Solving Timetabling problems using Genetic Algorithm Technique

H. M. Sani
Department of Mathematics,
Computer Science Unit,
Usmanu Danfodiyo University,
P. M. B 2346, Sokoto-Nigeria

M. M. Yabo
Department of Computer Science,
Shehu Shagari College of Education,
P.M.B. 2129, Sokoto-Nigeria

ABSTRACT
The timetabling problem is always a difficult task which comes up every calendar year in educational institutions. More especially if it has to be done manually. Various institutions of learning across the country are being faced with a lot of difficulties preparing examination timetable. Most of this problems are usually attributed to the constant increase in the number of students and courses while having limited resources (exam classes) to use in scheduling. The aim of this paper is to propose the use of genetic algorithm technique to develop an easier, effective and efficient timetable using in order to ease the problems faced during scheduling of examination. Although, there are various scheduling techniques, but the use of Genetic Algorithm was based on the fact that, the algorithm are robust there by properly fits into complex problem space. The new method is aimed at providing a more flexible timetable representation and proved to be efficient in real life applications..

General Terms
Computer- Genetic Algorithm

Keywords
Timetable scheduling, Genetic Algorithm, Constraints

1. INTRODUCTION
Timetabling problem is a special case of scheduling that has to do with allocating time slots to constraint [1]. It is one of the common scheduling problems, which can be described as the allocating of resources for events under predefined set of constraints so that it maximises the possibility of allocation of the available resources or minimises the violation of the constraints [2]. Timetabling scheduling problems are often complicated and difficult task in educational institutions, especially in higher institutions like the universities and colleges of educations. In institutions timetabling problem, the number of courses offered have to be scheduled into a number of time slots and venues/halls available while satisfying various constraints. Timetabling has become much more difficult to find the general and effective solution due to the diversity of the problem, the variance of constraints, as well as particular requirements from one institution to another. Scheduling examination timetable in Shehu Shagari College of Education (SSCOE) has always been a difficult task and the staff involved in the scheduling process often experience a lot of difficulties due to the large number of students and the complex nature of the courses being offered in the college. One major problem that contributes to the difficulties faced by the institution over the years is mostly attributed to the diverse nature of the courses being offered, constant increase in the number of students ,the limited resources(examination halls) available and above all, the scheduling is done manually. However, due to these difficulties being faced . the paper is aimed at proposing the use of scheduling component of Genetic Algorithm (GA) to analyze the timetabling scenarios of SSCOE so as to find a satisfactory ,effective and more efficient way of scheduling exams timetables with limited resources available and thereby eliminating the issue of exam clashes. The main constraint in this problem is that, no students should take 2 exams at the same time(i.e. there should be no clashes) . The paper is organized in 5 sections as follows: Section 1 provides an introductory aspect of the paper. Section 2 provides an overview of timetabling problems and Genetic Algorithm technique. In section 3, the method and materials used in the proposed timetabling was presented. The result and discussion carried out in this work was presented in section 4. And section 5 presents the conclusion and future recommendations.

2. TIMETABLING PROBLEM AND GENETIC ALGORITHM

2.1 Timetabling
Timetabling problems arise in many forms, most usually in an educational context, and basically involve resource allocation. It is essentially a special case of resource scheduling that deals with the allocation of time slot which must suit a number of constraints. These constraints are almost universally employed by people directly dealing with timetabling problems [1]. Constraints are almost universally broken into two categories: soft and hard constraints. Hard constraints are constraints, of which, in any working timetable, they cannot be violated. For example, a student cannot be in two places at once [2]. Soft constraints are constraints which may be violated, but of which breaches must be minimized. For example, exams should not be scheduled too close so the students can have more intervals for preparation.

A simple example of examination timetabling include: a set of examinations, a set of rooms and a set of available time periods, together with a list of the students who take each of the examinations. The objective is to allocate a room and a time to each examination, subject to various constraints. Example, clashes must be avoided by scheduling every pair of examinations to be taken by the same student at different times. Moreover, if two examinations are taken by the same set of students, they must be separated in time, in order to allow the students sufficient preparation time in between.

Several researchers have shown the used GAs to solve timetabling problems [3][11][5][6][4][7].

SSCOE runs a range of courses covering the various schools and departments . Each student belongs to a School and particular department. Each particular department runs a numbers of courses during each semester and each course has an examination at the end of a semester. The College Examination Office(CEO) is responsible for creating an

examination timetable for each department . All examination are timetabled over a 2 weeks period and normally scheduled on Monday-Saturday . During the scheduling stage, CEO allocate a time slot to each examination. There are three possible time slots (08:00-10:00 ,12:00-02:00 and 04:00-06:00) on each day during the examination period. This gives a total of 36 available slots Monday – Saturday. The main difficulty that CEO have is with the scheduling phase. It is important to ensure that no student is asked to sit two exams in the same slot, i.e. that exams do not clash. It is also preferable that students are not asked to sit two exams on the same day. SSCOE has recently encouraged the use of electives which allow students some choice of courses. Electives cause a problem in examination timetabling because it increases the number of potential examination clashes as electives will typically be available to students studying many other courses.

2.2 Genetic Algorithm(GA)

Genetic algorithms are a class of powerful and general purpose search algorithms based which model problems based on the principles from Darwin biological evolution [8][9]. Since its formulation, it has been growing in use, particularly for solving combinatorial problems with large irregular search space of possible solution [10][7][11]. They are often capable of finding optimal solutions even in the most complex of search spaces. They operate on population of coded parameter space to search for an optimal solution by combining set of genetic operator (crossover) or altering (mutating) current individuals [9][12]. The fitness is derived from the problem objective function. The population of a GA is a subset of a larger set of individuals whose members include all the possible solutions to the problem . Initial population is generated randomly. The basic Genetic Algorithm working cycle as well as pseudo code is shown in figure 1 and 2 respectively.

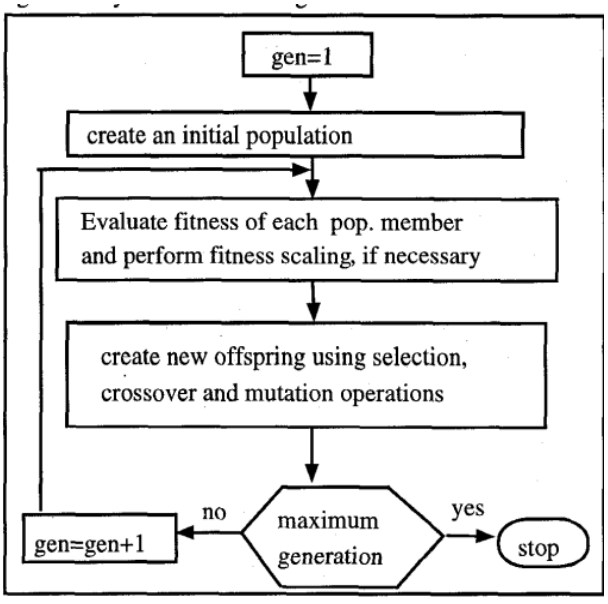


Figure 1. Basic Genetic Algorithm working cycle

InitializePopulation P
for each sol_i from P
calculateFitness (sol)
repeat
select two parents $sol1$ and $sol2$ from P
$child$ = crossover(sol_1, sol_2)
mutate($child$) calculateFitness($child$)
replaceSome(P,$child$)
until stop condition not reached

Figure 2. Pseudo code of genetic algorithm

The major steps involved in the GA circle are ; the generation of a population of solutions, finding the fitness function and the application of genetic operators(selection, crossover and mutation) .

2.2.1 Population Size and Initialisation

A genetic algorithm is a population based search technique that derives its power from the fact that it advances its search based on feedback obtained from a number of potential solutions to the problem, each searching different regions of the search space at the same time [12] . The initial population of solutions is usually generated randomly. The population of a GA is a subset of a larger set of individuals whose members include all the possible solutions to the problem. Hence, the size of the population is one of the major GA control parameters. [12].

The initialization procedure creates at random a population of feasible solutions 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.

2.2.2 Fitness function/Evaluation

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. A genetic algorithm conventionally searches for the optimal solution by maximising a given fitness function, and therefore an evaluation function which provides a measure of the quality of the problem solution must be provided.

2.2.3 Selection operator

The selection step of the GA cycle is the process of determining the number of copies of each individual parent that can participate in the reproduction or mating process. The selection operator is applied to the population to bread a new generation. Individual solution are selected via the chromosome fitness process. There are several ways of implementing the selection mechanism. Some examples include: stochastic remainder selection, roulette wheel' selection and tournament selection [12]. In this paper, tournament selection method is used . Fitness scaling is

usually applied to the fitness values to prevent premature convergence, which is caused by a lack of diversity in the population due to a decrease in the variance of fitness.

2.2.4 Crossover and mutation operators

The crossover operator is mainly responsible for the global search property of the GA. The operator basically combines the substructures of two parent chromosomes from either the mother or the father to produce new structures based on a chosen probability. The process of combining the genes of the parent can be done in a number of ways. The most commonly used crossover methods are one point, two point and uniform crossover [12] [7] . The simplest method of combination is the single point cross over which is the chosen method used in this paper. In this method, a child chromosome can be produced from its parents as shown in figure 3. Here, a crossover point is randomly chosen from any point in the string of genes and all the genetic material before the crossing point is taken from one parent and the other is taken from the other parent after the crossing point to form children. While on the other hand, the main reason for using the mutation operator is to prevent the permanent loss of any particular bit value, as without a mutation there is no possibility of re-introducing a missing bit value.

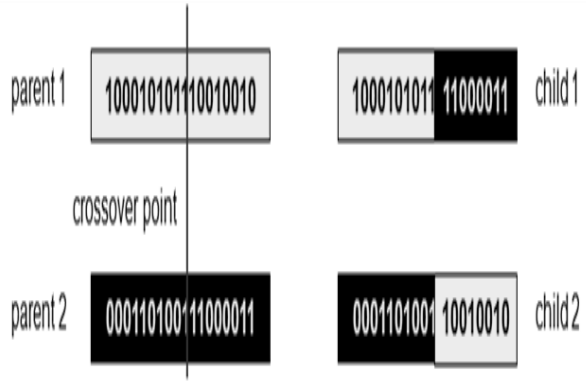


Figure 3. One-point Crossover structure

3. MATERIALS AND METHODS

3.1 Materials used

The tool used to carry out the experiment in this study is the ECJ toolkit. The ECJ as a GA Toolkit**,** is a software system for developing GAs that provides most of the standard components needed. The ECJ toolkit is being developed in Java by Sean Luke and others in the Evolutionary Computation Laboratory at George Mason University, USA. The ECJ was repackaged as a JAR file and enables the development of the new GA in a Net beans project environment.

3.2 Approach

3.2.1 Problem Encoding

An important characteristic of genetic algorithm is the coding of variables that describes the problem. The most common coding method is to transform the variables to a binary string or vector. This initial population formulation

process is critical. This step is also recognized as encoding process. The timetabling problem involves allocating slots to each of the exams .The time slots to each of the exam is an array of integers e.g. 1-20 or 1-24.The number of exams to be scheduled or allocated to specified time slot is also to be represented in an array of integers e.g. 1-40,1-100 and 1-200 exams. Since both the solution (time slots) and the exams to be scheduled can be represented as an array of integer numbers , therefore the encoding chosen for this problem is "integer encoding". The main constraint in this problem is that ,no students should take 2 exams at the same time(i.e. there should be no clashes).

3.2.2 Fitness Function

To apply GA to any problem, both the representation and evaluation function for the solutions most be specified. The evaluation function is used to evaluate the solution in order to get an optimal solution as described in the following code

```
int total = 0;
    IntegerVectorIndividual ind2 = (IntegerVectorIndividual)ind;
    for(int x = 0; x < ind2.genome.length; x++){
     for(int y =1; y < ind2.genome.length; y++){
      if( ind2.genome[x] == ind2.genome[y])
         total += (nodeClash(x,y) ? 1:0) ;
     }
     ((SimpleFitness)ind2.fitness).setFitness (state,
           // ...the fitness...
          //(float)((float)total/(float)ind2.genome.length),
           /// ... is the individual ideal?  Indicate here...
           (float)((1f/((float)1+(float)total)),
         total == 0);
      ind2.evaluated = true;
```

Figure 4. The Fitness Function code

In the above code, the variable total is set to zero. It is used to store the fitness of the solution(individual).The for loop performs the fitness testing on the exams data . At each position of x and y we decide to add 1 or 0 to the fitness. If x equal y and the position is 1, total is incremented indicating that there is a clash, if x equals y and value equals zero then total increments by 0 indicates that there is no clash. The fitness is then taken as 1/1+total this is to avoid dividing by zero. Here, total needs to return zero which indicates that there is no clash and therefore the optimal solution will return the fitness to be equal to 1.That is, for an ideal solution to be found ,fitness function is equal to 1.

3.2.3 Experimental Configuration

The testing was performed using sampled exams data of college of education located in Sokoto. The sampled data were grouped into three(3) different examination scenario data files to schedule 40, 100 and 200 exams respectively using different range of slots respectively. The 3 data files named: examination data 1,2,3 respectively. In each of the file, contain the no of courses(events) , total time slots available for the scheduling. As mentioned above, interger encoding was chosen for the representation of the problem. Such that the courses are represented as $C=\{c_1, c_2, c_3.....c_n\}$ to be scheduled on different time slot also represented in integer as $T=\{t_1,t_2,t_3,.....t_n\}$. Different parameters were explored in the experiment for the 3 instances of the examination data files. Figure 5 below shows some few parameters that were explored.

Operators	Range/Percentage
Population size	100-400
Selection methods:	Tournament, Fitness proportion selection and greedy selection
Crossover Type:	One, two and any
Crossover Rate:	0.5-1.0
Mutation Rate	0.001-0.1
Elitism	5-10
Selection size	2-4

Figure 5. Experimental Configuration Parameters

The figure above shows the different operators/parameters explored during the experiment. Elitism here means that some of the best solution (individuals) will always be copied into the next generation before selection. This is used because there is no guarantee that the best individual in one generation will appear in the next and this also helps to speed the process in finding the optimal solution than when compared to experiments without elites. Crossover types and crossover, mutation rates indicated in the table above were used to performed experiment with the three different selection methods on the 3 scenarios. At the end of the experimental exploration , crossover type one with rate 0.7 and mutation rate of 0.003 produced the best result using tournament selection method for the three scenarios and therefore they were used throughout the experiment. The selected final configurations used for each of the 3 scenarios are described in figure 7a,b, and c below

- pop.subpop.0.size	*= 200*	*population size set*
-pop.subpop.0.species.genome-size	*= 40*	
-pop.subpop.0.species.min-gene	*= 1*	
-pop.subpop.0.species.max-gene	*= 9*	
- pop.subpop.0.species.crossover-type	*= one*	
-pop.subpop.0.species.crossover-prob	*= 0.7*	
-pop.subpop.0.species.mutation-prob	*= 0.003*	
- select.tournament.size	*= 4*	
-breed.elite.0	*=5*	*Elitism/elite set to 5*

Figure 6a. Parameters for Examination data file2

-pop.subpop.0.size	*= 400*	*i.e population size set*
-pop.subpop.0.species.genome-size	*= 100*	
-pop.subpop.0.species.min-gene	*= 1*	*minimum slot set*
-pop.subpop.0.species.max-gene	*= 17*	*maximum slot set*
- pop.subpop.0.species.crossover-type	*= one*	
-pop.subpop.0.species.crossover-prob	*= 0.7*	
-pop.subpop.0.species.mutation-prob	*= 0.003*	
-select.tournament.size	*= 4*	
-breed.elite.0	*=5*	*Elitism/elite set to 5*

Figure 6b. Parameters for Examination data file2

- pop.subpop.0.size	*= 400*	*population size set*
- pop.subpop.0.species.genome-size	*= 200*	
-pop.subpop.0.species.min-gene	*= 1*	*minimum slot set*
-pop.subpop.0.species.max-gene	*= 20*	*maximum slot set*
- pop.subpop.0.species.crossover-type	*= one*	
-pop.subpop.0.species.crossover-prob	*= 0.7*	
-pop.subpop.0.species.mutation-prob	*= 0.003*	
-select.tournament.size	*= 4*	
-breed.elite.0	=5	Elitism/elite set to 5

Figure 6c Parameters for Examination data file3

The same crossover rate of 0.7, mutation of 0.003 and selection method was used to optimized the 3 examination scenarios.

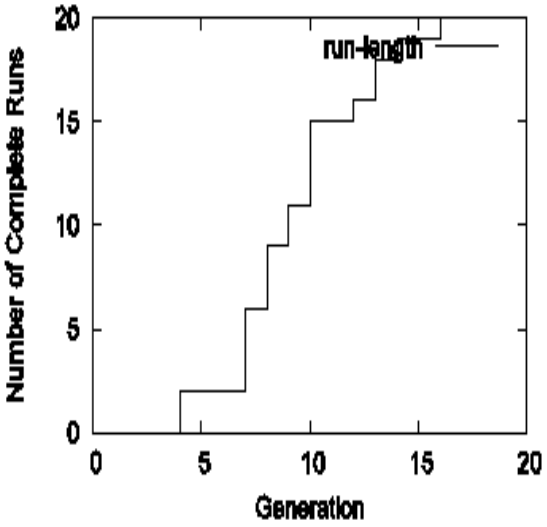
The examination data file1 finishes at about generation 16 with the 200 generations used, the examination data file2 finishes at about generation 51 with the 400 generations while the examination data file3 finishes at about generation 47 with the 400 generation used. Observations was also made with increment on selection size to 4 and adding 5 % elitism because it makes the search more efficient to generate an optimal solution than when compared with experiment without elitism and selection increment.

4. RESULTS AND DISCUSSION

This section describes the results produced by the GA with the different parameters used in running the experiments.

4.1 Examination data file1

The experiment carried out on the examination data file1 shows that a minimum slot of 9 can be used to schedule the exams without the exams clashing.



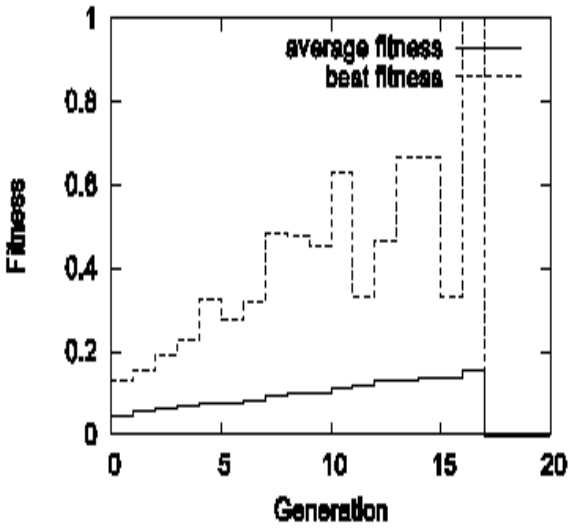


Figure 7a and b. Examination data file1 Output graphs

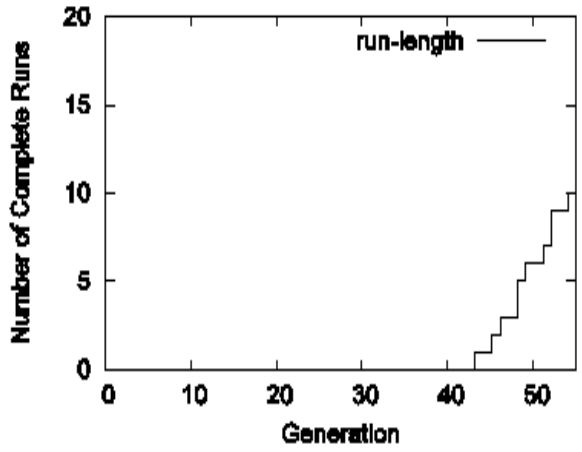
The diagram shows that the algorithm performed well on the 40 timetable exam data file. This is because the fitness rises to the highest fitness of 1 and showing that a more viable time table has been achieved and even with a much fewer slots than expected. Figure 7c shows the output showing the 9 slots used for the scheduling of 40 exams.

5 6 8 4 9 1 8 9 2 8 9 8 7 5 8 7 2 6 4 2 1 6 3
4 6 3 7 3 8 4 5 3 2 8 3 5 7 9 6 1

Figure 7c . Sample Output of (1-9) slots used to Schedule 40 exams

4.2 Examination data file2

The experiment shows that, the core courses can also be scheduled in a much faster time and lesser time slots since all the exams can be schedules using 17 slots without clashes.



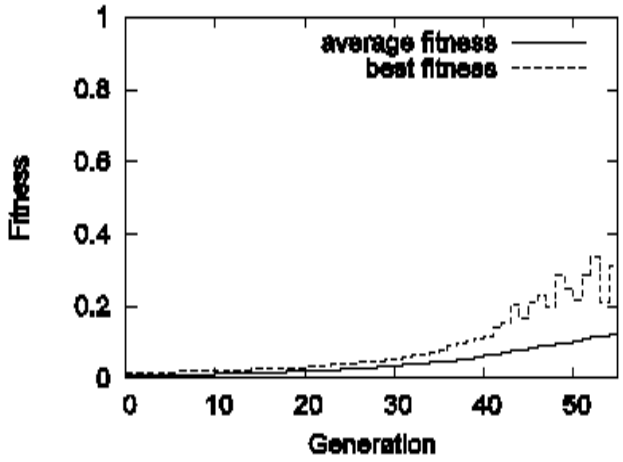


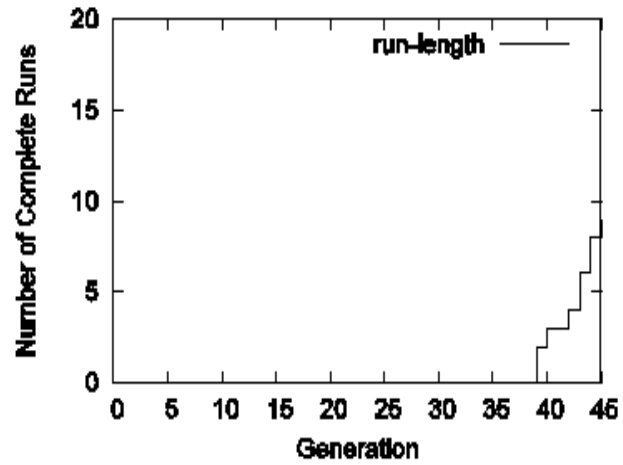
Figure 8a and b. Examination data file2 Output graphs

2 11 3 12 17 1 4 8 8 11 13 6 9 11 10 8 1 5 15 14
16 2 3 12 9 4 2 9 6 14 2 7 2 5 16 8 5 16 8 4 17 13 5
16 5 11 4 15 9 17 3 8 7 1 6 1 14 6 3 7 13 14 17 17 7 8
10 13 15 12 17 11 14 10 9 17 10 12 10 1 16 7 2 5 17
14 13 9 5 2 6 1 7 2 15 13 11 12 11 13

Figure 8c . Sample Output of (1-17) slots used to schedule 100 exams

4.3 Examination data file3

The experiment shows that, even with a minimum number of 20 slots the exams can be scheduled. This shows that the exams can still be scheduled without clash according to the experiment if the courses were introduced. The only implication it has is that the exams will be scheduled too close and students will not have much time to study in between exams.



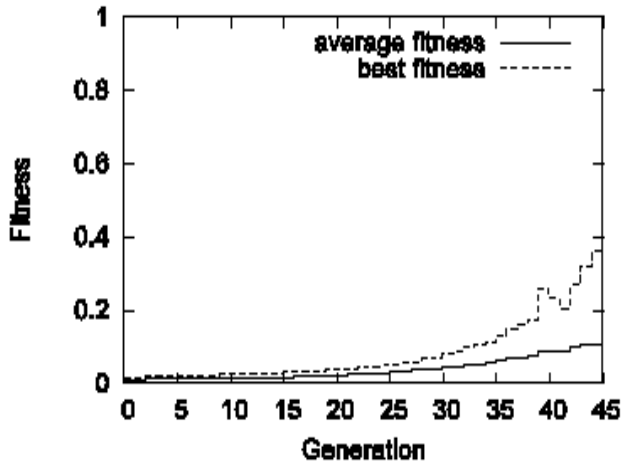


Figure 9a and b. Examination data file3 Output graphs

3 13 9 17 7 19 9 3 20 2 15 20 11 8 5 4 6 13 10 11 18 17 10
2 6 4 13 11 6 7 8 12 10 14 13 12 16 8 19 2 16 5 8 7 1 14 13 14
17 9 19 20 11 17 12 18 6 19 3 20 20 3 4 11 18 8 2 9 3 19 6 14 3
17 12 9 4 18 1 15 6 8 4 8 17 1 6 3 5 14 5 10 1 6 13 1 4 4 1 6 2
20 14 19 11 12 7 1 3 15 3 6 7 10 2 12 19 4 6 16 9 16 5 16 14 6
10 7 13 17 8 4 7 15 16 15 11 18 18 7 5 14 7 3 19 8 3 5 8 19
15 18 17 19 12 14 8 1 8 11 3 7 4 7 13 11 13 9 13 5 1 15
11 12 6 6 10 18 12 14 20 6 1 7 12 20 15 14 10 4 2 18 15 7 16
17 2 2 7 19

Figure 9c. Sample Output of (1-20) slots used to schedule 200 exams

5. CONCLUSIONS

Exam timetable schedules is indeed a much difficult task in any institution of learning especially if it is manually prepared. It can take days and even weeks to prepare. This study proposed the use of genetic algorithm approach/technique to solving timetable problem .

Although, the experimental result indicates that with appropriately configured Genetic Algorithm , a more efficient and reliable timetable scheduling can be achieved. This will provide a good examination timetables that will not have any clashing exams and in a much faster time. However, there are some limitations in this study. Firstly, the fitness does not penalise for two exams on the same day for as long as the exams doesn't clash. .Secondly, the study only tries to take care of hard constraint (i.e no clashing of two exams) as a result students might end up writing two exams on the same day. The future work is to extend the study to include solving soft constraint that will make sure that no student should write two exams too close.

6. REFERENCES

[1] Burke, E. and Ross, P. (Eds) 1996. Lecture Notes in Computer Science 1153 Practice and Theory of Automated Timetabling First International Conference, Edinburgh, U.K., Selected Papers. New York: Springer-Verlag Berlin Heidelberg.

[2] Thanh, N. D.,2006. Solving timetabling problem using genetic and heuristics algorithms Journal of Scheduling,9(5): 403–432, 2006

[3] Erben, W.,and Keppler, J., 1995. A genetic algorithm solving a weekly course timetabling problem. Proc. of the 1st Int. Conf. on Practice and Theory of Automated Timetabling, LNCS 1153, pp. 198-211, 1995.

[4] Lewis, R., and Paechter, B., 2005. Application of the Grouping Genetic Algorithm to University Course Timetabling Proc. of the 5th European Conf. on Evol. Computer in Combinatorial Optimization (EvoCOP 2005), LNCS 3448, pp. 144-153, 2005

[5] Abdullah, S., and Turabieh, H., 2008. Generating university course timetable using genetic algorithm and local search. Proc. of the 3rd Int. conf. on Hybrid Information Technology, pp. 254-260, 2008.

[6] Pongcharoen, P., Promtet, W., Yenradee, P.,and Hicks, C, 2008. Schotastic Optimisation Timetabling Tool for University Course Scheduling. International Journal of Production Economics, 112: 903-918, 2008.

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

[8] Davis, L. (1991) "Handbook of Genetic Algorithms" Van Nostrand Reinhold

[9] Negnevitsky, M. 2005. Artificial Intelligence, A Guide to Intelligence System (2nd ed.), Addison Wesley, pp. 222-245, IBN 0-321-20466-2, Harlow, England

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

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

[12] Zhao, Q. 2007. An Introduction to Genetic Algorithms.