EXAM TIMETABLING PROBLEM USING GENETIC ALGORITHM

Sujit Kumar Jha[1]

[1]*Engineering Department, Ibra College of Technology, Sultanate of Oman*

Abstract

Timetabling is an extremely difficult problem which is an element of the wider-field of scheduling. Scheduling is generally defined as the problem of the allocation of resources over time to perform a set of tasks and is NP-hard problems. Scheduling exam timetables in departments of college/university is a complex problem. This is usually done by hand taking several days or weeks of iterative repair after feedback from staff and students. This paper proposed an algorithm to solve the problem of exam timetabling, which is a good candidate to generate a timetable using genetic algorithm (GA). This paper details the implementation of a computer program which employs GA for an optimal solution of solving a timetable problem and generate exam timetable by using real student data from engineering department courses at ICT.

Keywords: *Chromosome, Crossover, Genetic Algorithm, Scheduling, Timetable.*

---***---

1. INTRODUCTION

According to Collins Concise Dictionary a 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. Consequently, a timetable specifies which people meet at which location and at what time. Educational timetabling is the sub-class of timetabling for which the events take place at educational institution/university. Examples of events in this sub-class are:

- Test or examinations at college or university (examination timetable)
- Lecturers in course offered at college or university (course/class timetable)
- Meeting between a class and a teacher at college (college timetable)

This paper has considered exam timetable construction problems in engineering department at ICT, Ibra. For these timetabling problems, the events are the subjects. Timetable problems are mainly allocating resources, i.e. invigilators, students, rooms, and time slots, to subjects. Three of these decisions, viz. invigilator assignment, room assignment, and time slot assignment made first to each subject than assigning students to subject. It is impossible for an invigilator or student to attend more than one exam simultaneously. Therefore, every invigilator and student can have at most one subject exam at the same time, this requirement can be considered as invigilator constraint and student constraint. Similarly, room constraints are only satisfied if each room is used for only one subject at a time. For exam timetabling problems, the event is a subject, invigilated by an invigilator to a group of students, in a room. The objective of this research is to schedule subjects to fully utilize available resources by assigning the subject to invigilator at correct time and place to appropriate event. Timetable constraints are many and varied. In this research, genetic algorithm approach has been applied for solving exam timetabling problem. The Figure 1 represents a central concept to which all other concepts, most of them demonstrating the necessary resources, are related.

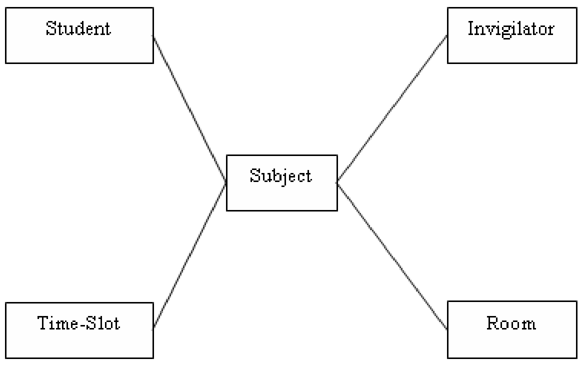


Fig -1: The relationship between different concepts for timetabling.

In section 2, Exam-Timetabling problem description has been depicted in full detail. The genetic algorithm, the underlying chromosome representation, the genetic operators, and the evaluation function are presented in section 3. Some performance aspects of timetabling and computational results are described in section 4. Finally, section 5 presents conclusions.

2. EXAM-TIME TABLING PROBLEM DESCRIPTION

Timetabling problems arise in many real-world circumstances like nurse rostering described by Burke et al. [1], sports timetabling presented by Easton et al. [2] and university timetabling problems by Carter and Laporte [3, 4]. 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 described by Burke et al. [5, 6]. 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 presented by (Burke and Newall; Carrasco and Pato) [7, 8]. Exam-timetabling problem involves scheduling of courses, invigilators, and rooms to a number of time-periods or time-slots in a week. This paper has constructed a timetable for the two year courses like diploma at Ibra College of Technology, Ibra, Oman. The course has divided into three semesters (fall, spring, and summer). In first two semesters the course contains 5 subjects and the third semester contains only 2 subjects. Some subjects should be allocated 2 time period and some 3 time periods like machine drawing. Each day of the week is divided into 2 periods (of 2 hour's duration). There are five working days per week. Hence, the set P of periods consists of 10 elements.

In academic year 2011-2012, engineering department of Ibra college of Technology was one of the first department to introduce automated exam timetable. This paper has presented the exam timetable such that many students and invigilators are happiest overall with this timetable. To create such a timetable will take many days by hand. Simple algorithms performing exhaustic search on real case patterns are also not an option, due to complexity of the problem. This paper has focused on a solution based algorithm known as genetic algorithm detailed describe by many researchers like Carter and Laporte, Burke et al. [4, 5, 6]. This paper has described its concert on a real case scenario, first the algorithm has been trial for academic year 2010-2011. After the successful trial of the algorithm, it has implemented from 2011-2012 and onwards in the department. The paper has considered the following case. There are more than 1400 students-course enrolments with 36 courses giving in average of 25 students in one section of the offered course. Final Exams for 36 courses should be scheduled in a two weeks period. Furthermore, in each day there are two time-slots available (09h – 11h and 12h – 14h), and in each week there are five working days, which gives a total of 2×5×2 = 20 time-slots. A number of possible time-slots with these parameters are roughly bounded by 20^{34}. This clearly indicates that any attempt to solve this problem that is based on exhaustive search is not option.

2.1 Problem Definition

Following are participants of exam timetable:
K invigilators $l_1, l_2, l_3, \ldots, l_k$ and
D days $c_1, c_2, c_3, \ldots, c_d$ 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 $p_1, p_2, p_3, \ldots, p_p$

K is the set of all invigilators (lecturers, technicians, etc.). A subject having duration of 2 hours and denoted by R. L is the set of rooms available in the college for exam. P is the set of all periods in a day of duration 2 hours. This scheduling problem consists of a number of days on which exams can be scheduled is the set of Days D. The problem formulation has been also presented by Rawat and Rajamani, Jain et al. [9, 10], where an assignment is a 5 tuple $\langle l, c, s, r, p \rangle$

2.2 Constraints Involved

In exam timetabling problems, given exams have to be assigned to number of periods such that there is no violation of hard constraints. Hard constraints are those to which timetable has to adhere in order to be satisfied. These are following:

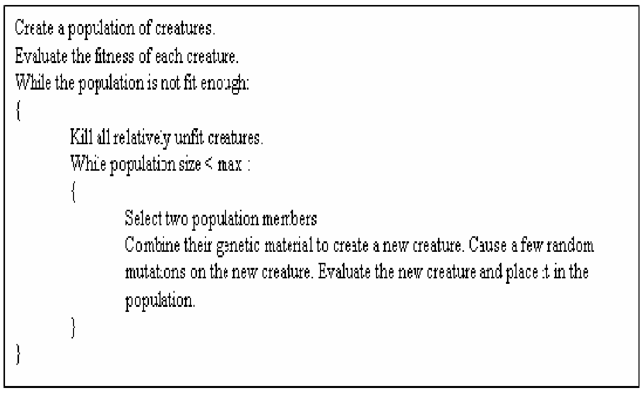
- Each subject is scheduled to exactly one period.
- No student may have two exams in the same period.
- No room should be double booked..
- All allocated rooms are large enough to hold the students.

Violating the above constraints will cause the timetable to be unfeasible. In addition, the paper would also like to satisfy as many soft constraints as possible in order to produce a good quality timetable. Soft constraints for this constrained optimization problem are actually to find a schedule that makes the students and invigilators happiest overall.

This paper has constructed an exam timetable by using 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. A chromosome is made up of groups as genes. It can be represented as: ({1, 3, 5}, {4, 6}, {2}, {8, 9}, …) indicating that the exams of subjects 1, 3 and 5 are scheduled in first period and for subjects 4 and 6 in second period. In that way to avoid building illegal timetables, and are not in need of any 'repair algorithm'. This is in contrast to approaches described in other papers Colorni et al., Ling [11, 12, 13]. A recent approach to the timetable problem is to use the genetic algorithms as a powerful method of solving difficult timetabling problems by Burke et al. [5].

3. 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 stated by Buckles and Petry [14]. 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 Figure 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.
        }
}
```

Fig- 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 described by Colorni et al., Rawat and Rajamani [12, 9]. 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 written in c. 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. Initial population is generated randomly. Figure 3 describes the genetic algorithm working cycle. Initial population is generated randomly.

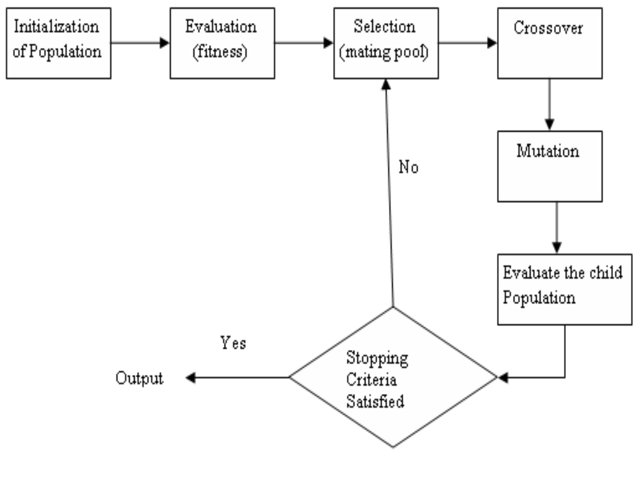

Initialization of Population
Evaluation (fitness)
Selection (mating pool)
Crossover
Mutation
Evaluate the child Population
No
Stopping Criteria Satisfied
Yes
Output

Fig.-3: The Cycle of GA

3.1 Chromosome Representation

In a ‘classical’ genetic algorithm chromosomes are represented as bit strings. However, this paper believes that problem-specific knowledge should be incorporated in the representation of solutions to the timetabling problem, and the chromosome representation should be natural. It should contain all the relevant information and be close to the original problem. In this sense it is straightforward to define a timetable to be a map

$$f : D \times K \times R \times L \times P \rightarrow \{0,1\}$$

Where f (l, c, s, r, p) = 1 if and only if subject r and invigilator l have to meet for an exam in room r at period p. Such a mapping is easily translated into the form timetable (Subject, Invigilator, 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).

Recall that the input data of our system specifies for each lesson a unique teacher. Therefore, we can simplify the definition of a chromosome slightly:

$$f : D \times K \times R \times P \rightarrow \{0,1\}$$

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 described by Evan et al. [15]. For a timetable to be generated, those subjects that are fairly limited as to their possible allocation are considered first. Subjects are selected in random order, and each subject 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 presented by Goldberg [16]. 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 defining the P_{stud} and P_{room} to determine how much each student or room likes their timetable. This paper is assigning a penalty value which contributes to the fitness function for each constraint violated.

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 constraint, and w_i an attached weight.

This paper assigned n_1 (f) = 50 point penalties for inconsistent schedules for students. n_2 (f) = 10 could be the component measuring the point of penalty, when students are assigned multiple exams on the same day. This paper assigned n_3 (f) = 1 when student does not have a rest day between exams. Similarly for rooms inconsistency schedule n_4 (f) = 50 points, n_5 (f) = 10 points for hosting exams in consecutive time-slots and n_6(f) = 1 point for any days without exams between start to end days of exams. Note that this research does 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 constraints should be considered to be more important than others. In this paper we are more concerned with the students than the rooms.

3.4 Selection

Reproduction (or selection) is an operator that makes more copies of better strings in a new population. Reproduction 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 presented by Goldberg and Corne et al. [16, 17].

3.5 Mutation

Mutation adds new information in a random way to the genetic search process 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. 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 L (f, п) is formed. A mutated timetable $f^{'}$ is produced by assigning new periods or rooms only within the 'time window' п and by leaving the rest unchanged. The paper has Considered п and L (f, п), instead of P and L, as input to the problem.
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. N 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.

3.6 Crossover

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 stated by Davis [18]. A child chromosome can be produced using single point crossover, as shown in Figure 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.

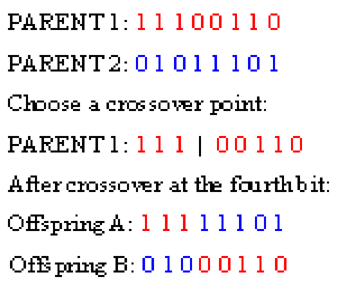

PARENT1: 11100110
PARENT2: 01011101
Choose a crossover point:
PARENT1: 111 | 00110
After crossover at the fourth bit:
Offspring A: 11111101
OffSpring B: 01000110

Fig-4: An Example of Crossover with Encoded Genes

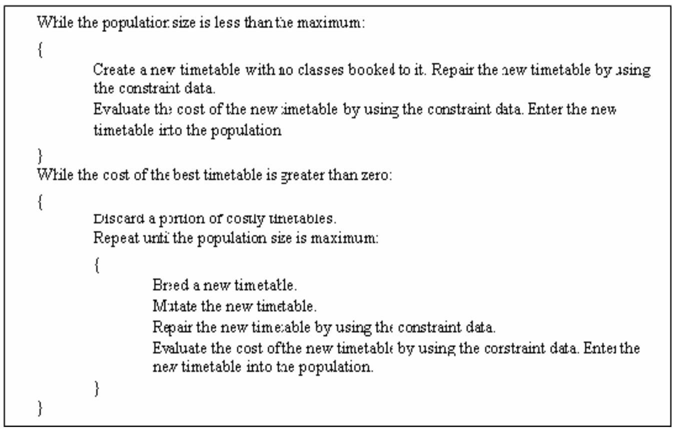
In this research, one site crossover has been used de to its advantages over the conventional method, where it can easily exchanged the information in the timetable and makes it optimal and effective as per the requirements. By selecting two parent timetables f and g, and considered them mother and father respectively and further build offspring in such a way that each invigilator, time-slot 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.

3.7 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 described by Rawat and Rajamani, Michalewics [9, 19]. Thus timetables are evolved with a minimum number of constraint violations. A top level description of the program is provided in Figure 5.

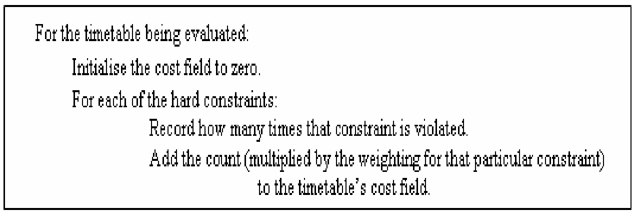


```
While the population size is less than the maximum:
{
        Create a new timetable with no classes booked to it. Repair the new timetable by using
        the constraint data.
        Evaluate the cost of the new timetable by using the constraint data. Enter the new
        timetable into the population
}
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 of the new timetable by using the constraint data. Enter the
                new timetable into the population.
        }
}
```

Fig-5: Top Level Description of Program.

3.8 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 6. 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.
```

Fig-6: Pseudo Code for Using Multiple Constraints to Evaluate a Timetable.

4. COMPUTATIONAL RESULTS

The exam timetable can be represented in a two dimensional array. The array's column represents individual invigilators and the array's row represents events on particular periods. In the Table 1, periods are arranged 1, 2,…, 20 as there are 20 periods in two weeks exam and events are scheduling of all courses for diploma and Higher Diploma specialization in engineering department to 54 invigilators. Each individual event represents allocation of invigilators to which room and which subject represented by R/S. For Example R2/S5 in period 5 represents third day first period in room 2 and subject 5 of invigilator allocation for that period. All subjects are arranged in table with serial number for simple representation in the table. For every subject exam there are two invigilators duty placed in a room in the same period. The results of the program has been presented in Table 1, by considering the problem of 54 lecturers, 36 courses, 12 rooms and 20 periods for end semester exam.

Table – 1: A Solution Timetable representing events at various periods.

S. No.	Lecturer / Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Sujit		R8/ S2			R2/ S5		R2/ S8					R2/ S15					R9/ S29			
2	Pramod	R3/ S1			R12 / S9							R9/ S16								R11/ S19	
3	Shiv Kumar		R8/ S2			R2/ S5				R8/ S18											
4	Shylesh	R3/ S1								R9/ S20								R5 1/S 33			R9/ S31
5	Subhash			R3/ S13			R5/ S18							R8/ S21		R9/ S32					
6	Dhanraj		R6/ S7					R5/ S23											R7/ S30		
7	Nitin								R7/ S1			R9/ S16			R9/ S34						R9/ S31
.																					
.																					
.																					
53	Amit	R4/ S2		R1/ S3						R4/ S14								R9/ S29			
54	Seetharam				R12 /S9			R2/ S8					R2/ S15							R11/ S19	

5. CONCLUSIONS

This research has concentrated on exam scheduling using a genetic algorithm for solving this problem. The paper has tried to show that genetic algorithm is a powerful method for solving timetabling problem, which are solving through conventional methods and time period is fixed for invigilators and subject. 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 enhance the results in various aspects such as through selection we can select the best chromosomes on the basis of fitness function from the pool of chromosomes and similarly through crossover exchange of the information of the timetable as per requirements. The future work of this research will attempt to use this genetic approach technique for solving College real-world exam timetabling problems.

REFERENCES

[1] Burke E.K., De Causmaecker P., Vanden Berghe G., Van Landeghem H., "*The State of the Art of Nurse Rostering*," Journal of Scheduling, Vol. 7(6), 2004, pp. 441-499.

[2] Easton K., Nemhauser G., Trick M., Sports Scheduling, Handbook of Scheduling: algorithms, models and performance analysis, CRC Press, 2004.

[3] Carter M.W., Laporte G., "*Recent developments in practical examination timetabling*," Practice and Theory of Automated Timetabling, Vol. 1153, 1996, pp. 1-21.

[4] Carter M.W., Laporte G., "*Recent developments in practical course timetabling*," Practice and Theory of Automated Timetabling, Vol. 1408, 1998, pp. 3-19.

[5] Burke E.K., Jackson K.S., Kingston J.H., Weare R.F., "*Automated timetabling: the state of the art*, " The Computer Journal, Vol. 40(9), 1997, pp. 565-571.

[6] Burke E.K.. Petrovic S, "*Recent research directions in automated timetabling*," European Journal of Operational Research, Vol. 140(2), 2002, pp. 266-280.

[7] Burke E.K., Newall J.P., "*A Multi-Stage Evolutionary Algorithm for the Timetable Problem*," IEEE Transactions on Evolutionary Computation, Vol. 3(1), 1999, pp. 63-74.

[8] Carrasco M.P., Pato M.V., "*A multiobjective genetic algorithm for the class/teacher timetabling problem*," Practice and Theory of Automated Timetabling III, Vol. 2079, 2001, pp. 3-17.

[9] Rawat S.S., Rajamani L., "*A Timetable Prediction for Technical Education System using Genetic Algorithm*," Journal of Theoretical and Applied Information Technology, Vol. 13(1), 2010, pp. 59-64.

[10] Jain A., Jain S., Chande P.K., "*Formulation of Genetic Algorithm to generate good quality course timetabling*," International Journal of Innovation Management and Technology, Vol. 1(3), 2010, pp. 248-251.

[11] Colorni A., Dorigo M., Maniezzo V., "*Genetic Algorithms-A new approach to the Timetable Problem*," NATO ASI Series, Combinatorial Optimization, Lecture Notes in Computer Science, F(82), 1990, pp. 235-239.

[12] Colorni A., Dorigo M., Maniezzo V., "*Genetic Algorithms and highly constrained problems: The time-table case*," Parallel Problem Solving from Nature, Vol. 496, 1991, pp. 55-59.

[13] Ling S.E., "*Integrating Genetic Algorithms with a Prolog Assignment Program as a hybrid Solution for a Polytechnic Timetable Problem*," Parallel Problem Solving from Nature, Vol. 2, 1992, pp. 321-329.

[14] Buckles B.P., Petry F., Genetic Algorithms, The IEEE Computer Society Press, 1992.

[15] Evan S., Itai A., Shamir A., "*On the Complexity of Timetable and Multi Commodity Flow problems*," SIAM Journal of Computing, Vol. 5, 1976, pp. 691-703.

[16] Goldberg D.E., Genetic Algorithms in search, Optimization, and Machine Learning, Addison-Wesley Professional, 1989.

[17] Corne D., Fang H.L., Mellish C., "Solving the Modular Exam Scheduling Problem with Genetic algorithms," Proceeding of the 6th International Conference on Industrial and Engineering Applications of artificial intelligence and expert systems, 1993, pp. 370-373.

[18] Davis L., Handbook of Genetic Algorithms, New York, Van Nostrand Reinhold, 1991.

[19] Michalewics Z., Genetic Algorithms+Data Structure = Evolution Programs, 2nd Edition, Springer, 1994.