CRISSCROSS CONSTRAINED PATTERN GENERATION ALGORITHM FOR UNIVERSITY TIMETABLE

Dr.B.Chandra Sekaram
System Analyst,
Information & Communication Technology Resource Centre,
Rashtriya Sanskrit Vidyapeetha, Tirupati-517501,Andhra Pradesh, India

Abstract : Preparing course time table manually for a university with many constrains is a herculean task. Researchers tested many possible approaches to generate automatic time table for University and working for better possible solution. Several technical papers discussed about theoretical approach to solve this problem very few of them come with particle approach which is point to specific use case. Timetable problems are different for different universities. Computerized way of generating automated time tables demands a need for a way of efficient representation of such automatically generated timetables for properly storing and accessing. Time table problem involves optimum utilization of student, Paper and lecture or Teacher schedule them accordingly. This paper is intended to propose an algorithm which generates automatic timetables with efficient storage representation and processing. This also proposes algorithmically an easy way of applying all possible time table constraints. Further it also proposes how this algorithm is flexible in applying stake holder centric constraints (stake holders being teacher/class/student).

IndexTerms – **UniversityTime Table, crisscross, pattern generation,**

Introduction: Time table preparation/planning is an inevitable time-consuming herculean process that every academic institution has to take it up at the beginning of every academic year. Quickest and proper way of taking up this process ensures a good beginning of academic process. This process consumes lot of human hours of teaching faculty. The difficulty in making time table for University is because of need for applying many constraints and bigger search space[11].conflicts usually arise as to the availability of teachers, classes and classrooms.

Generally, university timetable is prepared manually [5] by administrative staff based on their experience. While preparing the time table they need to consider available faculty, courses, rooms and laboratories, minimum number of working days, curriculum compactness. Moreover, the teacher's time and time of course are important constraints to handle. As stated in [12] and [13] the problem of time tabling is very complex because of several components that includes factors like elective courses, teachers, the lecture hall and teacher time slots with focus to the limitations and specific conditions that must be met. Therefore, based on all the mentioned constraints, preparing course timetabling is a very exhaustive and time-consuming task.

Problem: Generating time tables for the courses offered by the Vidyapeetha which satisfies the stake-holder centric constraints (stake holders being student, teacher, classroom, timeslots). A student opting a course will be free during specific time slots and depending on the course he/she has to attend the lectures specific course. Student opting a course will have to study two kinds of papers known as compulsory papers and optional papers. Two students pursuing different courses may have to study some common papers. Teachers can handle multiple papers relating to different courses. There is a limit on number of lecture halls with respect its accommodating capacity. There is a limit on maximum number of timeslots a teacher has to handle in a week. The task is to bind a free time slot of a student, unassigned timeslot of a teacher, and a suitable lecture hall subject to many constraints like students who opted same paper should never be divided and handled separately. Finding an optimal solution to binding time slot of teacher's availability, rooms and student with same group (same class, same selection, and same paper) is time tabling problem.

The task requires tables like student-paper, teacher-paper, room-capacity, teacher-timeslots, student-timeslots, roomtimeslots. These tables are generated during the admission process of the student.

Time slots of students with same group SA{Sa_1,SA_2,Sa_3…}

Time slots of Teacher's availability T{t_1,t_2,t_3….t_n}

Time slots of Lecture halls availability R={r_1,r_2… r_n}

Time slots of a class or course TS{ts_1,ts_2,t_s..ts_n}

Time table= $\boldsymbol{T} \cap \boldsymbol{TS} \cap \boldsymbol{SA} \cap \boldsymbol{R}$

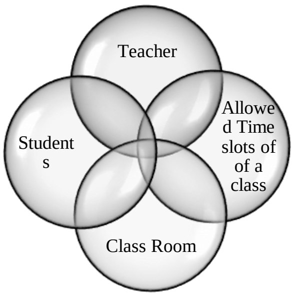


Figure 1

Following issues are considered while taking up automatic time table preparation

- Teacher's priority with respect to time slot in handling a specific class
- Avoiding two consecutive time slots to a paper if otherwise required
- Considering the maximum limit on the number of time slots per week
- Considering the maximum number of desired time slots required to complete a course
- Creating the new sections of a class depending upon maximum limit on the class strength
- Creating the new sections when student teacher ratio exceeds desired value
- Allowing mutual swapping of time slots teachers request basing on the recommendation of academic authorities
- Generate number of random time tables for university with a scheme to uniquely identify them
- Preparing the time table and communicate to student and teacher
- Allowing priority facility to have their time slots Swapped / changed
- Automatic timetable system facilitates generating timetable on weekly, monthly and yearly basis

State of Art:

So many theory's and algorithm are proposed to address this problem will talk about their solution and Limitations. Genetic algorithm approach to generate course timetables for universes this solution are subjected to GA operations (crossover and mutation) repeatedly until a near-optimal solution is achieve desire result which is highly time consuming **[1][2][3][7][8][9][10].** Implementation of Genetic algorithm is very complex in nature. Mobile based algorithm talks more about managing college time table with fixed number of teachers, courses and class. This does not talk about situation like how time table will be organized if substitute faculty rejected the request or not fee during the proposed time. **[4].** Bullet Timetable Education (BTTE) . This paper presents about BTTE research by considering end to end solution to making the timetable but not discussed about implementation approach **[6].**

Proposed method/ Crisscross constrained pattern generation algorithm: Time table problem has multiple solutions. Normally, number of timeslots in a day are fixed and therefore in a week also fixed. The syllabus of each and every paper of a course are to be taught subject to maximum stipulated timeslots. Computationally, paper-wise timeslots are calculated per week, as the timetable is devised to be handled on weekly basis. All desired papercodes of the subject paper relating to a course/student are stored in array/string-pattern. The course wise string patterns are generated by the special algorithm developed. These stringpatterns are made up of desired timeslots required for each papercode plus remaining leisure timeslots. First requirement to implement this algorithm will be to calculate total number of time slots required in a week and which is represented as T_1 to T_N and paper codes are represented as 3 char string where first character will represent paper type (compulsory or Optional) whereas 2 other characters are sequential numbers from 0-99 C01-C0N and O01-O0N. For example, if total timeslots in a week are 40 and 30 timeslots/week are desired then leisure timeslots are 10. Three lettered string is used to code a paper and LLL is used to code leisure timeslot. Accordingly, generated string pattern may be like "P01P01P01P01P01P01P02P02P02P02P02P02……LLLLLLL". The generated example string is of length 120. If there are 30 classes, the resultant university timetable made up of string of length 3*40*30. The benefit of storing timetable as a string is the ease of using regular expressions. All constraints/rules are applied/verified using regular expression, which increased efficiency of algorithm.

Crising & Pattern Generation Algorithm

```
for course in courselist_of_university
paperpatten=List(),
totaltimeslotsinweek =periods per day* Number of working days
           for   course in courselist_of_university
        papercodes=list()
           papercodes=getpapercodeFromcourse(course)
   for papercode in papercodes
                       timeslots=read_timeslots_for_papercode(papercode )
                    for  j=0; j<timeslots;j++

                       paperpattern.add(papercode) //addding same paper code

                    endfor
    endfor
    leisuretimeslots= totaltimeslotsinweek-  paperpattern.size()
    for  i0; i<lisuretimeslots;i++
        paperpattern.add(lisuretime)
     endfor
     while(true){
           Count=0;
           shuffel(paperpattern)
           for m=0;i<paperpattern.size();m=m+Periods_per_day
             sublist=list();
            sublist=paperpattern.getsubList(m,(Periods_per_day+m));

                    if ( frequency(papercode,sublist) >=2))
                 continue
               else
                  count++
                         if(count==numberofpapercodeforaclass)
                            break;
            endfor
     endwhile

  coursepattern[course]=paperpattern
endfor
```

The following prerequisites are observed before testing the pattern generation

1. Each class (or) Course has set of papers
2. Each paper must be assigning a unique code
3. Each student must be identified uniquely
4. Each faculty must be given a unique code
5. Each paper must have unique code
6. Paper code and teacher are bound together in a table
7. Paper code and students are bound together in a table

Input:

- Number of periods (or) Time Slots in a day
- Number of working days in a week **Days**
- Total Number of Hours Required in a week for a paper code
- Max number of allowed hours for a teacher in a week.

tAll types of constraints/rules are applied using regular expressions as it facilitates faster evaluation of the expression. The algorithms shown here applies arrayed patterns to evaluate. This algorithm is experimented by applying paper constraints horizontally, teacher constraints vertically. Teacher constraints are applied after transforming the generated paper patterns to teacher-coded patterns.

Paper Constraints Cris (apply Horizontally)

- Same paper code should not appear consecutively in same day
- Ensure all paper codes appear at least once in a day depending subject to desired time slots

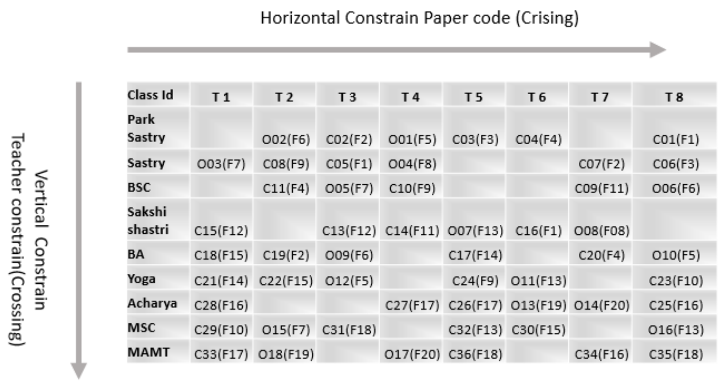
CROSS-pattern algorithm: The resultant paper-code patterns obtained from previous algorithm is subject Cross patter algorithm for applying rules relating particular timeslot where **cris** algorithm was applied to verify constraints

relating to day. For carrying out timeslot based constraint application, the resultant pattern converted teacher coded patterns. The algorithm ensures that never more than once a teacher code appears in any time slot . If by chance, a teacher code appears more than once, in any course pattern string that specific timeslot of that course is swapped with next time slot in the same course pattern and the same is applied paper code string also. This process is repeated till all course patterns verified with non-pre assigned teacher code in the string.

```
Application of constraints by crossing pattern
//convert paper pattern to teacher pattern
teacherPattern=list()
for course in coursepattern
        teacher_list=List()
        paperpattern=course
        for papercode in paperpatten
                teacher_list.add(get_avabile_teacher(papercode))
                update_teacher_avilability(teacher_l,papercode)
        endfor
        teacherPatter[course]=(teacher_list)
endfor
//conversion of paper code pattern to teacher patter complete
//crossing algorithm to apply teacher constraints on the pattern
for time in time_slot
        previousTeacher=""
        for course in coursepattern
                currentteacher=teacherPatter[coursre][timeslot]
                if(previousTeacher==teacher)
                        if(coursepattern[course][timeslot+1] is valid)
                                swap(coursepattern[course][timeslot], coursepattern[course][timeslot+1])
                        endif
                endif
                previousTeacher=currentTeacher
        endfor
endfor
```

Teacher and room Constraints (applied vertically)

1. Teacher code should not appear more than once in any time slot.


Horizontal Constrain Paper code (Crising)
Vertical Constrain
Teacher constrain(Crossing)

Class Id	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8
Park Sastry		O02(F6)	C02(F2)	O01(F5)	C03(F3)	C04(F4)		C01(F1)
Sastry	O03(F7)	C08(F9)	C05(F1)	O04(F8)			C07(F2)	C06(F3)
BSC		C11(F4)	O05(F7)	C10(F9)			C09(F11)	O06(F6)
Sakshi shastri	C15(F12)		C13(F12)	C14(F11)	O07(F13)	C16(F1)	O08(F08)	
BA	C18(F15)	C19(F2)	O09(F6)		C17(F14)		C20(F4)	O10(F5)
Yoga	C21(F14)	C22(F15)	O12(F5)		C24(F9)	O11(F13)		C23(F10)
Acharya	C28(F16)			C27(F17)	C26(F17)	O13(F19)	O14(F20)	C25(F16)
MSC	C29(F10)	O15(F7)	C31(F18)		C32(F13)	C30(F15)		O16(F13)
MAMT	C33(F17)	O18(F19)		O17(F20)	C36(F18)		C34(F16)	C35(F18)

C - Compulsory Paper Code
O - Optional Paper Code
F - Faculty Code

Once all paper codes and time slots are bind together, Check Same teacher will not assign Multiple paper codes in same time slot

- If teacher is assigning more than one Paper code Fist conflicted time slot will be swapped by next time slot and next with other and so on by following other constrain

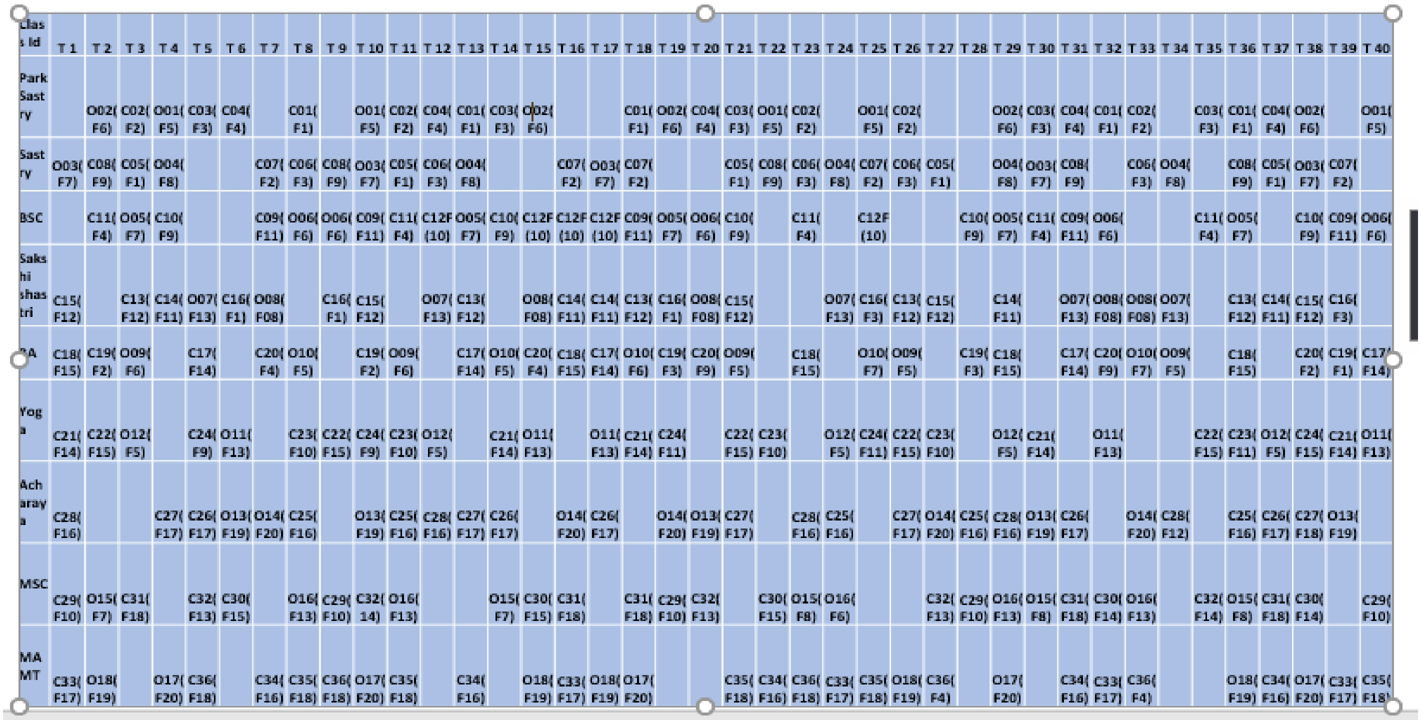
This algorithm was tested for 10 different courses offered in Rastriya Sanskrit Vidyapeetham in Tirupati with following inputs and results are satisfactory

Input

Number of periods per in a day =8
Number of working days =5
Number of paper codes per class=6(C01, C02,CO3,C04 ,O01,0O2)
Number of classes required for each paper code for a week=5
Number of Faculty available to teach these subjects are store in table Format as

Teacher	**Paper code**	**Max Period allowed In a week**
F1	**C01,C05,C16,C19**	14
F2	**C02,C07,C19,C20**	14
F3	**C03,C06,C16,C19**	14
F4	**C04,C11,C20**	14
F5	**O01,O12,O09,O10**	14
F6	**O02,O06,O09,O10,O16**	14
F7	**O03,O7,O10,O15**	14
F8	**O04,O08,O15**	14
F9	**C08,C10,C20,C24**	14
F10	**C12,C29,C23**	14
F11	**C9,C14,C23,C24**	14
F12	**C13,C15,C28,C32**	14
F13	**O07,O11,O16**	14
F14	**C17,C21,C30,C32**	14
F15	**C18,C22,C24,C30**	14
F16	**C25,C34,C28**	14
F17	**C26,C33,C27**	14
F18	**C27,C31,C35,36,27**	14
F19	**O13,O18**	14
F20	**O17,O14**	14

After applying Crisscross algorithm, the final time table is shown below for 10 Different classes



Class Id	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9	T 10	T 11	T 12	T 13	T 14	T 15	T 16	T 17	T 18	T 19	T 20
Park Sastry		O02(F6)	C02(F2)	O01(F5)	C03(F3)	C04(F4)		C01(F1)		O01(F5)	C02(F2)	C04(F4)	C01(F1)	C03(F3)	O02(F6)			C01(F1)	O02(F6)	C04(F4)
Sastry	O03(F7)	C08(F9)	C05(F1)	O04(F8)			C07(F2)	C06(F3)	C08(F9)	O03(F7)	C05(F1)	C06(F3)	O04(F8)			C07(F2)	O03(F7)	C07(F2)		
BSC		C11(F4)	O05(F7)	C10(F9)			C09(F11)	O06(F6)	O06(F6)	C09(F11)	C11(F4)	C12F(10)	O05(F7)	C10(F9)	C12F(10)	C12F(10)	C12F(10)	C09(F11)	O05(F7)	O06(F6)
Sakshi shastri	C15(F12)		C13(F12)	C14(F11)	O07(F13)	C16(F1)	O08(F08)		C16(F1)	C15(F12)		O07(F13)	C13(F12)		O08(F08)	C14(F11)	C14(F11)	C13(F12)	C16(F1)	O08(F08)
BA	C18(F15)	C19(F2)	O09(F6)		C17(F14)		C20(F4)	O10(F5)		C19(F2)	O09(F6)		C17(F14)	O10(F5)	C20(F4)	C18(F15)	C17(F14)	O10(F6)	C19(F3)	C20(F9)
Yoga	C21(F14)	C22(F15)	O12(F5)		C24(F9)	O11(F13)		C23(F10)	C22(F15)	C24(F9)	C23(F10)	O12(F5)		C21(F14)	O11(F13)		O11(F13)	C21(F14)	C24(F11)	
Acharaya	C28(F16)			C27(F17)	C26(F17)	O13(F19)	O14(F20)	C25(F16)		O13(F19)	C25(F16)	C28(F16)	C27(F17)	C26(F17)		O14(F20)	C26(F17)		O14(F20)	O13(F19)
MSC	C29(F10)	O15(F7)	C31(F18)		C32(F13)	C30(F15)		O16(F13)	C29(F10)	C32(14)	O16(F13)			O15(F7)	C30(F15)	C31(F18)		C31(F18)	C29(F10)	C32(F13)
MA MT	C33(F17)	O18(F19)		O17(F20)	C36(F18)		C34(F16)	C35(F18)	C36(F18)	O17(F20)	C35(F18)		C34(F16)		O18(F19)	C33(F17)	O18(F19)	O17(F20)		

Class Id	T 21	T 22	T 23	T 24	T 25	T 26	T 27	T 28	T 29	T 30	T 31	T 32	T 33	T 34	T 35	T 36	T 37	T 38	T 39	T 40
Park Sastry	C03(F3)	O01(F5)	C02(F2)		O01(F5)	C02(F2)			O02(F6)	C03(F3)	C04(F4)	C01(F1)	C02(F2)		C03(F3)	C01(F1)	C04(F4)	O02(F6)		O01(F5)
Sastry	C05(F1)	C08(F9)	C06(F3)	O04(F8)	C07(F2)	C06(F3)	C05(F1)		O04(F8)	O03(F7)	C08(F9)		C06(F3)	O04(F8)		C08(F9)	C05(F1)	O03(F7)	C07(F2)	
BSC	C10(F9)		C11(F4)		C12F(10)			C10(F9)	O05(F7)	C11(F4)	C09(F11)	O06(F6)			C11(F4)	O05(F7)		C10(F9)	C09(F11)	O06(F6)
Sakshi shastri	C15(F12)			O07(F13)	C16(F3)	C13(F12)	C15(F12)		C14(F11)		O07(F13)	O08(F08)	O08(F08)	O07(F13)		C13(F12)	C14(F11)	C15(F12)	C16(F3)	
BA	O09(F5)		C18(F15)		O10(F7)	O09(F5)		C19(F3)	C18(F15)		C17(F14)	C20(F9)	O10(F7)	O09(F5)		C18(F15)		C20(F2)	C19(F1)	C17(F14)
Yoga	C22(F15)	C23(F10)		O12(F5)	C24(F11)	C22(F15)	C23(F10)		O12(F5)	C21(F14)		O11(F13)			C22(F15)	C23(F11)	O12(F5)	C24(F15)	C21(F14)	O11(F13)
Acharaya	C27(F17)		C28(F16)	C25(F16)		C27(F17)	O14(F20)	C25(F16)	C28(F16)	O13(F19)	C26(F17)		O14(F20)	C28(F12)		C25(F16)	C26(F17)	C27(F18)	O13(F19)	
MSC		C30(F15)	O15(F8)	O16(F6)			C32(F13)	C29(F10)	O16(F13)	O15(F8)	C31(F18)	C30(F14)	O16(F13)		C32(F14)	O15(F8)	C31(F18)	C30(F14)		C29(F10)
MA MT	C35(F18)	C34(F16)	C36(F18)	C33(F17)	C35(F18)	O18(F19)	C36(F4)		O17(F20)		C34(F16)	C33(F17)	C36(F4)			O18(F19)	C34(F16)	O17(F20)	C33(F17)	C35(F18)

Time taken to generate time table for each class is shown below

Class	Time taken to generate valid pattern Sec
1	26.781
2	17.063
3	3.484
4	47.157
5	205.953
6	14.767
7	69.885
8	189.577
9	155.330
10	125.120

Average time taken per Class = 85.5117 ~ 90 sec

Minimum time taken per Class= 3.484 ~ 4 sec

Maximum time taken per Class= 205.953 ~ 206 sec

Conclusion: This algorithm was experiment to generate automated time tables for Rashtriya Sanskrit Vidyapeetha. Because of automating time tabling system we could easily generate time table at any number of times. This automated time tabling system facilitates administrator to frequently change time tables and help to avoid monotonic time tables all the academic year. The scheme of storing the time table as patters facilitated to communicate the whole time table to stakeholders as a plain text . Further, this algorithm is being extended to work with web interface in integration with mobile platform.

Acknowledgement: I gratefully acknowledge the encouragement and direction provided by our honourable Vice Chancellor Prof.V.Muralidhara Sharma. Further I extend my thankfulness to Sri Prakhya Phani Kumar for extending all the support during the testing phase.

References

[1] Mohammad A. Al-Jarrah1, Ahmad A. Al-Sawalqah2 and Sami F. Al-**Hamdan3 DEVELOPING A COURSE TIMETABLE SYSTEM FOR ACADEMIC DEPARTMENTS USING GENETIC ALGORITHM** Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 3, No. 1, April 2017.
[2]Chambers L. (1995), Practical Handbook of Genetic Algorithms: Application
[3]Maxfield, C. (1997), Genetic Algorithms: programs that boggle the mind. EDN
[4] Shraddha Shinde, Saraswati Gurav, Sneha karme **Automatic Timetable Generation using Genetic Algorithm.** Publish International Journal of Scientific & Engineering Research Volume 9, Issue 4, April-2018 19ISSN 2229-5518
[5] J. Soria-Alcaraz, E. Özcan, J. Swan, and G. Kendall, "**Iterated local search using an add and delete hyper-heuristic for university course timetabling,"** Appl. Soft Comput.,2016
[6] P. Fernandes, C. Pereira, and Armando Barbosa, "**BTTE – An automated timetabling software for Higher Education,**" Cist. 2013
[7]T. Lutuksin and P. Pongcharoen, "**Best-worst ant colony system parameter investigation by using experimental design and analysis for course timetabling problem,"** 2nd Int. Conf. Comput. Netw. Technol. ICCNT 2010, pp. 467–471, 2010.
[8]N. Pillay, "**Evolving Construction Heuristics for the Curriculum Based University Course Timetabling Problem,"** Proc. 2016 IEEE Congr. Evol. Comput. (CEC 2016), pp. 4437–4443, 2016.
[9]S. A. F. Oliveira and A. R. R. Neto, "**An improved Genetic Algorithms-based Seam Carving method**," 2015 Latin-America Congr. Comput. Intell. LA-CCI 2015, 2016.
[**10]** A. H. Karami and M. Hasanzadeh, "**University course timetabling using a new hybrid genetic algorithm**," 2012 2nd Int. eConference Comput. Knowl. Eng., pp. 144–149, 2012.
[11] Ross-Doria O and Paechter B, "A memetic algorithm for University Course Timetabling" Comb.Optim. 56, pp.54-70
[12] Simon D, Wang Y, Tiber O, Du D, Filev D and Michelini J "Trajectory optimization with memetic algorithms: Time-to-torque minimizations of turbocharged engines ", 2016 IEEE International Conference on Systems, Man and Cybernetics, SMC 2016- Conference Proceedings 3(2), pp 983-988
[13] Islam MM Sing, HK Ray T and Sinha A, "An Enhanced Memetic Algorithm for Single-Objective Bilevel Optimization Problems", Evol. Comput 25, pp607-42