



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Better Resource Utilization in Exam Scheduling using Graph Coloring

Meena Bharti¹, Navdeep Lata², Manish Goyal³

¹Department of Computer Engineering, Thapar University, Patiala, Punjab, India

²Department of Computer Engineering, MIMIT Malout, Punjab, India

³Department of Computer Engineering University College Ghudda, Bathinda, Punjab, India

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, first with student can have exam of two or more subjects consecutively and in second, student can't have exam of two or more subjects consecutively. In this paper it is attempted to do comparison between two ways to schedule exam based on graph coloring approach for both cases mentioned above.

Keywords: Exam scheduling, Time table, Graph coloring, Resource utilization, Constraints

I. Introduction

Time table problem is basically a resource allocation problem. It consists of various set of subjects which are needed to allocate according to availability of students, teacher or instructor and classrooms. Time table is a scheduling to do work or can say is set of slots to do particular work. It is a combinatorial problem consisting of large set of constraints to satisfy. Time table problem has three sub parts [1]:

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

There is difference between these three types of time table as in exam scheduling student has to give one exam of each subject. Moreover there is no constraint of teachers that which teacher is present in which class. While in time table of schools and universities there are lectures of more than one subject in a week and particular teacher can teach a particular subject. Also there is difference between time table of school and university like in case of schools every class contains almost same number of students and have students of one class study same subjects while in college or universities number of students can vary from few to hundred and above and also students of same class can study different subjects.

Exam scheduling is part of time table problem. In case of exam scheduling the exams are scheduled around the slots. There can be one or more slots in a day. In some cases exam of two subjects of same student can be in adjacent slots like in it can be in the case of internal exams while in case of external exams in most of cases exam of two subjects of a same student can't be at adjacent. The exams should be completed in minimum slots.

In this paper exam scheduling, problem is solved using graph coloring method. Graph coloring is a method to color the graph with different colors such that no two nodes got same color. Graph coloring is basically used to find the dependency list. The lists we got by graph coloring are dependent on each other and within a list elements are independent of each other. In graph coloring, the nodes represent the subjects we need to color or we need to schedule and edges represent the conflicts i.e. if there is at least one student common in two subjects there will be edge in those subjects. Thus an edge between two subjects shows that we can't schedule those two subjects at same time.

In 1994 algorithm based on graph coloring was given [2] in which coloring of graph was done and list was picked and allocated room according to the number of room. If number of students is more, then some subjects were removed from graph and left subjects were added on graph again and graph is colored again starting with subjects left to schedule in previous session.

In 2002, various directions of automated time tabling were provided [3]. List of various constraints, sequential methods, cluster methods, constraint based methods and meta-heuristic methods were explained.

In 2006, universal method for time table construction, based on evolutionary approach, was explained [4]. Two levels of solution were provided. In first level solution is find out on basis of evolutionary method based on given set of parameters and in second level those parameters were established.

In 2006, alternative type of course conflict graph and method to color it was provided [5]. Although the course conflict graph was complicated but it provided room assignment as well as slot assignment.

In 2007, comparison between genetic algorithm and evolutionary programming was done [6] for cutting stock problem. Conclusion was provided that group based GA is better than order based GA. It was observed that operators used in EP are simpler than that of GA but replacement strategy used in GA is better. So combination of both methods could be used to provide better solution.

In 2008, a new exam scheduling algorithm, using graph coloring, was provided [7] in which subjects were picked according to their degree and subject with highest degree was scheduled first. While picking next exam, first the proper slot, where exam can be scheduled, was found by checking, if there was any exam in that slot already such that those subject's dependant exam should not be dependant of exam to be scheduled.

II. Understanding Exam Scheduling [1]

Exam scheduling problem contains some input values; there are some constraints that are needed of follow to get the desired output. The input, desired output and constraints are listed in this section.

A. Notations

Here are some notations used in this paper. $\mathbb{B}$ denotes set of all Booleans, i.e. $\mathbb{B} := \{\text{true}, \text{false}\}$, 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.

B. Input

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

Definition 2.2.1. **(Time Input)**. The pair $\Gamma_T = \langle T, BLOCK \rangle$ 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 $t_1, t_2 \in T$ 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 2.2.2. **(Student Input)**. The pair of tuple $\Gamma_U = \langle U, CON \rangle$ defines the student input, where

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

Next input denotes number of students in each subject.

Definition 2.2.3. **(Subject Input)**. The pair of tuple $\Gamma_S = \langle S, U_s \rangle$ defines the subject input where

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

Each room contains particular number of seats.

Definition 2.2.4. **(Room Input)**. The four tuples $\Gamma_R = \langle R, N, A, N_A \rangle$ defines the room input where

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

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 2.2.5. **(Conflict Input)**. The pair of tuple $\Gamma_C = \langle S, S_s \rangle$ defines the subject conflict input where

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

C. Decision variables

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

Definition 2.3.1 **(Time slot assignment)**. The time slot assignment contains assignment of subjects to slots $t_s: S \rightarrow T$.

D. 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 2.4.1. (Subject constraint). If a subject $s \in S$ has exam in time slot $t \in T$, it implies that every student holding that subject has to give exam on that time slot.

$$T_s(s) := \{t \in T \mid t_s(s) = t\}$$

holds if $s_1 \neq s_2 \Rightarrow t_s(s_1) \neq t_s(s_2)$ for all subjects $s_1, s_2 \in S'$, 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 2.4.2. (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 $s \in S$ and $N_A(r)$ returns number of seats in room r . The seats constraint hold is

$$N_A(R) := \{\exists s \in S'' \mid \forall r \in R \mid \sum N_A(r) \leq \sum N_U(s)\}$$

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

Definition 2.4.3. (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

$$BLOCK_U(u) := \{\exists t_1, t_2 \in T_U(u) \mid CON_U(false) \Rightarrow BLOCK(t_1, t_2) = false\}$$

III. Algorithm

Algorithm is divided in 4 parts:

1. **Color Graph:** 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.
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.

Suppose there is a graph G with vertices x, y . Now x and y can be merged into a single vertex if there exists a vertex, 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]:

$$\begin{aligned} &(x, z_{(1,1)}, y_1) \\ &(x, z_{(1,2)}, y_1) \\ &(x, z_{(1,m1)}, y_1) \\ &\dots \\ &(x, z_{(i,1)}, y_i) \\ &(x, z_{(i,m1)}, y_i) \\ &\dots \\ &\dots \\ &(x, z_{(n,mn)}, y_n) \end{aligned}$$

This algorithm is based on Dutton and Brigham's [8] algorithm and another algorithm by Tehrani [9].

Color 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. $Construct\ all\ the\ triples\ as\ above\ and\ find\ y. If\ no\ such\ vertex\ exists\ (i.e.\ m_i = 0) choose\ a\ vertex\ of\ maximal\ degree\ non - adjacent\ to\ v_j.$
 4. $Merge\ v_j\ and\ x\ into\ v_j\ and\ repeat\ from\ 3\ until\ no\ choice\ is\ possible\ then\ start\ again\ from\ 2\ with\ H \leftarrow H - \{v_j\}, j \leftarrow j + 1.$
 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.

Remove subjects from list starting from subject with minimum number of students

Input: $S_1, S_2, S_3, \dots, S_n$ is list of subjects in increasing order according to number of students, $r_1, r_2, r_3, \dots, r_m$ 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. $Let\ TSeats \leftarrow \sum_{i=1}^m r_i, TStudents \leftarrow \sum_{j=1}^n S_j.studetns.$
 2. $Let\ difference \leftarrow TStudents - TSeats.$
 3. $Repeat\ for\ i\ while\ difference \geq 0$
 - a. $If\ S_i.degree \neq maxdegree\ then$
 $difference = difference - S_i.studetns\ and\ remove\ S_i\ from\ list.$
-

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.

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

Input: $s_1, s_2, s_3, \dots, s_n$ is list of subjects in increasing order according to number of students, $r_1, r_2, r_3, \dots, r_m$ 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. $Let\ TSeats \leftarrow \sum_{i=1}^m r_i, TStudents \leftarrow \sum_{j=1}^n S_j.studetns.$
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*
 - a. *If $S_i.degree == maxdegree$ then*
 - b. *difference = difference - $S_{i-1}.studentns$ and remove S_{i-1} from list .*
 - c. *Otherwise, difference = difference - $S_i.studentns$ and remove S_i from list.*
6. *if $S_i > difference$ then*
 - a. *if $S_i.degree == maxdegree$ then*
 - b. *difference = difference - $S_{i-1}.studentns$ and remove S_{i-1} from list .*
 - c. *Otherwise, difference = difference - $S_i.studentns$ and remove S_i from list.*

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.

Adjust

Input: $s_1, s_2, s_3, \dots, s_n$ 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 sub such that, $sub \notin S$ or $sub \notin alist$.*
4. *if $sub.studentns \leq leftSeats$ then*
 - a. *$S = \{S\} \cup sub$.*
 - b. *$leftSeats = leftSeats - sub.studentns$*
5. *if $leftSeats \neq 0$ repeat from step 1.*

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 Consecutive

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.

Non-consecutive

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. *$Nlist = \{TotalSlist\} - \{\{Slist\} \cup \{alist\}\}$.*

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.

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 1.

	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

Table 1: Different categories

CatA contains remove subjects with equal or greater number of students with non consecutive exams; CatB contains remove subjects starting with minimum number of subjects with non consecutive exams; CatC contains remove subjects with equal or greater number of students and with consecutive exams and CatD contains remove subjects starting with minimum number of subjects with consecutive exams.

IV. Conclusion

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. 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.

References

- [1] Willemen Robertus J., "School timetable construction Algorithms and complexity", IPA Dissertation Series, 2002 – 05.
- [2] 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.
- [3] 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.
- [4] Norberciak Maciej, "Universal Method for Timetable Construction based on Evolutionary Approach", World Academy of Science, Engineering and Technology, Vol. 15, pp 91-96, 2006.
- [5] Timothy A. Redl, "University Timetabling via Graph Coloring: An Alternative Approach", Congressus Numerantium, Vol. 187, pp. 175 – 184, 2006.
- [6] 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.
- [7] Malkawi Mohammad, Hassan Mohammad A., and Hassan Osama A., "A New Exam Scheduling Algorithm Using Graph Coloring", The International Arab Journal of Information Technology, Vol. 5, No. 1, pp 80 – 87, 2008.
- [8] Dotton R.D. and Brigham R.C. "A New Graph Coloring Algorithm" Comp. Jnl., Vol 24, No. 1, pp 85 – 86, 1981.
- [9] Tehrani A., "Un Algorithme de Coloration", Cahiers du centre d'études de Recherche Opérationnelle, Vol 17, No. 2 – 3 – 4, pp 395 – 398, 1975.