DOI: 10.21917/ijsc.2022.0382

EXAMINATION SCHEDULER USING A LINEAR-TIME GRAPH COLORING ALGORITHM

Debabrata Datta, Rush Guha, Neelabha Banerjee, Sohan Adhikary and Anal Acharya
Department of Computer Science, St. Xavier's College, India

Abstract

The primary aim of the study aims to provide a solution for scheduling examinations for most of the universities and colleges across India which follow the Choice Based Credit System (CBCS) using a graph coloring algorithm. Nowadays, due to the flexibility of opting various subjects, and many students taking up different courses in their colleges and universities, it becomes difficult to schedule these examinations. Creating an examination schedule can be quite challenging and time-consuming for controlling the body of an examination. Our research work focuses on reducing the efforts for scheduling such examinations. With the knowledge of graph theory and graph traversing and coloring algorithms, our algorithm with the help of a few assumptions gives an efficient solution to the examination scheduling problem. A detailed correctness proof along with performance analysis has been done. The efficiency of our proposed algorithm is then compared to that of the coloring algorithm using backtracking.

Keywords:

Examination Scheduling, Graph Coloring Algorithm, Bipartite Graphs, NP-Complete Problem, Linear Time Complexity, Meta-Graph

1. INTRODUCTION

The current flexible educational setup of our country allows students to opt for various subjects from various fields according to their choice. This leads to the cluttering of students among various subjects and is a challenge for the administrative body of the institutions to schedule the examinations in a manner such that courses having common students do not fall on the same date. Courses with no common students however can be evaluated on the same day. Many students having a variety of subjects might overwhelm the officials to arrange the exams within a particular span such that there is no clash of timings for any student. Without which some students may suffer due to lack of proper evaluation. Also, a lengthy schedule may result in increase of expenses for the education body and the students may slack off.

Our algorithm strives to solve this problem by scheduling examinations within minimum days without clash of timings. The students will be able to give their examinations for their respective chosen subjects easily and without time clashes, making it flexible and easy for them to prepare for the same. We are going to use the celebrated coloring properties of a graph. With vertices representing the courses and edges representing whether there are any common students between two courses, a graph can be constructed. Now, coloring that graph with minimum colors (based on the chromatic number of the respective graph) and applying partitioning based on the independent sets will allow us to schedule the examinations in an efficient way and hence solve the problem. This will not only make computation elegant and faster but also avoid unnecessary elongation of the examination period.

The objective of the algorithm is:

- To find out which subjects have common students, thus they cannot be scheduled together.
- To schedule examinations within minimum days.

The main purpose of developing such an algorithm is two-fold:

- We aim to make choosing subjects more flexible for students so that they can learn whatever they want, without facing the problem of exam time clashes.
- We wish to reduce the workload of the administrative body of the educational institutes to schedule the examinations without much fuss.

2. MATHEMATICAL PREREQUISITES

2.1 GRAPH DEFINITION

An undirected graph G is an ordered pair (V,E) where V is a set of vertices and E is a set of non-directed edges between vertices, such that $E[V]^2$, i.e. the elements of E are 2-element subsets of V. A graph can be represented using adjacency lists or adjacency matrix.

2.2 CONNECTIVITY

A walk is a list v_0, e_1, v_1,..., e_k, v_k of vertices and edges such that, for $1 \leq i \leq k$, the edge e_i has end vertices v_{i-1} and v_i.

A u-v path is a path in a non-empty graph $G = (V,E)$ of the form $V = x_0, x_1,..., x_k$ and $E = x_0, x_1, x_2,..., x_{k-1}, x_k$, where xjis are all distinct.

If $P = x_0, x_1,...,x_{k-1}$, is a path and $k \geq 3$, then the graph $C := P + x_{k-1}x_0$ is called a cycle. The length of a cycle is its number of edges (or vertices), the cycle of length k is called a k-cycle and denoted by C_k. As stated by West [10], the following lemma is useful in this connection:

Lemma 2.1. Every u-v walk contains a u-v path.

2.3 BIPARTITE GRAPHS

Now, let us characterize bipartite graphs using cycles. A walk is odd or even as its length is odd or even. As in Lemma 2.1, a closed walk contains a cycle C if the vertices and edges of C occur as a sub-list of W, in cyclic order but not necessarily consecutive. We can think of a closed walk or a cycle as starting at any vertex, the next lemma requires this view point.

A bipartition of G is a specification of two disjoint independent sets in G whose union is $V(G)$. The statement 'Let G be a bipartite graph with bipartition X, Y' specifies one such partition. An X, Y - bigraph is a bipartite graph with bipartition X, Y. The sets of bipartitions are partite sets. As mentioned by Konig [5], we have the following theorem:

Theorem 2.1 (Konig's Theorem). A graph is bipartite if and only if it has no odd cycle.

2.4 GRAPH COLORING

A coloring of a graph is an assignment of colors to its vertices so that no two adjacent vertices have the same color. The set of all vertices with any one color is independent and is called a color class.

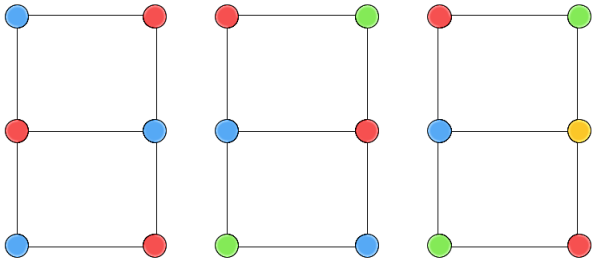

Fig.1. Three colorings of a Graph

The chromatic number $\chi(G)$ is defined as the minimum number 'n' for which G has an n-coloring. A graph G is n-colorable if $\chi(G)n$ and is n-chromatic if $\chi(G) = n$.

Since G obviously has a p-coloring and a $\chi(G)$ - coloring, it must also have an n-coloring whenever $\chi(G)<n<p$. The graph of Fig.1 is 2-chromatic; n-colorings for $n = 2, 3, 4$ are displayed, with different colors.

2.4.1 Chromatic Partitioning:

As mentioned by Liu [11], the proper coloring of a graph G induces a partitioning of its vertex set $V(G)$ into different subsets based on its colors. In Fig.1(b), it is seen that the vertices can be partitioned into three sub-sets based on the colors red, blue and green.

It may be observed that no two vertices in any of the set three subsets are adjacent. Such a subset is called an independent set.

A set of vertices in a graph is said to be independent set if and only if no two vertices in the set are adjacent.

A maximal independent set is an independent set in which no other vertex can be added without affecting its independence property. Our primary concern in this paper i.e., an optimal scheduling of examinations reduces to find the independent sets in the input graph G which belongs to the NP-hard class.

3. RELATED LITERATURE

General graph coloring algorithms have become common and have already been extensively studied by researchers. Various approaches have already been used to solve the scheduling problems.

Akbulut and Yilmaz [3] have considered a system aimed to schedule different exams in the same halls simultaneously. The main motive was to use hall 'capacity more efficiently and decrease cheating attempts considerably. In their work, they have considered the final exam weeks of the universities as their main problem to be solved. The students appearing for different exams can sit beside each other so that the capacity of the hall will increase. However, at this point, the coordination problem of the students' seating positions will arise. The solution that they came up with is to identify each student with an Radio Frequency Identification (RFID) tag. The card reader reads the scheduled exams of the particular student and displays them on a screen. They made use of two algorithms: A Hybrid Approach and their Graph Coloring Algorithm.

Malkawi and Hassan [2] aimed at solving the exam scheduling problem by using the node graph coloring technique. In their paper, they tried to achieve the objectives of fairness, accuracy, and optimal exam time period with respect to exam scheduling. The noteworthy point is that they considered some important assumptions and constraints, closely related to the general exam scheduling, and mainly driven from the real-life requirements collected through the experience at various universities. Such assumptions and constraints were distinct from those present in more general graph coloring problems.

Leighton [9] has introduced a new Recursive Largest First (RLF) coloring algorithm in his paper and compared it to various known algorithms. Various other existing coloring procedures were presented and their performance comparison with respect to the RLF algorithm was done on a wide range of test data. He came up with a procedure for generating random graphs with known chromatic numbers. This provided a standard method for testing the accuracy of graph coloring algorithms.

On the other hand, Mehta [7] in his paper expanded the objective of examination scheduling to not just deriving a conflict free minimum time period schedule. He proved that when the number of minimum time frame solutions become greater than the number of time frames in which the examinations are required to be scheduled then there will be a problem of finding a schedule with a minimum number of conflicts. This paper explains one of the faster heuristic procedures for scheduling semester examinations for a particular college.

Bharti [4] under the supervision of Kumar considered the construction of the exam schedule as a part of the time table problem which is basically are source allocation problem. Time tabling is basically a procedure to schedule a set of slots for a particular work and the table itself is a series of events arranged according to their schedule. In this thesis they attempted to compare the two cases of exam scheduling (i.e., one with consecutive exams when syllabus is less and the other being no consecutive exams to be set when syllabus is more) based on graph coloring approach.

In any educational institution, course time tabling and exam time tabling are the two most common academic scheduling problems. Ganguli and Roy [8], had collaborated in a paper in which their sole focus was on college course time tabling where both hard and soft constraints had been considered. After properly coloring the course conflict graph and transforming the coloring into conflict-free time slots of courses, they constructed the graph with courses as nodes and edges drawn between conflicting courses i.e., having common students. Since there is no fixed algorithm to solve a scheduling problem whose complexity is directly proportional to the number of constraints involved, they considered a typical honor (major) and general (minor) course combination scheduling problem under university curriculum.

4. PROPOSED METHODOLOGY

In this part we come to the proposed algorithm for scheduling the exams (in section 4.2.2). A small test case is shown in section

in order to explain the step-by-step execution of the algorithm. Later its correctness proof has been done along with the analysis of its running time in section 5. This coloring algorithm is then compared with the Naïve Graph Coloring algorithm using backtracking (as proposed in section 4.1), where it is observed that although both the algorithms strive to solve the same problem, the coloring algorithm proposed in this paper (in section 4.2.2) has an edge over the running time and efficiency of the naïve algorithm. In order to achieve fairness, as discussed in the Introduction section, a few constraints have been taken into consideration:

a) A student can be assigned at most one examination in the same time period.
b) A student can have only one examination scheduled on the same day.
c) The number of students taking up the exam is less than the number of examination halls present in the college.
d) There is a fixed number of time slots available on a particular day for examinations to be scheduled.
e) A student is not doing a double major.

4.1 GRAPH COLORING ALGORITHM USING BACKTRACKING

Input: An undirected graph $G=(V,E)$, where the vertices denote the set of courses and there is an edge between any two vertices if and only if there is any common student between the two courses. The cardinality of the vertex set V is $n = |V|$ and that of the edge set E is $m = |E|$.

Output: A sequence of scheduled timeslots.

Pseudocode:

```
C := An array containing the 'v' colors used for graph coloring
Procedure getColor (starting vertex s)
{
  Select color c from C
  colors := c
  delete c from C
  for every vertex v of the remaining n-1 vertices by backtracking
    select color k from C
    if k is already used to color the adjacent vertices of v
      color v := the next color in C
    else
      color v := k
  end loop
}
```

In the above algorithm, the for loop is executed at most n-1 times and as there are 'v' colors available in C, the total number of possible color configurations are v^n. Thus, the time complexity of this algorithm is $O(nv^n)$, i.e., exponential, which is very large.

The major drawbacks of this algorithm are:

a) It does not always use a minimum number of colors as this problem belongs to the NP-complete class of problems.
b) The number of colors used, sometimes depends on the order in which the vertices are processed.

4.2 PROPOSED GRAPH COLORING ALGORITHM USING BIPARTITE PROPERTY OF GRAPH

4.2.1 Preliminaries:

Following are few assumptions that are required for the algorithm to function correctly:

1. Each student in a respective semester is assumed to take an equal number of Core and General Elective (G.E.) and other compulsory papers. For example, as discussed in the test case in Section 7.1, for each undergraduate student in the third semester, the number of papers must be 4 (3 Core papers, 1 G.E. paper).
2. The degree of all the Core vertices is assumed to be the same for all departments.
3. A student does not have any double majors.
4. There may exist at least one course which is taken by each and every student in the college.
5. Each department has its respective G.E. courses which must be taken by at least one student from at least two departments.
6. A student can take only one G.E. paper in a particular semester.

The algorithm follows these basic steps:

i) If assumption 4 holds, i.e., there is at least one course which is taken by each and every student, then that respective vertex is assigned a color first. Then that vertex is deleted from the graph.
ii) Next, the graph G is converted into a meta-graph $M(G)$ which takes the set of vertices for the core papers of one department as one single super vertex with respect to the Department Core papers and the respective G.E. papers as its vertices. So, we map the total edges incident from a respective department on each G.E. to a single edge.
iii) The meta-graph $M(G)$ obtained is claimed to be bipartite and the two independent sets consisting of Core and G.E. can be quite easily detected through a single Breadth-First-Search method.
iv) Then, we color the vertices present in G.E. set with a single color, i.e., they can be scheduled in a single day.
v) And for the Core set, we make an observation: Let us have 'k' meta-core nodes and suppose the i^{th} meta-core node has mi vertices in it. Then, the total number of nodes to be colored is $\sum_{i=1}^{k} m_i$.

4.3 ALGORITHM

Input: An undirected graph $G = (V,E)$, where the vertices denote the set of courses and there is an edge between any two vertices if and only if there is any common student between the two courses. The cardinality of the vertex set V is $n = |V|$ and that of the edge set E is $m = |E|$. The number of core papers (d) in a semester.

Output: A sequence of scheduled time slots.

Pseudocode:

C: = An array containing the colors used for graph coloring

d: = The number of Core papers in a semester

M: = Meta Graph of G

H: = An empty array for final scheduling purpose

Procedure Scheduling (*G*)

{

If there is a vertex v with degree = *n*-1 //for a compulsory paper

Select a color *c* from *C*

color *v* := *c* //assigning color to compulsory paper

obtain graph *G*' by deleting *v* from *G*

delete color *c* from *C*

create a meta-graph *M*(*G*')

get partition of G.E. and Core vertices of M

select a color *p* from *C*

for every vertex v in set G.E. //assign the same color to all G.E. papers

color *v*: = *p*

end loop

delete color *p* from *C*

select a color subset *K* of size *d* from *C*

for every vertex u in set Core //using '*d*' number of colors to color vertices of Core

for every vertex *u*' in *u*

assign a distinct color *t* from *K* to *u*'

end loop

end loop

get the chromatic partitions *x* of the vertices and perform scheduling

map every element of *x* to *H* //mapping all vertices to H based on their color

return *H*// *H* contains the sequence of vertices ready for scheduling

end procedure Scheduling

5. CORRECTNESS

In order to prove that the algorithm runs correctly, it is required to prove a few necessary observations as stated below:

Claim 5.1. The input graph *G* is connected.

Proof. Let us assume if possible, the graph is disconnected.

Thus, $\nexists u$-v path $\forall u,v \in V(G)$

Now define two sets X_u and Y_v such that X_u: $=\{x|\exists u\text{-}x_{path}\}$, Y_v:$=\{y|\exists v\text{-}y_{path}\}$.

Clearly, *X*, *Y* are non-empty (∵ u∈*X*, v∈*Y*).

Also, $X \cap Y = \varphi$ (∵ we assumed the graph is disconnected)

But since assumption 4 in section 4.2.1 holds, i.e., there exists a course which is taken by all the students,

$\Rightarrow \exists a$ node, $n \in V(G)$ such that $(x,n) \in E(G)\ \forall\ x \in V(G)$.

$\Rightarrow X \cap Y \neq \varphi$ (∵$n \in X \cap Y$).

This is a contradiction to our assumption that the graph is disconnected. Hence, it is proved that the input graph G is connected.

Claim 5.2. In the metagraph *M*(*G*) constructed by the algorithm, the meta core vertices consist of the set of vertices that correspond to a sub-graph *H*(*G*), which is a complete graph.

Proof. It is to be shown that $(u,v) \in E(H)\ \forall u,v \in V(H)$. For this, we perform induction on the number of vertices.

Let our induction hypothesis,

P(*n*): = For a graph *H* consisting of '*n*' vertices, i.e., the number of edges is

$$|E(H)| = n(n\text{-}1)/2\ \forall\ n \in \mathrm{N} \quad (1)$$

Basis Step (*n*=2): *P*(2) is true, as when there are just two vertices in *H* and number of edges is $|E(H)| = 2(2\text{-}1)/2 = 1$ which is true (by Claim 5.1).

Inductive Step: Assume that *P*(*n*) is true, i.e., Eq.(1) holds for some non-negative integer *n*. Then adding another vertex in *H*, we have, $|E| = \frac{n(n+1)}{2} + n$ (∵adding a vertex means adding a Core course which must have students in common to all other Core courses of a respective department).

$$|E(H)| = \frac{n(n+1)}{2} + n = \frac{n(n-1)+2n}{2} = \frac{n^2+n}{2} = \frac{n(n+1)}{2}$$

This proves *P*(*n*+1) is true. So, it follows by induction that *P*(*n*) is true $\forall n \in N$.

Claim 5.3. The meta-graph *M*(*G*) constructed by the proposed algorithm in Section 4.2.2 is bipartite.

Proof. Let us define two sets *X*, *Y* such that,

X: = {u| u is a Core vertex}

Y: = {v| v is a G.E. vertex}

Clearly, $X \cap Y = \varphi$ (∵a vertex cannot be both Core and G.E.). Also, since, the meta-graph consists of only meta-Core and G.E. vertices, $X \cup Y = V(M)$.

Now clearly, $\nexists (x,y) \in E(M)\ \forall x,y \in X$ (∵as stated by assumption 3). Also, $\nexists (a,b) \in E(M)\ \forall a,b \in Y$ (∵as stated by assumption 6).

The *X*, *Y* are independent sets. So, the meta-graph *M* can be partitioned into two independent sets, which in turn proves our claim.

Now, that we have proved the meta graph *M*(*G*) is bipartite, we are just left with assigning colors to the corresponding independent sets.

Claim 5.4. The proposed algorithm in Section 4.2.2 correctly assigns colors for any input graph *G* taking all the assumptions into consideration.

Proof. From Claim5.3, it has already been proved that the meta-graph *M*(*G*) constructed by the algorithm is bipartite. Thus, from Theorem 2.2, the meta-graph is bi-colorable. Now, we'll have to prove that the proposed algorithm assigns colors to both the independent sets correctly. The proposed algorithm first colors all the elements of the G.E. set with a single color and deletes that color from the list. For, the Core set, clearly the number of Core papers in a semester will be the number of colors required. Since, the number of colors (*d*) is already provided with the input, the algorithm chooses a set of *d* colors from the Color array, and it colors each of the elements present in the Core meta-vertex using two for loops and hence all the vertices are colored correctly.

6. RUNNING TIME ANALYSIS

Claim6.1. Considering the necessary assumptions and every choice of vertices and colors for every input graph $G=(V,E)$, the scheduling algorithm runs in $\theta(n+m)$ time, where $n = |V|$ and $m = |E|$.

Proof. Before analyzing the algorithm rigorously, let us see what happened in the proposed algorithm. Primarily, we checked for vertices having the highest degree and if found, we assigned it a respective color and deleted that vertex from the corresponding graph and the assigned color is also deleted from the color set. Checking for the degree can be done in linear-time by just maintaining an extra array while creating the adjacency list for the input graph. Now, creating the meta-graph can also be done in linear time as while taking the input, we already store the Core and G.E. vertices and so creating the array of vertices can be done in linear time. Since the meta-graph will always be bipartite according to Claim 5.3, so detecting it and obtaining the independent sets can be done by a single breadth-first search loop which again can be easily implemented in linear time. If the number of Core meta-vertices (i.e. the number of departments) formed from the graph is k and the number of vertices in i^{th} meta-vertex be m_i, then the number of vertices representing the Core courses, $|V_{core}|= \sum_{i=1}^{k} m_i$, which is the number of times the iterations from line 16-21 is executed in the proposed algorithm. Next the coloring of Core and G.E. vertices is just assigning colors to the vertices which again can be implemented in linear time, because while doing so, the vertices are traversed at most once. After coloring all the vertices we get the chromatic partitions from the graph *G* and perform the scheduling. Therefore, as each vertex is visited once and each edge is traversed, both are done in O(1) time, thus the proposed algorithm runs in $\theta(n+m)$ time.

7. RESULTS AND FINDINGS

7.1 CASE STUDY

Most undergraduate colleges in India offer a variety of subject combinations to its students under the CBCS curriculum. In streams like B.A or B.Sc. students can take one subject as Honours (Core) and two subjects as General Elective (G.E.) papers. In the following subsection, we have presented a typical case of the scheduling problem and its conflict free solution timetable, maintaining equity.

Table.1. Core-G.E. Subject Combination

List of Core Papers	G.E. Subject Combination
Computer Science (CS)	M, P
Mathematics (M)	CS, P
Physics (P)	M, CS

Considering each course as a vertex, edge between two vertices is drawn only if there is common student between two courses. The following graph *G*, as seen in Fig.2 has been created.

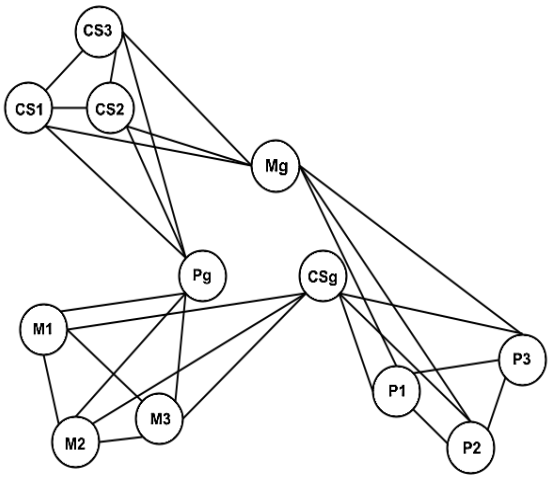

Fig.2. Graph of the case study

In Fig.3, the adjacency list representation of G is shown.

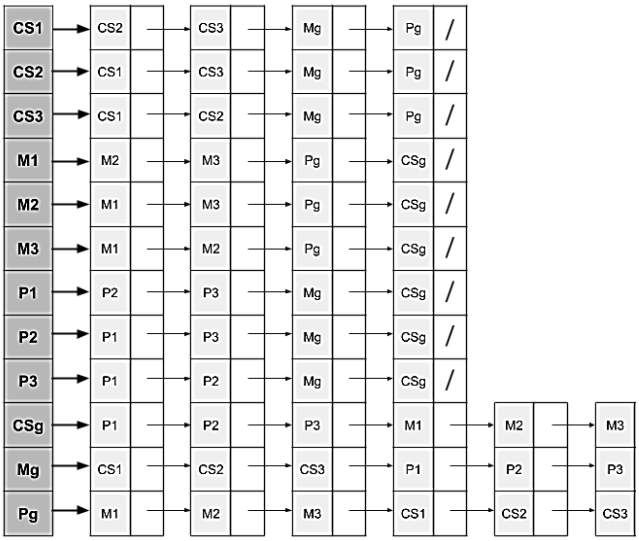

Fig.3. Adjacency List of the Graph of Fig.3

The graph *G* has been considered for semester 3 examinations, where we have the following representations of the vertices:

- CS1, CS2, CS3 - Core papers offered by the Department of Computer Science
- M1, M2, M3 - Core papers offered by the Department of Mathematics
- P1, P2, P3 - Core papers offered by the Department of Physics
- CSg - G.E. paper offered by the Department of Computer Science
- Mg - G.E. paper offered by the Department of Mathematics
- Pg - G.E. paper offered by the Department of Physics

In Fig.4, the corresponding meta-graph $M(G)$ is shown.

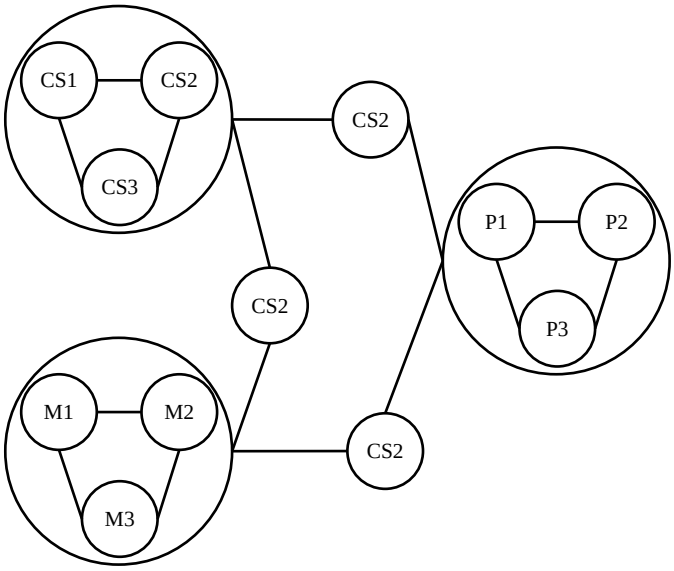

Fig.4. After Meta-Graph Construction

7.2 RESULTS

The coloring of the graph *G* in Fig.6 has been plotted in the table shown in Table.3. The chromatic partitions of *G* have been done based on the proposed algorithm where the set of colors = {pink, blue, yellow, green}. Now, each of these colors represents the day on which the exams need to be scheduled.

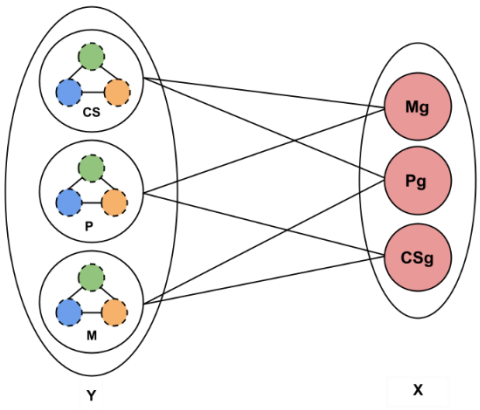

Fig.5. After bipartite construction

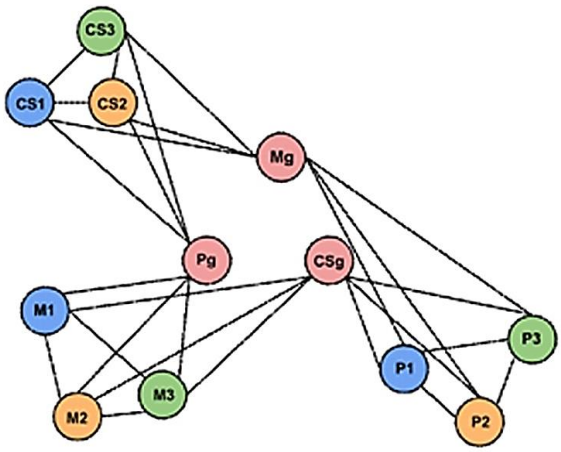

Fig.6. Final graph after applying the proposed algorithm

After applying the proposed graph coloring algorithm, the resultant graph is seen in Fig.6 is properly colored with chromatic number 4. This is the minimum number of non-conflicting timeslots required for scheduling for all given courses.

Table 2: The chromatic partitions of *G*

Pink	Blue	Yellow	Green
Mg	M1	M2	M3
Pg	P1	P2	P3
CSg	CS1	CS2	CS3

The Table.2 shows the chromatic partitions for the given graph, *G* and the final complete schedule is obtained and shown in Table.3.

Table.3. Examination Schedule

Day 1	Day 2	Day 3	Day 4
Mg	M1	M2	M3
Pg	P1	P2	P3
CSg	CS1	CS2	CS3

8. CONCLUSION AND FUTURE SCOPE

The proposed research work tried to give the solution for the graph coloring problem by adopting a completely new approach with the help of traditional data structures available. Although the coloring of a graph is an NP-complete problem, using the greedy algorithm does not always give the correct output, i.e., the minimum number of required colors. Blind search and brute force algorithm using backtracking is used as a hybrid approach for the starting point. But the proposed algorithm in this paper is specifically designed keeping in mind the symmetry of the CBCS curriculum followed in India. Considering a few assumptions and constraints, the pro-posed algorithm gives a blazing fast solution running in linear time. These two algorithms have been applied to the same data sets and different results are interpreted. The results have been examined depending on the process speed and efficiency. We have tried to address the graph coloring problem and proposed a solution that will work efficiently in the case of any kind of graph either a dense graph or a sparse graph as long as the symmetry and assumptions in the examination structure is maintained. Using the basic idea of graph coloring, at first, we applied the backtracking algorithm to find an initial solution to this problem restricted to a particular case involving certain constraints. Then we move on to use the bipartite property of graphs to implement our proposed methodology into an algorithm. The correctness and the running-time of this algorithm have been analyzed along with a case study to explain its working. Later the proposed algorithm is found to be more efficient than the algorithm using backtracking that has been used initially. The model can be useful for scheduling examinations in colleges as well, which can decrease their employees' workload significantly. Universities can schedule their mid-semester as well as semester examinations easily in a short span of time using the same. The students will be able to give their examinations for their respective chosen subjects easily and without clash timings, making it feasible for them.

With the vast growth and spread of knowledge leading to introduction of newer subjects now and then it seems that scheduling examinations will be more complex, and this research work has a wide scope ahead. We are currently investigating a modification of the proposed algorithm, which will achieve the minimum for a certain graph. Loosely speaking, later more generalized algorithms may be developed, that can schedule examinations for every college and university under the CBCS curriculum in India, using Artificial Intelligence and Machine Learning methodologies.

REFERENCES

[1] F. Harary, "*Graph Theory*", Addison-Wesley Publishing Company, 2001.

[2] M. Malkawi, M.A.H. Hassan and O.A.H. Hassan, "A New Exam Scheduling Algorithm using Graph Coloring", *International Arab Journal of Information Technology*, Vol. 5, No. 1, pp. 1-14, 2008.

[3] A. Akbulut and G. Yilmaz, "University Exam Scheduling System using Graph Coloring Algorithm and RFID Technology", *International Journal of Innovation, Management and Technology*, Vol. 4, pp. 66-78, 2013.

[4] M. Bharti and R. Kumar, "Better Resource Utilization in Exam Scheduling using Graph Coloring", Ph.D. Dissertations, Department of Computer Science and Engineering, Thapar University, pp. 1-57, 2012.

[5] D. Konig, "*The Infinite and Infinite Graphs*", Reprinted Chelsea, 1950.

[6] J.D. Ullman, "NP-Complete Scheduling Problems", *Journal of Computer and System Sciences*, Vol. 10, pp. 384-393, 1975.

[7] N.K. Mehta, "The Application of A Graph Coloring Method to An Examination Scheduling Problem", *Interfaces*, Vol. 11, pp. 57-65, 1981.

[8] R. Ganguli and S. Roy, "A Study on Course Timetable Scheduling using Graph Coloring Approach", *International Journal of Computational and Applied Mathematics*, Vol. 12, pp. 469-485, 2017.

[9] F.T. Ceighton, "A Graph Coloring Algorithm for Large Scheduling Problems", *Journal of Research of The National Bureau of Standards*, Vol. 84, pp. 489-506. 1979.

[10] D. West, "*Introduction to Graph Theory*", Prentice Hall, 2001.

[11] C.L. Liu, "*Elements of Discrete Mathematics: A Computer Oriented Approach*", Tata McGraw-Hill, 2008.

[12] J. Kleinberg, "*Algorithm Design*", Pearson India Education Services Pvt Ltd, 2014.

[13] K. Rosen, "Discrete Mathematics and Its Applications with Combinatorics and Graph Theory", McGraw-Hill, 2012.

[14] R. Diestel, "*Graph Theory*", Springer, 2017.

[15] J.A. Bondy, "*Graph Theory with Applications*", Elsevier, 1976.