DOI: http://dx.doi.org/10.26483/ijarcs.v14i1.6943

ISSN No. 0976-5697

Volume 14, No. 1, January-February 2023

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

ALGORITHM TO GENERATE GLOBAL ACADEMIC CLASS SCHEDULE

Md. Mizanur Rahman
Department of Arts and Sciences,
Bangladesh Army University of Science and Technology,
Saidpur, Bangladesh

Abstract: This paper describes an algorithm to generate global academic class schedule of an educational institution. In this method course teachers are assigned to courses manually. Rooms are also assigned with a further option which provides a searching technique for available rooms. This algorithm also considers the restrictions or conditions that define a specific period or a specific room for a particular teacher or for a particular course or for a particular batch/section. Depending on all assignments, restrictions and conditions global academic class schedule is generated automatically by this algorithm.

Keywords*:* Scheduling Algorithm, Timetabling, Global Routine Table.

1. INTRODUCTION

Generating of global class schedule in an educational institutionis one of the major troublesome tasks where inter departmental courses aremanaged,that means, teacher of one department takes courses of another department.It is very difficult to make the class routine of all the departmental courses within a department as well as to make the routine of the courses of other departments. It was seen that when a department gave a class to a teacher of another department, that teacher had a class in his own department or in another department. As a result, routines often overlap. This is why, it is important to create a global routine through computer programming so that no overlap occur.There are some algorithms to generate timetable of classes in academic institution [1-4]. This algorithm is general and useful in any case and widely described.

2. MATERIALS AND METHODS

Every program requires some variables and functions according to the task performed by the program. In this program, some array variables are used for different objects. These variables with dimension and symbol used for and corresponding objects' names are described in the following table.

Table 1

List of variables used in this algorithm

Object Name	Variable Name (Identifier)	Dimension
Teacher	T	1
Section	S	1
Room	R	1
Course	C	2
Assigned Course	AC	3
Teacher-Students Facing	TSF	3
Global Routine Table	GRT	3

The proposed method is described as follows-

Teacher (T)
Teachers are denoted by

$$T_i(i = 1,2,3,\cdots,TN)$$

TN is an integer that describes total no. of teachers. As example,

$$T_1 = \text{"MR"};$$
$$T_2 = \text{"NA"};$$
$$T_3 = \text{"DMR"};$$
$$\vdots$$

Section (S)
A group of students who seat in a class together at a time. Sections are denoted by

$$S_i(i = 1,2,3,\cdots,SN)$$

SN is an integer that describes total no. of sections. As example,

$S_1 = "CSE - 1st - A"$;(Dept. of Computer Science & Engineering, 1st Batch, Section A)

$S_2 = "CSE - 1st - B"$; (Dept. of Computer Science & Engineering, 1st Batch, Section B)

$S_3 = "EEE - 1st - A"$; (Dept. of Electrical &Electronic Engineering, 1st Batch, Section A)

⋮

Room (R)

Rooms are denoted by

$$R_i (i = 1,2,3, \cdots, RN)$$

RN is an integer that describes total no. of rooms. As example,

$$R_1 = "1001" ;$$
$$R_2 = "1002" ;$$
$$R_3 = "Circuit\ Lab\ 01" ;$$
$$\vdots$$

Course (C)

Courses are denoted by

$$C_{i,j} (i = 1,2,3, \cdots, CN; \quad j = 1,2,3)$$

$CN$is an integer that describes total no. of courses. First index i indicates the course and second index j indicates its attributes. Index $j = 1$ describes course code (CC), $j = 2$ describes course hour (CH) that means, how many periods or hours are taken in a week for the course and $j = 3$ describes course length (CL) that means, how many periods are contiguous.

Course Code (CC)

A string by which a course is identified. It also may be called Course ID. As examples, "$MATH$ 1101".

Course Hour (CH)

Course hour is an integer that describes how many periods or hours are taken for the course in a week.

Course Length (CL)

Course length of a course is an integer that describes how many hours or periods are contiguous that means how many hours or periods are taken at a time.

A course with course hour 3 and course length 1 means there are 3 different classes with length 1 hour or, 1 period for the course over week. Most of the theory classes are of this type. A course with course hour 3 and course length 3 means there is a single class with length 3 hours or, periods for the course over week. Most of the lab classes are of this type.

As example,

$C_{1,1} =$ "CSE101", $\quad C_{1,2} = 3, \quad C_{1,3} = 1;$
$(CC: CSE101, \quad CH: 3, \quad CL: 1)$

$C_{2,1} =$ "CSE103", $\quad C_{2,2} = 3, \quad C_{2,3} = 1;$
$(CC: CSE103, \quad CH: 3, \quad CL: 1)$

$C_{3,1} =$ "EEE102", $\quad C_{3,2} = 3, \quad C_{3,3} = 3;$
$(CC: EEE102, \quad CH: 3, \quad CL: 3)$

$\vdots$

There are three types of courses depending on course hour and course length.

1. **Concrete Course:** course length is equal to course hour.
2. **Discrete Course:** course length is less than course hour and greater than 1.
3. **Total Discrete Course:** course length is equal to 1.

In this algorithm, we have considered course length as an integer. But we may consider it as a string as future plan. As Example, $C_{1,1} = "CSE101"$, $C_{1,2} = 3$, $C_{1,3} = "111"$. Here"CSE101" is the code of 1st course whose course hour is 3 and course length is "111" that means the course is taken one hour in 3 different days. Then this course is **Total Discrete Course**. Maximum theory course is of this kind. If course length = "12", then the course is taken one hour in a day and two hours together in a different day. Then this course is **Discrete Course**. If course length = "3", then the course is taken three hours together in a day. Then this course is **Concrete Course**. Maximum lab course is of this kind. Variations in course length of a course with course hour 4 are "1111", "13", "22"or,"4". If course length = "4", then this course is a Concrete Course, if course length = "13" or, "22", then the course is a Discrete Course and if course length = "1111", then the course is Total Discrete Course.

Assigned Course (AC)

Assigned courses are denoted by

$$AC_{i,j}$$
$$(i = 1,2,3, \cdots, ACN; \;\; j = 1,2,3,4,5; \; 5 \; is \; optional)$$

ACN is an integer that describes total no. of assigned courses. First index i indicates the assigned course and second index j indicates its attributes.

We will assign all the courses of all the sections as follows-

$AC_{1,1} = S_i(any\ section);$
$AC_{1,2} = C_i(any\ course\ of\ the\ section);$
$AC_{1,3} = T_i(assigned\ teacher\ of\ the\ course);$
$AC_{1,4} = R_i(assigned\ room)\ or\ \{R_i, R_j, R_k, \cdots\}$
$(list\ of\ available\ rooms);$
$AC_{1,5} = 0\ (for\ anytime)\ or$
$1\ (not\ for\ after\ launch)\ or\ others;$

$AC_{2,1} = S_i(any\ section);$
$AC_{2,2} = C_i(any\ course\ of\ the\ section);$
$AC_{2,3} = T_i(assigned\ teacher\ of\ the\ course);$
$AC_{2,4} = R_i(assigned\ room)\ or\ \{R_i, R_j, R_k, \cdots\}$
$(list\ of\ available\ rooms\ );$
$AC_{2,5} = 0\ (for\ anytime)\ or$
$1\ (not\ for\ after\ launch)\ or\ others;$

$\vdots$

Since, C is a two-dimensional array. So, $AC_{i,2,1}$ is course code of the assigned course. In this way AC is a three-dimensional array. When $AC_{i,5} = 1$, then this course will not be taken after launch. When $AC_{i,5} = 0$, then this course may be taken any time. This attribute is optional. We can define values of this optional attribute according to conditions and restrictions of the institution.

As example,

$$AC_{1,1} = S_1 ;$$
$$AC_{1,2} = C_1 ;$$
$$AC_{1,3} = T_3 ;$$
$$AC_{1,4} = R_2 ;$$
$$AC_{1,5} = 0 ;$$

$$AC_{2,1} = S_1 ;$$
$$AC_{2,2} = C_2 ;$$
$$AC_{2,3} = T_1 ;$$
$$AC_{2,4} = \{R_1, R_2\} ;$$
$$AC_{2,5} = 0 ;$$

$$AC_{3,1} = S_1 ;$$
$$AC_{3,2} = C_3 ;$$
$$AC_{3,3} = T_2 ;$$
$$AC_{3,4} = AllTheoryRooms;$$
$$AC_{3,5} = 1 ;$$
$$\vdots$$

$AllTheoryRooms, SecondFloor, FirstFloor$ etc. are list of all theory room's name or number, list of all theory rooms in the 2nd floor, list of all theory rooms in the 1st floor respectively.

Example: $AllTheoryRooms = \{R_1, R_2, R_3, \cdots\}$

In the routine generated algorithm, serially all rooms of this list from first to last are checked. If a room of the list is available then the room is selected for the course. If no room of this list is available then no room for the course is allotted.

Teacher-Students Facing (TSF)

Teacher-students facing means when course teacher and students are faced/meet in a theory class or lab class. As example, for a lab course a teacher meets students one time in a week which continues 3 periods. But for a theory course a teacher meets students 3/4 times in a week according to credit of the course and takes one period at a time only. Suppose, in a section there are 3 theory courses with course hour 3 and course length 1 and 2 lab courses with course hour 3 and course length 3. So, there are 5 assigned courses in this section. But there are $3 \times 3 + 2 \times 1 = 11$ teacher-students facings.

Teacher-students facings are denoted by

$$TSF_{i,j}$$
$$(i = 1,2,3, \cdots, TSFN;\ \ j = 1,2,3,4,5;\ 5\ is\ optional)$$

$TSFN$ is an integer that describes total no. of teacher-students facings. It is almost same as Assigned Course (AC). Manually we assign a teacher with a theory course only. But this theory course has two, three or four classes in a week according to credit of the course. In TSF these classes are generated separately. Actually, this is the unit part or element of the GRT (Global Routine Table) which is described later. TSF are generated by the algorithm described in the next session.

Generating TSF

By following flow chart TSF is generated from AC. To understand the following flow chart, we remind here that $AC_{i,2}$ is an assigned course and $AC_{i,2,2}$ is course hour of the course and $AC_{i,2,3}$ is course length of the course. TSF is generated based on the course hour and course length of the courses. If AC is concrete course, then one TSF is generated form a AC. If AC is discrete or total discrete course, then multiple TSF are generated from the AC. Suppose, course length is 1 and course hour is 3 for an AC. Then 3 $TSFs$ are generated from the AC. In this algorithm, only concrete courses and total discrete courses are considered.

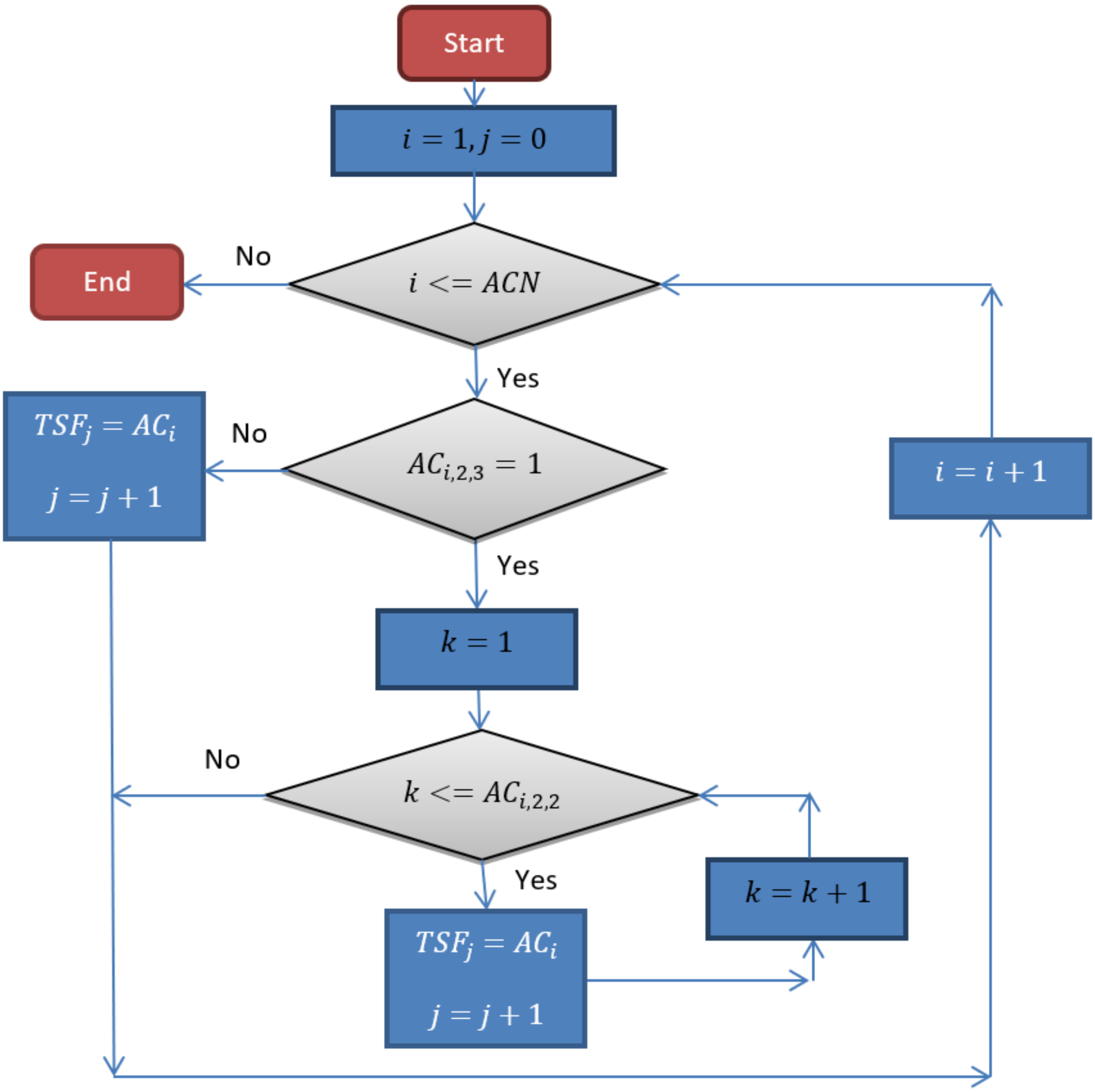

Figure 1:TSF generating flow chart from AC

Global Routine Table (GRT)

Global routine table (GRT) is a 3 dimensional array. Each row indicates a room, each column indicates a weekly period and each cell has 3 layers that means each cell contains threeinformation- teacher, course and section. Here we have assumed that there are five days from Sunday to Thursday and every day there are 9 periods. So, there are 45 periods in a week. So, there are 45 columns in GRT and number of rows in GRT is the number of rooms that means RN. After 3rd period there is a tea break and after 6th period there is a launch break. A TSF will be inserted in a cell.

	Day and Period																						
	Sunday											…	Thursday										
Room	1	2	3	Tea Break	4	5	6	Launch Break	7	8	9	…	1	2	3	Tea Break	4	5	6	Launch Break	7	8	9
R_1												…											
R_2												…											
R_3												…											
⋮	⋮	⋮	⋮		⋮	⋮	⋮		⋮	⋮	⋮	…	⋮	⋮	⋮		⋮	⋮	⋮		⋮	⋮	⋮
R_{RN}												…											

Figure 2: Layout of Global Routine Table

Charactesistics of GRT:

1. Every column of GRT contains a teacher only once. That is, a teacher must not stay in a column twice.
2. Every column of GRT contains a section only once. That is, a section must not stay in a column twice.
3. 9 columns of a day of GRT contains a course of a section only once.

Elements of GRT are denoted by

$$GRT_{i,j,k}$$

$$(i = 1,2,3,\cdots,RN; \quad j = 1,2,3,\cdots,45; \quad k = 1,2,3;)$$

First index i indicates the rows of the table, second index j indicates the columns of the table and third index k indicates information of the cells in the table. Index $k = 1$ describes section, $k = 2$ describes course and $k = 3$ describes teacher.

Besides the array variables there are some variables used in this method. The variables are listed below with their used values

$NotContCourse = 3;$
$HourPerDay = 9;$
$DayPerWeek = 5;$
$HourPerWeek = HourPerDay \times DayPerWeek;$

Inserting Elements of TSF to GRT

Elements of TSF are inserted to GRT serially as they are positioned after shuffling the TSF. To insert elements of TSF to GRT there are five testfunctions-

1. to avoid overlapping of a section (a section should not have a class with two teachers at the same time),
2. to avoid occurrence of a course with a section in a day twice,
3. to avoid overlapping of a teacher (a teacher should not have a class with two sections at the same time),
4. to avoid 3 contiguous classes of a teacher in a triplet,
5. to maintain uniform load distribution ofa teacher over a week.

The test functions are described below-

First three test functionsare mandatory. But the 4th and 5th test functions are optional. 4th and 5thtest functions are used for better routine.

1. $\boldsymbol{SectionTest(s,j)}$

Input: 1st argument is a section s and 2nd argument is a column j of GRT.

Action: This function checks the repetition of the section s in the j^{th} column of GRT. Because, a section cannot be present two different class rooms at the same period.

Output: If repetition occurs, then returns 1. Otherwise, returns 0.

Algorithm:

Start

Step 1: $Section\_Count = 0;$

Step 2: $i = 1;$

Step 3: **Start Loop 1:**$While\ i \leq RN$

$If\ GRT_{i,j,1} = s, then\ Section\_Count++;$

$i++;$

End Loop 1.

Step 4: $If\ Section\ \_Count > 1, then\ return\ 1;$

$else\ return\ 0;$

End.

2. $\boldsymbol{CourseTest(s,c,j)}$

Input: 1st argument is a section s, 2nd argument is a course c and 3rd argument is a column j of GRT.

Action: This function checks the repetition of the course c with the section s in a day containing the j^{th}column. Because, usually a theory course cannot be taken twice in a day with the same section.

Output: If repetition occurs, then returns 1. Otherwise, returns 0.

Algorithm:

Start

Step 1: $Course\_Count = 0;$

Step 2: $day = Ceiling(j/9);$

Step 3: $k = 9 \times (day - 1) + 1;$

Step 4: **Start Loop 1**: $While\ k \leq 9 \times (day - 1) + 9$

$i = 1$;

Start Loop 1.1: $While\ i \leq RN$

$If\ GRT_{i,k,1} = s\ and\ GRT_{i,k,2} = c,$

$then\ Course\_Count++$;

$i++$;

End Loop 1.1.

$k++$;

End Loop 1;

Step 5: $If\ Course\_Count > 1, then\ return\ 1$;

$else\ return\ 0$;

End.

3. $\boldsymbol{TeacherTest(t, j)}$

Input: 1st argument is a teacher t and 2nd argument is a column j of GRT.

Action: This function checks the repetition of the teacher t in the j^{th} column of GRT. Because, a teacher cannot be present two different class rooms at the same period.

Output: If repetition occurs, then returns 1. Otherwise, returns 0.

Algorithm:

Start

Step 1: $Teacher\_Count = 0$;

Step 2: $i = 1$;

Step 3: **Start Loop 1:** $While\ i \leq RN$

$If\ GRT_{i,j,3} = t, then\ Teacher\_Count++$;

$i++$;

End Loop 1.

Step 4: $If\ Teacher\_Count > 1, then\ return\ 1$;

$else\ return\ 0$;

End.

Definition of Triplet: Triplet is the bundle of three colums of GRT those are seperated by tea break, launch break and day break.

4. $\boldsymbol{ContTeacherTest(t, j)}$

Input: 1st argument is a teacher t and 2nd argument is a column j of GRT.

Action: This function checks the number of occurrences of the teacher t in a triplet containing the j^{th} column. If j is the first column of a triplet, then the next two periods or columns along with the j^{th} column will be checked, if j is the middle column of a triplet, then the previous one period and the next one period along with the j^{th} column will be checked and if j is the last column of a triplet, then the previous two periods along with the j^{th} column will be checked to find number of occurrences of the teacher t in the triplet containing the j^{th}column.

Output: If $NotContCourse = 2$ and teacher t has two classes contiguously in a triplet containing j^{th} column, then return 1.

Else if $NotContCourse = 3$ and teacher t has three classes contiguously in a triplet containing j^{th}column,then return 1.

Otherwise, returns 0.

$NotContCourse = 2$ means a teacher will not have 2 theory courses contiguously in a triplet.

$NotContCourse = 3$ means a teacher will not have 3 theory courses contiguously in a triplet.

First triplet consists of first 3 periods before tea break, second triplet consists of 4th, 5th and 6th periods after tea break and before launch break, third triplet consists of 7th, 8th and 9th periods after launch break.

Algorithm:

Start

Step 1: $Teacher\_Count1 = 0$; $Teacher\_Count2 = 0$; $Teacher\_Count3 = 0$;

Step 2:

$If\ j\%3 = 1$ (i.e., when j is the first column of a triplet), then

$i = 1$;

Start Loop A: $While\ i \leq RN$

$If\ GRT_{i,j,3} = t, then\ Teacher\_Count1 = 1$;

$If\ GRT_{i,j+1,3} = t, then\ Teacher\_Count2 = 1$;

$If\ GRT_{i.j+2,3} = t, then\ Teacher\_Count3 = 1$;

$i++$;

End Loop A.

$Else\ if\ j\%3 = 2$ (i.e., when j is the middle column of a triplet), then

$i = 1;$

Start Loop B: $While\ i \le RN$

$If\ GRT_{i,j-1,3} = t, then\ Teacher\_Count1 = 1;$

$If\ GRT_{i,j,3} = t, then\ Teacher\_Count2 = 1;$

$If\ GRT_{i.j+1,3} = t, then\ Teacher\_Count3 = 1;$

$i++;$

End Loop B.

$Else\ if\ j\%3 = 0$ (i.e., when j is the last column of a triplet), then

$i = 1;$

Start Loop C: $While\ i \le RN$

$If\ GRT_{i,j-2,3} = t, then\ Teacher\_Count1 = 1;$

$If\ GRT_{i,j-1,3} = t, then\ Teacher\_Count2 = 1;$

$If\ GRT_{i,j,3} = t, then\ Teacher\_Count3 = 1;$

$i++;$

End Loop C.

Step 3: $If\ NotContCourse = 2\ and$

$$\{(Teacher\_Count1 = 1\ and\ Teacher\_Count2 = 1)\ or\ (Teacher\_Count2 = 1\ and\ Teacher\_Count3 = 1)\}, then\ return\ 1;$$

$Else\ if\ NotContCourse = 3\ and\ (Teacher\_Count1 = 1\ and\ Teacher\_Count2 = 1\ and\ Teacher\_Count3 = 1), then\ return\ 1;$

$else\ return\ 0;$

End.

5. $\boldsymbol{LoadTeacherTest(t, j)}$

Input: 1st argument is a teacher t and 2nd argument is a column j of GRT.

Action: This function checks the daily teacher load of the teacher t. $DailyTeacherLoad$ of a teacher is evaluated by dividing total credit of the teacher by 5 (number of days in a week).

Output: If daily teacher load of the teacher exceeds value of the variable $DailyTeacherLoad$ of the teacher, then returns 1. Otherwise, returns 0.

Algorithm:

Start

Step 1: $Teacher\_Count = 0;$

Step 2: $day = Ceiling(j/9);$

Step 3: $k = 9 \times (day - 1) + 1;$

Start Loop 1: $While\ k \le 9 \times (day - 1) + 9$

$i = 1;$

Start Loop 1.1: $While\ i \le RN$

$If\ GRT_{i,k,3} = t, then\ Teacher\_Count++;$

$i++;$

End Loop 1.1.

$k++;$

End Loop 1.

Step 4: $If\ Teacher\_Count > DailyTeacherLoad\ of\ t, then\ return\ 1;$

$else\ return\ 0;$

End.

Besides the above five test functions another function is used to find room index.

$\boldsymbol{RoomIndex(r)}$

Input: The argument is a room r.

Output: This function returns the index of the room r i.e. no. of row in GRT which contains the room r.

Algorithm:

Start

Step 1: $i = 1;$

Step 2: **Start Loop 1:** $While\ i \le RN$

$If\ R_i = r, then\ return\ i;$

$i++;$

End Loop 1.

End.

The following algorithm to calculate daily teacher load of a teacher for uniform distribution which is used in $LoadTeacherTest(t, j)$ function.

Algorithm:

Start

Consider an array variable named $DailyTeacherLoad$.

Step 1:$i = 1;$

Step 2:Loop 1:$While\ i \leq TN$

$DailyTeacherLoad\ of\ T_i = 0;$

$j = 1;$

Loop 2:$While\ j \leq ACN$

$If\ AC_{j,3} = T_i, then$

$If\ AC_{j,2,3} = 2\ or\ AC_{j,2,3} = 3, then$
$DailyTeacherLoad\ of\ T_i = +3;$

$[a = +b\ \ means\ b\ is\ added\ to\ a.]$

$Else\ if\ AC_{j,2,3} = 1,\ then$
$DailyTeacherLoad\ of\ T_i = +AC_{j,2,2};$

$j + +;$

End of Loop 2.

$i + +;$

End of Loop 1.

$DailyTeacherLoad\ of\ T_i$

$= Ceiling(DailyTeacherLoad\ of\ T_i /\ DayPerWeek);$

End.

Evaluation of total periods or, hours:

$tp = 0;$

$k = 1;$

Start Loop:$While\ k \leq TSFN$

$tp\ =\ tp\ +\ TSF_{k,2,3}\ ;$

k is increased by 1;

End Loop.

Main Algorithm to insert elements of TSF to GRT

Start

Start Loop 1:While all the elements of TSF are not inserted to GRT, i.e., $ip\ \neq\ tp$

Shuffle the array TSF.

$ip = 0;$ [No. of inserted periods to GRT]

Flash the array GRT. That means, remove all data from GRT.

$k = 1;$

Start Loop 2: While $k <= TSFN$

For right alignement of TSF in GRT let $p = 0$.

Else for left alignment of TSF in GRT let $p = 1$.

Else for justify alignment of TSF in GRT let $p = 1 - p$.

[We will use one of the above three alignments. Right alignment means $TSFs$ are aligned from last column of GRT to frist. Left alignment means $TSFs$ are aligned from first column of GRT to last.]

$j = p + (1 - p) \times HourPerWeek;$

Make an array $CourseRoom$ from the list $TSF_{k,4}$.

$$r = 1;$$

Start Loop 3:

Get the row number i of GRT where the room $CourseRoom_r$ exists by the function.

$$i = RoomIndex(CourseRoom_r);$$

If$TSF_{k,2,3} = 1\ and$ (j is within first period and sixth period and $TSF_{k,5} = 1$ or j is within first period and ninth period and $TSF_{k,5} = 0$) and $GRT_{i,j,1}$ is empty, then

$$GRT_{i,j,1} = TSF_{k,1}$$

$$GRT_{i,j,2} = TSF_{k,2,1}$$

$$GRT_{i,j,3} = TSF_{k,3}$$

Now after inserting TSF_k into GRT-

If any of the test functions SectionTest$(TSF_{k,1}, j)$ or, CourseTest $(TSF_{k,1}, TSF_{k,2,1}, j)$ or, TeacherTest $(TSF_{k,3}, j)$ or,ContTeacherTest $(TSF_{k,3}, j)$ or, LoadTeacherTest$(TSF_{k,3}, j)$ returns 1, then erase the inserting TSF_k from GRT.

Otherwise, the insertion of the TSF_k is successful,

$ip = ip + TSF_{k,2,3}$;

and exit **Loop3**.

Else If$(TSF_{k,2,3} = 2\ or, TSF_{k,2,3} = 3)\ and$ (j is within first period and sixth period and $TSF_{k,5} = 1$ or j is within first period and ninth period and $TSF_{k,5} = 0$) and $GRT_{i,j,1}$ is empty and $GRT_{i,j+1,1}$ is empty and $GRT_{i,j+2,1}$is empty and j is first period or fourth period or seventh period, then

$GRT_{i,j,1} = TSF_{k,1}$, $GRT_{i,j,2} = TSF_{k,2,1}$, $GRT_{i,j,3} = TSF_{k,3}$,

$GRT_{i,j+1,1} = TSF_{k,1}$, $GRT_{i,j+1,2} = TSF_{k,2,1}$, $GRT_{i,j+1,3} = TSF_{k,3}$,

$GRT_{i,j+2,1} = TSF_{k,1}$, $GRT_{i,j+2,2} = TSF_{k,2,1}$, $GRT_{i,j+2,3} = TSF_{k,3}$.

Now after inserting TSF_kinto GRT-

If any of the test functions SectionTest$(TSF_{k,1}, j)$ or, SectionTest $(TSF_{k,1}, j+1)$ or, SectionTest $(TSF_{k,1}, j+2)$ or, TeacherTest $(TSF_{k,3}, j)$ or, TeacherTest$(TSF_{k,3}, j+1)$ or, TeacherTest$(TSF_{k,3}, j+2)$ or, LoadTeacherTest$(TSF_{k,3}, j)$ returns 1, then erase the inserting TSF_k from GRT.

Otherwise, the insertion of the TSF_k is successful,

$ip = ip + TSF_{k,2,3}$;

and exit **Loop 3**.

When insertion of the TSF_kin a cell is failed:

If $p = 1$,then j is increased by 1 to insert the TSF in the next cell.

After increasing the value of j if $j > HourPerWeek$, then initialize j i.e. $j = 1$ and r is increased by 1 to check next room of the list $CourseRoom$.

After increasing the value of r if $r > number\ of\ rooms\ in\ the\ listCourseRoom$, then there is no room for the TSF i.e. the TSF is not inserted in GRT. Therefore, this permutation of $TSFs$ is not perfect. So, exit **Loop 3** and **Loop 2** for new permutation of $TSFs$.

Else if $p = 0$, then j is decreased by 1 to insert the TSF in the previous cell.

After decreasing the value of j if $j < 1$, then $j = HourPerWeek$ and r is increased by 1 to check next room of the list $CourseRoom$.

After increasing the value of r if $r > number\ of\ rooms\ in\ the\ list\ CourseRoom$, then there is no room for the TSF i.e. the TSF is not inserted in GRT. Therefore, this permutation of $TSFs$ is not perfect. So, exit **Loop 3** and **Loop 2** for new permutation of $TSFs$.

End of **Loop 3.**

k is increased by 1 for next TSF;

End of **Loop 2.**

Print:ip periods are inserted to GRT out of tp periods \n (new line);

If **Loop 1** is executed for 1000 times or all $TSFs$are inserted to GRT successfully, i.e., $ip = tp$ then exit **Loop 1**.

End of **Loop 1**.

End.

By the above algorithm we can generate GRT. If all $TSFs$ are not inserted within 1000 times then we have to run this program again. We can change this repeatation number (1000) as ourselves. The array TSF is shuffled randomly by the built-in-function of the language used to create program as like as “shuffle” function of php.

From the GRT we can generate all teachers’ individual routine and all sections’ individual routine by the following two algorithms.

The followings array variables are used in the next two algorithms-

$dayhour_1 = 1st$

$dayhour_2 = 2nd$

$dayhour_3 = 3rd$

$dayhour_4 = 4th$

$dayhour_5 = 5th$

$dayhour_6 = 6th$

$dayhour_7 = 7th$

$dayhour_8 = 8th$

$dayhour_9 = 9th$

$day_1 = Sun$

$day_2 = Mon$

$day_3 = Tue$

$day_4 = Wed$

$day_5 = Thu$

Algorithm to generate all teachers' individual routine:

Start

$t = 1;$

Start Loop 1: While $t \leq TN$

$j = 1;$

Start Loop 2: While $j \leq HourPerWeek$

tt_j = ""; [alltt_jvariables are initialized to blank string.]

j is increased by 1;

End Loop 2;

$j = 1;$

Start Loop 3: While $j \leq$ HourPerWeek

$i = 1;$

Start Loop 4: While $i \leq RN$

If $(T_t = GRT_{i,j,3}) tt_j = GRT_{i,j,2} \backslash n \ GRT_{i,j,1} \backslash n \ R_i$

[Here, $\backslash n$ means new line.i represents row and $j$represents column of GRT and a column contains a teacher only once.Each row in a column is checked. If teacher T_t is found in a row then other information of that cell like course no. and section no. along with room no. is assigned to tt_j. And if teacher T_t is not found in any row of that column then tt_jremains blank string.]

i is increased by 1;

End Loop 4.

j is increased by 1;

End Loop 3.

Print Table:

Print table headings by teacher name T_t;

Drawing a table with 6 rows and 10 columns as follows-

T_t

	1st	2nd	3rd	4rt	5th	6th	7th	8th	9th
Sun									
Mon									
Tue									
Wed									
Thu									

[The cell of the above table contains course code, section, room no.]

Print a blank cell;

$j = 1;$

Start Loop 5: While $j \leq 9$

Goto next column;

Print $dayhour_j$;

j is increased by 1;

End loop 5.

$i = 0;$

Start Loop 6: While $i < 5$

Goto next row;

Print day_{i+1};

$j = 1;$

Start Loop 7: While $j \leq 9$

Goto next column;

Print $tt_{i\times 9+j}$;

j is increased by 1;

End Loop 7.

i is increased by 1;

End Loop 6.

t is increased by 1;

End Loop 1.

End.

Algorithm to generate all sections' individual routine:

Start

$t = 1;$

Start Loop 1: While $t \leq SN$

$j = 1;$

Start Loop 2: While $j \leq HourPerWeek$

$tt_j = ""$; [all tt_j variables are initialized to blank string.]

j is increased by 1;

End Loop 2;

$j = 1;$

Start Loop 3: While $j \leq$ HourPerWeek

$i = 1;$

Start Loop 4: While $i \leq RN$

If $(S_t = GRT_{i,j,1}) tt_j = GRT_{i,j,2} \backslash n\ GRT_{i,j,3} \backslash n\ R_i$;

[Here, $\backslash n$ means new line. i represents row and j represents column of GRT and a column contains a section only once. Each row in a column is checked. If section S_t is found in a row then other information of that cell like course no. and teacher along with room no. is assigned to tt_j. And if section S_t is not found in any row of that column then tt_j remains blank string.]

i is increased by 1;

End Loop 4.

j is increased by 1;

End Loop 3.

Print Table:

Print table headings by section name S_t;

Drawing a table with 6 rows and 10 columns as follows-

S_t

	1st	2nd	3rd	4rt	5th	6th	7th	8th	9th
Sun									
Mon									
Tue									
Wed									
Thu									

[The cell of the above table contains course code, teacher, room no.]

Print a blank cell;

$j = 1;$

Start Loop 5: While $j \leq 9$

Goto next column;

Print $dayhour_j$;

j is increased by 1;

End loop 5.

$i = 0;$

Start Loop 6: While $i < 5$

Goto next row;

Print day_{i+1} ;

$j = 1;$

Start Loop 7: While $j \leq 9$

Goto next column ;

Print $tt_{i \times 9 + j}$;

j is increased by 1;

End Loop 7.

i is increased by 1;

End Loop 6.

t is increased by 1;

End Loop 1.

End.

By the above two algorithms we can easily create individual teachers' routine and individual sections' routine at the same time of creating $\boldsymbol{GRT}$.

3. RESULTS AND DISCUSSIONS

Using the above algorithm,we have successfully created global academic routine in BAUST with52 teachers, 33 sections, 260 assigned courses, 600 $TSFs$, 756 periods and 51 rooms. From GRT we can easily generate individual teachers' routine and all sections' routine.

4. CONCLUSIONS

Displaying GRT we can see the whole scenario of an institution at a glance. A single table describes whole institution. Seeing a column, we can see at this period which teachers are in classes and seeing the row we can find out in which room the teacher is taking class. Besides, individual teachers' routine and all sections' routine are also be generated from GRT by this algorithm.

ACKNOWLEDGEMENTS

We acknowledge to the VC of BAUST who inspired me to do this task to prepare a global academic routine for BAUST. I thank to all department heads of BAUST who help me providing necessary information about assigned courses. Then I start to think and try to generate an algorithm to build a program to accomplish this job.

REFERENCES

1. Mary Almond,An algorithm for constructing University timetables, Mathematics Department, Queen Mary College, Mile End Road, London. (Extracted on 25 June 2019). Source: https://academic.oup.com/comjnl/article-abstract /8/4/331/400750
2. Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole, An Algorithm to Automatically Generate Schedule forSchool Lectures Using a Heuristic Approach, International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012.
3. Akshay puttaswamy, H M Arshad Ali Khan, Chandan S.V, Parkavi.A, A Study on Automatic Timetable Generator, International Journal of Science and Innovative Engineering & Technology, May 2018.
4. Łukasz Antkowiak,Parallel algorithms of timetablegeneration, School of ComputingBlekinge Institute ofTechnologySE-371 79 KarlskronaSweden.