Scientific Journal Impact Factor (SJIF) : 3.518
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

DATA REPRESENTATION APPROACH FOR CLASSTIME TABLING PROBLEM WITH GENETIC ALGORITHM FOR ENGINEERING COLLEGE

Shafaque M Islam[1] and Uday Bhave[2]
[1] *Computer Engineering,* Shah And Anchor Kutchhi Engineering College
[2] *professor Computer Engineering,* Shah And Anchor Kutchhi Engineering College

Abstract—The construction of class timetables for academic institutions is a very difficult problem with a lot of constraints that have to be respected and a huge search space to be explored. The time table problem has exponential number of the possible feasible timetables, even if the size of the problem input is not significantly large. Due to huge search space a linear method or algorithm cannot be employed to handle class time table problem, hence the usage of a heuristic method. The heuristic method to be used in this study is the genetic algorithm. The genetic algorithm is one that seeks to find the most optimal solutions where the search space is great and conventional methods are inefficient. It works on a basis of the Darwinian evolution theory. Before a genetic algorithm can be put to work on any problem, a method is needed to encode potential solutions to that problem in a form that a computer can process. The study focused 2 data representation approaches to generate solution of class time table: 1. Matrix form 2. Vector form.

Keywords—hard constraints, soft constraints, optimization, chromosome, genetic algorithm, crossover, mutation.

I. INTRODUCTION

Scheduling class time tables for large number of classes is a complex problem. The problem is a combinatorial optimization problem belonging to NP-hard class where the computational time grows exponentially as the number of variables increases. A lecture timetable problem is concerned with finding the exact time allocation within limited time period of number of events (courses-lectures) and assigning to them number of resources (teachers, students) while satisfying some constraints. The constraints are classified into Hard Constraints and Soft Constraints. Hard constraints are those that must be adhered to, while Soft constraints can be violated if necessary [3]. In course timetabling the hard constraint will be that at a time lecturer cannot take lecture in two different classes [2].

The timetabling problem is considered as an optimization problem because it searches for optimal solutions in the search space of possible timetables, for which Genetic Algorithm (GA) as an optimization technique, can be used to solve it. The advantage of GA is that they can explore the solution space in multiple directions at once [1]. Some of the characteristics of the timetabling problem that make it amenable to genetic search are [4]:

- A timetable can be represented as a chromosome.
- The fitness function can be defined to assess the worth or quality of a particular solution, thus distinguishing the good timetable from the worse one based on the constraints of the timetabling problem.
- Very large and multi-modal search space of timetables makes the GA a particularly good tool for navigating such space.

In paper, the objective of data representation approach is to minimize complexity during genetic algorithm reproduction process. The main focused is on generation of class time table therefore the lab slots are taken as input from end user. These lab slots allocation are keep static throughout the time table generation process.

II. DATA REPRESENTATION

The first step in prototyping a Genetic Algorithm is to define the data stored and how to encode it. For the timetable management problem, a Chromosome is a set of timetables for different classes and a Gene is a part of chromosome.

Input data module of class time table is described as:

- Class : data describe semester
- Subject : data describe name of subjects in respective class (semester)
- Faculty : data describe name of lecturer
- Slot : it is a time period
- Days : data describe working days

The structure of class time table consists of Input Data Module, relation between input data module, time slot module, day module, applying constraints and Genetic algorithm then extract reports.

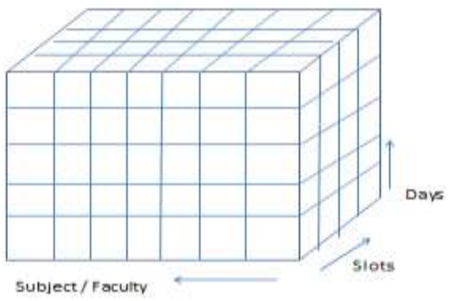
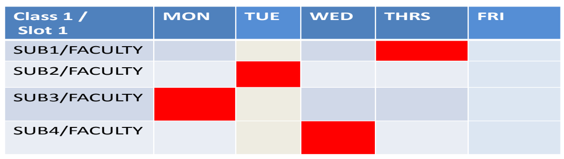


Figure 1. Time Table Presented as 3D structure

The paper proposed two approaches to represent data.

1. **Matrix form approach:** for each class a separate 2D slot wise matrix is created where column indicate days of week and rows indicate subject with faculty. In this representation exactly one row per column must be filled in order to avoid conflict. A filled cell represents subject with faculty scheduled on some day at specific slot.



Class 1 / Slot 1	MON	TUE	WED	THRS	FRI
SUB1/FACULTY				🟥	
SUB2/FACULTY		🟥			
SUB3/FACULTY	🟥				
SUB4/FACULTY			🟥		

Figure 2. Data Represented in Matrix form

In Figure[2] , Filled cell indicate subject with faculty allocation on slot 1 of each day of class 1.

2. **Vector form approach:** chromosome is represented simply as a vector of slots (9.00 to 4.00) , for each day for different classes. Where each time table that represent solution is a number of genes and individual gene represent one subject with respective faculty.

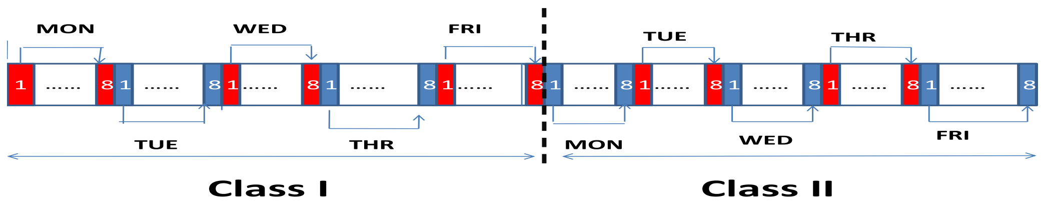


Figure 3. Data Represented in Vector form

III. CONSTRAINTS

Time Table problems are mainly classified as constraint satisfaction problems, where the main goal is to satisfy all problem constraints, rather than optimizing a number of objectives [5]. The constraints involved in building the class time table for engineering college (through the study of constraints requirements of engineering colleges in india) are outlined below:

3.1. Hard Constraint:

- Total number of Lectures of each Subject should be allotted as per curriculum.
- Same faculty should not be assigned to a same period in different class.
- No two faculties have lectures in the same period, in the same class.
- Empty periods should not be present on any day.

3.2. Soft Constraint:

- Consecutive lectures of same faculty in the different class for same subject.
- No consecutive lectures of same faculty on different classes for different subjects.
- No consecutive lectures of different faculty in the same class for same subject.
- Not more than one lecture of same faculty on same subject on same day in same class.
- If there is a load at 9.00 am then there should not be any load beyond 4.00 pm.
- Maximum 2 hours lecture load per day but not more than 5 hours total load per day.

IV. GENETIC ALGORITHM (GA)

A genetic algorithm is a programming technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a fitness function that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them.

A genetic algorithm (GA) has several genetic operators that can be modified to improve the performance of particular implementations. These operators include parent selection, crossover and mutation [6]. Genetic algorithm maintained a population of chromosomes, each representing a possible solution to given problem. Each chromosome is coded a finite length vector of genes (variables).The GA then evaluates each chromosome according to the fitness function. In a pool of randomly generated chromosomes, of course, most will not work at all, and these will be deleted. However, purely by chance, a few may hold promise - they may show activity, even if only weak and imperfect activity, toward solving the problem. These promising solutions are kept and allowed to reproduce.

Steps of the GA

```
Choose initial Population
Repeat
  Evaluate population
  Select best-evaluated individuals for reproduction
  Apply crossover/mutation reproduction
  Set new population
Until ending condition (when a individual get the desired evaluation or a specified number of generations has been done).
```

4.1. Initial Population

Initiate the population is the first step in GA. It consists into creating an amount of random individuals using hard constraints. The choice of the population depends on the needs of the user. A small amount of population will get smaller and destroy the full population after some generations because of evolution can give invalid chromosomes. On opposite a large amount of population will give better results but will require more resources and will be slower.

4.2. Evaluation of the Population – Fitness function

A way of determining the states of generated solutions i.e. calculating how well or bad the individual solutions within the population are.
Fitness: The fitness of a solution is the estimation of how good the solution is, using soft constraints. At this range, the solution is valid. The fitness function can give this information by different ways:

- The fitness can be given using penalty value. Each soft constraints violation give a certain amount of weight (penalty) based on importance of soft constraint, to respective chromosome. In this way the chromosome having greater fitness value will be the worst solution among others.
- The fitness can be given using reward value. Each soft constraints satisfied give certain amount of weight (reward). The chromosome with greater fitness value will be considering the best one among others.

4.3. Selection

The selection strategy addresses on which of the chromosomes in the current generation will be used to reproduce offspring in hopes that next generation will have even higher fitness. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected [7].

4.4. Reproduction

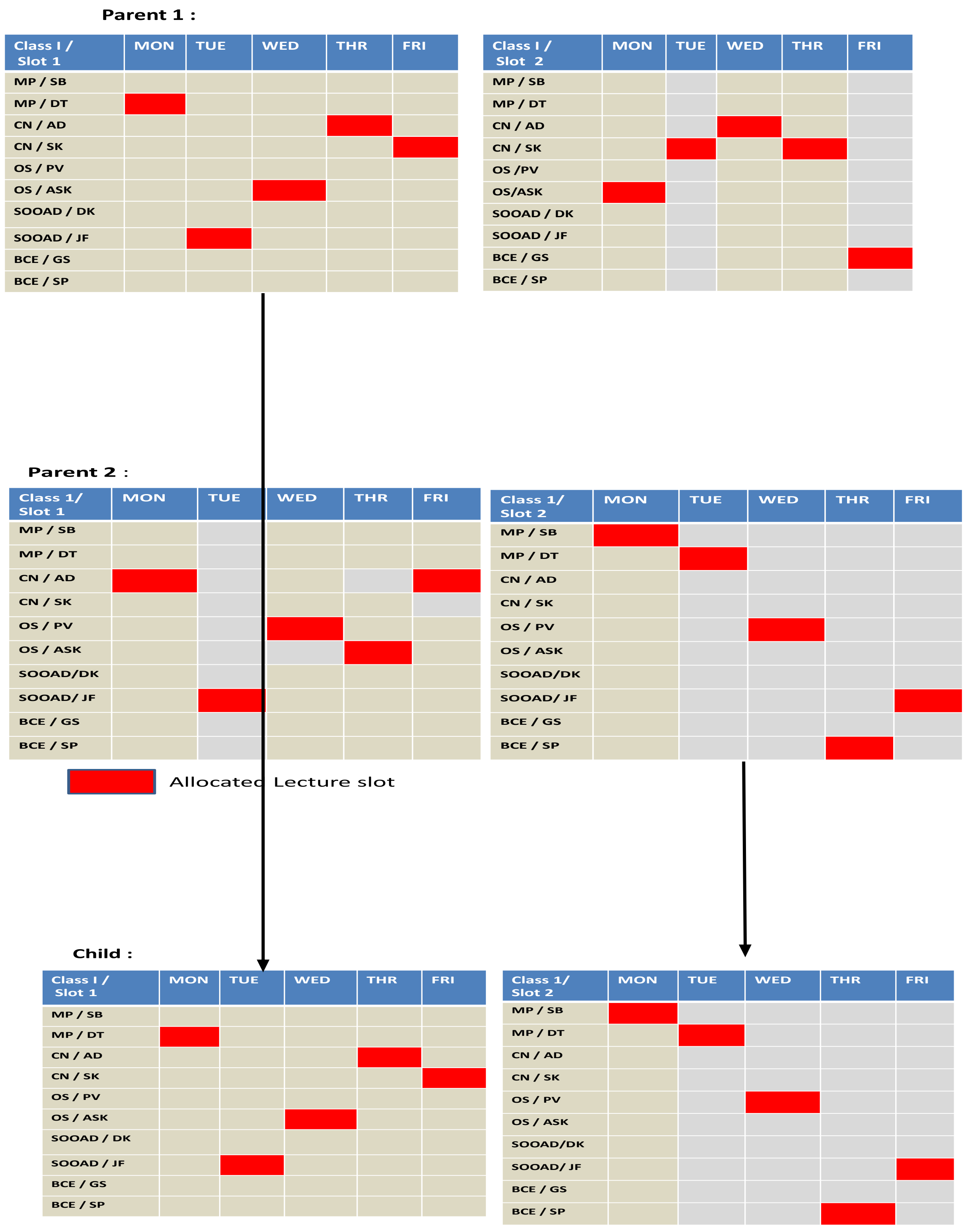
This step generates new chromosomes for next generation. For generating new chromosomes, the algorithm can use both crossover and mutation.

4.4.1. Crossover

A method for mixing fragments of the better solutions to form new, on average even better solutions. Since only the the good solutions are picked for breeding, during the selection procedure, the crossover operator mixes the genetic material, in order to produce children with even greater fitness Two types of Crossover: 1. One point Crossover 2. Two Point Crossover.

1. One Point Crossover: Chromosomes are dissected at a selected point at random to make a head and tail portions. The dissected parts are then moved around and matched to where the head of the first chromosome is combined with the tail of the second one and vice versa creating two new chromosomes [4].

Fig [4] shows one point crossover, applied on chromosome represented in matrix form. Parent1 and parent2 has two 2D matrixes that represent slot1 slot2 of each day. In parent1 , on Monday first slot is assigned to subject MP with faculty SB. Like this in parent2 , on Monday first slot is filled with subject CN and faculty AD. Slot1 matrix of parent1 and slot2 matrix of parent2 are combined to create new chromosome (child). Slot1 matrix of parent1 become head in child and slot2 matrix of parent2 become tail in child.



Parent 1 :

Class I / Slot 1	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT	🟥				
CN / AD				🟥	
CN / SK					🟥
OS / PV					
OS / ASK			🟥		
SOOAD / DK					
SOOAD / JF		🟥			
BCE / GS					
BCE / SP					

Class I / Slot 2	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT					
CN / AD			🟥		
CN / SK		🟥		🟥	
OS /PV					
OS/ASK	🟥				
SOOAD / DK					
SOOAD / JF					
BCE / GS					🟥
BCE / SP					

Parent 2 :

Class 1/ Slot 1	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT					
CN / AD	🟥				🟥
CN / SK					
OS / PV			🟥		
OS / ASK				🟥	
SOOAD/DK					
SOOAD/ JF		🟥			
BCE / GS					
BCE / SP					

Class 1/ Slot 2	MON	TUE	WED	THR	FRI
MP / SB	🟥				
MP / DT		🟥			
CN / AD					
CN / SK					
OS / PV			🟥		
OS / ASK					
SOOAD/DK					
SOOAD/ JF					🟥
BCE / GS					
BCE / SP				🟥	

🟥 Allocated Lecture slot

Child :

Class I / Slot 1	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT	🟥				
CN / AD				🟥	
CN / SK					🟥
OS / PV					
OS / ASK			🟥		
SOOAD / DK					
SOOAD / JF		🟥			
BCE / GS					
BCE / SP					

Class 1/ Slot 2	MON	TUE	WED	THR	FRI
MP / SB	🟥				
MP / DT		🟥			
CN / AD					
CN / SK					
OS / PV			🟥		
OS / ASK					
SOOAD/DK					
SOOAD/ JF					🟥
BCE / GS					
BCE / SP				🟥	

Figure 4. One Point Crossover on Parents represented in Matrix form

2. Two Point Crossover : A two-point crossover randomly selects two points of the parent chromosomes to exchange and match the segments between them [4]. Figure[5] illustrates the process of the two-point crossover on chromosome represented in vector form. In Parent1 on Monday first slot is assigned to subject CN with faculty AD. A slot filled with blue color indicate fixed lab slot provided from end user. These static lab slots are considered while allocating lecture slots.

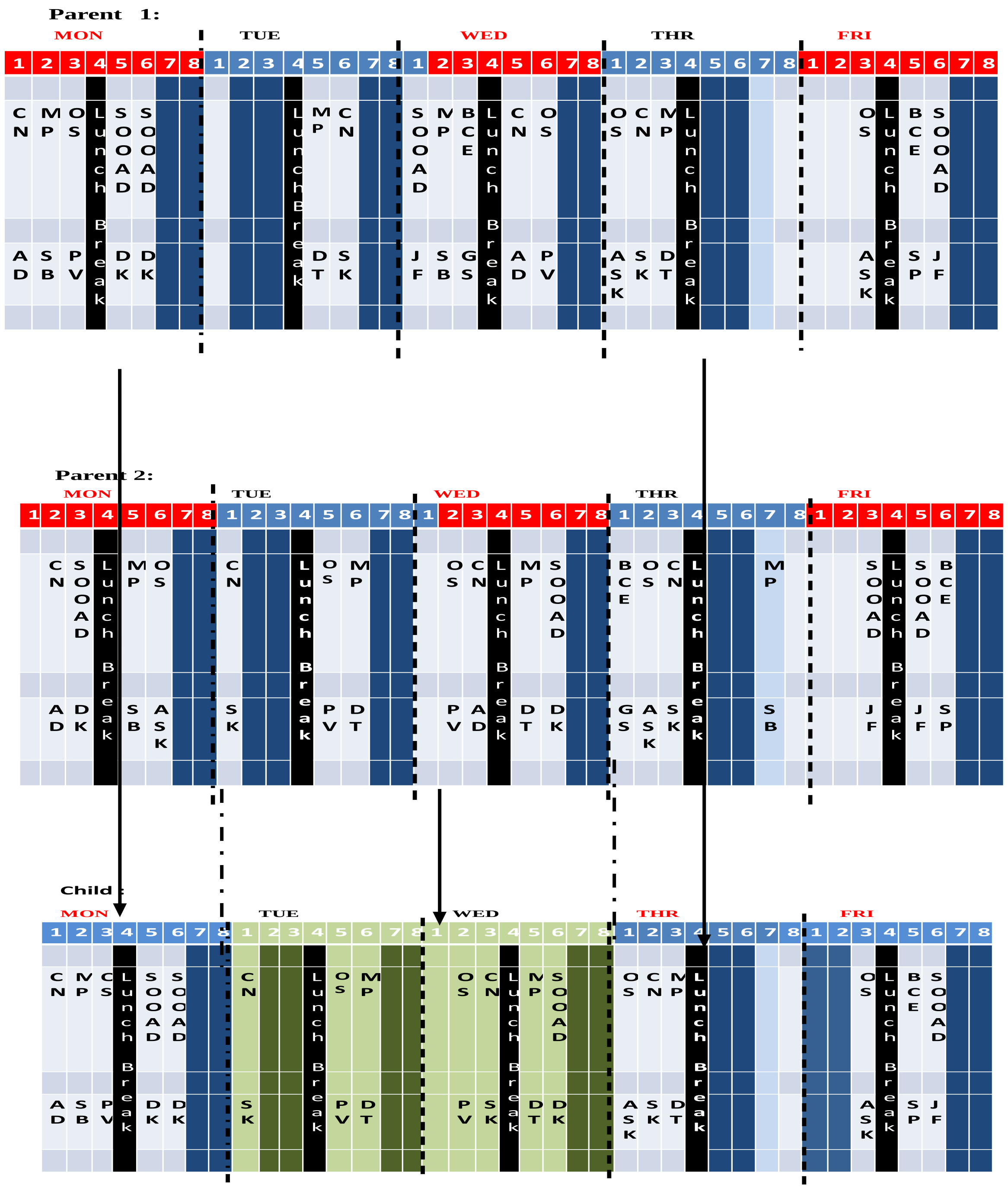


Figure 5. Two Point Crossover on Parents represented in Vector form

4.4.2. Mutation

Mutation is used to get the algorithm moving. It consists of changing the values of a gene randomly, permitting an evolution resulting in a new unexpected solution. These solutions offer a new point of view for the fitness function. The mutation changes only the chromosome, without affecting others solutions. It allows the algorithm to avoid local minima by preventing the population of chromosome from becoming too similar to each other [1].

Parent :

Class 1 /Slot 2	MON	TUE	WED	THR	FRI
MP / SB	🟥				
MP / DT					
CN / AD			🟥		
CN / SK				🟥	
OS / PV		🟥			
OS / ASK					
SOOAD/ DK					🟥
SOOAD/ JF					
BCE / GS				▨	
BCE / SP					

Class 1 / Slot 4	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT		🟥			
CN / AD					
CN / SK					▨
OS / PV					
OS / ASK	🟥				
SOOAD/ DK					
SOOAD/ JF			🟥		
BCE / GS					🟥
BCE / SP				🟥	

Child :

Class 1 /Slot 2	MON	TUE	WED	THR	FRI
MP / SB	🟥				
MP / DT					
CN / AD			🟥		
CN / SK					
OS / PV		🟥			
OS / ASK					
SOOAD/ DK					🟥
SOOAD/ JF					
BCE / GS				⬛	
BCE / SP					

Class 1 / Slot 4	MON	TUE	WED	THR	FRI
MP / SB					
MP / DT		🟥			
CN / AD					
CN / SK					⬛
OS / PV					
OS / ASK	🟥				
SOOAD/ DK					
SOOAD/ JF			🟥		
BCE / GS					
BCE / SP				🟥	

Figure 6. Mutation on Parents represented in Matrix form

In Figure [6], mutation is applied on slot 2 and slot4 matrix of class 1. Slot2 lecture of Thursday is swap with slot4 lecture of Friday result in to child generation (offspring). In child cell filled with black indicate result after mutation.

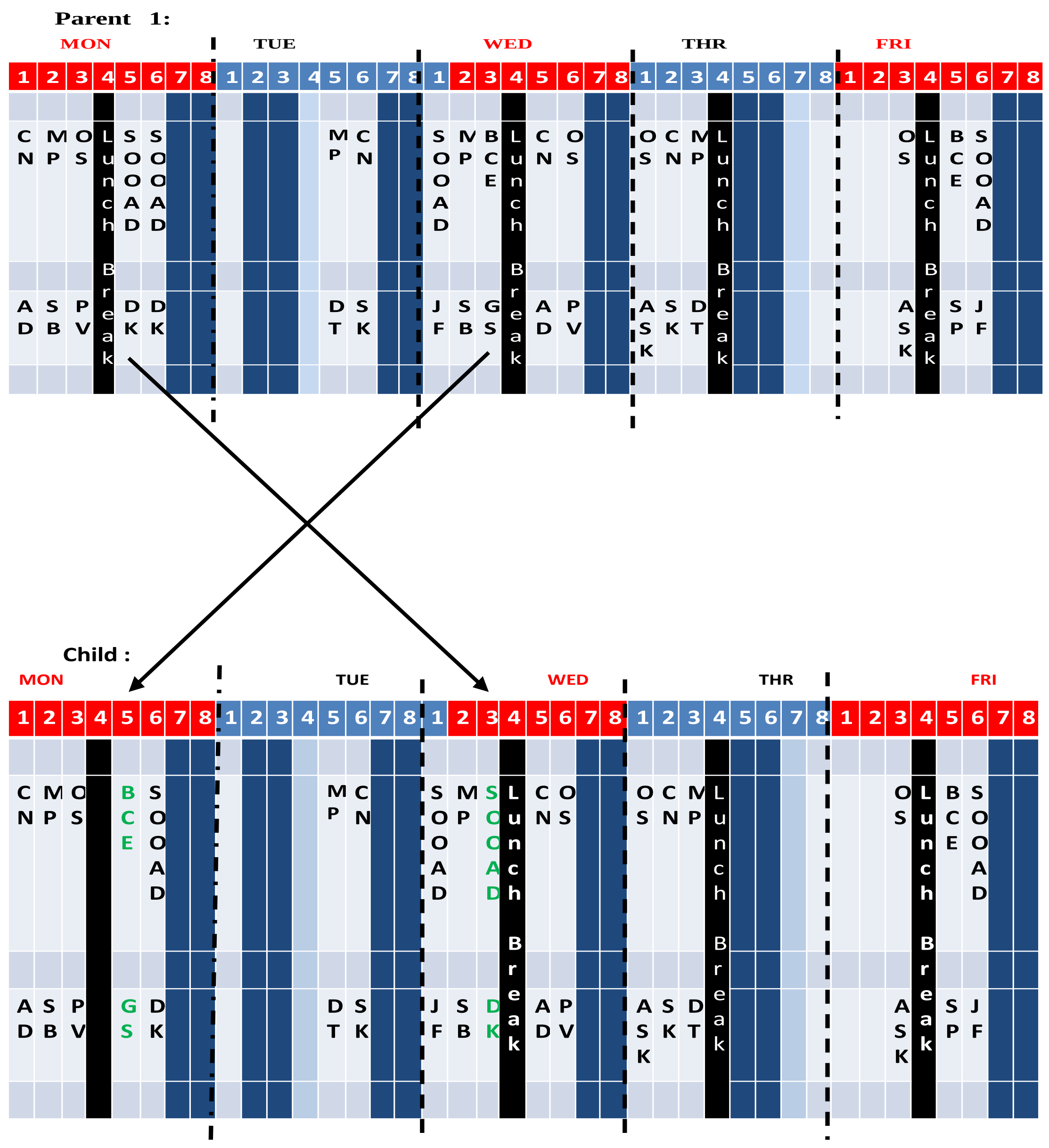


Figure 7. Mutation on Parent represented in Vector form

Figure [7], illustrate mutation example on parent represented in vector form. In parent, slot5 lecture of Monday
(subject :SOOAD with faculty: DK)swapped with slot3 lecture of Wednesday(subject: BCE with faculty : GS.

V. CONCLUSION

Class timetabling is a major administrative activity for a wide variety of engineering colleges in india and usually consists of hard and soft constraints. Satisfactions of all hard constraints ensure feasible solution where as soft constraints satisfactions ensure optimal solution. Since class time table is produced by considering provided static lab slot allocation therefore this result into reduction in collision. This paper described two data representation approaches for class time table problem.1.matrix representation 2.vector representation. In matrix representation, if there are eight slots per day than eight different 2D matrix has to be created for each class. Where as in vector representation, vector is created of size = (number of days * number of slots * number of class).

Genetic algorithm can be applied on data represented in either of above two approaches. Crossover operator applied on two different chromosomes to generate offspring. Through mutation operator, individual chromosome swaps only those genes of same class that improve fitness value. Chromosome must be represented in such a form that it become simple to apply genetic operators on genes. In Vector representation, gene(subject with faculty allotment) is represented in vector so crossover and mutation operator can be applied efficiently on gene by keep tracking of respective vector horizontally . In matrix representation since a separate 2D matrix is created for each slot of each class, individual gene can be refer by keep tracking of row and column of respective matrix. Therefore data represented in matrix form increase complexity during genetic algorithm reproduction process.

REFERENCES

[1] Asif Ansari , Prof Sachin Bojewar,” Genetic Algorithm to Generate the Automatic Time-Table – An Over View”,
[2] International Journal on Recent and Innovation Trends in Computing and Communication , Volume: 2 Issue: 11 November 2014.
[3] Ashish Jain, Dr. Suresh Jain, and Dr. P.K. Chande, “Formulation of Genetic Algorithm to Generate Good Quality Course Timetable ”, International Journal of Innovation, Management and Technology, Vol. 1, No. 3, August 2010.
[4] M. O. Odim, B. O. Oguntunde, O. O. Alli , ” On the Fitness Measure of Genetic Algorithm for Generating Institutional Lecture Timetable”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 4, No. 4 April 2013.
[5] Omar Ibrahim Obaid, MohdSharifuddin Ahmad, Salama A. Mostafa, Mazin Abed Mohammed, “Comparing Performance of Genetic Algorithm with Varying Crossover in Solving Examination Timetabling Problem”, Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO.10, Oct 2012 .
[6] Sandeep Singh Rawat , LakshmiRajamani , “A TIMETABLE PREDICTION FOR TECHNICAL EDUCATIONAL SYSTEM USING GENETIC ALGORITHM”, Journal of Theoretical and Applied Information Technology,2005.
[7] Noraini Mohd Razali, John Geraghty, “Genetic Algorithm Performance with Different Selection Strategies in Solving TSP”, Proceedings of the World Congress on Engineering 2011 Vol II , July 6 - 8, 2011.
[8] Pratibha Pajpai, Dr. Manoj Kumar , “Genetic Algorithm - an Approach to Solve Global Optimization Problems”,Pratibha Bajpai et al./Indian Journal Of Computer Science and Engineering Vol 1 No 3 199-206.