete graph and bipartite graph (section 1.1). In 1852, Thomas Gutherie found the famous four-color problem. The first results about graph coloring deal exclusively with planar graphs in the form of the coloring of maps [3]. Even though the four-color problem was invented it was solved only after a century by Kenneth Appel and Wolfgang Haken [1]. In 1890, Heawood proved the five-color theorem, saying that every planar map can be colored with no more than five colors [2]. In 1912, George David Birkhoff to study coloring problems in algebraic graph theory introduced the chromatic polynomial [4][5]. Graph Coloring has many real-time applications including map coloring, scheduling problem, parallel computation, network design, sudoku, register allocation, bipartite graph detection, etc [3][4]. Graph coloring has considerable application to a large variety of complex problems involving optimization [11]

While constructing a schedule of courses at a college or university, it is obvious that courses taught by the same professor and courses that require the same classroom must be scheduled at different time slots. Furthermore, a particular student or group of students may be required by a curriculum to take two different but related courses (e.g., physics and Mathematics) concurrently during a semester. In such cases too, courses need to be scheduled in a way to avoid conflicts. Thus, the problem of determining a minimum or reasonable number of time slots that can successfully schedule all the courses subject to restrictions is a typical graph coloring problem [14][16][17].

## II. LITERATURE SURVEY

Solving timetabling problems through application of computers has a long and varied history. In 1967, the problem of course scheduling was applied to graph coloring [6]. In 1967 Welsh and Powell [10] illustrating the relationship between timetabling and graph coloring, and developed a new general graph coloring algorithm to solve (or approximately solve) the minimum coloring problem more efficiently. They were also successful in coloring graphs that arise from timetabling problems, more specifically examination timetabling problems.

In 1969, Wood's graph algorithm [14] operated on two n × n matrices, where n denotes the number of vertices in the graph; a conflict matrix C was used to illustrate which pairs of vertices must be colored differently due to constraint restrictions in the problem and a similarity matrix S was used to determine which pairs of vertices should be colored the same. Dutton and Bingham in 1981 introduced two of the most popular heuristic graph coloring algorithms. Considering each color one by one, a clique [4] is formed by continually merging the two vertices with the most common adjacent vertices. On completion, identical coloring is applied to all the vertices which are merged into the same.

In A Study on Course Timetable Scheduling using Graph Coloring Approach 473 1991, Johnson, Aragon, McGeoch and Schevon [20] implemented and tested three different approaches for graph coloring with a simulated annealing technique, observing that simulated annealing algorithms can achieve good results, but only if allowed a sufficiently large run time. In 1992, Kiaer and Yellen in a paper [21] describes a heuristic algorithm using graph coloring approach to find approximate solutions for a university course timetabling problem. The algorithm using a weighted graph to model the problem aimed at finding a least cost k-coloring of the graph (k being number of available timeslots) while minimizing conflicts.

In 1995, graph coloring method was introduced aiming at optimizing solutions to the timetable scheduling problems [11]. Bresina (1996) was among the early researchers who used this approach and made several modifications in the manual approach conducted at universities [23].

In 2007, for university timetabling an alternative graph coloring method was presented that incorporates room assignment during the coloring process [6]. In 2008, the Koala graph coloring library was developed which includes many practical applications of graph coloring, and is based on C++ [7]. In 2009, automata-based approximation algorithms were proposed for solving the minimum vertex coloring problem [8].

## III. PROBLEM STATEMENT

Exam Time-Table Generator is simple and easy to use application which takes Semester Subjects and Maximum number of slots as input from the user and displays the Schedule for examination with no subjects of same semester clashing in slot. The Application utilizes the Concepts of Graph and Graph coloring algorithm to generate a schedule for examination.

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 vast numbers of offered courses makes it difficult to schedule exams in a limited epoch of time and often leads to clashing subjects of Subjects for similar semesters.

## IV. METHODOLOGY

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. For Every Adjacent Vertex we gave it different color so that it doesn't clashes in same slot.

Time Complexity of Algorithm: O(n2)

Space Complexity of Algorithm: O(n2) , where, n is number of Subjects

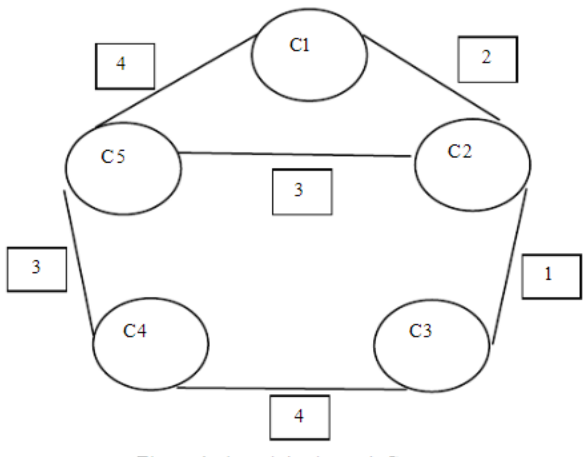


**Fig 1. Graph representation**

The Mathematical Model

Various steps that need to be performed to schedule an exam timetable are discussed. The overall workflow diagram for this work.

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.

Method-1:

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

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

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

Step 3.1 If Vi ϵ Li then all the subjects in Li are adjacent to Vi , 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 ={c1, c2, c3….ck}. 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 Vi 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.

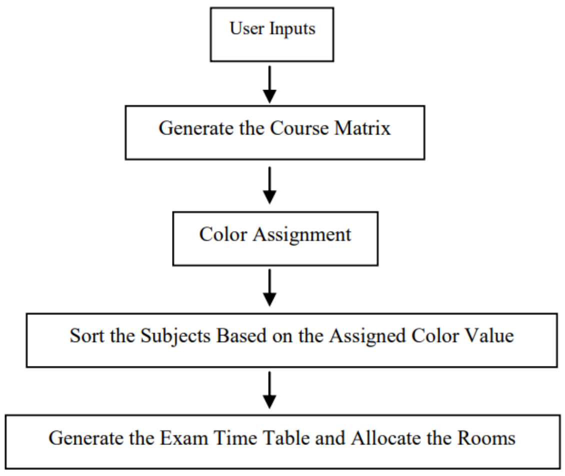

User Inputs
Generate the Course Matrix
Color Assignment
Sort the Subjects Based on the Assigned Color Value
Generate the Exam Time Table and Allocate the Rooms

**Fig 2. Flowchart**

System Architecture:

The system architecture for an exam time table scheduler using the backtracking algorithm can be divided into several components:

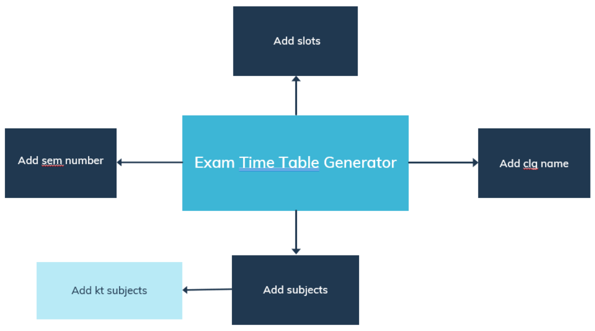


**Fig 3. System Architecture**

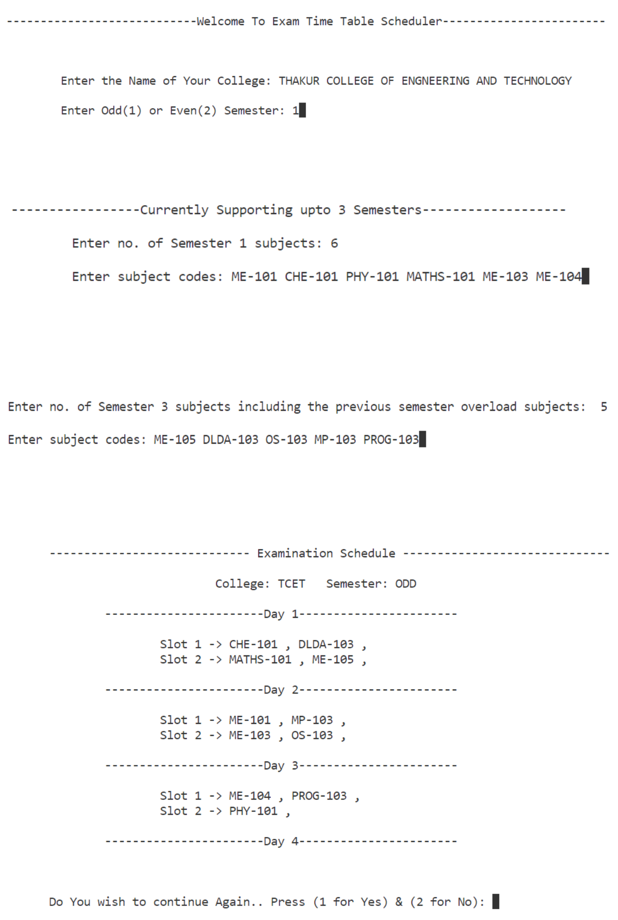
1. User interface: The user interface component provides an interface for users to interact with the system, such as entering exam data, viewing the exam schedule, and making changes to the schedule.
2. Data management: The data management component handles the storage and retrieval of data related to the exam schedule, such as student data, course information, and exam dates and times. This component can use a database management system to store and manage the data.
3. Scheduling engine: The scheduling engine is the core component of the exam time table scheduler, which uses the backtracking algorithm to generate an optimal exam schedule based on the input data. The scheduling engine takes into account various constraints, such as classroom availability, invigilator availability, and student preferences.
4. Reporting and analytics: The reporting and analytics component generates reports and analytics based on the exam schedule, such as the number of exams scheduled per day, the number of exams for each course, and the overall efficiency of the scheduling process.
5. Integration: The integration component integrates the exam time table scheduler with other systems used by the educational institution, such as student information systems, course management systems, and learning management systems.

## V. RESULT AND DISCUSSION

Our backtracking algorithm works by assigning exams to time slots one by one, starting from the first day and proceeding to the last day. The algorithm uses a recursive approach, which explores all possible solutions and backtracks when a conflict arises.

The algorithm maintains a list of unassigned exams and a list of available time slots for each exam. It then selects an unassigned exam and iterates through all available time slots. If a time slot is found that does not conflict with any other assigned exams, the exam is assigned to that time slot, and the algorithm moves on to the next unassigned exam. If no feasible time slot is found, the algorithm backtracks to the previous exam and tries a different time slot.

The algorithm continues this process until all exams are assigned to time slots or until it determines that no feasible solution exists. In the latter case, the algorithm backtracks to the previous exam and tries a different time slot until all possibilities are exhausted.



```
---------------------------Welcome To Exam Time Table Scheduler------------------------

        Enter the Name of Your College: THAKUR COLLEGE OF ENGNEERING AND TECHNOLOGY

        Enter Odd(1) or Even(2) Semester: 1


 -----------------Currently Supporting upto 3 Semesters-------------------

          Enter no. of Semester 1 subjects: 6

          Enter subject codes: ME-101 CHE-101 PHY-101 MATHS-101 ME-103 ME-104



Enter no. of Semester 3 subjects including the previous semester overload subjects:  5

Enter subject codes: ME-105 DLDA-103 OS-103 MP-103 PROG-103



      ----------------------------- Examination Schedule ------------------------------

                         College: TCET   Semester: ODD

              -----------------------Day 1-----------------------

                      Slot 1 -> CHE-101 , DLDA-103 ,
                      Slot 2 -> MATHS-101 , ME-105 ,

              -----------------------Day 2-----------------------

                      Slot 1 -> ME-101 , MP-103 ,
                      Slot 2 -> ME-103 , OS-103 ,

              -----------------------Day 3-----------------------

                      Slot 1 -> ME-104 , PROG-103 ,
                      Slot 2 -> PHY-101 ,

              -----------------------Day 4-----------------------


      Do You wish to continue Again.. Press (1 for Yes) & (2 for No):
```

**Fig 4. Model representation**

## VI. FUTURE SCOPE

The problem of exam time table scheduling is a challenging task that has been studied extensively in the past. However, there is still room for improvement and research in this field. Some possible future directions for this research include:

1. Hybrid Approaches: Hybrid approaches that combine multiple algorithms such as Genetic Algorithm and Tabu Search can potentially lead to better solutions. For example, a hybrid algorithm can use Genetic Algorithm to generate initial solutions and then use Tabu Search to refine them further.
2. Constraint Programming: Constraint Programming is a technique that has shown promising results in solving scheduling problems. It can be used to model the constraints and requirements of the problem and solve it efficiently. Constraint Programming can be used in conjunction with Backtracking Algorithm to improve the efficiency of the algorithm.
3. Machine Learning: Machine Learning techniques such as Reinforcement Learning can be used to learn from past solutions and improve the efficiency of the algorithm. For example, a Reinforcement Learning-based algorithm can learn from past solutions and adapt to new problem instances.

## VII. CONCLUSION

The complexity of a scheduling problem is directly proportional to the number of constraints involved. There is no fixed algorithm to solve this class of problem. Here we have studied a typical honours(major) and general(minor) course combination scheduling problem under university curriculum. Uniqueness and optimality are the main concerns in this scheduling. For the same chromatic number, there are many alternative solutions, and thus it is not unique. Although all the solutions can be claimed optimal when solved using minimum number of colors, a better schedule is one which maximizes satisfaction of soft constraints among its alternative solutions. In addition, we have also studied a teacher-subject scheduling problem where two alternative graph coloring methods (edge coloring using bipartite graph and vertex coloring using line graph) were applied and a complete solution is provided. The dynamic nature of scheduling problem challenges to further experiment with large data sets and complex constraints. An algorithm which can evenly distribute resources among available time-slots without conflict, create unique and optimized schedule and satisfy all hard and maximum number of soft constraints can be called ideal. Finding such algorithm is surely an evolving area of further research.

## VIII. ACKNOWLEDGEMENT

We gratefully acknowledge the support, guidance, and encouragement of my Dissertation Guide Assistant Professor Ms. Nikki Modi ma'am for this novel work.

## REFERENCE

[1] Kenneth, A., and Wolfgang, H., 1977, "Every Planar Map is Four Colorable. I. Discharging," Illinois Journal of Mathematics, 21(3), pp. 429–490.

[2] Heawood, P. J., 1980, "Map-Colour Theorems, 1980," Quarterly Journal of Mathematics, Oxford, 24, pp. 332–338.

[3] Birkhoff, G.D., 1912, "A determinant formula for the number of ways of coloring a map," Annals of Mathematics, 14, pp. 42-46. 484 Runa Ganguli and Siddhartha Roy

[4] Deo, N., 1990, "Graph theory with applications to engineering and computer science," Prentice Hall of India.

[5] Bondy, J. A., 1969, "Bounds for the chromatic number of a graph," Journal of Combinatorial Theory, 7, pp. 96-98.

[6] Redl, T.A., 2007, "University Timetabling via Graph Coloring: An Alternative Approach," Congressus Numerantium, 187, pp. 174-186.

[7] Dobrolowski, T., Dereniowski, D., and Kuszner, L., 2008, "Koala Graph Coloring Library: An Open Graph Coloring Library for Real World Applications," Proceedings of the 2008 1st International Conference on Information Technology, IT, Gdansk, Poland, pp. 1-4.

[8] Torkestani, J.A., and Meybodi, M.R., 2009, "Graph Coloring Problem Based on Learning Automata," International Conference on Information Management and Engineering, pp. 718-722.

[9] Klotz, W., 2002, "Graph Coloring Algorithms," Mathematic- Bericht 5, TU Clausthal, pp. 1-9.

[10] Welsh, D.J.A., and Powell, M.B., 1967, "An Upper Bound for the Chromatic Number of a Graph and it's Application to Timetabling Problems," The Computer Journal. 10(1), pp. 85-86.

[11] Miner, S.K., Elmohamed, S., and Yau, H.W., 1995, "Optimizing Timetabling Solutions using Graph Coloring," NPAC, Syracuse University, pp. 99-106.

[12] Garey, R. D., and Johnson, S., 1979, "Computers and intractability: A Guide to the Theory of NP-Completeness," Freeman.

[13] Burke, E. K., Elliman, D. G., and Weare, R., 1993, "A university timetabling system based on graph coloring and constraint manipulation," Journal of Research on Computing in Education, 27(1), pp. 1-18.

[14] Wood, D. C., 1968, "A System for Computing University Examination Timetables," The Computer Journal, 11, pp.41-47.

[15] Karp, R. M., 1972, "Reducibility among Combinatorial Problems," in Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, Eds., The IBM Research Symposia Series, New York, NY: Plenum Press, 1972, pp. 85-103.

[16] Bondy, S., 2002, "Final Examination Scheduling," Communications of the ACM, 22(7), pp. 494-498.

[17] Peck, J. E. L., and Williams, M. R., 1966, "Examination Scheduling," Communications of the ACM, 9(6), pp. 433-434.

[18] Wood, D. C., 1969, "A Technique for Coloring a Graph Applicable to Large Scale Timetabling Problems," The Computer Journal, 12, pp. 317