Course Scheduling with Graph Colouring Algorithm

Ammar El Hassan Ph.D.
College of Computer Engineering & Science
Prince Mohammad bin Fahd University
AL-Khobar, Saudi Arabia
aelhassan@pmu.edu.sa

Abstract. In this paper, the graph-colouring algorithm for scheduling courses to time-room slots is implemented using a 3-way approach that varies the selection order of courses to serve. The results are then contrasted and compared for completeness and time-efficiency. The approach utilizes Conflict Matrices that are used as input for the coloring algorithm in order to produce colored groups of compatible courses. The different sorting styles are analyzed to select the optimum approach include standard (alphabatical) such that the course with least conflicts is selected first, by Conflict_Weight and by Conflict_Weight Descending such that the course with most conflicts is selected first.

Keywords: Course Conflict, Instructor Conflict, Graph Coloring Conflict Matrix, Conflict Weight.

1. INTRODUCTION

Scheduling algorithms for courses or exams have been the subject of much research including surveys of various techniques and algorithms [3, 4, 11]. AbdulRahman et al. proposed an exam-time tabling (ETA) algorithm in an open credit system [12]. Computer applications have also been developed as implementations of some of the research theories [5].

This interest is due, in part, to the significance of schedule automation and the potential man-hours a successful theory and application can save. Exam time table generation is considered an NP-Hard problem; meaning that no algorithm has been developed to solve it in reasonable (polynomial) amount of time [1, 2].

There are many approaches to this common problem; mostly based on graph coloring [6, 7, 8] or on Genetic Approaches [9]. Asham et al [10] propose a solution to the exam timetable problem that utilizes a hybrid approach based on Graph Coloring and Genetic algorithms wherein these two approaches are studied and compared to a new (hybrid) algorithm. The hybrid algorithm fared better than the genetic algorithm in terms of time-to-process performance and better than the graph coloring algorithm in terms of fitness performance.

What is proposed in this paper is also a hybrid methodology wherein the Conflict_Matrix which forms the pool of courses to schedule may be interrogated according to Conflict-Weight or Alphabetical course title or randomly, the resulting course sets are then schedules in one of several approaches within the Sequential Graph Coloring (SGC) (greedy algorithm) once by grabbing the courses from the list of unscheduled courses in alphabetical order, and then by using the conflict_weight of a course as an indexer, in both descending and ascending order.

The performance of the three approaches, is compared and contrasted. All algorithms are implemented using C# Loops and Microsoft Visual Studio Recordset/Dataset filtering functionality for increased efficiency in conjunction with Oracle as the main database. The results are then compared and contrasted in terms of both time-to-process and fitness performance.

General Algorithm Requirements

Any new algorithm that is sought in this work needs to eventually solve the exam scheduling problem within the following parameters:

- Courses which enroll the same set of students cannot be scheduled for the same timeslot; this is the definition of a standard course conflict.
- Maximum number of exams per day per student should not exceed a certain number, this is set to two.
- Two or more courses taught by the same instructor cannot be scheduled for the same timeslot. This is instructor conflict. Instructors must be available at the times their courses are scheduled.
- If two or more courses require a specific room/hall, these cannot be scheduled for the same timeslot, this is room conflict. No classroom sharing. i.e., two or more courses scheduled for the same timeslot cannot be assigned the same classroom.
- Each course must be scheduled for exactly one timeslot and one room, to remain constant throughout the scheduling period.
- Each course must be scheduled in an available classroom that can accommodate its size.
2. Methodology

The methodology proposed in this work is a 4-phase process as illustrated in Fig 1. The first phase is the production of the Conflict Matrix, which is extracted from the Course List data table.

2.1 Conflict Matrices

Course List

The Course List is a simple database table showing Student_ID for each course (module section combination) taught at PMU. Fig 2 shows a subset of the course list. The course names (identifiers) in both the Course List and the conflict matrix is a combination of the module name plus the section number; for example, section 202 of module GEIT1411 is listed in the matrix as GEIT1411_202.

With this naming convention, each section of each module will be treated as an independent entity for scheduling purposes; this is a “semantic” requirement/assumption by the institute. The conflict matrix used in this work comes in three variants, explained below.

In all matrices used here, the order of the column headings is the same as the order of the row headings; hence all conflict matrices are symmetric with respect to the main diagonal. The SQL used to build the matrices is available by contacting the authors.

Standard (Alphabetical) Conflict Matrix

Like all matrices to follow, this contains the list of courses as row and column indices, with each cell showing the number of shared/common students between the 2 courses at the row-column indices. Fig 3 shows a simplified instance of this matrix. The image clearly shows the alphabetical sorting of course names, the composition of course names (module concatenation with section number) and a typically high conflict value of 6 between course “ACCT3312_201” and “ACCT3311_201”, meaning there are 6 common/shared students between these two courses.

Conflict-Weight

The next matrix to use is the Conflict-Weight matrix. This is a variation of the standard conflict matrix, above, with an added column showing the total number of shared students within each course. The image in Fig 4 shows the Conflict-Weight column; this matrix is sorted by this column, hence the column-row indices are no longer in alphabetical order, rather they are in conflict-weight followed by alphabetical order.
order, see the two courses with a conflict-weight of 5 for example.

![Conflict-Weight Matrix](image)

**Conflict-Weight Descending**

The Conflict-Weight Descending matrix is based on the conflict-weight matrix (above) sorted by descending conflict-weight. The column and row indices with the identical conflict-weights are still sorted alphabetically.

2.2 CONFLICT GRAPH

The conflict matrices, above, form the bases for creating the Conflict Graphs [13]. With each Conflict Graph G, there are a set of n courses \( \{c_1, c_2, \ldots, c_n\} \) to be scheduled.

Each course \( c_i \) will be represented by exactly one vertex \( v_i \) in \( G \), hence, \( G \) contains \( n \) vertices:

\[ V(G) = \{v_1, v_2, \ldots, v_n\}. \]

Edges between vertices \( v_i \) and \( v_j \) where \( v_i \in V(G) \) and \( v_j \in V(G) \) indicate that the two courses \( c_i \) and \( c_j \) cannot be scheduled at the same timeslot, these occur if any one or more of the following conditions is true:

**Course Conflict:**

\( c_i \) and \( c_j \) have at least one common student

**Instructor Conflict:**

\( c_i \) and \( c_j \) have a common instructor

**Room Conflict:**

\( c_i \) and \( c_j \) require the same room/hall

2.3 GRAPH COLORING ALGORITHM

With the Conflict Graph complete, the process of coloring the vertices can commence. The algorithm works by coloring a pair of vertices \( v_i \) and \( v_j \) with distinct colors if there is an edge between them. Vertices that do not share an edge may be colored with different colors, or they may be colored the same color (see limited rooms above).

Suppose we are working with a symmetric conflict matrix \( CM \) of size \( N \times N \), where \( N \) is the number of courses requiring exams slots, the process, as illustrated in Fig 6 is:

i. Assign color1 to the vertex corresponding to the course with highest conflict weight, call it \( v_i \). (In the case of the Conflict-Weight Descending matrix above, this will be the first course)

ii. Also assign color1 to any vertex that is not (directly) connected to \( v_i \) and not connected with other vertices that are of color1.

iii. Assign color2 to the next uncoded vertex with the next highest conflict weight.

iv. Repeat step (ii) for color2

v. If the number of the colored courses in one group exceed the NUMBER_OF_ROOMS use a new color

vi. Repeat step (ii) until all courses are colored.

vii. Assign a timeslot for each color

viii. Assign a room for each course within a color group

![New Graph Coloring Algorithm](image)

**Algorithm Pseudo Code**

Pseudo Code for the New Graph Coloring Algorithm is shown in appendix
2.4 ROOM ALLOCATION

Each set of colored vertices can be assigned a distinct timeslot in the timetable.

Let \( k \) be the number of groups or colors for the entire graph \( G \), hence the schedule for \( G \) will require \( k \) timeslots.

Let \( d \) denote a day in the exam period.

Let \( D \) denote the total number of exam days.

Let \( t \) be a timeslot on any day \( d \).

Let \( e \) denote the number of exams that have been assigned to a timeslot \( t \).

Let \( r \) denote the number of rooms that are available to hold exams during \( t \).

To be able to distribute courses to timeslots correctly, \( e < t \) must be true and to be able to find rooms for all exams \( e < r \) must also be true.

The process of distributing the sets of courses to rooms can be represented in 3D as in Fig 7.

The exam days \( (d) \) are mapped along the \( Y \)-axis. Each exam day \( d \) consists of \( n \) timeslots, along the \( X \)-axis. This work is restricted to a special case of 2 timeslots per day.

Each set of courses will be mapped to a single timeslot-multiple room combination along the \( Z \)-axis. This work uses the max number of rooms available as a limit for group sizes.

Each set of non-conflicting exams (represented by a single color) can be assigned to one timeslot along the \( Z \)-axis. Each exam-room-timeslot combination will occupy one unit/piece in the 3D structure of Fig 7.

![Room Allocation Diagram](image)

Fig 7. Room Allocation

3. RESULTS - EFFECT OF CONFLICT MATRIX TYPE AND SORTING

Using our graph coloring algorithm, we constructed a conflict graph with 357 vertices/courses. A C# application was then written to produce the colored groups from the 3 types of conflict matrices above. The group sizes (number of courses) for 100 and 20 rooms are shown in table 1 and table 2 with the corresponding graphs in Fig 8 and Fig 9 respectively.

![Graph 8](image)

Fig. 8. Groups for 100 Rooms

Table 1. Comparative Group Size per Conflict Matrix Type – 100 Rooms available

<table>
<thead>
<tr>
<th>Group</th>
<th>Standard</th>
<th>Conflict Weight Desc</th>
<th>Conflict Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>53</td>
<td>37</td>
<td>58</td>
</tr>
<tr>
<td>Group 2</td>
<td>49</td>
<td>34</td>
<td>50</td>
</tr>
<tr>
<td>Group 3</td>
<td>37</td>
<td>37</td>
<td>43</td>
</tr>
<tr>
<td>Group 4</td>
<td>26</td>
<td>26</td>
<td>28</td>
</tr>
<tr>
<td>Group 5</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Group 6</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Group 7</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

![Graph 9](image)

Fig. 9. Groups for 20 Rooms

Table 2. Comparative Group Size per Conflict Matrix Type – 20 Rooms available

<table>
<thead>
<tr>
<th>Group</th>
<th>Standard</th>
<th>Conflict Weight Desc</th>
<th>Conflict Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Group 2</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Group 3</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Group 4</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Group 5</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Group 6</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
5. CONCLUSION AND FURTHER WORK

This paper presented an algorithm for ordering, sorting, grouping and scheduling time-space allocations for course exams using data provided by PMU for the Fall semester of 2013. Conflict matrices were used as input for the coloring algorithm in order to produce colored groups of compatible courses. Matrices with different sorting were analyzed to select the optimum approach.

This work has shown that the optimum approach for scheduling is to select courses in Descending Conflict-Weight order. For the 2-slots restriction, above, we propose that in addition to the two (G1 & G2) groups be mapped to two slots of the day, a third group (G3) of courses can be mapped to the 3rd slot such that none of courses within G3 forms a Complete Graph [14, 15] with any of the elements of G1 and G2. Such a group can be used to fill the 3rd timeslot and so on. This idea can be extrapolated for other timeslots as well. A model, formal algorithm and example will be attempted in future work. For the Instructor Conflict constraint, our work above treats these in a similar way to the standard Conflicts, meaning that two courses that have no student conflict other than the instructor conflict will look for all intents and purposes like two that share at least one student. No attempt or algorithm have been attempted that work otherwise.

APPENDIX A

Pseudo Code for the New Graph Coloring Algorithm is shown below

Let N be the number of exams;
Let CM[i][j] be the exams conflict matrix where i ∈ [1,...,N], j ∈ [1,...,N]
Let D be the number of exam days
Let d_i be an exam day, where a ∈ [1,...,D]
Let T be available timeslots per day
Let t_s be a timeslot, where s ∈ [1,...,T]
Let R be the number of Rooms
Let r_t is timeslot where f ∈ [1,...,R]
Let k is the number of color and set k=0
Let loc is the number of exams per group and set loc=0
Let CG[i] be a colors matrix

1. For each c_i in CM:
   a. If G[c_i][c_j] = 0 where i ≠ j //
      i. If c_j is already colored.
         Go to step 2
      ii. If course c_i not conflict the
          courses in group CG[k]
         Add c_i to CG[k][loc] // add the
          course to the group
         Increment loc
         If loc = R // reach the maximum
          number of exams in one TS
          Increment k
          // create new color group
         Go to step 1
   b. If j <= N // find not conflicted and not
      colored course
      Continue step 2
Else
   Increment k

The combination of the input data we used and our algorithm indicate that when the number of rooms is 100, the maximum number of rooms required is always below 59. When the number of rooms is restricted to 20, all of them will be required. It is our conclusion that the best approach for scheduling is one that starts with a Descending Conflict-Weight matrix, as this yields the minimum number of colored groups and group size combination, as illustrated in table 3.

Table 3 – Summary Info for 100 & 20 Rooms

<table>
<thead>
<tr>
<th>Conflict Matrix</th>
<th>100 Rooms</th>
<th>20 Rooms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of Groups</td>
<td>Max Group Size</td>
</tr>
<tr>
<td>Standard</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>Conflict-Weight</td>
<td>26</td>
<td>58</td>
</tr>
<tr>
<td>Conflict-Weight Desc</td>
<td>20</td>
<td>37</td>
</tr>
</tbody>
</table>

4. RESTRICTIONS & ASSUMPTIONS

The algorithm proposed in this paper and the results from it assume a limit of two exam slots per day. Although the number of rooms available for exams is set to a maximum limit of 100, the results show that the optimum configuration yields a requirement that is satisfied by 40 rooms.

The instructor conflict is not catered for as an autonomous factor; the resolution for this is overlooked for now, to be covered in the future work. Having said that, we do propose a shortcut solution to this constraint; see the Further Work section below.

Finally, although we have been working with the aforementioned conflict matrices, there are other ways of sorting and grouping that have not been explored yet, and those shall be attempted in future work.

Fig. 9. Groups for 20 Rooms
// create new color
group
Go to step 1

// assign courses to rooms
3. For each d, in D:
4. For each t, in T:
5. For each G_k in CG:
   If (t < 2) // maximum 2 exam per a day
      Assign G_k to t // assign G_k courses to
   rooms
      Remove G_k from CG
   Else
      Find group with courses not conflicted
      with the conflicted courses of the
      selected grouped in day d,
      If (group is found)
         Assign G_k to t // assign group G_k
      to timeslot t,
      Remove G_k from CG
   Else
      Go to step 3
   If (Selected grouped < MAX_TIME_SLOT)
      Go to step 5
   Else
      Go to step 4

REFERENCES

[1] D. Brelaz
1979 New Methods to Color the Vertices of a Graph Comm. A.C.M. 7, 494-498.

1964 Final Examination Scheduling Comm. A.C.M. 7, 494-498.

1986 A Survey of Practical Applications of Examination Timetabling Algorithms
OR Practice 34, 193-202.


1981 A New Graph Coloring Algorithm Comp. Jnl. 24, 85-86.

1989 A Las Vegas Coloring Algorithm Comp. Jnl. 32, 474-476.


2011, Trans Genetic Coloring Approach for Timetabling Problem
IJCA Special Issue on “Artificial Intelligence Techniques - Novel Approaches & Practical Applications” AIT, 2011

1995 A Survey of Automated Timetabling.
Technical Report CS-R9567,

1993 A university timetabling system based on graph coloring and constraint manipulation

[13] Complete Graph definition: Wikipedia, the free encyclopedia
Link valid on 25 March 2014
http://en.wikipedia.org/wiki/Complete_graph