A New Approach to Generate Time Table

Prof. Swapna Borde*, Ms. Ekta Shah**, Ms. Priti Rawat***, Ms. Vinaya Patil****

(Department of Computer engineering, MUMBAI University, India
*Email: swapnaborde@yahoo.com
**Email: shah.ekta13@gmail.com
***Email: priti.vcet@gmail.com
****Email: vinaya.vcet@gmail.com

ABSTRACT

The annual construction of a timetable is a common problem for all institutions of higher education. In this paper, we present a hybrid algorithm for University Timetabling of courses which is combination of if else Algorithm and Graph Coloring Algorithm. First algorithm is based on simple if else statement which can be used very easily in any programming language and second algorithm is based on connected graphs. This hybrid algorithm is being introduced to overcome individual method's disadvantages and provide more efficient time table generating algorithm.

Keywords-**Hybrid algorithm, time table, graph coloring, scheduling, if-else ladder, practical, lecture**.

1. INTRODUCTION

The time-tabling problem (TTP) is basically the scheduling and assignment of the lessons into appropriate time-slots and resources respectively, without causing time clashes for the students and the teachers, as well as the resource clashes[1,2]. The drawing up of university timetables is a slow, laborious task, performed by people working on the strength of their knowledge of resources and constraints of a specific institution. Generating university time table is a tedious job with lots of constraints to be satisfied and different requirements by different departments or universities thus generating time table is being considered as a complex problem. Traditionally, the problem was solved manually, but it is likely to miss far better solutions. These uncertainties have motivated for the scientific study of the problem, and to develop an automated solution technique for it. Over the past few years, a wide array of techniques have been proposed for solving the course timetabling problem and it's variants. Several techniques have been developed for automated timetable generation. These techniques include using neural networks, graph coloring method, simulated annealing, constraint based programming, evolutionary algorithms etc. These algorithms are well feasible and effective. In this paper, we propose a method which uses the combination of two methods to reduce the complexity of individual algorithms.

2. THE TIMETABLING PROBLEM

Timetabling is the allocation, subject to constraints, of given resources to objects in space-time domain to satisfy a set of desirable objectives as nearly as possible. Particularly, the university timetabling problem for courses can be viewed as fixing in time and space a sequence of meetings between instructors and students, while simultaneously satisfying a number of various essential conditions or constraints. The time tabling problem can be modeled as a constraint satisfaction problem with many parameters and loose constraints. There are basically two constraints: hard constraints and soft constraints. These constraints have to be modeled in a format that can be handled efficiently by the scheduling algorithm. In order to obtain feasible time table all the hard constraints needs to be satisfied but in order to obtain a high quality time table all the soft constraints also needs to be satisfied. These hard and soft constraints are based on the various requirements of teachers, students and resources.

The hard constraints: These are constraints that must be satisfied in order to obtain a feasible timetable. The hard constraints in this problem model can be viewed from 3 different perspectives as follows[1]:

Student's perspective - All my lessons must not have any time slot clashes.

Resource manager's perspective - No two or more rooms could use a lecture hall or tutorial room during a common time slot.

Teacher's perspective - I cannot teach more than one lesson during any time slot.

The soft constraints: These are constraints of lower priorities to be satisfied. The violation of the soft constraints will not cause the timetable to lose its feasibility. The soft constraints in this problem model can be viewed from 2 different perspectives as follows[1]:

Student's perspective - I want my week to be short and I don't want to have too many empty slots in between lessons.

Teacher's perspective - I cannot teach for more than 2 hours continuously.

The objective of scheduling algorithm is to obtain the best time table while satisfying all these constraints.

3. IF-ELSE LADDER

The if-else ladder[4] is common programming construct that is based upon nested ifs. For generating timetable, set of if-else ladders can be used. Mainly such algorithm has 2 major parts:

- Assigning subject practical for all batches of students
- Assigning subject lectures for whole class

Assigning subject practical for all batches

In this part, practical sessions are assigned for each batch of student. The main constraint checked using if-else ladder is that the practical session should have two consecutive hours. Also in a practical slot all batches should have practical i.e. no batch should have free slot when other batches are having their practical. Here each slot is checked whether it is optimized or not. If not that slot is optimized using another if-else ladder. The slot is called optimized if it has practical for all the batches, no repetition of same subject practical in a slot.

Assigning subject lectures to whole class

This is the next step of this algorithm. In this step lectures are assigned. Here for assigned lectures another if-else ladder is used. The assignment of lectures is according to the number of teaching hours given to a subject by the university syllabus. Also while assigning the lectures, care must be taken that the professor of the subject does not have any lecture at that time on any other class as well as the number of lectures assigned to that professor does not exceed the number of lectures that a professor have to take in a week. No professor should have more or less lectures than fixed by the university. For checking this, a separate if-else ladder is used.

Advantages:

- It is simple to implement.
- It satisfies all the constraints.

Disadvantages:

- It requires laborious work
- It is difficult to modify the code

4. GRAPH COLORING ALGORITHM

Other method for implementing timetable generator is using graph coloring algorithm. In this method subjects are considered as vertices and subjects which conflict in any manner are connected to each other using edges.

Graph coloring problem[3] is basically to color all the vertices of the graph in such a way that no two adjacent vertices have the same color. In this problem color of the vertices represent time slot allotted to that subject.

Consider an example in Fig. (1), where subjects of two different years have been considered to make the graph. Time table is to be generated for this two years, here edge between nodes are used to connect subjects which are conflicting in any manner.

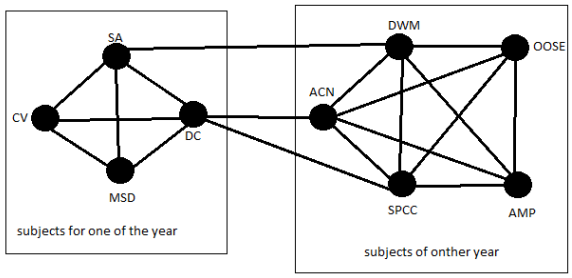

Fig. (1)

In the Table (1) shown below first column represents the time slots of the course and the first row shows the working days of the students. There are 50 slots in the Table(1) shown below. We assign color numbers to the nodes (i.e. to the subjects). Thus one subject will get more than one color, as in a week; there is more than one lecture for a subject.

	Mon	Tues	Wed	Thurs	Fri
9.00-10.00	1	8	15	22	29
10.00-11.00	2	9	16	23	30
11.15-12.15	3	10	17	24	31
12.15-1.15	4	11	18	25	32
2.00-3.00	5	12	19	26	33
3.00-4.00	6	13	20	27	34
4.00-5.00	7	14	21	28	35

Table (1)

In this way using graph coloring algorithm, university timetable can be generated.

Advantages:

- This method is simpler as compared to other methods available.
- This method can be very efficiently used for generating exam time table.

Disadvantages:

- In case of university time table this method does not work well when considering practical session.
- As practical requires two hours session so same color needs to be applied to two adjacent slots which is not possible because it does not satisfies basic constraint of graph coloring problem.

5. HYBRID ALGORITHM

As seen, the above two methods have some shortcomings. In order to overcome them, we propose a combination of the two algorithms: the if-else ladder and the graph coloring method. This hybrid algorithm initially assigns the practical sessions in the timetable using the if-else ladder. We saw in the third section that the if-else ladder implementation method involves two steps: the first, assigning the practical session for each batch and second, assigning theory lectures to a particular class.

In our algorithm, we implement only the first step of assigning practical session to a class using if-else ladder. Generally in Mumbai University, practicals are performed by batch of students and in a class, there are four batches which perform different practical in the same time slot. As explained in earlier section, we implement the if-else ladder for assigning practical. It is a two step process. In first step the generator assigns subject practicals randomly in any time slit considering all the constraints.

As we can see not all time slots are not optimized properly. Hence this output is again given to another set of if-else ladder for optimization. Result of initially generated time table after optimization.

Once we have assigned the practical to all batches, then we can design a graph which includes nodes for the lectures as well as practical for a particular subject, and thus for all the subjects. A sample graph is shown in Fig (2).

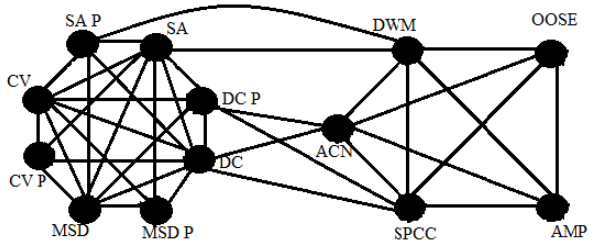

Fig (2).

In Fig. (1), we designed a graph for all the subjects for two different classes. To that graph we have added the practical of each subject as shown in Fig(2). In the Fig(2), we have shown the practical session (CV P indicates CV practical and so on) for only one class to reduce complexity. According to the graph coloring algorithm, we connect those nodes which conflict in some way, like the same professor who conducts a practical session for a class might take a theory lecture in another class, so these two nodes conflict and are thus connected. We then use the graph coloring algorithm to assign the subjects as explained in the fourth section.

The implementation is done as follows: Firstly when the practical sessions are assigned using the if-else ladder, a timetable is generated in which only practical are assigned i.e. only those slots are filled. Consider that on Monday, from 9.00 a.m. to 11.00 a.m., we have a practical session of four different subjects to four different batches. Now, according to Table(1), the color 1 and 2 are assigned to all the practical nodes and similarly, all those slots which are already filled by practical sessions are assigned to the respective nodes. Then the remaining slots are assigned using the graph coloring algorithm as explained in the fourth section. Finally, all the slots are filled and the timetable is generated.

Advantages:

- This method is simple to implement.
- It overcomes the limitations of the earlier discussed two methods.

Disadvantages:

- It is not flexible i.e. it is difficult to modify the code if there are any changes in the database.
- Modification can be done by technical experts only.

6. IMPLEMENTATION AND RESULTS

We implemented the first part of our new approach i.e. the step in which the practical session is assigned. After implementing the first step, we get an output timetable with only practical sessions assigned.

In Table(2), we show the output that is obtained when implementing the if-else ladder. Here we can see that the practical sessions are not assigned in an optimized way. Not all practical sessions are assigned in the same time slot.

	Monday	Tuesday	Wednes	Thursday	Friday
9-10	ACN, SEMI SPCC, AMP	----------	----------	SPCC, AMP, SEMI, DWM	OOSE----
10-11	ACN, SEMI SPCC, AMP	----------	----------	SPCC, AMP, SEMI, DWM	OOSE----
11.15 12.15	----------	OOSE, DWM SPCC, SEMI	----------	----------	-------
12.15 -1.15	----------	OOSE, DWM SPCC, SEMI	----------	----------	-------
	B	R	E	A	K
2-3	----------	----------	----------	---------	---------
3-4	OOSE, SPCC SEMI, DWM	ACN----- ,AMP--------	----------	ACN---------	OOSE, AMP DWM, ACN
4-5	OOSE, SPCC SEMI, DWM	ACN----- ,AMP--------	----------	ACN---------	OOSE, AMP DWM, AcN

Table(2)

Thus the given timetable is optimized using conditions and then the optimized timetable with practical sessions assigned is as shown in Table(3)

	Monday	Tuesday	Wednes	Thursday	Friday
9-10	ACN, SEMI, SPCC, AMP	----------	----------	ACN, AMP, SEMI, DWM	---------
10-11	ACN, SEMI, SPCC, AMP	----------	----------	ACN,AMP, SEMI, DWM	---------
11.15 - 12.15	----------	OOSE, DWM, SPCC, SEMI	----------	----------	---------
12.15 -1.15	----------	OOSE, DWM, SPCC, SEMI	----------	----------	---------
	B	R	E	A	K
2-3	----------	----------	----------	----------	---------
3-4	OOSE, SPCC SEMI, DWM	ACN, OOSE, AMP, SPCC	----------	----------	OOSE, AMP DWM, ACN
4-5	OOSE, SPCC SEMI, DWM	ACN, OOSE, AMP, SPCC	----------	----------	OOSE, AMP DWM, AcN

Table(3)

CONCLUSION

In this paper, we discussed the timetabling problem and the various constraints that are needed to be considered while generating a timetable. This work is usually done manually, but there are some techniques with which timetabling problem can be solved. Here, we have discussed two such techniques. Both the techniques (if-else ladder method and the graph coloring method) are efficient in many ways. We have implemented the if-else method. Baki Koyuncu, Mahmut Secir,[3] have implemented timetable generation system using graph coloring algorithm. But both these techniques have some limitations. Thus, we have come up with the idea of a hybrid algorithm in order to generate a timetable. This method combines the two techniques and thus overcomes the disadvantage of the individual methods.

FUTURE SCOPE

The constraint regarding the number of consecutive lectures given to a professor is not addressed in this algorithm. This can be included using a new set of if-else ladder. Also in big universities, sometimes there are fewer classrooms

but more subjects or students have more than two elective subjects. For this, single class requires more than 2 classrooms, 1 for each elective subject at the same time. In this case, for generating timetable, the algorithm needs to consider the shared classrooms as well. This constraint is not addressed in this algorithm. This can be done using graph coloring algorithm by generating a graph for classrooms.

REFERENCES

Journal Papers:

[1] Dipti Srinivasan, Tian Hou Seow, Jian Xin Xu, Automated Time Table Generation Using Multiple Context Reasoning for University Modules, *IEEE 0-7803-7282-4/02 2002.*

[2] De Werra D, An Intoduction to TimeTabling, *European Journal Of Operation Research, Volume 19/1985, pp 151-162.*

[3] Baki Koyuncu, Mahmut Secir, Student TimeTable Generator using Graph Coloring Algorithm.

[4] Patrick Naughton and Herbert Schildt, *Java 2: The Complete Reference (*Tata McGraw Hill).