SOLVING A CLASS SCHEDULING PROBLEM USING GENETIC ALGORITHM

Sujit Kumar Jha[1], and Pramod K Shahabadkar[2]
[1]Engineering Department, Ibra College of Technology, Ibra, Sultanate of Oman
[2]Training & Placement Cell,
K.K. Wagh Institute of Engineering Education & Research, Nashik, India

ABSTRACT

Timetabling is a highly complex problem which is a part of the wider-field of scheduling. Scheduling is the allocation of resources over time to carry out a set of tasks and is NP-hard problems. Scheduling of large modular courses and teacher assignment is a complex problem which often has to be solved in the department of educational institutions/universities. This is usually done by hand taking several days or weeks of iterative repair after feedback from staff and students complaining that the timetable is unfair to them in some way. The timetabling problem combines both lecturer assignment and scheduling of the courses simultaneously. Being classed as NP-hard problem, this problem is a good candidate to generate a timetable using genetic algorithm (GA). The paper proposed a genetic algorithm technique to solve this complex problem. This paper presents details about the development and implementation of a computer program which employs GA for an optimal solution of solving a timetable problem by considering number of classes and courses offered in each semester and finally generate lecturer timetable. The program written in C includes a repair strategy for faster evolution. The performance of the algorithm further can be improved by using parallel computing approach.

KEYWORDS: *Chromosome, Evolution, Selection, Crossover, Mutation, Fitness, Genetic Algorithm, Mathematical Model, Scheduling, Timetable.*

I. INTRODUCTION

Timetabling concerns all activities with observe to making a timetable. Timetable is a table of events arranged according to the time when they take place. The events are usually meeting between people at a particular location. A timetable must meet a number of requirements and should satisfy the desires of all people involved simultaneously as well as possible. Timing of events should be such that nobody has more than one event at the same time. Since 1995, a large amount of research has been presented on this area.

In this paper, we have considered timetable construction problems in engineering department at ICT, Ibra. Timetable problems are mainly allocating resources, i.e. teachers, students, rooms, and time slots, to lessons. Three of these decisions, viz. teacher assignment, room assignment, and time slot assignment made first to each lesson than assigning students to lessons. For example, let us take any subject. There is a group of students that take lessons in this subject. If the number of students is small enough, all students fit in one room therefore can take same lessons in the subject. On the other hand, if the number of students is large, we must divide the group of students into number of groups that are small enough to accommodate in room also called students sectioning. In practice, college management decides on the number of sections for each subject. All sections of the same subject have the same number of lessons. Assigning a student to a section implies assigning the student to all lessons of the subject.

It is impossible for a teacher or student to attend more than one lesson simultaneously. Therefore, every teacher and student can have at most one subject at the same time. We call these requirements as teacher constraints and student constraint. Similarly, room constraints are only satisfied if each

room is used for only one lesson at a time. Teachers can be unavailable for teaching subjects at some time slots. Teacher availability constraints force us to find time slot assignments that satisfy these requirements. This paper also considers room compatibility constraints, which consider that some rooms are dedicated to specific subjects. This paper also consider educational constraints, which consider that subject of the same section must be assigned to different days such that students can absorb the material and are able to do homework for the next lesson. We also consider the idle time of a teacher or students, which can be minimized during start or end session. For example, if a student has fewer subjects less than the total number of time slots, he will have some idle time in his schedule. Students generally do not like idle time when it leads to gaps in their timetables. However, gaps in the student's timetable at the beginning or ending of the day are favourable.

For timetabling problems, the events are lessons in a subject, taught by a teacher to a group of students, sometimes referred to as a class, in a room. In figure 1 we show a central concept to which all other concepts, most of them representing the necessary resources, are related.

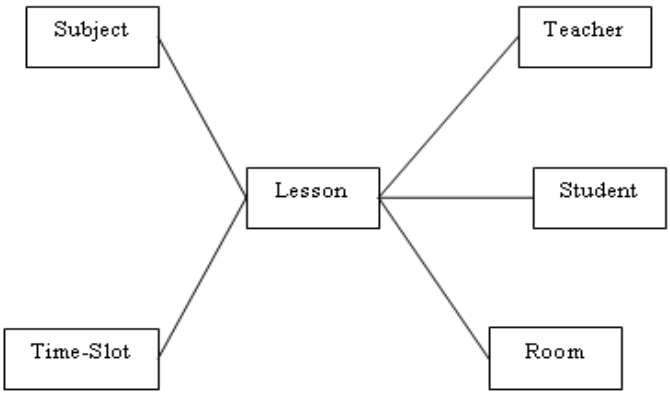

Figure 1: Shows the relationship between different concepts for timetabling.

Our aim is to schedule classes to fully utilize available resources. This is done by assigning the course to teacher at correct time and place to appropriate event. Timetable constraints are many and varied. In this research, genetic algorithm approach is applied for solving college timetabling problem. Genetic Algorithm is the way of addressing hard search and optimization problems which provides a good solution although it requires large execution time.

In section 2, Course-Timetabling problem description has been described in detail. The genetic algorithm, the underlying chromosome representation, the genetic operators, and the evaluation function has been presented in section 3. In section 4, Computational Results has been described for performance aspects of timetabling. Finally, Conclusions and Future work has presented in section 5.

II. COURSE-TIMETABLING PROBLEM DESCRIPTION

Timetabling problems arise in many real-world circumstances (e.g. nurse rostering [1] (Burke et al., 2004), sports timetabling [2] (Easton et al., 2004) and university timetabling problems [3, 4] (Carter and Laporte, 1996; Carter and Laporte, 1998). A general timetabling problem includes scheduling a certain number of events (exams, courses, meetings, etc.) into a limited number of time periods, while satisfying as many of the required constraints as possible. Timetabling problems have been very well studied for more than 4 decades [5, 6] (Burke et al., 1997; Burke and Petrovic, 2002). Recently a large amount of successful research has been carried out which has investigated meta-heuristic approaches for a variety of timetabling problems with evolutionary algorithms [7, 8] (Burke and Newall, 1999; Carrasco and Pato, 2002). These include tabu search [9, 10] (Costa, 1994; Gaspero and Schaerf, 2000), simulated annealing [11] (Dowsland, 1998) and evolutionary algorithms [12, 13] (Colorni et al., 1990; Colorni et al., 1991). In addition, fuzzy methodologies have recently been explored for both courses and exam timetabling [14, 15] (Burke et al., 2001; Dean, 2008).

Course-timetabling problem involves scheduling classes, teachers, course and rooms to a number of time-periods or time-slots in a week. In this paper, we have constructed a timetable for the two levels

of courses like diploma and higher diploma at Ibra College of Technology, Ibra, Oman. The course is divided into two semesters (fall and spring). In every semester the course contains 5 subjects in each level. Some subjects should be allocated 2 time period and some 3 time periods like machine drawing. Each day of the week is divided into 12 periods (of 1 hour's duration). There are five working days per week. Hence, the set P of periods consists of 60 elements. Our college offers a number of courses (e.g. Engineering Materials, Manufacturing Process, Applied Mechanics I and II, Fluid Mechanics I and II, Mechanics of Materials, Electrical Technology …) in Mechanical Engineering Department.

A class consists of all students studying a given course in the same semester. We denote M the set of all classes. For each class $c \in M$ the maximum number of students must be given as part of the input to the system**.** K is the set of all teachers (lecturers, tutors, etc.). Each lecturer requires a number of free-time slots (or even one or two free days) where he/she is unavailable. L is the set of rooms available in the college. A room can be a laboratory (e.g. for physics, or chemistry) or a lecturer theatre. A course module consists of one or more lessons. A lesson may be a lecture, a laboratory or a group exercise class, and it has duration of 1 hours. The set of all lessons is denoted by R. For each $s \in R$ the type of the lesson, its unique teacher and the coordinate list of classes must be present [17, 18, 19] (Van den Broek and Hurkens, 2010; Sandeep Singh Rawat and Lakshmi Rajamani, 2010; Ashish et al., 2010).

2.1 Problem Definition

Following are participants of lecturer timetable:

M classes $c_1, c_2, c_3, \ldots, c_m$ and
K lecturers $l_1, l_2, l_3, \ldots, l_k$ and
R subjects $s_1, s_2, s_3, \ldots, s_r$ and
L rooms $r_1, r_2, r_3, \ldots, r_l$ and
P periods $p_1, p_2, p_3, \ldots, p_n$

where an assignment is a 5 tuple $\{c, l, s, r, p\}$

2.2 Constraints Involved

In this paper, constraints are classified into two categories hard and soft constraint [7, 19] (Burke and Newall, 1999; Ashish et al., 2010). Hard constraints are those to which timetable has to adhere in order to be satisfied. These are following:

- Each lesson $s \in R$ is scheduled to exactly one period i.e. every class must be scheduled exactly once.
- There should be no clashes: Neither a class nor a teacher nor a room is assigned to more than one lesson in the same period.
- There cannot be more than 2 classes for a subject on one day.
- No room should be double booked.
- Lecturers/Teachers unavailability is considered.
- All allocated rooms are large enough to hold the students.

Violating the above constraints will cause the timetable to be unfeasible. In addition, we would also like to satisfy as many soft constraints as possible in order to produce a good quality timetable. Soft constraints are actually the students and lecturers preferences which are following:

Here are some examples:

- Lecturer having 3 theory subjects has no lab assignments.
- Lab classes may not be in consecutive hours.
- Students, as well as some teachers, do not like to have many idle periods between two lessons in their timetables.
- Lessons should be spread uniformly over the whole week, in general.
- Some teachers, by contrast, wish to have all their lessons scheduled to consecutive periods.
- Room should be just larger enough to hold the students.

In this paper, we consider 'K' no of lecturers, 'M' no of classes per subject per week. Each day has 'H' no of hours and we have 5 working days per week. Then scheduling of K*M no of classes to 5*H time slots [19] (Ashish et al., 2010). For our problem, we considered 7 lecturers and 13 subjects in the college for diploma and higher diploma levels are given in table 1.

Table 1: Courses allocation to lecture

LECTURER	SUBJECTS	No. of Sections
Sujit	Engineering Materials, Mechanics of Machines	3,1
Pramod	Manufacturing Process, Workshop Technology	2,2
Shiv Kumar	Applied Mechanics I, Applied Mechanics II	2,1
Shylesh	Fluid Mechanics I, Fluid Mechanics II	2,1
Subhash	Electrical Technology	2
Dhanraj	Machine Drawing, Engineering Drawing	2,1
Jawahar	Engineering Instrumentation & Industrial Control, Mechanics of Materials	2,1

2.3 Mathematical model for the problem

The mathematical model of the problem can be formulated as follows [16] (Ferland, 1985):
For given n classes and m time periods. Let x_{ij} (i=1,2,…,n and j=1,2,…,m) be the decision variable, that can be found as

$$x_{ij} = \begin{cases} 1 \; if \; class \, i \, is \; start \; at \; time \; period \; j \\ 0 \; else \end{cases}$$

For any i=1,2,…,n, denoted by J_i a set of time periods that are suitable for class i and J_{ijk} a subset of J_k which are in conflict with the assignment of lecture i to time period j.
Assignment constraints can be specified as follows:

$$\sum_{j=1}^{m} x_{ij} = 1, i = 1,2,...,n$$

Additional side constraints that describe the conditions for avoiding conflicts can be specified as follows:

$$x_{ij} + x_{kl} \le 1, \; i \in J_{ijk}, \; k \succ i, \; 1 \le i \le n, \; 1 \le j \le m$$

The goal is to determine an assignment for each lecturer to a time period in order to minimize total assigning cost and fulfil additional side constraints.
The cost of assigning class i to the time period j is denoted by C_{ij}. This cost is specified in terms of availability and preferences of corresponding lecture for class i. In this model, the preference of lecturers for each time period is ranging from 1 (highly preferred) to 4 (impossible). The cost structure is specified as follows:

$$C_{ij} = \begin{cases} k \; if \; time \; period \; j \; is \; ranked \; k \; for \; class \; i, 1 \le k \le 3 \\ n \cdot m + 1 \; if \; time \; period \; j \; is \; ranked \; 4 \; for \; class \; i \end{cases}$$

Then, our objective of minimizing the total cost of class assignment to lecture is given as:

Minimize: $F(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} \cdot x_{ij}$

Subject to:

(i) $\sum x_{ij} = 1, i = 1,2,...,n, x_{ij} \in \{0,1\}, i = 1,2,...,n, \; j = 1,2,...,m$

(ii) $x_{ij} + x_{kl} \le 1, \; i \in J_{ijk}, \; k \succ i, \; i = 1,2,...,n, \; j = 1,2,...,m$

This is the typical type of the class scheduling problem. The first constraint (i) makes sure that each class is assigned at exactly one period and constraint (ii) eliminating the conflict situation between every two classes.
We construct a timetable which is based on genetic algorithm techniques. It aims not only to find the feasible solution to the problem, but it also searches for timetables people can be happy within that. A 'natural' chromosome representation was chosen, and genetic operators we developed make use of knowledge specific to the particular problem. In that way we avoid building illegal timetables, and are not in need of any 'repair algorithm'. This is in contrast to approaches described in other papers [20, 21] (Ling, 1992; Burke et al., 1997). A recent approach to the timetable problem is to use

the genetic algorithms as a powerful method of solving difficult timetabling problems [22] (David, 1989).

III. GENETIC ALGORITHM

John Holland's original schema was a method of classifying objects, then selectively "breeding" those objects with each other to produce new objects to be classified [23] (Buckles and Petry, 1992). The programs followed a simple pattern of the birth, mating and death of life forms from Darwinian natural selection. A top level description of this process is given in fig. 2. A GA, as shown in figure requires a process of initializing, breeding, mutating, choosing and killing.

```
Create a population of creatures.
Evaluate the fitness of each creature.
While the population is not fit enough:
{
        Kill all relatively unfit creatures.
        While population size < max :
        {
                Select two population members
                Combine their genetic material to create a new creature. Cause a few random
                mutations on the new creature. Evaluate the new creature and place it in the
                population.
        }
}
```

Figure 2: Top Level description of a GA.

Genetic Algorithms (GAs) are a specialization of evolution programs, based on the principals of natural selection and random mutation from Darwin biological evolution. They were formalized in 1975 by John Holland and have been growing in popularity since, particularly for solving problems with a large irregular search space of possible solutions [13, 18] (Colorni et al., 1991; Sandeep Singh Rawat and Lakshmi Rajamani, 2010). The basic concept of a GA is that a population of individuals is maintained; each of these holds a chromosome which encodes a possible solution to the problem being solved. With the passing of time the members of the population interact and their content is passed on through generations of new individuals. Fitter solutions (those closer to the optimum) are more likely than their poorer counterparts to "breed", passing on parts of their genetic material (parts of their solution to the problem) to the individuals ("offspring") in the next generation [24] (Davis, 1991).

Each chromosome would be large, holding an allele for each class to schedule. The GA would assign a room and time slot to each class and its fitness would be a function of the number of constraint violations. The possible constraint violations would include assignment of classes to undersized rooms or rooms of wrong type in addition to clashes between classes. A population of feasible timetables is maintained. The fittest timetables are selected to form the basis of next iteration or generation. Basic operators such as selection, mutation and crossover are applied to get the best results. The initialization of a population, the evaluation, and the genetic operators were implemented and controlled by a program in c. Figure 3 describes the genetic algorithm working cycle. Initial population is generated randomly.

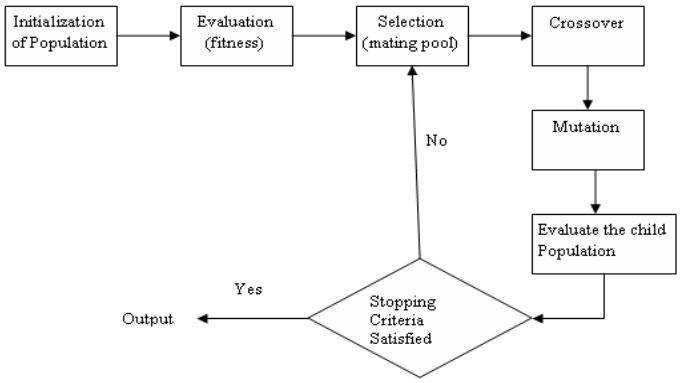

Figure 3: The Cycle of GA

3.1 Chromosome Representation

In a 'classical' genetic algorithm chromosomes are represented as bit strings. However, we believe that problem-specific knowledge should be incorporated in the representation of solutions to our timetabling problem, and the chromosome representation should be natural [24] (Davis, 1991). It should contain all the relevant information and be close to the original problem. In this sense it is clear-cut to define a timetable to be a map

$$f : M \times K \times R \times L \times P \to \{0,1\}$$

Where f(c, l, s, r, p) = 1 if and only if class c and teacher l have to meet for a lesson s in room r at period p. Such a mapping is easily translated into the form

timetable (Class, Teacher, Lesson, Room, Period).

A gene, in this representation may also be considered as an element of a 5-dimensional matrix, with an allele value of 0 (false) or 1 (true).

The input data of our system specifies for each lesson a unique teacher. So we can simplify the definition as:

$$f : C \times R \times L \times P \to \{0,1\}$$

We denote R_c as the set of all lessons which are offered to class c. Then for a timetable f to be feasible it is essential that

$$f(c,s,r,p) = \{0\},\ if\ s \notin R_c$$

$$f(c,s,r,p) = \{1\},\ if\ s \in R_c$$

For any timetable f and a subset π of the set P of periods, can be represented by R (f, π) the set of all lessons which are scheduled in the timetable f to a period $p \in \pi$.

3.2 Initialization

The initialization procedure creates at random a population of feasible solutions. Our objective, valid for the whole algorithm, is to start with legal timetables and never leave this search space. In fact, the timetable problem is known to be NP-hard [13] (Colorni et al., 1991). The member of this population, however, suffers from very poor fitness, in general: they contain a large number of 'idle' periods. This is because the initialization routine does not care at all about soft constraints. For a timetable to be generated, those lessons that are fairly limited as to their possible allocation are considered first. Lessons are selected in random order, and each lesson is assigned to a randomly chosen period and lecture room without violating any hard constraint.

3.3 Evaluation

Fitness in biological sense is a quality value which is a measure of the reproductive efficiency of chromosomes [18, 22] (Sandeep Singh Rawat and Lakshmi Rajamani, 2010; David, 1989). In genetic algorithm, fitness is used to allocate reproductive traits to the individuals in the population and thus act as some measure of goodness to be maximized. The fitness function of each chromosome is evaluated by examining the soft constrains. Each soft constraint is assigned a penalty value which contributes to the fitness function for each constraint violated. In order to determine which are fitter than others, each creature must be evaluated. The method of evaluating a

chromosome, we can either focus to generate the least costly population or the most fit. It is a question of minimizing cost or maximizing fitness. Our evaluation function is made up in the form

$eval(f) = \frac{1}{1+x}$, where x is a sum of weighted penalty values:

$$x = \sum_{i=1}^{k} w_i \cdot n_i(f),$$

here $n_i(f)$ is a penalty value imposed to the violation of a specific soft constraint, and w_i an attached weight. $n_1(f)$ could be the number of class clashes in the timetable. $n_2(f)$ could be the component measuring the costs of having scheduled lessons for lectures double booked during a day. $n_3(f)$ could be the number of room too small allocated to lessons attended by students, $n_4(f)$ could be the component of costs of having scheduled lessons for lecturers unavailable during given time slot, etc. Note that we do not need to impose penalties on violated hard constraints because the concept of our domain-specific genetic operators is to produce only feasible solutions.

The value of evaluation function range from 0 to 1, and our genetic algorithm aims at finding a timetable which maximizes this function. We are still experimenting with different settings $\{w_1, w_2, w_3, ..., w_k\}$ of weights for the components of the cost function. Often it is hard to decide which soft constraints should be considered to be more important than others.

3.4 Selection

The method by which individuals are chosen to contribute material to the next generation is known as selection. The aim is to give preference to individuals of a higher fitness in the hope that they pass the elements which make them better on to the next generation. The initial approach to this was a simple probability based system where the likelihood of an individual reproducing directly corresponded to its fitness relative to the rest of the population. This is known as Roulette Wheel Selection because its operation is similar to that of the selection of numbers on a roulette wheel. Roulette wheel selection provides an advantage over conventional method because through selection we can select the best timetable from the available feasible timetables. Fitness function is used as a measure for the selection of chromosomes (timetable).

Reproduction (or selection) is usually the first operator applied on a population. During each successive generation, a proportion of existing population is selected to breed a new generation. Individual solutions are selected through fitness-based process, where fitter solutions are typically more likely to be selected [14, 18] (Burke et al., 2001; Sandeep Singh Rawat and Lakshmi Rajamani, 2010). Selection methods rate the fitness of each individual and preferentially select the best solution. In this paper, we have used Roulette wheel selection method.

3.5 Crossover

The next step after selection is crossover which generates a second generation population of solutions from those selected through selection. A crossover operator is used to recombine two strings to get a better string. Once parents have been chosen, breeding itself can then take place. A new creature is produced by selecting, for each gene in the chromosome, an allele from either the mother or the father. The process of combining the genes can be performed in a number of ways. The simplest method of combination is called single point cross-over [24] (Davis, 1991). A child chromosome can be produced using single point crossover, as shown in fig 4. A crossover point is randomly chosen to occur somewhere in the string of genes. All genetic material from before the crossover point is taken from one parent, and all material after the crossover point is taken from the other.

PARENT 1: 1 1 1 0 0 1 1 0
PARENT 2: 0 1 0 1 1 1 0 1
Choose a crossover point:
PARENT 1: 1 1 1 | 0 0 1 1 0
After crossover at the fourth bit:
Offspring A: 1 1 1 1 1 1 0 1
Offspring B: 0 1 0 0 0 1 1 0

Figure 4: An Example of Crossover with Encoded Genes

Currently, in our research work we are using one site crossover. Crossover is advantageous over the conventional method because through crossover we can easily exchanged the information in the timetable which makes it optimal and effective as per the requirements.
Having selected two parent timetables f and g, let us call them mother and father respectively, we build offspring in such a way that each lesson and its time and room assignment comes from one of the parents. This is done by generating, for each class $c \in C$, a set $\pi_c \subseteq P$ of periods such that the timetable defined by

$$h(c,l,r,p) = \begin{cases} f(c,l,r,p), & iff \ \ p \in \pi_c \\ g(c,l,r,p), & else \end{cases}$$

is feasible. h is an offspring which has inherited some properties from mother f, others from father g. A second offspring is simply established by changing the roles of f and g.

3.6 Mutation

After crossover is carried out and before the child is released into the wild, there is a chance that it will under go mutation. The purpose of mutation is to inject noise, and in particular, new alleles, into the population and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators. Once a gene has been selected for mutation, in case of a binary string representation, mutation causes a single gene value to be 1 from 0 and vice-versa. Select a timetable f for mutation, a natural number m and a set $\pi \subseteq P$ consisting of m periods are chosen at random and the set $R\{f, \pi\}$ is formed. A mutated timetable f' is produced by assigning new periods or rooms only within the 'time window' π and by leaving the rest unchanged. Taking π and $R\{f, \pi\}$, instead of P and R, as input.
The window size m ranges between two values m_{min} and m_{max} which are parameters of this mutation operator. Clearly, if m is too small, the mutation operator might fail in finding a solution f' different from f. This is because even slight modifications of period or room assignments are likely to produce invalid timetables, and some points in the search space may be isolated. On the other hand, if m is too large, f' may loose similarities to f. We have tested mutation with different parameter settings and have seen best results for random numbers m between 4 and 15.

3.7 GA Implementation

In this paper, GA is employed to develop a program in C to perform Timetabling. The GA operates upon a population of timetables which are maintained in memory. Each timetable is evaluated by testing the number of times it breaches each constraint. Thus timetables are evolved with a minimum number of constraint violations [18] (Davis, 1991). A top level description of the program is provided in Fig 5. The structure is similar to the pseudo code given in Figure 2, with the major difference being the incorporation of a repair strategy.

```
While the population size is less than the maximum:
{
        Create a new timetable with no classes booked to it. Repair the new timetable by using
        the constraint data.
        Evaluate the cost of the new timetable by using the constraint data. Enter the new
        timetable into the population
}
While the cost of the best timetable is greater than zero:
{
        Discard a portion of costly timetables.
        Repeat until the population size is maximum:
        {
                Breed a new timetable.
                Mutate the new timetable.
                Repair the new timetable by using the constraint data.
                Evaluate the cost ofthe new timetable by using the constraint data. Enter the
                new timetable into the population.
        }
}
```

Figure 5: Top Level Description of Program.

3.8 Repair Strategy

The repair strategy is used to make certain that exactly one booking of each class is made in a week. For robustness, this is done in two stages. Firstly, any classes which appear more than once are altered such that they appear only once, as shown in Figure 6. Secondly, any classes which did not appear at all are booked to a spare space (regardless of room size, etc) as shown in Figure 7. If this repair strategy is applied to any empty timetable the result is a timetable with each class booked to a random time and place.

```
For each class:
        set the Count to 0.
        for each time:
                for each room:
                        If the current class is booked at this location:
                                Add 1 to the count.
                                Add the location of this class to a linked list.
    If the class occurred more than once then:
                keep doing the following until there is only one booking left:
                        randomly choose one of the bookings.
                        turn it into a NULL booking.
        Free the linked list.
```

Figure 6: Pseudo Code for the first stage of the Repair Strategy.

```
For each class:
For each time:
        Look in each room until either the class is seen or you get to the end.
        If you got to the end without finding the class then randomly find a
                NULL booking and book that class to it.
```

Figure 7: Pseudo Code for the Second stage of the Repair Strategy.

The repair strategy is used to make sure that each class is booked exactly once. Hence, the number of hard constraints which must be considered when timetables are being evaluated is further reduced. The hard constraints "rooms must not be double booked" and "every class must be scheduled exactly once" must be satisfied by non genetic means.

3.9 Evaluation of Timetable

The remaining hard constraints are used in the evaluation of timetables. Each type of the hard constraint will be considered in turn, as shown in the pseudo code of Figure 8. This method could be extended to any amount of hard constraints. Soft constraints

```
For the timetable being evaluated:
    Initialise the cost field to zero.
    For each of the hard constraints:
        Record how many times that constraint is violated.
        Add the count (multiplied by the weighting for that particular constraint)
            to the timetable's cost field.
```

Figure 8: Pseudo Code for Using Multiple Constraints to Evaluate a Timetable.

3.10 Breeding Timetable

Timetables are randomly selected from the population and used for breeding. No favouritism is given to fitter timetables. A child timetable is bred by performing unity order based crossover on the parents [18] (Sandeep Singh Rawat and Lakshmi Rajamani, 2010). This means that each parent has an equal chance of providing each gene. In this paper method of mutation is shown in Figure 9. This means that the chance of any one gene undergoing mutation is approximately twice the mutation rate divided by one thousand.

```
There is a fixed mutation rate. For each gene
{
    Randomly choose a number between 1 and 1000.
    If the number is less than the mutation rate then
    {
        Randomly choose a gene from the current timetable and swap it with the current gene.
    }
}
```

Figure 9: Pseudo Code for Method of Mutation.

IV. COMPUTATIONAL RESULTS

The timetable for a department is a two dimensional array as shown in table 2. Each field describes some event of that particular period as cost or number of breaches. Times at which no class booked hold a NULL booking. The result listed below is applied to the problem of 60 time periods to schedule all courses for diploma and higher diploma specialization in mechanical engineering department. The timetable stores information about which lecturers are booked for concerned classes, at any hour of the day or any day of the week. List of lecturers are placed in column and all events taken place in rows.

Table 2: A solution timetable representing events at various periods.

Lecturer/ Period	**1**	**2**	**3**	**4**	**…**	**10**	**11**	**…**	**p**	**…**	**59**	**60**
Sujit						e					e	
Pramod		e					e					
Shiv Kr.				e								
Shylesh											e	
Subhash		e	e									
Dhanraj			e									
Jawahar				e								e

The result listed above shown in table 2 is applied to the problem of 7 lecturers, 13 courses, 4 rooms, 5 days and 12 periods (5×12=60).

The timetable represented in a two dimensional array A_{ij} where i=1,2, …,K and j= 1,2, …, P such that each element of array represents a 3 tuple $\{R, M, L\}$. A_{ij} can be easily mapped to the weekly lecturer timetable. The column represents individual lecturers and the array's row represents events on particular periods. In above table 60 periods are arranged 1, 2,…,60 and events are represented by

e. Each individual event (e) represents allocation of lecturer to concerned subject (R) of lesson (M) in room (L) for a particular period.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have completed the work according to the need of the department. We have concentrated on course scheduling problems by implementing a genetic algorithm. The program produced optimal timetables void of hard constraint violation and the result were promising. We have tried to show that genetic algorithm is a powerful technique for solving timetabling problem, which we are solving through conventional methods and time period is fixed for faculty and subject later very difficult to change it. These problems of conventional methods can be eliminated by making use of fitness function provided by genetic algorithm. The various genetic operators such as selection, mutation and crossover improve the results in different phases.
This research has some limitation, which can be improved by incorporation of the institute timetable instead of department timetable. A parallel computing approach can be applied to check the large search problems and the quality of solutions can be improved further.

REFERENCES

[1]. Burke E.K.,, De Causmaecker P., Vanden Berghe G. and Van Landeghem H., 2004. "The State of the Art of Nurse Rostering". Journal of Scheduling, **7**(6): 441-499.
[2]. Easton K., Nemhauser G. and Tick M., 2004. "Sports Scheduling. In: Leung JYT (ed.)" Handbook of Scheduling. Chapter 52. CRC Press LLC.
[3]. Carter M.W. and Laporte G., 1996. "Recent developments in practical examination timetabling", International Conference on Practice and Theory of Automated Timetabling I, Springer, Vol. 1153, pp. 1- 21.
[4]. Carter M.W. and Laporte G., 1998. "Recent developments in practical course timetabling", International Conference on Practice and Theory of Automated Timetabling II, pp. 3- 19.
[5]. Burke E.K., Jackson K.S., Kingston J.H. and Weare R.F., 1997. "Automated timetabling: the state of the art", The Computer Journal, 40(9): 565-571.
[6]. Burke E.K. and Petrovic S., 2002. "Recent research directions in automated timetabling", EJOR, 140(2): 266-280.
[7]. Burke E.K. and Newall J.P., 1999. "A Multi-Stage Evolutionary Algorithm for the Timetable Problem", IEEE Transactions on Evolutionary Computation. 3(1): 63-74.
[8]. Carrasco M.P. and Pato M.V., 2002. "A multi-objective genetic algorithm for the class/teacher timetabling problem", International Conference on Practice and Theory of Automated Timetabling III, Springer, Vol. 2079, pp. 3-17.
[9]. Costa D., 1994. "A tabu search algorithm for computing an operational timetable", EJOR, 76: 98-110.
[10]. Gaspero L.D. and Schaerf A., 2000. "Tabu search techniques for examination timetabling", International Conference on Practice and Theory of Automated Timetabling III, Springer, Vol. 2079, pp. 104-117.
[11]. Dowsland K.A., 1998. "Off the peg or made to measure? Timetabling and Scheduling with SA and TS", International Conference on Practice and Theory of Automated Timetabling III, Springer, Vol. 1408, pp. 37-52.
[12]. Colorni A., Dorigo M., Maniezzo V., 1990. "A GenetiC Algorithm to Solve the Time- table Problem", Technical Report No. 90-060, Politecnico di Milano, Italy.
[13]. Colorni A., Dorigo M., Maniezzo V., 1991. "GenetiC Algorithms and Highly Con- strained Problems: The Time-Table Case", In: Schwefel, H.-P.; Männer, R. (eds.): Paral- lel Problem Solving from Nature. Proceedings of the First Workshop PPSN I, Dort- mund, Oct. 1-3. Springer, Berlin: 55-59.
[14]. Burke E.K., MacCarthy B.L., Petrovic S. and Qu R., 2001. "Case-based reasoning in course timetabling: an attribute graph approach", In: Aha DW, and Watson I (eds.): *Case-Based Reasoning Research and Development*. Lecture Notes in Artificial Intelligence 2080. pp. 90-104.
[15]. Dean, J., 2008. "Staff Scheduling by a Genetic Algorithm with a Two-Dimensional Chromosome Structure", Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling. Montreal, Quebec, Canada.
[16]. Ferland J.A., Serge Roy, 1985. "Timetabling Problem for University As Assignment To

Resources", Comput. & Opr. Re., Vol 12, N 2, pp. 207-218.
[17]. Van den Broek J.J.J., and.Hurkens C.A.J., 2010. "An LP-based Heuristic for the post Enrolment Course Timetabling problem of the ITC", Annals of Operation Research, DOI: 10.1007/s10479-010-0708-z.
[18]. Sandeep Singh Rawat and Lakshmi Rajamani, 2010. "A Timetable Prediction for Technical Education System using Genetic Algorithm", Journal of Theoretical and Applied Information Technology, Vol. 13. No.1, pp. 59-64.
[19]. Ashish Jain, Dr. Suresh Jain and Dr. P.K. Chande, 2010. "Formulation of Genetic Algorithm to Generate Good Quality Course Timetable", International Journal of Innovation, Management and Technology, Vol. 1, No. 3, pp. 248-251.
[20]. Ling, S.E., 1992. Integrating GenetiC Algorithms with a Prolog Assignment Program as a Hybrid Solution for a PolyteChniC Timetable Problem", In: Männer, R.; Manderick, B. (eds.): Parallel Problem Solving from Nature, Proceedings of the Second Conference on PPSN, Brussels, Sep. 28-30. North-Holland, Amsterdam: 321-329
[21]. Burke E.K., Jackson K., Kingston J. H. and Weare R.E., 1997. "Automated university timetabling: The state of the art", The computer journal, vol. 40, no.9.
[22]. David E. Goldberg, 1989. "Genetic Algorithms in search, optimization and machine learning".
[23]. Buckles B.P. and Petry F.E., 1992. "Genetic Al algorithms. Los Alamitos", The IEEE Computer Society Press.
[24]. Davis L., 1991. "Handbook of Genetic Algorithms", New York: Van Nostrand Reinhold.

AUTHORS BIOGRAPHY

Sujit Kumar Jha is a Faculty member of Engineering Department at Ibra College of Technology, Ibra, Oman. He has started his career as a Lecturer in Engineering College, India. He Graduated from Govt. Engineering College, Bihar and Post Graduate of Diploma in Thermal Power Plant Engineering from NPTI, Nagpur and M.Tech in Manufacturing Engineering from NIFFT, Ranchi, India. He has over 14 years of industry, teaching and research experience. His major area of research includes Assembly Line Balancing, Scheduling of Operations, Genetic Algorithm, Metal cutting, Production and operation management and Non-traditional Optimization. He has published and communicated more than 24 papers in International Journal. He has presented 12 papers at various International and National conferences

Pramod K Shahabadkar is a head of Training & Placement Cell, K. K. Wagh Institute of Engineering Education & Research, Nashik, India. He graduated in Industrial and Production Engineering and did his post graduation from Gulbarga University, Gulbarga. He was awarded Third rank in the Gulbarga University, Gulbarga in B.E final year examination. He is having a total of 22 years' experience in teaching. He is a Life member of Institution of Engineers (India) and Indian society for Technical education & Indian Institute of Production Engineers-India. He has presented the papers at various International and National conferences.