International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Syllabus and Timetable Generation System

Anuj Kinge[1], Aarti Ahluwalia[2], Hrithik P.B.[3]

[1,2,3]*Student, Dept of CSE, MIT School of Engineering, Pune, Maharashtra, India*

Abstract - *Time table generation is a tedious job for educational institutions concerning time and manpower. Providing an automatic timetable generator will help to generate a much-sophisticated timetable automatically in no time. The proposed system, "Syllabus and Timetable Generation System" will make the process of timetable generation fast and effective. Most educational institutions have resorted to the manual generation of their timetables which according to statistics takes much time to get completed and optimal. Even at the optimal stage of the manually generated timetable, there are still a few clashes and it is the lecturer that takes a clashing course that works out the logistics of the course to avoid the clash. Therefore, there is a great requirement for an application to distribute the course evenly and without collisions. A simple, easily understandable, efficient, and portable application is developed, which could automatically generate good quality timetables along with their syllabus within seconds.*

Key Words: Timetable Generation, Syllabus Generation, Faculty Scheduling, Recursion

1. INTRODUCTION

Time table generation is a complex task having to consider factors that keep changing from time to time. A system is developed which takes into the matter some of the factors which are intended to take into account.

The Syllabus and Timetable Generation System takes in essential information by the user to generate a syllabus and timetable that can be for multiple classes without any clashes. The class teacher is also allocated randomly to any class without any teacher being the class teacher of more than one class.

With the help of the Syllabus and Timetable Generation System according to the no. of lectures required in one day and the number of days, it should be conducted in a week can be controlled by the user and accordingly a timetable can be generated. The system is user-friendly and easy to use. It makes it easier for the user to get the required timetable by not asking the user to input the subjects every time the user needs to generate a timetable. It has a modern design – it displays the table generated exactly in the same way we do it on the paper i.e. in a tabular form. The system generates the timetable in just one click saving so much time the user invested in making the timetable manually. The timetable can be created for every class and the most interesting part, it is different and has not too much overlap.

2. LITERATURE SURVEY

To date, there had been various scheduling techniques and algorithms for solving timetable problems. Some of the techniques such as Simulated annealing (SA), Tabu Search, Genetic Algorithm (GA) were reviewed to solve these problems [3]. Genetic algorithm has been used a lot in the scheduling techniques because of its power of heuristics. Tabu Search which was proposed by Fred Glover in 1986 shows significant improvements when it is compared to a custom timetable [5]. Before Tabu Search, the Genetic algorithm happens to be the fastest algorithm for problem scheduling problems. However, when Tabu search was proposed it was observed that it is way faster than the Genetic algorithm. By using the technique of Backtracking, and Evolutionary algorithm, it has been extrapolated that almost all of the constraints were satisfied [4]. Markov chain Monte Carlo (MCMC) used along with Backtracking developed a solution for general university timetable scheduling problems [2]. The genetic parameters used for the automation of timetable scheduling gained positive results in the feasibility of applying an Evolutionary Algorithm to solve the timetable scheduling problem after running it 100 times on an AMD Athlon K7 550Mhz PC with 256MB of [1].

3. PROPOSED SYSTEM

In our proposed system, as per fig. 1. which is a use case diagram, there are two actors, actor 1 - a teacher(who is going to schedule the timetable) and actor 2 - a student. Actor 1 has all facilities such as deciding attributes, creating and viewing timetables, and syllabus, and modifying timetable whereas actor 2 has only two facilities, viewing timetables and syllabus.

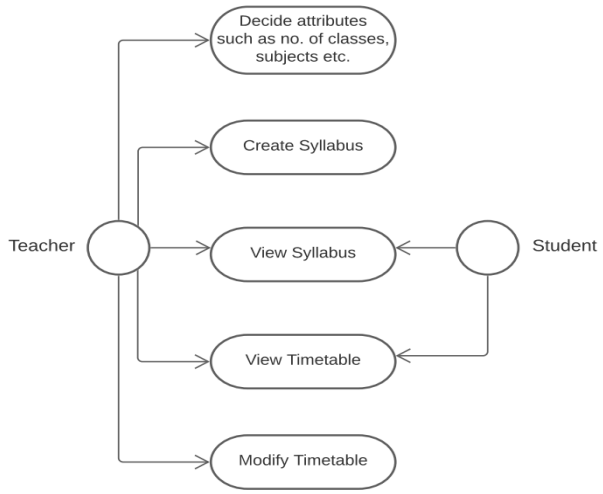

Fig -1: Use Case Diagram

The proposed system consists of a login page wherein teachers or timetable planners can log in to the application and enter the necessary details such as department name, institution name, number of classes, and number of subjects. In addition to this, they can enter information related to the syllabus such as credits or marks, subject names, and objectives and outcomes of every subject. As in fig. 2. we can see that after entering all this information, the teacher can get the print of the syllabus as well as the generated timetable.

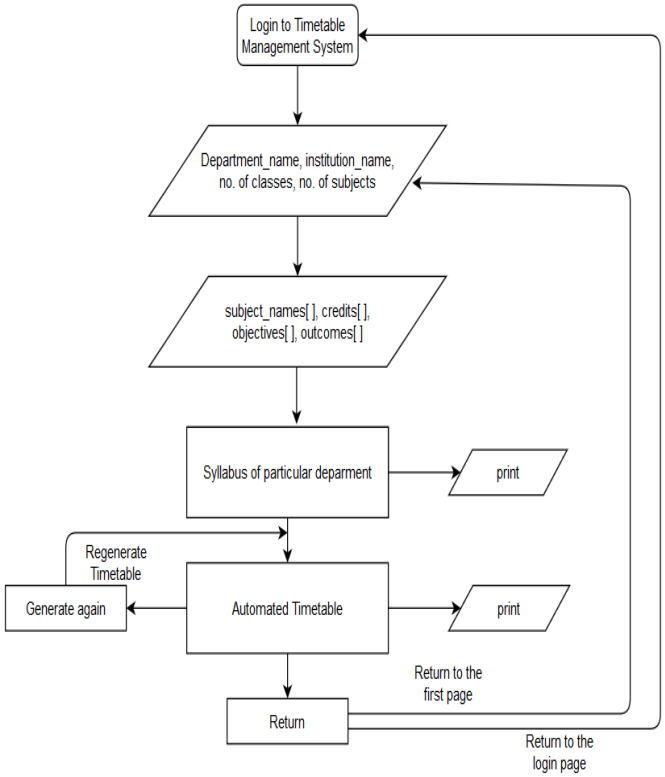

Fig -2: Proposed System

3.1 Algorithm

Step 1:

Our algorithm takes in the following attributes as input,

c: No. of classes,
x: No. of subjects,
w: No. of working days in a week,
l: No. of lectures in a working day,
f: No. of faculties for every subject,

So, the total no. of lectures in a week = w x l.

The appearance of a subject in a week means how many times a subject should appear throughout a week.

Thus, Σ Appearance of subjects = w x l.

Step 2:

After taking input from Step 1, the Class Timetable is generated using the below algorithm,

```
public class GeneratorClass {
 String[][] class1(int w, int l, int x, int[] ap, String[] subjects,
String[][] S) {
 int k,m;
 String y;
 Random rd = new Random();
 for (int i = 0; i < w; i++) {
 for (int j = 0; j < l; j++) {
 while(S[i][j] == null){
 m = rd.nextInt(subjects.length);
 y = subjects[m];
 for (k = 0; k < x; k++) {
 if (y.equals(subjects[k]) && ap[k] != 0) {
 S[i][j] = y;
 ap[k]--;
 break;
 }
 }
 }
 }
 }
 return S;
   }
```

where,
w: No. of working days in a week,
l: No. of lectures in a working day,

ap: Array of appearances of a particular subject,
subjects: Array of subjects,
S: Generated timetable.

Step 3:

The output of Step 2 is considered as input of Step 3 and here recursion takes place to prevent the clashes of the classes as shown in the below code,

```
if(S[i][j].equals(Sn[i][j])){
 generateClass(w, d, x, ap, subjects, Sn, S);
 }
```

where,
S: Timetable of the class generated in Step 2,
Sn: Timetable of other class generated without clash.

Step 4:

In Step 4, faculties are assigned through the Static Faculty Assignment.
f: no. of faculties for a particular subject.

When f = 1, i.e there is only one faculty for the subject, that faculty will be directly assigned to that subject.

When f = 2 or 3 i.e when a subject has 2 or 3 faculties, indexes are given to the faculties starting from 0,
(i)0,1 when f = 2
(ii) 0,1,2 when f = 3
A random integer is generated in the range of 0 to 1 when f = 2 and 0 to 2 when f = 3.

Assume,
Faculty at index 0 is selected then again random integer is to be chosen from 1 and 2. This process goes on until all faculties are assigned to the subject.

3.2 Illustration

Let us consider,
c(no. of classes) = 4,
x(no. of subjects) = 10,
w(no. of working days in a week) = 5,
l(no. of lectures in a working day) = 8,
f(no. of faculties for every subject) = 3,
and A,B,C,D,E,F,G,H,I,J as the subjects.

So, the total no. of lectures in a week (L) = w x l = 40.

Now, these inputs are given to the algorithm where recursion takes place and timetables are generated.

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture 1	J	J	J	G	J
Lecture 2	H	F	I	I	E
Lecture 3	F	H	H	A	D
Lecture 4	C	B	D	E	B
Lecture 5	H	D	A	B	E
Lecture 6	C	A	H	I	D
Lecture 7	G	I	G	A	E
Lecture 8	B	H	G	B	B

Fig -3: Class 1 timetable

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture 1	A	I	B	D	G
Lecture 2	E	C	A	E	H
Lecture 3	B	D	G	G	I
Lecture 4	B	F	A	D	H
Lecture 5	D	A	J	J	I
Lecture 6	F	B	C	E	H
Lecture 7	B	B	J	J	H
Lecture 8	H	G	H	E	I

Fig -4: Class 2 timetable

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture 1	D	H	G	B	I
Lecture 2	B	J	E	I	A
Lecture 3	H	H	H	A	G
Lecture 4	J	E	E	I	D
Lecture 5	H	B	E	D	C
Lecture 6	I	J	G	B	J
Lecture 7	F	G	H	F	A
Lecture 8	B	C	B	D	A

Fig -5: Class 3 timetable

	Day 1	Day 2	Day 3	Day 4	Day 5
Lecture 1	F	B	E	I	D
Lecture 2	E	A	F	B	J
Lecture 3	A	G	G	D	J
Lecture 4	D	B	A	H	B
Lecture 5	C	G	B	H	G
Lecture 6	H	C	J	E	H
Lecture 7	A	I	D	E	H
Lecture 8	J	I	I	B	H

Fig -6: Class 4 timetable

	Faculty for Class 1	Faculty for Class 2	Faculty for Class 3	Faculty for Class 4
Subjects	-	-	-	-
A	A3	A2	A2	A1
B	B2	B3	B3	B1
C	C3	C2	C2	C1
D	D3	D1	D1	D2
E	E2	E1	E1	E3
F	F1	F2	F2	F3
G	G2	G1	G1	G3
H	H2	H1	H1	H3
I	I1	I2	I2	I3
J	J3	J2	J2	J1

Fig -7: Faculty Assignment

After giving these inputs, we got the timetables where clashes do not occur as shown in fig. 3, fig. 4, fig. 5, and fig. 6. The faculties are assigned by the Static Faculty Assignment as shown in fig. 7.

This algorithm ensures that there are no clashes in the timetables of different classes. Wherever there is a clash of a subject, a different faculty is assigned. Hence, this algorithm helps us ease the task of scheduling.

4. CONCLUSION

We used our own algorithms along with recursion for generating multiple timetables for multiple classes. Although recursion seems a bit obsolete approach for generating timetables, it is still a good way to use it as it eases the complex tasks of scheduling problems. Moreover, recursion plays a significant role in our algorithm. Using this, we can generate timetables without clashes of the subjects. Though there are a lot of Timetable generators already which might be much more effective in their way, our Syllabus and Timetable Generator System is custom made for the needs of our department and the faculty. Future Enhancements in our proposed work can be considered by taking more factors into account which can help in deciding a much more effective and more towards the real-life situations like rescheduling of lectures due to the absence of the faculty, availability of classrooms, automatic notifications which can be sent to the faculty and the students of the new schedule or change in the timings.

REFERENCES

[1] Srinivasan, D. & Seow, Tian & Xu, Jian-Xin. (2002). Automated time table generation using multiple context reasoning for university modules. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002.2. 1751 - 1756. 10.1109/CEC.2002.1004507. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[2] Badoni, Rakesh & Gupta, D. & Lenka, Anil. (2013). A new approach for university timetabling problems. International Journal of Mathematics in Operational Research. 6. 10.1504/IJMOR.2014.059537.

[3] A, Parkavi. (2018). A STUDY ON AUTOMATIC TIMETABLE GENERATOR. International Journal of Innovative Research & Growth. May.

[4] S.Alagu Lakshmi, N.Durga, S.Sujitha, TIME TABLE AUTOMATION SYSTEM USING BACKTRACKING TECHNIQUE, International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST) Vol. 2, Special Issue 15, March 2016.

[5] Deeksha C S, A Kavya Reddy, Automatic Timetable Generation System, Journal of Emerging Technologies and Innovative Research (JETIR) April 2015, Volume 2, Issue 4.