International Journal of Advance Research in Science and Engineering
Volume No.07, Issue No.03, March 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Schedule Organizer for Educational Institutes

P. Surendra Varma[1], S. Sushma[2]

[1]Asst.Professor,Dept of CSE,Raghu Engineering College,Andhra Pradesh,(India)

[2]Asst.Professor,Dept of IT,Aditya Engineering College,

Surampalem,Andhra Pradesh,(India)

ABSTRACT

Generating a schedule for every course in educational institutes involves a lot of work, as number of permutations that need to be tested are high. Manually carrying out such works involves a lot of overhead and in some cases, clashes among subjects are almost impossible to be prevented due to various constraints.

So, an automated scheduler would reduce this overhead by verifying every possibility of generating a clash free time table.

I.INTRODUCTION

The basic idea behind generating an automated scheduler is that comparisons among two entities can be done by a machine easily and it would be more efficient than a human doing the same work.

Here, the two entities are the time table that is being generated and already existing time table. We perform comparisons between these two entities so that there would not be any risk of clash if same faculty were allocated to both old and new time tables.

So we take care such that if there is an entry “a” at position “x” in existing time table, then there would be the same entry “a” in newly generating time table at any position other than “x”. “a” cannot be at the same position in both the schedules.

In this way we try to create schedules for multiple sections of the same course where faculty can be same among different sections but there would not be any clash among two subjects of same class nor same subject for different classes.

II.EXISTING SYSTEM

At present schedules are being made by using pen and paper, where a person has to manually check all the possibilities of placing a subject at a particular time slot for a day where comparisons are very high.

Some Institutes also use spread sheets for allocating schedules, which is in a way better than doing it on a paper, but still all the comparisons need to be performed manually, the only advantage of this method is that a change can be undone easily, whereas the number of permutations are still the same.

III.DISADVANTAGES

The number of comparisons are very high, For example if we consider a simple time table with n days and x hours per day. We need to make a minimum number of n*x comparisons and this complexity increases linearly with increase in number of sections. Every subject needs to be maintained a count so that all subjects get equal number of classes per week.

Educational institutions often have various constraints such as utilization of labs, equal work distribution for faculty, and more number of hours for subjects which have more syllabus etc.

Managing all these constraints manually is an extra job done, staff could concentrate on improving their skills or could impart knowledge in students rather than wasting time on such documentation works.

IV.PROPOSED SYSTEM

The proposal mainly concentrates on a clash free schedule for multiple classes. The software would generate time tables based on the number of classes to be conducted per week for each subject. The same subject would not be allocated for two different sections on the same time period of same day.

This would be saving much time and effort for staff and management, because schedules are carried generated automatically, so the staff and management could use their time to work on self-improvement rather than wasting their time on such documentation works. Carrying out the same allocation using a pen and paper could have been a very complex task especially when schedules are made for multiple sections at once.

V.IMPLEMENTATION

The proposed system can be implement by following the below steps:

1. Getting data as per the requirement of scheduler. Read the following variables:

- No. of Theory Subjects into s.
- No. of Lab Subjects into l.
- No. of classes per each day into n.
- No. of sections to be allocated into c.
- A double dimensional array schedule[i][j] indicates the schedule for each section where i represents day of the week and j represents the class number in a particular day.
- Create a Double dimensional array sc[][] to maintain subject count of each subject.

2. Generate a Random Number in between 1 and l (No. of labs), check whether 3 continuous entries are free starting from this generated number (An entry is a value in schedule[i][j], it is said to be free if nothing is allocated). Here we are checking for three continuous entries because we assume a lab to be conducted for 3 classes.

3. After allocating labs, again generate a random number between 1 and s (No. of subjects), check whether the entry is free at generated number.

- If we find a free entry, we allocate a subject there.

International Journal of Advance Research in Science and Engineering
Volume No.07, Issue No.03, March 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

- If we find an allocated entry, we repeat step 3.

4. After allocating a lab or subject, increment the value of subject count in array sc[][] to maintain count of subjects allocated so that we ensure all subjects have equal number of classes per week.

5. Subjects and Labs are allocated this way, before every allocation, check whether schedule[i][j] is already filled at the generated number and after every allocation, increment value of subject count.

6. For n number of sections, repeat steps 2 to 5 n number of times to generate n schedule[i][j] arrays which are time tables of n sections respectively.

VI.TEST RESULTS

1. A screenshot of schedule for one section:

Section1	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Mon	Subject2	Subject5	Lab3	Lab3	Lab3	Subject6	Subject3
Tue	Subject1	Subject6	Subject2	Subject5	Subject3	Misc.	Subject1
Wed	Subject6	Lab2	Lab2	Lab2	Subject1	Misc.	Subject5
Thu	Subject3	Subject1	Subject6	Subject5	Subject2	Subject6	Subject3
Fri	Subject2	Subject5	Subject3	Subject1	Lab1	Lab1	Lab1
Sat	Subject6	Subject3	Subject1	Subject5	Subject2	Misc.	Subject2

2. Screenshot of schedule for second section:

Section2	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Mon	Subject3	Subject6	Subject1	Subject2	Lab1	Lab1	Lab1
Tue	Subject2	Subject5	Subject1	Subject1	Subject6	Subject5	Subject2
Wed	Subject3	Subject2	Subject6	Subject3	Subject5	Misc.	Subject3
Thu	Subject6	Subject2	Subject6	Lab2	Lab2	Lab2	Misc.
Fri	Subject5	Subject6	Subject1	Subject2	Subject5	Subject3	Misc.
Sat	Lab3	Lab3	Lab3	Subject3	Subject1	Subject5	Subject1

3. Screenshot of schedule for both the sections to compare for any clashes:

Section1	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Mon	Subject2	Subject5	Lab3	Lab3	Lab3	Subject6	Subject3
Tue	Subject1	Subject6	Subject2	Subject5	Subject3	Misc.	Subject1
Wed	Subject6	Lab2	Lab2	Lab2	Subject1	Misc.	Subject5
Thu	Subject3	Subject1	Subject6	Subject5	Subject2	Subject6	Subject3
Fri	Subject2	Subject5	Subject3	Subject1	Lab1	Lab1	Lab1
Sat	Subject6	Subject3	Subject1	Subject5	Subject2	Misc.	Subject2

Section2	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Mon	Subject3	Subject6	Subject1	Subject2	Lab1	Lab1	Lab1
Tue	Subject2	Subject5	Subject1	Subject1	Subject6	Subject5	Subject2
Wed	Subject3	Subject2	Subject6	Subject3	Subject5	Misc.	Subject3
Thu	Subject6	Subject2	Subject6	Lab2	Lab2	Lab2	Misc.
Fri	Subject5	Subject6	Subject1	Subject2	Subject5	Subject3	Misc.
Sat	Lab3	Lab3	Lab3	Subject3	Subject1	Subject5	Subject1

VII.SCOPE FOR EXTENSION

- The proposed algorithm searches each time slot one by one in a linear fashion, which would increase time complexity. Using an efficient searching technique could reduce the time complexity
- To allocate a subject we are using a random number between 1 and N(Number of subjects), a procedural approach instead of a random number would make it better.
- This scheduler assumes that same subject is dealt by same faculty for any number of section. This has been done to avoid clashes but its efficiency can still be improved in a case where same subject is allocated among different teachers.
- This approach might not be suitable best for generating schedules for 3 or more classes at once, which can be improved.

REFERENCES

[1.] ProblemSolvingandProgramDesigninC,Hanly,Koffman,7th ed,PERSON

[2.] IntroductiontoAlgorithms,secondedition,T.H.Cormen,C.E.Leiserson,R.L.RivestandC.Stein,PHIPvt.Ltd.

[3.] CbyExample,NoelKalicharan,Cambridge

[4.] ClassicDataStructures,2/e,Debasis,Samanta,PHI,2009

[5.] DataStructuresusingC,ReemaThareja,Oxford

[6.] TheCprogrammingLanguagebyDennisRichieandBrianKernighan

[7.] ProgrammingwithC,Bichkar,UniversitiesPress

[8.] ProgramminginC,SecondEditionPradipDeyandManasGhosh,OXFORDHigherEducation

[9.] Datastructuresandalgorithmanalysisin C,2nd ed,markallenweiss

[10.] FundamentalsofDataStructureinC,2/e,Horowitz,Sahni,AndersonFreed,UniversityPrees

[11.] IntroductiontotheDesignandAnalysisofAlgorithms,AnanyLevitin,PEA

[12.] AlgorithmDesign,Foundation,AnalysisandinternetExamples,MichelTGoodrich,RobertoTamassia,Wiley.

[13.] FundamentalsofComputerAlgorithms,EllisHorowitz,SatrajSahniandRajasekharam,UniversitiesPress.

[14.] https://www.youtube.com/watch?v=yRM3sc57q0c&list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V

[15.] https://www.youtube.com/watch?v=YWnBbNj_G-U

[16.] https://www.youtube.com/watch?v=gOKVwRIyWdg

[17.] https://www.youtube.com/watch?v=ENWVRcMGDoU