The Possible Slot Algorithm

Mayank Keoliya
Documentation

---

1. Key Terms Used

a. Class/Grade - an collection of students reading at the same grade (example - class/grade six)

b. Section - a subdivision of a class signified by an alphabet (example - 6A,6B, or 12C,12D and so on)

c. Period - a time slot ranging from 35 to 60 minutes. Typically, 8 such periods make up a given school day. Each period is defined by the subject taught, teacher conducting and the section taught

d. Double Period - 2 consecutive periods of the same subject conducted by the same teacher

e. Mixed Period - a period where students of two or more sections are split up into groups(depending on language or other subject-choice) and taught by different teachers

f. Possible Period Timetable - a depiction of the periods that will be taught to a class on a particular day, **in no particular order** and in accordance with the required number of periods per week for each subject

g. No-Constraint Period - a period where the given subject can be taught or conducted by any of a given selection of teachers (example - General Knowledge,Art or Library)

2. Motivation

Formulating timetable schedules in India is among the most challenging,repetitive and vital functions in schools in a country where there is a severe paucity of skilled teachers, class sizes exceeding capacity and a lack of basic infrastructure to cope with the same. It is primarily accomplished **manually** in the country, either through the use of chart-papers or spreadsheet applications like Microsoft Excel. Further, constraints like a lack of zero periods (where no teacher is assigned to a class), a rigid requirement of periods per subject per week mandated by curriculum and an emphasis of classes on mathematics and the sciences lead to a unique problem not particularly well served by existing algorithms. Further, the constraints are often incredibly complex in nature - and a timetable which fails to satisfy even one of them is redundant.

A salient feature of the Indian school system is the classroom and student-centric approach - it is the students who are assigned a classroom and teachers who are assigned to the students. There is also the additional constraint of "class-teachers" assigned to each class - who take primary charge of students and usually teach the first period of the day. Due to a diversity in teacher qualifications, allotment of subject teachers is also rigid (for example,a teacher without a Masters' in his/her chosen subject is not permitted under government mandate to teach in certain classes for the senior secondary school).

Further, most scheduler programs lack transparency in run-time with respect to the end-user, with

the possibility of the resulting timetable diverging from a viable option for the given school. There also exists the problem of assigning of no-constraint periods (which any of several teachers can teach) and the additional information that the class teacher and various subject teachers of a class are pre-defined by the user.

Finally, it is often found that 70-80 percent of periods assigned are single teacher, single class and single classroom (i.e there is no subdivision into groups or splitting up of a class). It is often viable in such a case to assign certain double-periods or split-periods directly and immutably, with the other periods being allotted using the PS Algorithm.

3. Description of the Problem and Constraints

The problem of generating a timetable can be stated thus : -

Given a list of : -

a. *All teachers employed in the school*

b. *All classes and classrooms*

c. *Subjects taught to individual classes and their number per week*

d. *Subject Teachers for individual classes*

e. *No-Constraint periods*

f. *Mixed Periods and their predefined slots*

g. *Additional Constraints*

a timetable for the high school is to be designed which is to be used in a weekly cycle, and generated every year according to changing parameters

The constraints used for the test-project are : -

a. 8 periods per day [1]

b. 5 days per week (Mon-Fri) [2]

c. Maximum of 28 periods/week for any Teacher

d. Maximum of 6 periods/day for any Teacher

e. Maximum of 3 consecutive periods (recess included) for any Teacher

f. Special Constraint - A Teacher (Srabani Mondal) could only teach on Mondays and Wednesdays

g. Special Constraint - Music could be held only in the last period

h. Special Constraint - Mathematics cannot be held in the last period

4. System Information and Software Used

[1] Denotes the parameter can be changed according to input
[2] Denotes the parameter can be changed according to input

(a) Dell Inspiron,Intel Core i3 with 1.90 GB RAM [3]

(b) Operating System - Windows 10 (64-bit)

(c) JDK Version - 1.8.0.60

(d) IDE - NetBeans IDE 8.2 (Custom install - no additional plugins installed except local history for code backup)

(e) Microsoft Office 2007 - Microsoft Excel (Macro-Enabled)

(f) Additional Library - ApachePOI 4.0.0

5. Basic Algorithm and Snippets

(a) Our first objective and priority is to satisfy constraint of each class, and fill each day up with the list of 8 subjects to be taught that day in no particular order (called "Possible Slots") hence making each day's timetable independent of the rest while finding permutation/combinations. This simplifies production of 10^6 timetables in 2 minutes, excluding time for I/O operations.Note that care is taken to ensure that periods inserted into the timetable manually using Excel (hard-coded) are unchanged, and the program runs dynamically in this aspect.

(b) Class 1 : Section - contains static arrays and other information regarding possible slots, final slots, number of periods to be taught, list of teachers teaching the class,etc.

(c) Class 2 : Teacher - contains Section objects and static arrays regarding possibke slots, the classes the teacher is supposed to teach, and intermediate ArrayLists used for calculations in periods left to be assigned per iteration.

(d) Class 3 : TimeTable - the main, public class from where main() is called. A TimeTable object has dynamic number of teachers,classes,etc. Calls **build** functions to store data in required objects, and then calls the **buildTimeTable** function to generate as many timetables as needed until a given day has no clashes, and is perfectly scheduled. [4]

(e) Another important function is the buildPossibleSlots and doThePossibleSlotCalculation function, which is the backbone of the process used to make a day's functioning independent from other days. Rudimentary datatypes like Strings were chosen over ArrayLists and Lists as it provided better time complexity using coded arithmetic operations.[5]

(f) After the possible slots are generated, we keep generating timetables using the same copy of possible slots until each day has been resolved perfectly. [6]

6. Flow of Work - The initial Excel files and coding formulae in VB was started on 28th April, 2018 and during the summer (May-July) the Java process for automation was initiated, with

[3]The lack of availability of adequate RAM did not affect testing, as maximum file sizes tended to be 22KB
[4]See Appendix A
[5]See Appendix B
[6]See Appendix C and SchoolTimeTable.xlsm in this folder

approximately 3-4 hours of coding each day in this period. Documentation was started in October 2018 and finished in late November 2018

7. **Future Endeavours/Goals** - Future goals include addition of features such as a rich GUI (Graphical User Interface) using Swing and substituting Excel for a native .exe application, as well as incorporating more constraints automatically (such as double-periods) into the program, generating informational charts of time-usage by subject and by day, and finally to release the packaged software as open-source.

```
public void buildTimeTable(int size){
    int minimum[]=new int[5];
    for(int i=0;i<5;i++)
        minimum[i]=99;

    ArrayList teacherList=new ArrayList<Integer>();
    Random random=new Random();
    double total=0;
    for(int mega=0;mega<size;mega++){
        if(minimum[0]==0 && minimum[1]==0 && minimum[2]==0 && minimum[3]==0 &&
            minimum[4]==0)
            return;
        for(int day=0;day<5;day++){
        TimeTable tt=new TimeTable();
        tt.swap(this);
        tt.sum=0;
        int count=0;

        for(int i=1;i<this.teachers.length-2;i++)
            teacherList.add(i);

        tt.doIniDay(tt.teachers[52],day);
            //tt.doLastDay(tt.teachers[54],day);
            //tt.doLastDay(tt.teachers[53],day);
        tt.doIniDay(tt.teachers[23],day);
        tt.doIniDay(tt.teachers[25],day);
        tt.doIniDay(tt.teachers[32],day);
        tt.doIniDay(tt.teachers[34],day);
        tt.doIniDay(tt.teachers[45],day);
        tt.doIniDay(tt.teachers[46],day);
        tt.doIniDay(tt.teachers[50],day);


        for(int i=1;i<this.teachers.length-2;i++){
            int teacherNum= random.nextInt(53-i) + 1;
            if(teacherNum>=teacherList.size())
                teacherNum=teacherList.size()-1;
            teacherNum=(int)teacherList.get(teacherNum);
            teacherList.remove(new Integer(teacherNum));
            if(i%2==0){
                tt.doIniDay(tt.teachers[teacherNum], day);
                count++;
            }
            else
```

```
                tt.doLastDay(tt.teachers[teacherNum],day);
            }

            for(int i=1;i<tt.teachers.length-2;i++)
                tt.sum = tt.sum + tt.teachers[i].remainingSubjects[day].size();

            if(tt.sum<minimum[day]){
               minimum[day]=tt.sum;
               System.out.println(days[day] + ":" + mega + ":" + minimum[day]);
            }
             if(tt.sum==0){
                 minimum[day]=0;
                 for(int i=0;i<8;i++){
                     for(int t=1;t<this.teachers.length;t++)
                       this.teachers[t].teacherSlots[day][i]=tt.teachers[t].teacherSlots[day][i];
                     for(int t=1;t<this.sections.length;t++)
                         this.sections[t].sectionSlots[day][i]=tt.sections[t].sectionSlots[day][i];
                 }
            }
          }
        }
    }
```

```
public void buildPossibleSlots(){
    for(int i=33;i>=1;i--){
        for(int j=0;j<sections[i].subjectTeacher1.length;j++){
            Teacher teacher = getTeacherObject(sections[i].subjectTeacher1[j]);
            int periodsToAllot = sections[i].remainingPeriods[j];
            doThePossibleSlotCalculation(i,sections[i],teacher,periodsToAllot,sections[i].subjects[j]);
        }
    }
}

public void doThePossibleSlotCalculation(int sectionNo,Section section, Teacher teacher,
     int periodsToAllot,String subjectName){

    String daysSort[]=new String[5],combo;
    int maxPeriods=6;
    int avail=5;
    if(teacher.teacherName.equals("Art") || teacher.teacherName.equals("GK") )
        maxPeriods=3;
    if(teacher.teacherName.equals("Library"))
        maxPeriods=8;
    for(int i=0;i<5;i++){
        if(alreadyDone(subjectName,section,i)){
            avail--;
            daysSort[i]="Error : Already allotted today";
            continue;
        }
      daysSort[i]=section.noOfPossiblePeriods[i]+ "" + teacher.noOfPossiblePeriods[i] + i;
      if(teacher.noOfPossiblePeriods[i]==maxPeriods){
          daysSort[i]="Error : 6 periods for teacher exceeded";
          avail--;
      }
      else{
      if(i!=4){
          if(section.noOfPossiblePeriods[i]==8){
              daysSort[i]="Error : 8 periods for class exceeded";
              avail--;
          }
      }
      else{
          if(sectionNo<26){
              if(section.noOfPossiblePeriods[i]==8){
                  daysSort[i]="Error : 8 periods for class exceeded";
                  avail--;
```

```
                }
            }
            else{
                if(section.noOfPossiblePeriods[i]==5){
                    daysSort[i]="Error : 5 periods for class exceeded";
                    avail--;
                }
            }
        }
        }
    }

    if(avail<periodsToAllot){
        System.out.println("You");
        System.out.println(section.sectionName + ":" + teacher.teacherName + ":" +
            subjectName);
    }

    Arrays.sort(daysSort);


    for(int i=0;i<periodsToAllot;i++){
        if(daysSort[i].length()<4){
            int day=daysSort[i].charAt(2)-48;

            int sectionIndex=section.noOfPossiblePeriods[day];//section index is the
                current slot where possible period is to be inserted
            section.noOfPossiblePeriods[day]++; //one more period added
            section.possibleSlots[day][sectionIndex++]=subjectName;

            int teacherIndex=teacher.noOfPossiblePeriods[day];
            teacher.noOfPossiblePeriods[day]++;//one more period added
            teacher.possibleSlots[day][teacherIndex]=section.sectionName + "," +
                subjectName;
            teacher.remainingSubjects[day].add(subjectName);
            teacher.remainingSections[day].add(section.sectionName);
        }
        else{
            //System.out.println("Imperfect Generation);
        }
    }
}
```

Appendix C : Output Profiler

```
                    Monday:0:8 // the format of "day:iteration number:number of conflicts"
                        is followed, 0 conflicts indicate the day has been scheduled
                        perfectly.
Tuesday:0:7
Wednesday:0:7
Thursday:0:3
Friday:0:7
Monday:1:7
Wednesday:1:5
Tuesday:4:5
Wednesday:4:4
Monday:5:6
Wednesday:5:1
Friday:8:6
Friday:11:5
Monday:12:4
Tuesday:13:4
Friday:15:4
Tuesday:50:2
Thursday:75:2
Monday:81:2
Friday:174:3
Thursday:175:1
Tuesday:190:1
Friday:512:2
Wednesday:773:0 // Wednesday has been scheduled
Friday:1985:0 // Friday has been scheduled
Monday:2236:1
Tuesday:7666:0 // Tuesday has been scheduled
Thursday:11640:0 // Thursday has been scheduled
Monday:25760:0 // Monday has been scheduled
```