Better Resource Utilization in Exam Scheduling Using Graph Coloring

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Meena Bharti
(Roll No. 801031018)

Under the supervision of:
Mr. Ravinder Kumar

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2012

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Better Resource Utilization of Exam Scheduling Using Graph Coloring"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Ravinder Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Meena Bharti

Meena Bharti

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Mr. Ravinder Kumar
Assistant Professor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgment

I express my sincere and deep gratitude to my guide Mr. Ravinder Kumar, Assistant Professor in Computer Science & Engineering Department, Thapar University Patiala, for the invaluable guidance, support and encouragement. He provided me all resource and guidance throughout thesis work.

I am heartfelt thankful to Dr. Maninder Singh, Head of Computer Science & Engineering department Thapar University Patiala, for providing us adequate environment, facility for carrying out thesis work.

I would like to thank to all staff members who were always there at the need of hour and provided with all the help and facilities, which I required for the completion of my thesis. At last but not the least I would like to thank God and mine parents for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Meena Bharti
801031018

Abstract

Time table is required in every school, college, university and various departments. Time table is basically a scheduling to do work or can say is a set of slots to do particular work. Exam scheduling is part of time table problem with bit differences in both like in case of time table problem one lecture can be needed to schedule 3 or more times a week while in exam scheduling one exam is needed to schedule only once.

In case of exam scheduling there are two cases,

- Student can have exam of two or more subjects consecutively. This type of case is mostly in case of internal exams where syllabus is less.
- Student can't have exam of two or more subjects consecutively. This type of case is mostly in case of final or external exams where syllabus is more.

In this thesis it is attempted to do comparison between two ways to schedule exam based on graph coloring approach for both cases mentioned above. The algorithm presented in this paper is divided in four parts:

- Graph coloring which is taken from previous build algorithm.
- Removal of subject is done if number of students is more than the number of seats available.
- Adjustment of subject is done is after removing subjects number of seats remain vacate then subject with less number of students can be adjusted
- Next graph is made after removing subjects which are scheduled already.

Also an example is explained to show the comparisons of different categories and explain the algorithms.

Table of Contents

List of Figures

List of Tables

Chapter1
Introduction

1.1. Overview

This thesis is concerned for construction of exam schedule, which is part of time table problem. Time table construction problems are interesting to study because neither modeling nor solving them is straightforward. It is difficult to make clear cut distinction between feasible time table and non feasible time table because of large set of constraints. Each constraint has its own characteristic aspects that add complexity to time table problem.

Timetabling means all activities which are required to make a feasible time table. According to Collins Concise Dictionary (4^{th} Edition), a time table is a table of events, arranged according to the time, when they take place. The events usually are, meeting between people at particular location and at what time. A time table must meet a number of requirements and should satisfy the desire of all people involved simultaneously. The timing of events must be in such a way that everyone has exactly single event or null event at a single time. A dependency of event on various objects in a general time table is shown in Figure 1.

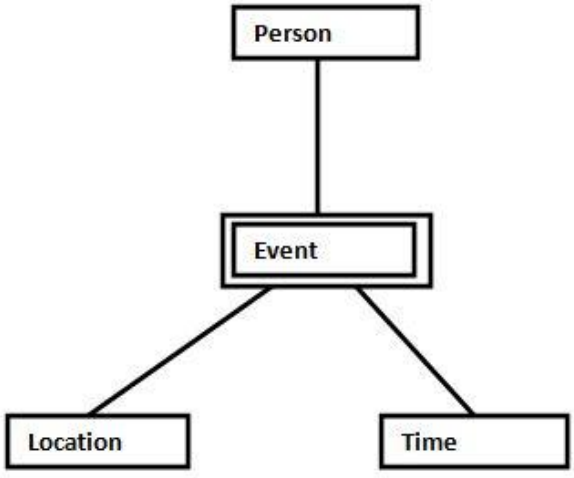

Figure 1.1: A general time table [29]

In a time table, there are various factors which affect the occurrence of an event. Location of event, at what time the event will occur and which persons will be involved in the

event are basic factors of it. Time table should be set in such a way that at same location and at same time one and only one event can occur and persons participating in that event should have a single event at a time.

1.2. Educational time table

Educational time table is required in every school, college, universities and many of the other places. Time table problem is a resource allocation problem. It consists of various set of subjects which are required to allocate according to the availability of students, teacher or instructor and classrooms. Time table is a scheduling to do work or can say is a set of slots to do some particular work. It is a combinatorial problem consisting of a huge set of constraints to satisfy. Time table problem has three sub parts [29]:

- Exam scheduling- consisting of scheduling of subjects around the rooms.
- Time table of universities- consists of weekly time table of the university.
- Time table of schools- consists of weekly time table of schools.

There is a difference between these three types of time table problem.

1.2.1. Exam Scheduling

In case of exam scheduling, the exams are scheduled around the slots. In this case, one subject is required to schedule only once. There is no restriction about teachers i.e. any teacher can be assigned for any subject. However if a teacher who is teaching that subject to the class is assigned to that class in which that subject is scheduled, then it will be beneficial because sometimes there can be some printing mistakes or some other corrections in the question paper, that will become easy for the teacher to do such corrections. The theoretical exam of students can be scheduled in any room according to availability of room but sometimes there are practical exams which need special rooms. Like practical of physics, chemistry and biology can be done in respective labs only because of availability of materials required for practical. In case of exam scheduling, there can be a case that whether a student can have exam of two subjects consecutively or not. It means if a student have consecutive exam of two subjects, then a student will not get much time for preparations. This is the case when syllabus for exam is less like in

case of internal exams or can say sessionals. But, in case, where student can't have two exams consecutively, is a case when a student has to take the exam from more syllabuses like external exams.

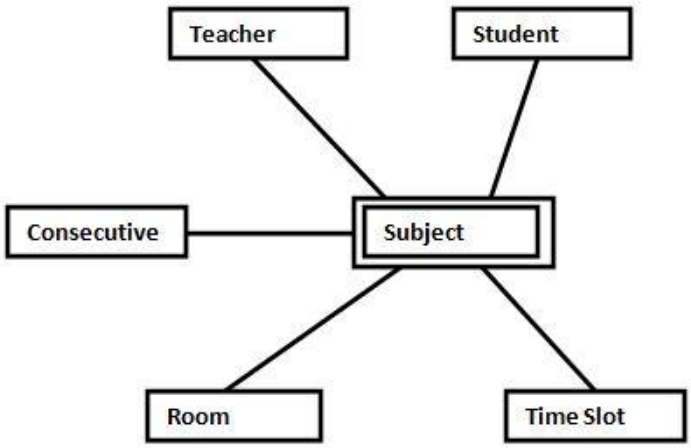

Figure 1.2: Exam scheduling [29]

In case of exam scheduling, subject is an event i.e. exam of which subject should be scheduled at what time and in which room. A student can have one exam at a time and number of seats depends on the number availability of rooms and teacher. Consecutive constraint means whether a student can have consecutive exam of two subjects or not.

1.2.2. Time table

Time table problem is basically a resource allocation problem. It consists of various sets of subjects, which are required to allocate according to availability of students, teacher or instructor and classrooms. In case of time table problem, lecture of one subject might be required to schedule one or more times a week i.e. time table construction is done for a week. Moreover, some subjects require special rooms as like the subjects of physics and chemistry. For these subjects, their respective laboratories are required. A particular teacher or can say a subject teacher can teach that subject. This is just like only a Physics teacher can teach Physics and only a Mathematics teacher can teach Mathematics. There is difference between both time table of university and schools.

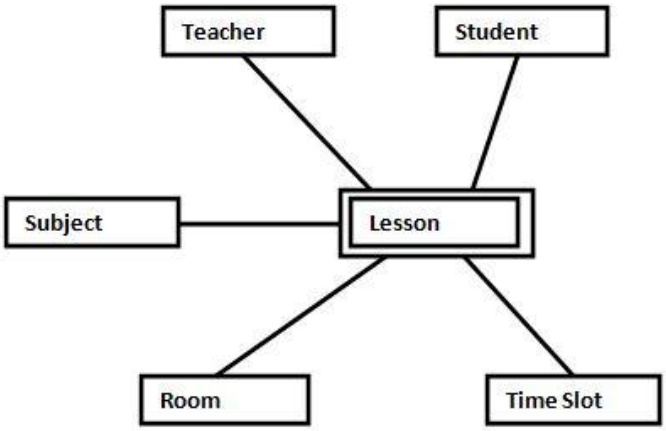

Figure 1.3: Educational time table [29]

In educational time table, lesson is event. It depends on that which lesson of which subject has to be taught at what time and in which room, which teacher will teach that subject and to which students. While making time table it should be taken care that schedule of teachers should be made according to availability of teachers and a teacher can teach only one subject at a time. Moreover students can study only one subject at a time. In one room there should be lecture of one subject only. And room should choose according to size of class.

1.2.2.1. Time table of university

In case of university and colleges, some subjects are elective which means that whole class doesn't read the same subject. Also, different teachers are required for different subsets (subsets of class are created because of different elective subjects chosen by different students). Moreover, due to elective subjects, sometimes it happens that for popular subjects, number of students often exceeds the capacity of classroom; planners frequently need to divide students into smaller groups. For some subjects, that class is like a group but for elective subjects there are sub groups of same class. In case of university, number of students in a class varies from 2-3 to hundred and above.

1.2.2.2. Time table of school

In case of schools there are not any elective subjects. All classes are of same size almost. Sometimes, when number of students in a class increases from some limit then the sections are divided with equal number of students these sections also study same subjects but teacher teaching that subject can be different.

Exam scheduling requires less number of constraints than schedule of class time table like in class time table, classes more than one in a week while in exam time table one course exam has to schedule once. In class time table there should be proper teacher to teach a particular subject while in exam time table any teacher can give duty in any room.

1.3. Different techniques to solve time table problem

Different algorithms to solve time table problem are:

- Heuristics algorithm
- Evolution algorithm
    - Genetic algorithm
    - Evolution strategies

1.3.1. Heuristics algorithm

'Heuristics' is a term used for algorithms, which work on previous experience and find possible solution but not the best one. They may be considered as approximately but not the accurate algorithms. These algorithms find the close solution to best one. Graph coloring is an assignment of labels traditionally called 'colors' to elements of a graph subject to certain constraints [30]. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges share the same color.

In time table problem nodes of graph represent the subjects or courses and edges represent the conflicting subjects that can't be scheduled concurrently, means those

subjects which have at least one student in common. Color means scheduled time. Like in case of exams different color means different time slots or different day of exam.

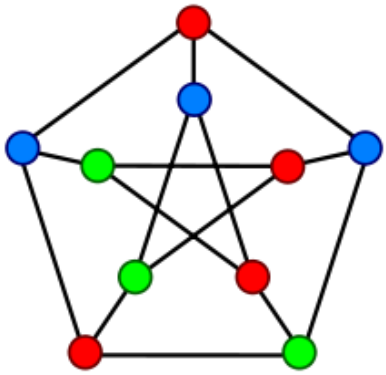

Figure 1.4: Graph with colored vertices [30]

1.3.2. Evolution algorithm

Two classes of algorithms: Evolution Strategy (ES) and Genetic Algorithms (GAs) are grouped under the name of "Evolutive Algorithms". Both classes of methods are derived from an Idea of Darwinian evolutive adaptation: an instance of the data set to be optimized is seen as an individual, who must fight against similar individuals to survive and reproduce. Best fitted individuals (i.e.: individuals that are well-adapted to the environment) reproduce and transmit to their wards some of their characteristics. The fitness of an individual is defined as a function of the data set, represented by the individual. Finding the fittest individual is equivalent to maximizing the fitness function.

1.3.2.1 Genetic algorithm

A population of individuals (timetables) is generated randomly.

- In every algorithm's iteration, best fitted individuals reproduce themselves, while worst ones die and are eliminated from the population. Survived individuals mate, exchange genetic material (part of their data) and eventually undergo a random mutation.

Internal data representation must be as a flat string of characters: a timetable is defined as a sequence of integers, where every integer represents a teacher or a free hour (hour where there aren't any lessons: some courses may require less hours than some others). The sequence of numbers represents the succeeding of teachers in classes: a first

subsequence of integers represents all lessons of first class, and then there is a second substring for all lessons of second class, and so on. We decided to use symbols from a non-binary alphabet (the alphabet used has a symbol for every teacher, plus one for free hours) to avoid lots of illegal individuals generated (if we define a timetable as a string of bits, and a lesson represented by a substring, and the number of teachers wasn't a power of two minus one, simple crossover would generate lots of individual where a given substring of bits doesn't represent any of teachers).

In mating phase, reproduction is handled with the so-called simple crossover: parents are selected with probability directly proportional to their fitness, a cut point is selected at the interior of a string, randomly and with uniform distribution; first child inherits from first parent all symbols before the cut point and from second parent all symbols after the cut point; other child is generated symmetrically.

1.3.2.2. Evolution strategies

The algorithm named Evolution Strategy is a simplification of a genetic algorithm: a population of only two members is taken, and it evolves exclusively by means of selection (at every iteration, only the best individual survives and reproduces) and mutation (the child, that will compete with the parent for reproduction at next round, is different from the parent only for a random mutation).

A mutation consisted in changing one of vector components by a random amount chosen with normal distribution with zero mean; the variance of this distribution were dynamically adjusted for a faster exploration of the entire search space (when a lot of children were better than parents, the variance was increased to explore new zones; otherwise it was reduced, suspecting that a maximum were not too distant and then refining search). Recently the method was adapted to work also on non-continuous problems.

This method was adapted to the permutation problem by defining an individual as a class/hour matrix of lessons and mutation as exchanging two lessons at the interior of a class or as inversion of the timetable for a class. Data structure is identical to the structure used by the Complex Genetic Algorithm and the same fitness simplifications are used.

Chapter 2

Literature Survey

2.1. Constraints in time table

Every time table is not feasible, to make time table feasible there are some constraints which are needed to follow. The constraints in time table problem can be essential and preferential. Both of them are explained following.

2.1.1. Essential timetabling [27]

Essential time tabling conditions (also called as hard constraints) are conditions or constraints that must be satisfied so as to produce a feasible timetable. These constraints are almost same everywhere. Examples of essential timetabling conditions include the following:

- Instructor can't be present at two places i.e. instructor's two lecturers can't be scheduled at same time [21].
- Similarly two or more subjects can't be scheduled in same classroom at same time [28].
- Subjects should be scheduled according to availability of instructor [11].
- Two or more subjects of a student can't be scheduled at same time like if a student have physics and mathematics both subjects the lecture of these subjects can't be scheduled at same time [25].
- Some labs or subjects can held in particular rooms like physics lab or chemistry lab.
- Some subjects have fixed number of times a week like 3 periods a week or 4 periods a week [25].
- Each subject must be scheduled in such a room that can accommodate its size [15].
- Same subject having 2 or more lectures in a week should not have 2 or more lectures on same day. This constraint signifies that students can do their work of previous lecture before having next lecture.

2.1.2. Preferential timetabling [27]

Preferential timetabling conditions (also called as soft constraints) are additional conditions or constraints that are needed to fulfill to make better or a comfortable time table for students or teacher but need not necessarily be satisfied to produce a feasible timetable. These constraints can be used in correspondence to their priority. More the preferential conditions will be satisfied, better will be time table. Examples of preferential timetabling conditions include the following:

- Instructor should have preferences of time i.e. he can decide at what timings his lectures should be scheduled and at what timings he should be free [28].
- In colleges or universities time slots are more than the scheduled lectures so free lectures should be either in morning or in evening so that student can either come late for college or go early from college [11].
- Similarly for instructor free periods should be either in morning or evening [25].
- Lectures of an instructor should be near his room or department [28].
- Lectures should be in morning session and labs should be in evening session.
- Classrooms should be just large enough to accommodate students of a subject. Means subject taken by small number of students should be held in small room to avoid presence of unused empty space [15].

2.2. Method to color the graph [10]

As stated earlier, in this thesis exam scheduling problem is solved by graph coloring method. To color the graph first start with subject with highest degree then find triples such x is adjacent to y and y is adjacent to z but x is not adjacent to z. Then combine x and z into one node.

Suppose there is a graph G with vertices x, y. Now x and y can be merged into a single vertex if there exist say z between x and y which is adjacent to both x and y but x and y are not adjacent to each other. x and y can be merged to produce x' or y'.

For example for any given vertex x, triples of vertex are as follow [2]:

$(x, z_{(1,1)}, y_1)$

$(x, z_{(1,2)}, y_1)$

$(x, z_{(1,m1)}, y_1)$

..

$(x, z_{(i,1)}, y_i)$

$(x, z_{(i,m1)}, y_i)$

..

..

$(x, z_{(n,mn)}, y_n)$

Where the first vertex is adjacent to second and second vertex is adjacent to the third but first vertex is not adjacent to third. This means that the third vertex is adjacent to an adjacent to the first. Then first and third vertexes are combined together to a new vertex and which contains all the edges from first and third vertexes.

This algorithm is based on Dutton and Brigham's [14] algorithm and another algorithm by Tehrani [26].

Algorithm 1: Color the Graph

Input: A Graph G which is needed to color.

Output: List of independent subjects.

1. $Let\ j \leftarrow 1, H \leftarrow G.$
2. $Let\ v_j\ be\ vertex\ of\ maximal\ degree\ in\ H.$
3. $Construct\ all\ the\ triples\ as\ above\ and\ find\ \mathbb{y}.\ If\ no\ such\ vertex\ exists\ (i.e.\ m_i = 0) choose\ a\ vertex\ of\ maximal\ degree\ non-adjacent\ to\ v_j.$
4. $Merge\ v_j\ and\ \mathbb{x}\ into\ v_j\ and\ repeat\ from\ 3\ until\ no\ choice\ is\ possible\ then\ start\ again\ from\ 2\ with\ H \leftarrow H - \{v_j\}, j \leftarrow j + 1.$
5. $When\ there\ are\ no\ vertices\ left, stop\ and\ color\ all\ vertices\ merged\ into\ v_i (1 \leq i \leq j) color\ i, this\ will\ be\ a\ j-coloring\ of\ graph\ G.$

In the above algorithm, triples are found as shown above and then those vertices are colored with one color and hence getting the independent list of subjects.

Example

Suppose we have twelve exams which need to be timetabled and that they have the following conflict graph:

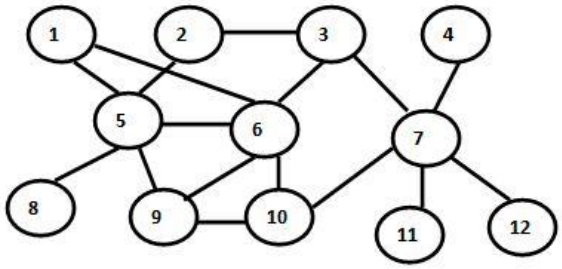

Figure 2.1: Conflict graph [10]

To color this graph starts with node with highest degree. Vertices 5 and 6 have same degree so any vertex from 5 and 6 can be chosen arbitrarily. Let's start with vertex 5.

Now make triples such that any vertex y is not adjacent to 5 but adjacent to adjacent 5. 3 and 10 are such vertexes. So these 3 vertices will be merged to make a new vertex say V1. First merging node 3 and 10 we will get

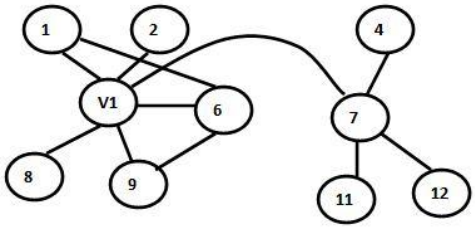

Figure 2.2: Graph after iteration 1 of merging

Now, again in this graph, nodes have to find which are adjacent to adjacent to V1. Nodes 4, 11, 12 are such nodes which are adjacent to adjacent to V1 and are not adjacent to each other. So merging V1 with 4, 11 and 12 the graph got will be

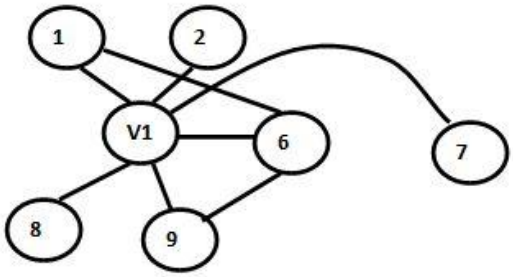

Figure 2.3: Graph after iteration 2 of merging

Now, no such vertex exists which is adjacent to adjacent to V1. So these vertices will be colored with one color i.e. vertex 3, 4, 5, 10, 11, 12 will got one color and next graph will be colored in same way choosing different color starting from node 6. Here to show coloring of graph different shapes are taken like circle, triangle and square. So colored graph for this example is shown in Figure 8.

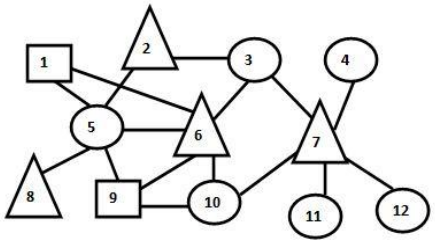

Figure 2.4: Graph after coloring subjects

2.3. Finding rooms for each exam [10]

Consider exams and rooms be listed according to size of students and number of seats in a room respectively in increasing order. Exams are denoted by e_1, e_2, ……. e_n and rooms are denoted by R_1, R_2, ….. R_m. It is assumed that largest exam can fit in largest room.

Algorithm 2: Find room for each exam

Input: List of exams in increasing order by e_1, e_2, ……. e_n and list of rooms in increasing order R_1, R_2, ….. R_m.

Output: Rooms are outputted that which exam will held in which room for that session

1. $Let\ i \leftarrow 1, j \leftarrow 1, D_1 \leftarrow \emptyset.$
2. $If\ |e_i| \leq |R_j| then\ place\ e_i\ in\ R_j$

   $Otherwise\ j \leftarrow j + 1, repeat\ step\ 2.$
3. $Let\ k \leftarrow j.$
4. $If \sum_{e_i\ in\ R_k} |e_i| \geq |R_k|\ then\ find\ the\ set\ of\ exams\ in\ room\ R_k\ with\ maximal$

   $combined\ size\ such\ that\ they\ fit\ in\ R_k.$

   $If\ k = m\ then\ D_{i+1} \leftarrow D_i \bigcup \{remaining\ exams\}\ otherwise\ put\ them\ in$

   $R_{k+1}, let\ k \leftarrow k + 1, D_{i+1} \leftarrow D_i\ and\ repeat\ step\ 4.$

5. $Let\ i \leftarrow i + 1.$

   $If\ i \leq n, repeat\ from\ step\ 2.$

In the above algorithm, the subject with minimum number of students is adjusted in largest room. It is because by doing this large space remain vacant in which another subject can hold.

This algorithm starts with smallest exam, adjusting it in smallest room and continuing through list, algorithm puts each exam in smallest room it will fit. If there are more students scheduled to be in the room than the room can held, the algorithm displaces the minimum number of students to the next bigger room so that the combined size of the exams remaining is less than the size of the room, repeating with the next bigger room and so on.

Example

Consider example above, exams 3, 4, 5, 10, 11 and 12 got same color means these exams can be scheduled at same time. Suppose numbers of students in these exams are 71, 37, 42, 73, 28 and 96 respectively. Also, consider that in some institution, there are four rooms with sizes 40, 60, 80 and 100. Denoting an exam E of size S by E^S, the ordering of exams in increasing order is:

$$11^{28} \quad 4^{37} \quad 5^{42} \quad 3^{71} \quad 10^{73} \quad 12^{96}$$

Exam11 is adjusted in room 40 and the placing of rooms continues (from left to right) as follows:

Table 2.1: Placement of subjects in different rooms [10]

Room size	Positioning of exams at each stage					
DUD						12^{96}
100				11^{28}	$11^{28}, 3^{71}$	$11^{28}, 3^{71}$
80			11^{28}	3^{71}	10^{73}	10^{73}
60		11^{28}	5^{42}	5^{42}	5^{42}	5^{42}
40	11^{28}	4^{37}	4^{37}	4^{37}	4^{37}	4^{37}

As shown in table, exam 11 is pushed up each time a new exam is scheduled until it is not in room 100 where it is small enough to share a room with exam 3 which has just been knocked up by exam 10. Exam 12 goes straight into the DUD list as it is smaller than the combination of exams 11 and 3. Thus the exams are fitted into their respective rooms.

2.4. Creating the exam scheduling [10]

Two algorithms are explained earlier. First is to color the graph and second is to find room for each exam. Now, here is algorithm which combines these two algorithms to provide a proper method to schedule the exams

---

Algorithm 3: Create schedule of exam

Input: Set of rooms R, a set of exams and their associated conflict Graph G,

Output: Exam scheduling

1. $Let\ p \leftarrow 1.$
2. $Use\ algorithm\ 1\ on\ G\ to\ produce\ set\ I\ of\ non-conflicting\ exams.$
3. $With\ I\ as\ input, use\ algorithm\ 2\ to\ place\ the\ exams\ in\ I\ in\ respective\ rooms$ $set\ R, leaving\ a\ list\ of\ unscheduled\ exams\ DUD. This\ assignment\ is\ taken\ to$ $the\ time\ table\ for\ period\ p.$
4. $Let\ p \leftarrow p + 1.$
5. $Delete\ all\ vertices\ (and\ adjacent\ edges) in\ I\ but\ not\ in\ DUD\ from\ G\ to\ prod$ $If\ G'is\ the\ null\ graph\ then\ timetable\ is\ complete$
6. $Starting\ with\ exams\ in\ DUD(merged\ together\ in\ the\ graph\ to\ form\ a\ single$ $vertex)\ or\ most\ conflicting\ exam\ if\ DUD\ is\ empty, use\ algorithm\ 1$ $to\ produce\ new\ set\ I\ from\ G'.$
7. $Let\ G \leftarrow G'. Repeat\ from\ step\ 3.$

Let's continue with example shown above. Exam 12 was left unscheduled i.e. exam 12 was left in DUD. The exams scheduled in session 1 are 3, 4, 5, 10 and 11. So after removing these subjects from graph G the left graph G' is shown in Figure

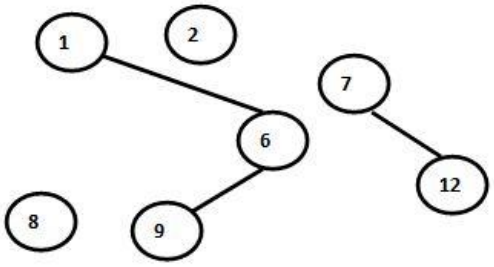

Figure 2.5: Subjects left unscheduled after iteration 1 [10]

Now, starting next iteration for graph shown in figure above starting from exam left in previous session i.e. exam 12 and repeating until all exams are not scheduled. The final result of scheduling subjects is shown in table below.

Table 2.2: Exam schedule [10]

Room	**Sessions**		
	1	**2**	**3**
100	11^{28},3^{71}	12^{96}	9^{50}
80	10^{73}	8^{75}	2^{69}
60	5^{42}		7^{54}
40	4^{37}	6^{37}	1^{36}

2.5. Exam scheduling problem using Graph coloring algorithm

An undirected graph G with {V, E} where V represent vertices and E represent nodes. The graph coloring is a well known problem [1, 3, 4, 5, and 8]. Adjacent nodes are nodes in which there is an edge. Node coloring assigns colors to nodes such that no two adjacent nodes have same color. Node coloring graph can be used to solve many problems such as exam scheduling problem. General graph coloring algorithms are well known and have been extensively studied by researchers [1, 3, 6, 8, 9, 10, 12, 16, 17, 18, and 19].

Exam scheduling is a challenging task in many colleges and universities. Because we have to schedule many exams in a minimum possible time. Exam scheduling should avoid conflicts like no student can have two exams concurrently.

List of constraints that are followed while doing scheduling for time table:

- There should be no conflict of examination hall i.e. there should not be schedule which contains two or more concurrent exams at same time [20].
- A student can't have two or more exams at same time as a single student can't sit in two exams concurrently [20].
- User can set number of time slots for a day. That how many exam slots should be there in one day. Like if we have an exam of three hours and we have to take exams from time 9am to 5pm then there will 2 time slots per day [20].
- Number of concurrency of exam sessions depends on number of halls available and number of teachers available to take exam. It is pre determined by registrar office [20].
- Concurrency of course means number of classes having same subject like a subject C1 is taught in 5 classes then exam of this subject will be on same time and we need to 5 concurrent session for the exam[20].
- Concurrency of course can also depend on number of students in a class and number of students that can sit in an examination hall. Like if a class is of 30 students and number of students that can sit in exam hall are 15 then we need two concurrent sessions of exam.
- Student can't have more then (y) number of exams per day. Like some time it is necessary to have only on exam in a day or some time (like during msts or monthly tests) number of exams are 2 per day [20].
- Student could not have gap of more than (x) days between two times. Sometimes if required to have such
- A constraint the student can't have gap of more than say 1 or 2 days between two exams [20].
- Schedule should be completed in minimum possible time i.e. we have to do maximum use of resources so that we can complete exams in minimum possible number of days [20].

A list of students and list of courses have opted then draw graph and then find the weight matrix.

- e_{ij} is the edge between two nodes. Edge will exit between two nodes only if there is at least one student common in both courses.
- w_{ij} is the weight of edge i.e. total no of students present in both courses.

Some terms used in algorithm:

- C is the list of courses offered. C_i refers to specific course.
- Degree of course is number of edges associated with the course. Degree represents that large number of conflicts with this course or we can say this course is taken by large number of students and such type of course should be scheduled first.
- R_{ab} is the color. Where day index of exam and b is time slot. i.e. color is nothing but the exam session [20].
- CL is concurrency limit i.e. no of exams can held concurrently or no of exams required concurrently for one subject like CL (R_{ab}) means no of exam halls available on day a and time slot b. and CL (C_i) means no of halls required concurrently for course C_i like one course is taught in two classes then we need to have exam of that course in two halls concurrently.
- M is array of subjects which are adjacent to course Ci i.e. those subjects in which there is at least one student in common.
- Internal distance (D1): This is the distance between two colors (R_{ij}) and R_{ik}) with the same index(I) and indexes J and K , and defined by

  D1 = |K-J| [20]

  D1 represents the exam scattering on the same day I for the same set of students.
- External distance (D2) : This is the distance between two colors (R_{ij}) and (R_{kl}) , and defined by

  D2 = |K-I| [20]

  D2 represents the exam scattering a cross different days

2.6. Algorithm [20] for exam scheduling

A. Build Weight Matrix and Graph

1. Locate the files for students' listings in all the courses, which need to be scheduled for the exam. Each course corresponds to a node in the matrix. Set the concurrency level of each node to the number of sections for the given course.

2. For each node (course) find the set of adjacent nodes, and the weight of the edges connecting the node to its adjacent nodes. Fill the weight matrix W with weight values w.

3. Create an undirected graph using the weight matrix.

4. Find the degree for each node.

B. Color the Graph

Sort the nodes in the weight matrix in a descending order based on the degree of nodes. Nodes with similar degrees are ordered based on the largest weight w in its adjacency list. Nodes with similar degrees d and weights w are ordered based on their node ID (smallest ID first).

Set C = The sorted list of nodes mentioned in Step 1.

Set No-Of-Colored-Courses = 0

Repeat for i from 1 to C.length until all nodes are not colored

If i = 1 then

R_{ab} = get-First-Node-Color (ci)

If R_{ab} = null then Exit and finish.

Otherwise,

R_{ab} = get-Smallest-Available-Color (ci)

If R_{ab} != null then

Set Color (ci) = R_{ab}

No-Of-Colored-Courses = No-Of-Colored-Courses + 1

CL (R_{ab}) = CL (R_{ab}) - CL (ci)

Endif

Endif

Set Array M = get-Ordered-Adjacency- Courses-Of-ci ()

repeat For j from 1 to No-Of-Courses-In-Array-M

If Mj is not colored then

R_{cd} = get-Smallest-Available-Color (Mj)

If Color (cd) != null then

Set Color (Mj) = R_{cd}

No-Of-Colored-Courses = No-Of-Colored-Courses + 1

CL (R_{cd}) = CL (R_{cd}) - CL (Mj)

Endif

Endif

Endloop

Endloop

C. Color the neighbor

1. Description of Sub-routine "get-First-Node-Color":

Input : The course ci that needs to be Colored.

Output: The color assigned to ci or null.

Algorithm:

For j = 1 to Max-Schedule-Days do:

For k = 1 to No-Of-Time-Slots do:

If CL ($Color_{jk}$)) ≥ CL (ci) then

return Colorjk

return null

Description of Sub-routine "get-Smallest-Available-Color":

Input: The course ci that needs to be colored.

Output: The color assigned to ci or null.

Algorithm:

get AL(ci), the Adjacency-List of ci

Repeat for j from 1 to Max-Schedule-Days

Repeat for k from 1 to No-Of-Time-Slots

Let valid = true

Repeat for r from 1 to Length (AL (ci))

R_{ef} = Color (AL_r)

If Ref! = null then

If e! =j or f! =k then

If D2 {(R_{ef}), (R_{jk})} = 0 then

If D1 {(R_{ef}), (R_{jk})} <= 1 then

Let Valid = false

Exit loop

EndIf

EndIf

If CL (R_{jk}) <= CL (ci) then

Valid = false

EndIf

Exit loop

End

2.7. Example to explain algorithm

Make time table of 3 classes such that

- Class 1 contains 29 students and have subjects:- UML, S/w Architecture, S/w Engg , Research Methodologies, SPM.
- Class 2 contains 30 students and have subjects:- UML, S/w Engg , Adv Data Structure , Adv DBMS, Research Methodologies.
- Class 3 contains 2 optional subjects: NSD (15 students) or Client Server(17 students), N/w Security(23 students) or CBD(9 students).
    - NSD and N/w Security contains 10 students in common.
    - NSD and CBD contain 5 students in common.
    - N/w Security and Client Server contains 13 students in common.
    - CBD and Client Server contain 4 students in common.
- 5 concurrent exam sessions can be taken.
- One hall can contain 17 students.
- Two slots per day.
- A student can have 2 exams per day.

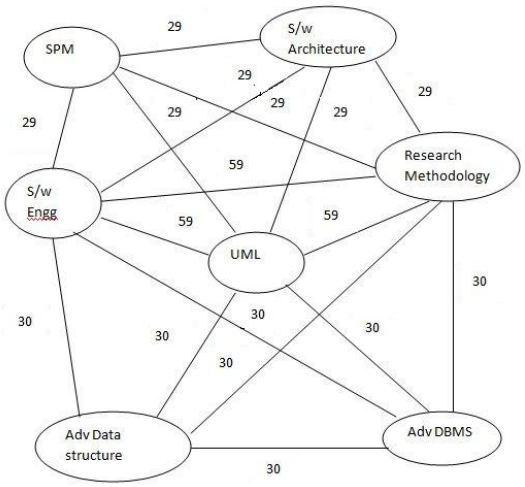

Figure 2.5(a): Dependency graph 1

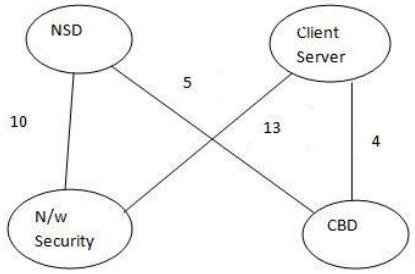

Figure 2.5(b): Dependency graph 2

Table 2.3: Weight matrix

	SPM	UML	S/w Arch	S/w Engg	R.M.	Adv DS	Adv DBMS	NSD	Client Server	N/w Security	CBD
SPM	0	29	29	29	29	0	0	0	0	0	0
UML	29	0	29	29	29	0	0	0	0	0	0
S/w Architecture	29	59	0	29	29	0	0	0	0	0	0
S/w Engg	29	59	29	0	59	30	30	0	0	0	0
Research Methodologies	29	59	29	59	0	30	30	0	0	0	0
Adv Data Structure	0	30	0	30	30	0	30	0	0	0	0
Adv DBMS	0	30	0	30	30	30	0	0	0	0	0
NSD	0	0	0	0	0	0	0	0	0	10	5
Client Server	0	0	0	0	0	0	0	0	0	13	4
N/w Security	0	0	0	0	0	0	0	10	13	0	0
CBD	0	0	0	0	0	0	0	5	4	0	0

Table 2.4: Subjects with their Concurrency limit, Degree and Weight

Subject	Concurrency Limit	Degree	Weight
SPM	2	4	116
S/w Architecture	2	4	116
S/w Engg	4	6	236
UML	4	6	236
Research Methodologies	4	6	236
Adv Data Structure	2	4	120
Adv DBMS	2	4	120
NSD	1	2	15
Client Server	1	2	17
N/w Security	2	2	23
CBD	1	2	9

The list of courses according to priority(i.e according to degree and weight):

- S/w Engg
- UML
- Research Methodologies
- Adv Data Structure
- Adv DBMS
- SPM
- S/w Architecture
- N/w Security
- Client Server
- NSD
- CBD

Time table that will be generated due to algorithm for above example is shown in table 5.

Table 2.5: Exam schedule on basis of days and slots

Day and time slot	Subject
Day 1 slot 1	S/w Engg and Client Server
Day 1 slot 2	UML, NSD
Day 2 slot 1	Research Methodologies , CBD
Day 2 slot 2	SPM, Adv data Structure
Day 3 slot 1	Adv DBMS ,S/w Architecture
Day 3 slot 2	N/w security

Chapter 3

Problem Statement

Exam scheduling is part of time table problem. There are various constrains which need to follow to make a feasible schedule. Exam scheduling take certain inputs, and provides output following the various constraints without which it is impossible to make a feasible exam schedule. One of such type of constraint is student can be present at two places i.e. a student can give one exam at a time. Here is the list of inputs, constraints and output for exam scheduling problem.

3.1. Notations

Here are some notations used in this section. $\mathbb{B}$ denotes set of all Booleans, i.e. $\mathbb{B}$:= {true, false }, and S denotes set of all subjects, U denotes set of all students, A denotes set of all seats and R denotes set of all rooms.

All functions names are of kind F_X. Capital letter F denotes range of function i.e. set or subset which will be returned, while small letter denotes that a single value is returned and X denotes domain of function.

3.2. Input

First time related input is defined. Exams of a same student should be in different slots.

Definition 3.2.1. **(Time Input).** The pair $\Gamma_T = < T, BLOCK >$ defines the time input, where

- T denotes a finite set of time slots.
- $BLOCK: T^{\times}T \rightarrow \mathbb{B}$, where block($t_1$, t_2) indicates whether two time slots $t_1, t_2 \epsilon T$ are consecutive.

Exams of a student can be consecutive or not depends on user selection. i.e. user can select whether a student can have two or more exams in consecutive slots.

Definition 3.2.2. **(Student Input).** The pair of tuple $\Gamma_T =< U, CON >$ defines the student input, where

- U denotes a finite set of students.
- $CON: U \to \mathbb{B}$, where con(U) denotes whether students can have consecutive exam or not.

Next input denotes number of students in each subject.

Definition 3.2.3. **(Subject Input).** The pair of tuple $\Gamma_S =< S, U_s >$ defines the subject input where

- S denotes a finite set of subjects.
- $U_s: S \to U$, where $U_S(s)$ denotes the students who has opted subject $s \epsilon S$

Each room contains particular number of seats.

Definition 3.2.4. **(Room Input).**The four tuples $\Gamma_R =< R, N, A, N_A >$defines the room input where

- R denotes finite set of rooms.
- N denotes the number.
- A denotes number of seats.
- $N_A : \to R$, where $N_A(r)$ denotes number of seats in room $r \epsilon R$.

There are many subjects which are hold by same students or subjects which have at least one student in common can't be held at same time. Such subjects should be listed in input.

Definition 3.2.5. **(Conflict Input).** The pair of tuple $\Gamma_C =< S, S_s >$ defines the subject conflict input where

- S denotes a finite set of subjects.
- $S_S: S \to S$, where $S_S(s)$ denotes set of subjects who can't schedule together with subject $s \epsilon S$.

3.3. Decision variables

Decision variables contain the variables which are required as output, like, in exam scheduling subjects is allotted a time slot.

Definition 3.3.1. **(Time slot assignment).**The time slot assignment contains assignment of subjects to slots $t_S: S \rightarrow T$.

3.4. Constraints

All solutions are not feasible in exam scheduling. Some constraints are needed to follow to make a feasible schedule like subjects having at least one student in common can't be scheduled at same time.

Definition 3.4.1.**(Subject constraint).** If a subject $s \epsilon S$ has exam in time slot$t \epsilon T$, it implies that every student holding that subject has to give exam on that time slot.

$$T_S(s) \coloneqq \{t \epsilon T | t_S(s) = t\}$$

holds if $s_1 \neq s_2 \Rightarrow t_S(s_1) \neq t_S(s_2)$ for all subjects $s_1,\ s_2 \epsilon S'$, where $S' \subseteq S$ be the set of subjects that have at least one student in common.

The second constraint is of seats constraints which means that the number of students giving exam should be less than or equal to total number of seats of all rooms.

Definition 3.4.2.**(Seats constraint).**Let $S'' \subseteq S$ where S''list of subjects that can be scheduled at same time. N_U(s) returns the number of students holding subject $s \epsilon S$ and N_A(r) returns number of seats in room r. The seats constraint hold is

$$N_A(R) \coloneqq \{\exists s \epsilon S'' \big| \forall r \epsilon R \mid \sum N_A(\mathrm{r}) \leq \sum N_U(\mathrm{s})\ \}$$

Finally, restriction is implemented on time slots and student. Student can't have consecutive exams if CON{false} case is selected.

Definition 3.4.3.**(BLOCK TIME Constraint).**Finally, restriction is implemented on time slots and student. Student can't have consecutive exams if Con{false} case is selected. T_U(u) returns the set of all time slots for a student u. Block time constraint holds if $BLOCK_U(u) \coloneqq \{\exists t_1, t_2 \epsilon T_U(u) | CON_U(false) \Rightarrow BLOCK(t_1, t_2) = false\ \}$

Chapter 4

Proposed Work

4.1. Algorithm

Algorithm is divided in 4 parts:

1. **Graph Coloring**: In this part, coloring of graph is done and an independent list of subjects will be found out; coloring will start from subject with maximum degree. This algorithm is explained in Chapter 2 Literature Survey
2. **Remove subjects**: In this part removal of subjects is done if total number of students is more than total number of seats available.

   It can be done in two ways:

   a. Removing subjects from list, starting from subject with minimum number of students.

   b. Removing subjects from list, starting from subjects with number of students equal or higher than the difference between total number of students in the list and number of seats.
3. **Adjust**: This part adjusts the subject i.e. after removing subjects from the list, subject can be adjusted if removed subjects have more number of students than the number of seats.

1. **Next graph**: In this part, list of subjects for next iteration is obtained. It again consists of two sub parts depends on whether CON ={true, false}:

   a. Consecutive (CON{true}): If consecutive case is selected, it means that exams of a single student of two or more subjects can be conducted on adjacent sessions.

   b. Non-consecutive (CON{false}): If non-consecutive case is selected, it means that exams of a single student of two or more subjects can't be conducted on adjacent sessions.

Algorithm 4: Remove subjects from list starting from subject with minimum number of students

Input: $s_1, s_2, s_3 \ldots\ldots s_n$ is list of subjects in increasing order according to number of students $r_1, r_2, r_3 \ldots r_m$ is list of rooms; maxdegree contains maximum degree contained in graph.

Output: List of subjects after removal of subjects if number of students in list is more than number of seats.

1. $Let\ TSeats \leftarrow \sum_{i=1}^{m} r_i, TSudents \leftarrow \sum_{j=1}^{n} S_j.students.$
2. $Let\ difference \leftarrow TStudents - TSeats.$
3. $Repeat\ for\ i\ while\ difference \geq 0$
4. $If\ S_i.degree \neq maxdegree\ then$
   $difference = difference - S_i.students\ and\ remove\ S_i from\ list.$
5. $End\ if.$
6. $Let\ i \leftarrow i + 1.$
7. End loop.

In the above algorithm, subjects are removed if TStudents are more than the number of seats. To remove a subject first, difference of TStudents and TSeats is taken and the list is arranged according to increasing number of students. Then subjects are removed from the starting of list until difference between TStudents and TSeats becomes zero or negative. One thing which is taken care is that if any subject has maximum degree in that graph, that subject will not be removed so as to reduce dependencies for next iteration.

Algorithm 5: Remove subjects from list starting from subject with equal or greater number of students

Input: $s_1, s_2, s_3 \ldots\ldots s_n$ is list of subjects in increasing order according to number of students, $r_1, r_2, r_3 \ldots r_m$ is list of rooms; maxdegree contains maximum degree contained in graph.

Output: List of subjects after removal of subjects if number of students in list is more than number of seats.

1. $Let\ TSeats \leftarrow \sum_{i=1}^{m} r_i, TSudents \leftarrow \sum_{j=1}^{n} S_j.students.$
2. $Let\ difference \leftarrow TStudents - TSeats.$

3. $Repeat\ while\ difference > 0.$
4. $Repeat\ for\ i = 1\ to\ S.length.$
5. $If\ i \geq S.length\ then$
6. $If\ S_i.degree == maxdegree\ then$
   $difference = difference - S_{i-1}.students\ and\ remove\ S_{i-1}\ from\ list.$
   $break.$
7. $Otherwise,$
   $difference = difference - S_i.students\ and\ remove\ S_i\ from\ list.$
   $break.$
9. $End\ If.$
10. $End\ If.$
11. $If\ S_i \geq difference\ then$
12. $If\ S_i.degree == maxdegree\ then$
   $difference = difference - S_{i-1}.students\ and\ reDove\ S_{i-1}\ from\ list.$
13. $Otherwise,$
   $difference = difference - S_i.students\ and\ remove\ S_i\ from\ list.$
14. $End\ If.$
15. $End\ If.$
16. $End\ loop.$
17. $End\ loop.$

In the above algorithm, again the subjects are removed if TStudents are more than the number of seats but in a bit different way. To remove a subject first, difference of TStudents and TSeats is taken and list is arranged according to increasing number of students. In this, the subject chosen to remove will either have equal number of students with the difference or if no such subject exists, then the subject having just greater number of students will be removed. To do this start from the ending of list and find a subject whose number of students are greater or equal to difference between TStudents and TSeats. In this case also care is taken that if any subject has maximum degree in that graph, that subject will not be removed so as to reduce dependencies for next iteration.

Algorithm 6: Adjust: Subjects

Input: s_1, s_2, s_3……s_n is list of subjects in increasing order according to number of students, leftSeats is number of seats left on which students can be adjusted.
Output: List of subjects after adjustment of subjects if any vacant seat is left.

1. $Let\ alist\ be\ list\ of\ adjacent\ subjects. Initially\ alist \leftarrow null.$
2. $For\ each\ subject\ S_i, alist\ \leftarrow \{alist\} \cup \{S_i.adjacentsubjects.\}$
3. $Find\ subject\ suN\ such\ that, sub\ \notin S\ or\ sub\ \notin alist.$
4. $If\ sub.students \leq leftSeats\ then$
   $S = \{\ S\ \} \cup\ sub.$
   $leftSeats = leftSeats - sub.students.$
5. $If\ leftSeats\ \neq 0\ repeat\ from\ step\ 1.$
6. $End\ If.$
7. $End\ If.$
8. $End\ loop.$

In above algorithm, after removing subjects sometimes seats are left vacant or can say difference between TStudents and TSeats become negative. In that case, above algorithm will run and will find the subject, if any, have students less than or equal to the number of seats left and add that subject in the list.

Next Graph

Algorithm 7: Consecutive subjects

Input : Slist is list of subjects taken for present session, TotalSlist is list of subjects which are unscheduled up to previous session.
Output: Nlist filled with subjects for making next graph.
1. $Nlist = TotalSlist - Slist.$

In above algorithm, list of students is extracted of next iteration. In case of CON={true} i.e. if consecutive exams of a student are allowed, then simply the subjects chosen for previous iterations are removed from the total list of subjects.

Algorithm 8: Non-consecutive subjects

Input: Slist is list of subjects taken for present session; TotalSlist is list of subjects which are unscheduled up to previous session.
Output: Nlist filled with subjects for making next graph.

1. $Let\ alist\ be\ list\ of\ adjacent\ subjects. Initially\ alist \leftarrow null.$
2. $For\ each\ subject\ S_i \in Slist, alist \leftarrow \{alist\} \cup \{S_i.adjacentsubjects\}.$
3. $Nlist = \{\ TotalSlist\ \} - \{\ \{\ Slist\ \} \cup \{\ alist\ \}\ \}.$
4. $End\ Loop$

In above algorithm, list of students is extracted of next iteration. In case of CON={false} i.e. if consecutive exams of a student are not allowed, then all the subjects chosen for previous iterations are removed from the total list of subjects and subjects adjacent to the last list will be removed for this iteration only.

Chapter 5

Testing and Experimental Results

Depending on different methods of removing subjects, if number of students are more than number of subjects and whether a student can have consecutive exam of two or more subjects or not, four categories are divided which are given in Table 6.

Table 5.1: Different categories

	Step2a	**Step2b**	**Step4a**	**Step4b**
Cat A	N	Y	N	Y
Cat B	Y	N	N	Y
Cat C	N	Y	Y	N
Cat D	Y	N	Y	N

CatA contains remove subjects with equal or greater number of students with non-consecutive exams; Cat B contains remove subjects starting with minimum number of subjects with non-consecutive exams; Cat C contains remove subjects with equal or greater number of students and with consecutive exams and Cat D contains remove subjects starting with minimum number of subjects with consecutive exams.

5.1. Illustration

Consider the dependency graph G shown in Figure 11.

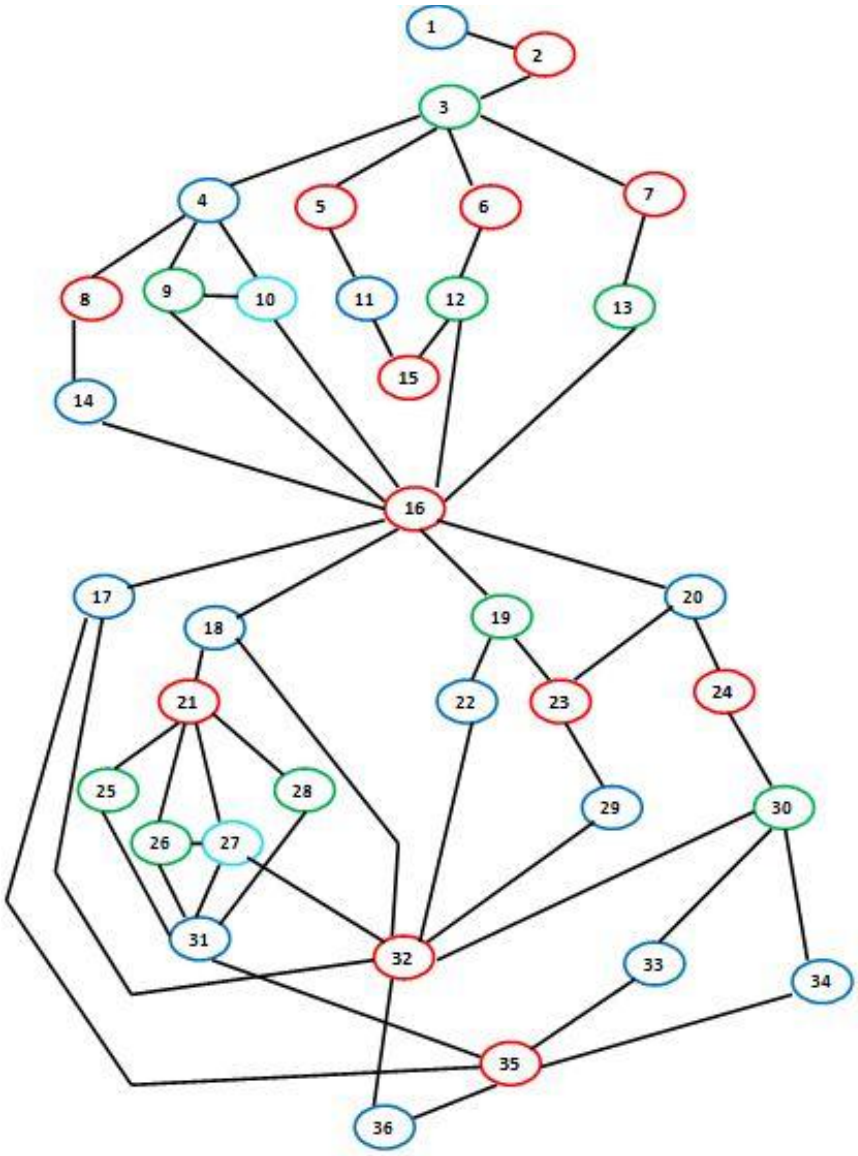

Figure 5.1: A Simple Graph G

Numbers of student in each subject are: 1^{20}, 2^{30}, 3^{10}, 4^{10}, 5^{15}, 6^{20}, 7^{40}, 8^{25}, 9^{10}, 10^{10}, 11^{30}, 12^{40}, 13^{15}, 14^{35}, 15^{30}, 16^{110}, 17^{30}, 18^{35}, 19^{30}, 20^{35}, 21^{40}, 22^{30}, 23^{20}, 24^{35}, 25^{40}, 26^{40}, 27^{40}, 28^{40}, 29^{45}, 30^{40}, 31^{40}, 32^{25}, 33^{10}, 34^{15}, 35^{20} and 36^{15}.

Total number of seats =300

5.1.1. For Non-consecutive and removing subjects starting with equal or greater number of students(Cat A)

Starting from node number 16(node with highest degree) the list of independent subjects in increasing order is:

5^{15}, 6^{20}, 35^{20}, 23^{20}, 8^{25}, 32^{25}, 2^{30}, 15^{30}, 24^{35}, 7^{40}, 21^{40} and 16^{110}

Total number of students in list is:

TStudents = 15 + 20 + 20 + 20 + 25 + 25 + 30 + 30 + 35 + 40 + 40 +110 = 410

Difference = 395 – 300 = 110.

As subject 16 has maxdegree so moving towards next the subject, 21^{40} will be removed.

Now difference = 110 – 40 = 70.

Again, subject 7 will be removed. Now difference = 70 – 40 =30.

Again, subject 15 will be removed. And difference becomes 30 – 30 = 0,as difference is 0 now, therefore no need to remove subject further. Hence subjects in session 1 will be:

Session 1 = 5, 6, 35, 23, 8, 32, 2, 16, 24

In case of non-consecutive, there is need to build alist to check which subjects are adjacent to subjects in list above which is shown in Table 7.

Table 5.2: List of adjacent subjects after Session 1 for Cat A

Name of subject	List of adjacent subjects
5	3,11
6	3,12
35	31,33,34,17,36
23	19,20,29
8	4,14
32	18,22,29,30,17,36,27
2	1,3
16	14,9,10,12,13,17,18,19,20
24	20,30

From above, alist contains subject: 1, 3, 4, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 22, 27, 29, 30, 31, 33, 34 and 36. And SList contains: 2, 5, 6, 8, 14, 16, 32 and 35.

Hence list for next graph = $Total\ subjects = Total\ subjects - \{\{SList\} \cup \{alTst\}\}$= 7, 15, 21,25,26,28. Graph for these subjects is shown in Figure 12.

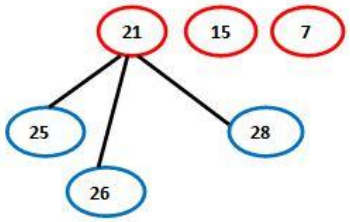

Figure 5.2: Non-dependent subjects after iteration 1 for Cat A

Similarly as iteration in1, subjects for session 2 are 15, 7 and 21. List of adjacent subjects for session 2 are shown in Table 8.

Table 5.3: List of adjacent subjects after Session 2 for Cat A

Name of subjects	List of adjacent subjects
15	11,12
7	3,13
21	18,25,26,27,28

Alist= 3,11,12,13,18,25,26,27,28 and SList= 7,15,21

List of subjects for iteration 3 is: 1, 4, 9, 10, 14, 17, 19, 20, 22, 29, 30, 31, 33, 34 and 36. Graph of these subjects is shown in Figure 13.

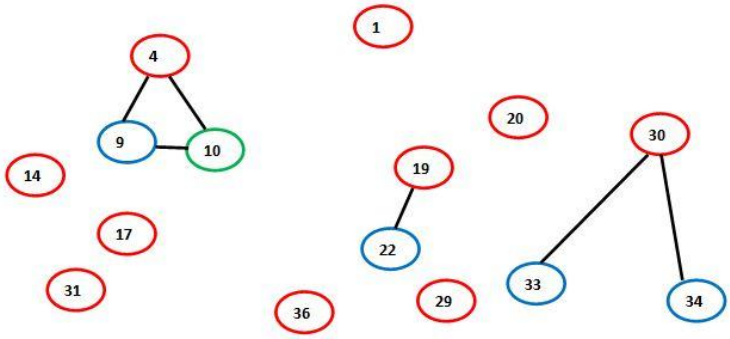

Figure 5.3: Non-dependent subjects after iteration 2 for Cat A

Starting from subject 4 lists of independent subjects in increasing order is:
4^{10}, 36^{15}, 1^{20}, 17^{30}, 19^{30}, 20^{35}, 14^{35}, 30^{40}, 31^{40} and 29^{45}.

TStudents= 10 + 15 + 20 + 30+30+35+35+40+40+45=300. Hence whole list will be taken for session 3

Session3=4, 29, 36, 1, 17, 19, 20, 14, 30 and 31.

Proceeding in this way, and iterating on subjects until all subjects get scheduled subjects for next sessions will be:

Session 4= 11, 12, 13, 18, 22.

Session 5= 3, 9, 25, 26, 28, 33, 34

Session 6=--

Session 7= 10, 27

Table 5.4: List of subjects for different sessions for Cat A

Session number	List of subjects
1	5, 6, 35, 23, 8, 32, 2, 16, 24
2	15, 7, 21.
3	4, 29, 36, 1, 17, 19, 20, 14, 30, 31.
4	11, 12, 13, 18, 22.
5	3, 9, 25, 26, 28, 33, 34
6	---
7	10, 27

5.1.2. For Non-consecutive and removing subjects starting with minimum number of students(Cat B)

Starting from node number 16(node with highest degree) the list of independent subjects in increasing order is:

5^{15}, 6^{20}, 35^{20}, 23^{20}, 8^{25}, 32^{25}, 2^{30}, 15^{30}, 24^{35}, 7^{40}, 21^{40} and 16^{110}

Total number of students in list is:

TStudents = 15 + 20 + 20 + 20 + 25 + 25 + 30 + 30 + 35 + 40 + 40 +110 = 410

Difference = 395 – 300 = 110. In this case removal of subjects will be from subject with minimum number of students hence subject 5 will be removed first

Now difference =110-15=95.

Again, subject 6 will be removed now difference will become 95 – 20 = 75.

Proceeding same way until difference becomes <=0.

Subjects removed from list will be: 5^{15}, 6^{20}, 35^{20}, 23^{20}, 8^{25} and 32^{25}.

And difference will be -15.

As difference is -15, subject with 15 or less students can be adjusted if that subject is not adjacent to subjects listed above.

Subject 36 can be adjusted in list

Hence Session1: 2, 7, 15, 16, 21, 24 and 36.

In case of non-consecutive, there is need to build alist to check which subjects are adjacent to subjects in list above which are shown in Table 10.

Table 5.5: List of adjacent subjects after Session 1 for Cat B

Name of subject	List of adjacent subjects
2	1,3
7	3,13
15	11,12
16	14,9,10,12,13,17,18,19,20
21	18,25,26,27,28
24	20,30
36	32,35

From above, alist contains subject: 1, 3, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 25, 26, 27, 28, 30, 32 and 35. And SList contains: 2, 7, 15, 16, 21, 24 and 36.

Hence list for next graph = Total subjects – {{SList} U {alist}} = 4, 5, 6, 8, 22, 23, 29, 31, 33, 34. Graph for these subjects is shown in Figure 14.

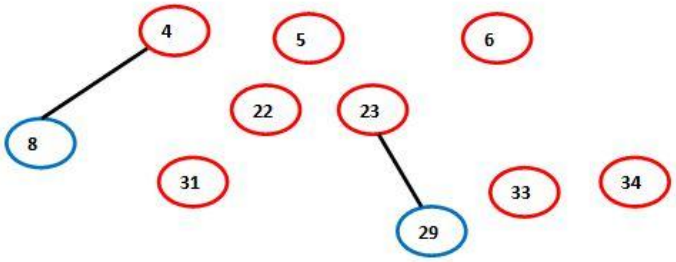

Figure 5.4: Non-dependent subjects after iteration 1 for Cat B

List of independent subjects from above graph in increasing order is: 4^{10}, 33^{10}, 5^{15}, 34^{15}, 6^{20}, 23^{20}, 22^{30} and 31^{40}.

TStudents = 10+10+15+15+20+20+30+40=160. As TStudents are less than TSeats (i.e. 300), whole list can be taken for next session.

Session 2= 4,5,6,22,23,31, 33 and 34. For alist adjacent subject list is shown in Table 11

Table 5.6: List of adjacent subjects after Session 2 for Cat B

Name of subject	List of adjacent subjects
4	3,8,9,10
5	3,11
6	3,12
22	19,32
23	19,29,20
31	25,26,27,28,35
33	30,35
34	30,35

Subjects left for next iteration are: 1, 13, 14, 17, 18 and graph of these subjects are shown in Figure 15.

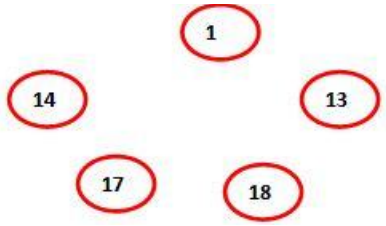

Figure 5.5: Non-dependent subjects after iteration 2 for Cat B

Session 3= 1, 13, 14, 17 and 18.

For alist adjacent subject list is shown in Table 12.

Table 5.7: List of adjacent subjects after Session 3 for Cat B

Name of subject	List of adjacent subjects
1	
13	
14	8
17	32,35
18	

Subjects left for next iteration are: 3, 9, 10, 11, 12, 19, 20, 25, 26, 27, 28, 29 and 30.

Graph for these subjects is shown in Figure 16.

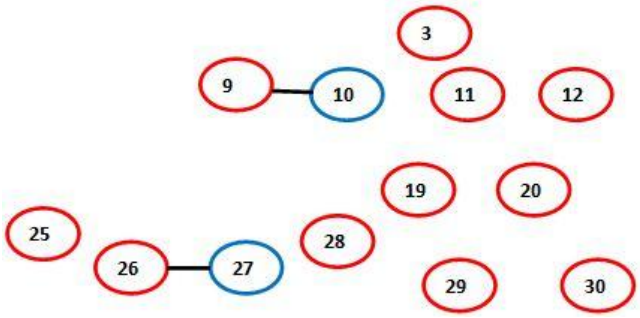

Figure 5.6: Non-dependent subjects after iteration 3 for Cat B

List of independent subjects in increasing order is:

3^{10}, 9^{10}, 11^{30}, 19^{30}, 20^{35}, 12^{40}, 25^{40}, 26^{40}, 28^{40}, 30^{40} and 29^{45}.

TStudents= 10 + 10 + 30 + 30 + 35 + 40 + 40 + 40 + 40 + 40 +45 = 360.

Difference = 360 – 300 =60.

Remove subjects 3, 9, 11 and 19. Now difference=60-10-10-30-30=-20

Adjust subjects 3, 9.

Hence subjects in session4 are: 3, 9, 12, 20, 25, 26, 28, 29 and 30.

Proceeding in same way, and doing iterations till all subjects got scheduled:

Session5= 8,11,19,35

Session 6=10, 27

Session7=--

Session8=32

Table 5.8: List of subjects for different sessions for Cat B

Session number	List of subjects
1	2, 7, 15, 16, 21, 24, 36.
2	4,5,6,22,23,31,33,34.
3	1, 13, 14, 17, 18.
4	3, 9, 12, 20, 25, 26, 28, 29, 30
5	8, 11, 19, 35
6	10, 27
7	----
8	32.

5.1.3. For consecutive and removing subjects starting with equal or greater number of students (Cat C)

As from Cat A first iteration will be same, hence

Session1: 2, 5, 6, 8, 16, 23, 24, 32 and 35.

As in this case there can be consecutive exams, hence no need for alist. Just graph for next iteration is made from subjects left to schedule. Graph of left subjects is shown in Figure 17.

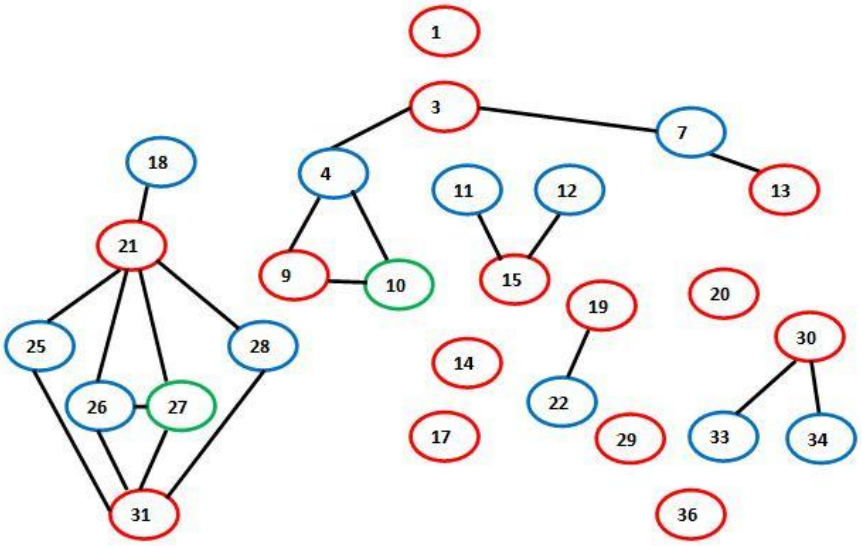

Figure 5.7: Subjects left after iteration 1 for Cat C

List of independent subjects in increasing number of students is: 3^{10}, 9^{10}, 13^{15}, 36^{15}, 1^{20}, 15^{30}, 17^{30}, 19^{30}, 20^{35}, 14^{35}, 30^{40}, 21^{40}, 31^{40} and 29^{45}.

TStudents=10 + 10 + 15 + 15 + 20 + 30 + 30 + 30 + 35 + 40 + 40 + 40 + 45 = 395.

Difference =395-300 =95. As difference is >0, so there is need of removing subjects. As it is case of removing subjects with more students, subjects 29^{45}, 30^{40}, 3^{10}.

Session2 = 9, 13, 36, 1, 15, 17, 19, 20, 14, 21 and 31.

Graph of subjects for next iteration is shown in Figure 18.

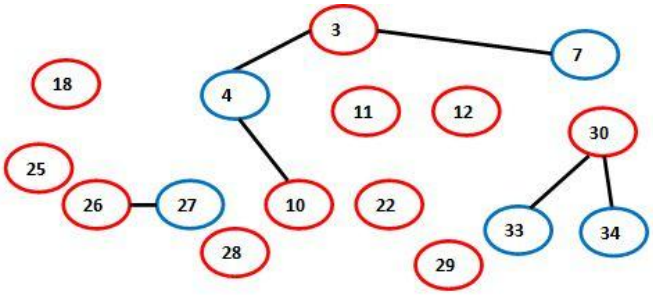

Figure 5.8: Subjects left after iteration 2 for Cat C

List of independent subjects in increasing order is: 3^{10}, 10^{10}, 11^{30}, 22^{30}, 18^{35}, 12^{40}, 25^{40}, 26^{40}, 28^{40}, 30^{40} and 29^{45}.

TStudents= 10+10+30+30+35+40+40+40+40+40+45= 360

Difference =360 -300. Remove = 29 and 28 (can't remove subject as subject 30 has maxdegree).

Session 3: 3, 10, 11, 22, 18, 12, 25, 26 and 30.

Graph for next iteration is shown in Figure 19.

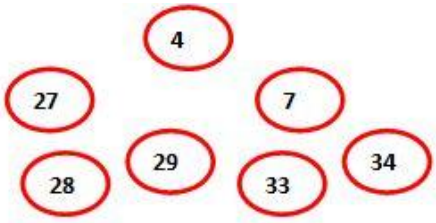

Figure 5.9: Subjects left after iteration 3 for Cat C

List of independent subjects in increasing order is: 4^{10}, 33^{10}, 34^{15}, 7^{40}, 27^{40}, 28^{40} and 29^{45}.

TStudents= 10 + 10+ 15+40+40+40+45=200. As TStudents is less than 300 so whole list will be taken

Session 4=4, 33, 34, 7, 27, 28, 29.

Table 5.9: List of subjects for different sessions for Cat C

Session number	List of subjects
1	2, 5, 6, 8, 16, 23, 24, 32 and 35
2	9, 13, 36, 1, 15, 17, 19, 20, 14, 21, 31
3	3, 10, 11, 22, 18, 12, 25, 26, 30
4	4, 33, 34, 7, 27, 28, 29

5.1.4. For consecutive and removing subjects starting with minimum number of students(Cat D)

Session 1: 2, 7, 15, 16, 21, 24, 36.

As in this case there can be consecutive exams, hence no need for alist. Just graph for next iteration is made from subjects left to schedule. Graph of left subjects is after iteration 1 is shown in Figure 20.

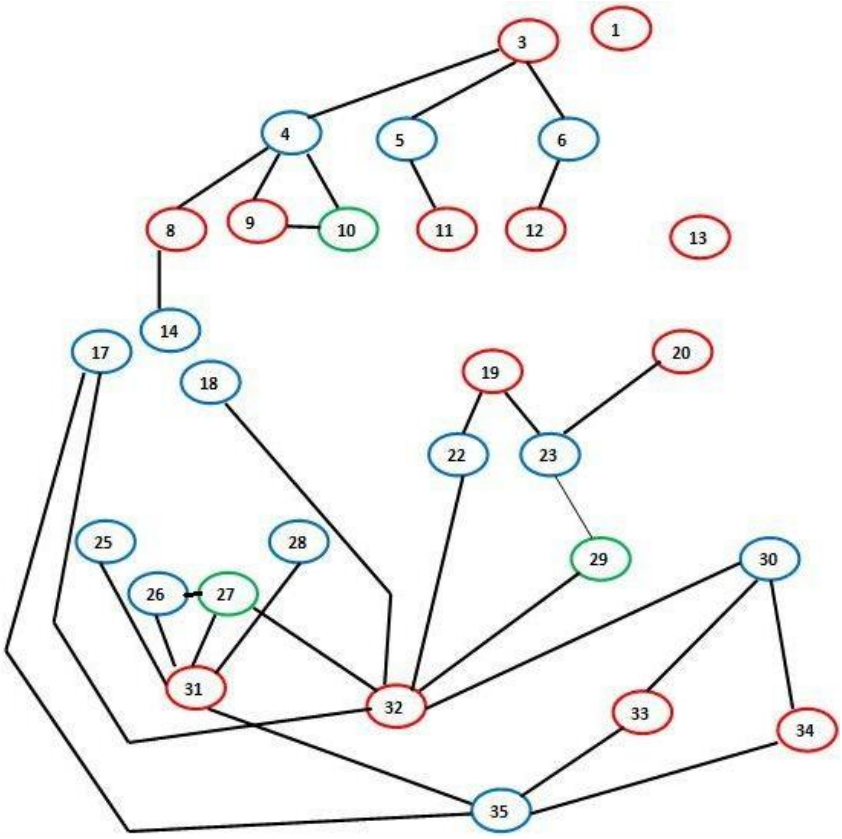

Figure 5.10: Subjects left after iteration 1 for Cat D

List of independent subjects I increasing order is: 3^{10}, 9^{10}, 33^{10}, 13^{15}, 34^{15}, 1^{20}, 8^{25}, 32^{25}, 11^{30}, 19^{30}, 20^{35}, 12^{40} and 31^{40}.

TStudents=10+10+10+15+15+20+25+25+30+30+35+40+40=305, Difference=305-300=5. As in this case subjects are removed with minimum number of students, subject 3 is removed.

Session 2= 9, 33, 13, 34, 1, 8, 32, 11, 19, 20, 12 and 31. Graph of left subjects is shown in Figure 21.

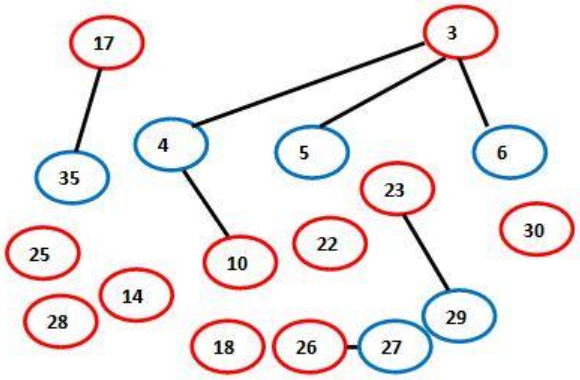

Figure 5.11: Subjects left after iteration 2 for Cat D

List of independent subjects in increasing order is: 3^{10}, 10^{10}, 23^{20}, 17^{30}, 22^{30}, 14^{35}, 18^{35}, 25^{40}, 26^{40}, 28^{40} and 30^{40}.

TStudents=10 + 10 + 20 + 30 + 30 + 35 + 35 + 40 + 40 + 40 + 40 = 330.

Similarly as above, Remove subject 10 and 23(Subject 3 can't be removes as subject 3's degree is equal to maxdegree of this graph).

Session 3: 3, 17, 22, 14, 18, 25, 26, 28 and 30.

Graph of left subjects after iteration 3 is shown in Figure 22.

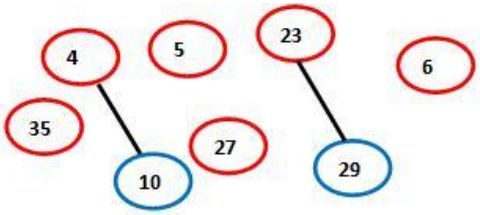

Figure 5.12: Subjects left after iteration 3 for Cat D

List of independent subjects is: 4^{10}, 5^{15}, 6^{20}, 23^{20}, 35^{20} and 27^{40}.

TStudents=10+15+20+20+20+40=125. As number of students is less than 300, whole list will be taken for next session.

Session 4= 4, 5, 6, 23, 35, 27.

Similarly Session 5= 10, 29.

Table 5.10: List of subjects for different sessions for Cat D

Session number	List of subjects
1	2, 7, 15, 16, 21, 24, 36
2	9, 33, 13, 34, 1, 8, 32, 11, 19, 20, 12 and 31
3	3, 17, 22, 14, 18, 25, 26, 28 and 30
4	4, 5, 6, 23, 35, 27
5	10,29

Chapter 6

Conclusion and Future work

As exam scheduling problem can be solved in different ways like exams can be consecutive or non-consecutive. Removing of subjects can also be done in two ways i.e. removing subjects from list starting from subject with minimum number of students and remove subjects from list starting from subject with equal or greater number of students than difference of TStudents and TSeats.. Number of sessions of different categories is shown in Table 16.

Table 6.1: Comparison of number of sessions for different categories.

Categories	**CatA**	**CatB**	**CatC**	**CatD**
Number of sessions	7	8	4	5

Hence, from above result, it is found that CatA and CatB are for non-consecutive(CON{false}) and CatA provides less number of sessions than CatB. CatC and CatD are for consecutive(CON{true}), CatC provides less number of sessions than CatD. So CatA and CatC are better i.e. removing subjects from list starting from subject with equal or more number of students than difference between TStudents and TSeats is better than removing subjects from list starting with subject having minimum students.

In future more work can be done on combination of exam scheduling and room allocation. Also teacher assignment and seat allocation algorithms be combined with exam scheduling algorithm.

References

[1] Alon N., "A Note on Graph Colorings and Graph Polynomials," Journal of Combinatorial Theory Series B, vol. 70, no. 1, pp. 197-201, 1997.

[2] Arvind. S. Babu , R.Chockalingam and S.Kavitha,"A Hybrid Genetic Algorithm Approach to departmental Class Timetabling Problem Using Efficient Data", *International Journal of Computer Applications."* , vol. 1, no. 17, pp. 99-103, 2010.

[3] Baldi P., "On a Generalized Family of Colorings," Graphs and Combinatorics, vol. 6, no. 2, 1990.

[4] Bang-Jensen J. and Gutin G., Digraphs: Theory, Algorithms and Applications, Springer- Verlag,2000.

[5] Bar-Noy A., Motwani R., and Naor J., "The Greedy Algorithm is Optimal for On-line Edge Coloring," Information Processing Letters, vol.44, no. 5, pp. 251-253, 1992.

[6] Batenburg K. and Palenstijn W., "A New Exam Timetabling Algorithm," Leiden Institute of Advanced Computer Science (LIACS), http://visielab.ua.ac.be/staff/batenburg/papers/bapa_bnaic_2003.pdf.

[7] B. McCollum, P.McMullan, A.J. Parkes, E.K. Bruke and R. Qu, "A New Model for Automated Examination Timetabling", *Annals of Operations Research,* vol. 194, no. 1, pp. 291-315, 2012.

[8] Bean D., "Effective Coloration," The Journal of Symbolic Logic, vol. 41, no.2, pp. 469-480, June 1976.

[9] Burke E. and Petrovic S., "Recent Research Directions in Automated Timetabling," European Journal of Operational Research (EJOR), vol.140, no. 2, pp 266-280, 2002.

[10] Burke E., Elliman D., and Weare R., "A University Timetabling System Based on Graph Coloring and Constraint Manipulation," Journal of Research on Computing in Education, vol. 27,no. 1, pp. 1-18, 1994.

[11] C.C. Gotlieb. The construction of class-teacher time-tables. Proc. IFIP Congress, 62:73- 77,1963.

[12] Christofides N., Graph Theory: An Algorithmic Approach, Academic Press, 1975.

[13] D.Woods and A Trenaman. Simultaneous satisfaction of hard and soft timetable

constraints for a university department using evolutionary timetabling. Department of Computer Science, National University of Ireland, Maynooth, Co. Kildare, Ireland, 1999.

[14] Dotton R.D. and Brigham R.C. "A New Graph Coloring Algorithm", *Comp. Jrnl.*, vol 24, no. 1, pp. 85 – 86, 1981.

[15] E.K. Burke, D.G. Elliman, and R. Weare, A genetic algorithm based university timetabling system. In Proceedings of the 2nd East-West International Conference on Computer Technologies in Education, volume 1, pages 35-40, 1995.

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

[17] Gross J. and Yellen J., Graph Theory and its Applications, 2nd ed., CRC Press, 2005.

[18] Husseini S., Malkawi M., and Vairavan K.,"Analysis of a Graph Coloring Based Distributed Load Balancing Algorithm," Journal of Parallel & Distributed Systems, vol. 10, no. 2, pp. 160-166, 1990.

[19] Husseini S., Malkawi M., and Vairavan K.,"Graph Coloring Based Distributed Load Balancing Algorithm and Its Performance Evaluation," in the 4th Annual Symposium on Parallel Processing, 1990.

[20] M. Malkawi, M. Al-Haj Hassan, and O. Al-Haj Hassan,A New Exam Scheduling Algorithm Using Graph Coloring,The International Arab Journal of Information Technology, Vol. 5, No. 1, pp 80-87, January 2008.

[21] M. Trick. Network resources for coloring a graph. http://mat.gsia.cmu.edu/COLOR/color.html

[22] Norberciak Maciej, "Universal Method for Timetable Construction based on Evolutionary Approach", *World Academy of Science, Engineering and Technology*, vol. 15, pp. 91-96, 2006.

[23] P.M. Cavanaugh and C. Tran. Private communication. University of Houston, Houston, TX.

[24] Raymond Chiong, Ooi Koon Beng, "A Comparison between Genetic Algorithms and Evolutionary Programming based on Cutting Stock Problem", *Engineering Letters*, vol. 14, no. 1, pp. 72 – 77, 2007.

[25] S.K. Miner, S. Elmohamed, and H.W. Yau. Optimizing timetabling solutions using graph coloring. 1995 NPAC REU program, NPAC, Syracuse University, 1995.

[26] Tehrani A., "Un Algorithme de Coloration", *Cahiers du centre d'e´tudes do Recherche Ope´rationnelle*, vol. 17, no. 2 – 3 – 4 , pp 395 – 398, 1975.

[27] Timothy A. Redl, "University Timetabling via Graph Coloring: An Alternative Approach", *Congressus Numerantium*, vol. 187, pp. 175 – 184, 2006.

[28] W. Erben and J. Keppler. A genetic algorithm solving a weekly course timetabling problem. In Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling, 1995.

[29] Willemen Robertus J., *"School timetable construction Algorithms and complexity"*, IPA Dissertation Series, 2002 – 05.

[30] http://en.wikipedia.org/wiki/Graph_coloring

List of Publications

Communicated

[1] Meena Bharti, Ravinder Kumar, "Better resource utilization of exam scheduling", The International Arab Journal of Information Technology.
(Impact factor: 0.065)

[2] Meena Bharti, Ravinder Kumar, "Efficient resource utilization of exam scheduling using graph coloring", Journal of computer science and technology.
(Impact factor: 0.656)