Article Citation Format
Ezike J.O.J. & Uwagbai D.O. (2020):
Automated Examination Timetabling with Genetic Algorithm:
Object-Orientation and Other Considerations
. Journal of Digital Innovations & Contemp Res. In Science., Engineering &
Technology. Vol. 8, No. 2. Pp 29-48

Article Progress Time Stamps

Article Type: Research Article
Manuscript Received: 21st April , 2020
Review Type: Blind
Final Acceptance: 19th June, 2020

Automated Examination Timetabling with Genetic Algorithm: Object-Orientation and Other Considerations

***Ezike J.O.J. & Uwagbai D.O.**
[1]Dept. of Computer Science and Information Technology,
Bells University of Technology
Ota, Nigeria
*** Corresponding Author Email**: joezike@bellsuniversity.edu.ng
Phone: +2348035057550

ABSTRACT

Genetic Algorithm (GA) is one of the leading approach for solving the Examination Timetabling Problem (ETP), however, how the GA-chromosome is represented and the crossover operator used have been noted to have impact on the solution process and quality. Following an object-oriented approach, this research represents the chromosome as composite objects, and adapted a hybrid heuristic crossover operator for solving the ETP in Bells University of Technology, Ota. Modeled as an optimization problem using constraints gathered from the university and implemented using Java, the ETP was solved following a three-stage process, which involve the optimization of the generated timetable using the GA. The GA was investigated on some key parameters to determine their impact on the generated timetable quality and evaluated using first Order Conflict Counts (OCC) and second OCC for students and invigilators respectively, and using its space complexity. The GA yielded average first OCC for both students and invigilators of 0.0 and 0.0. Similarly, GA yielded best second OCC of 356, and average of 454.78 for students. The average second OCC for invigilators is 0.0. The GA recorded empirical space complexity of O(n). When compared with previously obtained results on the dataset used, the GA's performance was poor; this was observed to be due to the crossover operator implemented, the key search operator of the GA, and not to the representation used. For solving large combinatorial optimization problems like the ETP, it is recommended that more effective crossover operator be considered.

Keywords: Genetic Algorithm, Object-Oriented, Automated Timetabling, heuristic crossover, optimization

1. INTRODUCTION

The manual university examination timetable scheduling process consumes lots of time and resources in its preparation, yet the output is usually not satisfactory to all the parties concern [1, 2]. This has led to extensive research efforts in automated timetabling as noted in [3-7], with the goal of producing quality timetables that meets the need of all stakeholders while effectively utilizing system resources.

Genetic Algorithm (GA), a population based technique, is among the leading paradigms for solving the ETP. This is because GAs delivers good enough solution in very reasonable time, making GAs attractive for use in solving optimization problems [8, 9]. GA solutions to a problem are encoded as chromosomes. In literature, different approaches has been used for chromosome representation for the ETP; these approaches have been classified as direct or implicit [10]. The direct representation encodes the actual timetable while the implicit encodes a set of instructions as to how the timetable should be built. However, how the GA-chromosome is represented have been noted to have impact on the solution process and quality [11]. In [8], it was also noted that most of the success or otherwise of the GA application comes from the representation used, since this determines the kinds of operators that might be chosen. In literature so far consulted on Examination Timetabling with GA, different schemes have been adopted in representing the GA chromosome for problem solving. These schemes includes Binary/Bit String representation, Real-Valued representation, Integer Representation, Permutation Representation, two-dimension (m x n) array of integer numbers as note in [11]. No literature consulted so far was observed to have used an object-oriented chromosome representation.

This research aim at solving the ETP using GA with an object-oriented chromosome representation, and observes the effect of such representation on the GA's performance.

2. REVIEW OF RELEATED WORKS

2.1 The Examination Timetabling Problem

A number of researchers have noted the Examination timetabling problem (ETP) as a well-known NP-Complete combinatorial problem present in universities with a large number of students and courses, especially if the courses are many and the student can choose from a wide range of electives [12-14]. Operations researchers have "identified the Examination timetabling problem as a scheduling problem with disjunctive and cumulative conjunctive constraints, classified as NP-complete, for which no classical operations research (OR) approach is directly applicable."[15]. The complexities and the challenges posed by the timetabling problem arise from the fact that a large variety of constraints, differing from institution to institution, some of which contradict each other, need to be satisfied in different institutions [14, 16]. The constraints fall into one of two category**:** Hard and Soft. Fulfilling all hard constraint make the generated Timetable feasible. The more the soft constraints fulfilled the more desirable the timetable becomes. Determining the quality of timetables generated is done by an objective function, formulated using the hard and soft constraints stipulated by the institution and implemented in the timetable generating algorithm.

2.2 Genetic Algorithm

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics; they mimics the evolutionary process in nature by manipulating and evolving a population of solutions within a search space so as to obtain optimized solutions to a given search problem over a number of generations [17, 18]. GA candidate solutions to a search problem encode the decision variables of the search problem into finite-length strings of alphabets of certain cardinality. The strings are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles [19].

2.3 GA and Search Space Exploration

GAs does not require knowledge of the problem domain during search space exploration except when a heuristic operator is used; however, it requires an objective function to evaluate the solution. Accumulated information is explored by the parent selection mechanism while genetic operators are used to explore new regions of the search space. Conventionally, the crossover or recombination operator is the principal operator and determines the performance of the GA. Carrying out exploitation, the operator seeks better solution from selected parents.

The mutation operator, operating as a background operator, is used to produce spontaneous random changes in chromosomes [20], thereby keeping the algorithm from converging on a popular solution [21]. However, when the population has substantially converge and crossover no longer has much effect, the mutation is used to perform extensive random search [11].

2.4. Advantages and Limitations of GA

Advantages that made GAs very popular include the following:

i. GAs does not require any derivative information (which may not be available for many real-world problems) [18, 22].
ii. GAs is faster and more efficient as compared to the traditional methods [8, 9]
iii. GAs have very good parallel capabilities [23, 24].
iv. GAs optimizes both continuous and discrete functions and also multi-objective problems [18, 24, 25] .
v. GAs provides a list of "good" solutions and not just a single solution [8, 11].
vi. GAs always gets an answer to the problem, which gets better over the time.
vii. GAs are useful when the search space is very large and there are a large number of parameters involved [26].
viii. GAs are robust, efficient and very general in their application and use [18, 27]

However, GAs suffers from the following limitations:

i. GAs are not suited for all problems, especially problems which are simple and for which derivative information is available [18, 22].
ii. Fitness value is calculated repeatedly which might be computationally expensive for some problems [18, 28].
iii. Being stochastic, there are no guarantees on the optimality or the quality of the solution.
iv. If not implemented properly, the GA may not converge to the optimal solution.

Regardless of the above limitation, GA is attractive for use in solving optimization problems because it delivers good enough solution in very reasonable time. GA is noted to be efficient, robust and versatile [18, 27]. Table 1 captures the pseudo code for a standard GA [29].

Table 1. Pseudo code for a standard Genetic Algorithm

```
Pseudo Code for Genetic Algorithm
1    Population ← randomly initialize population(t)
2    determine fitness of population(t)
3    repeat
4            select parents from population(t)
5            perform crossover on parents creating population(t+1)
6            perform mutation of population(t+1)
7            determine fitness of population(t+1)
8            employ replacement strategy (population)
9    until best individual is good enough || terminating condition
10   return Population
```

2.5 Examination Timetabling With GA: Some Issues worth Noting
[11] noted a number of issues may make solving the ETP using GA difficult. These include the nature of the problem, the difficulty of devising a suitable representation of the problem, the choice of GA operations, their implementations and parameters, the need to decide the kind of GA to implement (Steady State, Generational, Breeder, et-cetera) amongst other.

The following expatiate on some of these issues:

i. *Problem Type*: [11] note that GA might not be effective in solving some class of examination timetabling problem, those problem having sub-problems characterized with "sequences of clique."
ii. *Problem representation:* Representation can be direct or implicit [10]. The direct representation encodes the actual timetable while the implicit encodes a set of instructions as to how the timetable should be built. [11] note that GA with direct representation are prone to fails because of difficulties of coordinating the effort expended on solving different parts of the problem; such representation may not be appropriate in all situations when solving problems with GA, particularly if the problem involves only binary constraints as was also noted in [30]. However, [2] noted that by using a direct representation of timetable and a combination of specially developed heuristic operators, the GA can be guided towards the best sectors of the solution space, resulting in quality timetables. GA therefore may needs a sort of guiding mechanism when solving problems of certain types.
iii. *Initial Solution Technique***:** In [31] is was shown that the technique used in preparing the initial population or solution can greatly affect the quality of the final solution obtained using GA. Their used of a construction heuristics as against randomly creating potential solution in the initial population, improved the performance of the GA to produce better quality timetables and better results for different problems.
iv. *Issue of Repair Methods*: Solving Examination timetabling problem using GA often require the use of repair method or algorithms. Repair methods are required when the timetable produced using GA operators such as crossover and mutation results in an infeasible timetable. A survey of repair methods is given in [32]. However, the used of repair method can be avoided by a careful design and implementation of the GA operators in a manner to avoid infeasible solutions as was done in [11] and [33]. This is the approach adopted in this work.
v. *GA Operators Design*: [2] showed that by using hybrid crossover operators, which incorporated known graph coloring techniques, good quality timetables can be produced, even from extremely constrained problems with a GA. [2] also noted that for solving he ETP, the GA operators (crossover and mutation) may need redefining to prevent the generation of infeasible timetables.
vi. *Use of Penalty Function & Objective Function*: it is desirable that computation (of penalty and objective function) be fast [9]. [11] noted that using penalty-function method (involving the linear combination of weights) is "computationally cheap" and that choosing a penalty value is more of "common-sense" and not hard as GA folklore suggest. Linear combination or assignment of weights is the popular approach in literature as in [9, 10, 33]. This was the approach used in this research.
vii. *When to Use GA in Examination Timetabling:* [11] noted that GA is a good option for solving large (examination) timetabling problems having significant number of non-binary constraints as well as binary ones: performance is impeded if all constraints are binary as was also noted in [30, 34]. They suggested that it may be better to employ GA to search for a good algorithm than for the solution to a given problem.
viii. *Search Space Redefinition*: With respect to solving the ETP, [2] noted that it may be necessary to redefine the GA search space to prevent infeasible timetables so as to limit problems that may arise with the hard and soft constraints during scheduling.
ix. *Other challenges noted in using GA are* (i) None of the standard GA can backtrack when stock in a local minima [11]. (ii) The innate parallelism can cause problem (not sharing of information) [11].

3. MATERIALS & METHODS

Using Bells University of Technology, Ota, as a case-study, constraints for solving the ETP were elicited (Table 2) and subsequently used in modeling the ETP.

Table 2: Constraint Types and Penalty Values

Code	Definition
HC1	No student should write more than one exam at a time (that is, write two or more exams at a time)
HC2	No teacher (staff) should be scheduled to be in more than one room at any time.
HC3	No exam should be schedule more than one
HC4	All scheduled venues must have adequate capacity to contain the students that enrolled for the exam.
SC1	No student should be scheduled to sit for two non-consecutive exams in a given day.
SC2	No student should be scheduled to sit for two consecutive exams in a day.
SC3	No student should be scheduled to sit for three consecutive exams in a day.
SC4	No teacher should be scheduled to invigilate two non-consecutive exams in a day.
SC5	No teacher should be scheduled to invigilate two consecutive exams in a day.
SC6	No teacher should be scheduled to invigilate three consecutive exams in a day

In literature, HC1, HC2 and HC3 violations (see Table 2) are referred to as first-order conflicts [35, 36]. In this research, second order conflicts refers SC3 and SC6 violations, third order conflict to SC2 and SC5, while fourth-order conflicts to SC1 and SC4.

3.1 Mathematical Formulation of the ETP

Using the constraints in Table 2, the ETP is formulated as follows:

3.1.1. Resource Definition

The resources used in solving the ETP are:

P: A set of p periods (or time-slots), p_1, p_2, …, p_p.

D: A set of d days (i.e. examination duration), $d_1, d_2, d_3, \ldots, d_d$: a day comprise 1 to 3 periods.

E: A set of e examinations, $e_1, e_2, e_3, \ldots, e_e$

E_p: A set of β examinations scheduled in period p_p, that is, $e_1p_p, e_2p_p, e_3p_p \ldots e_\beta p_p$

S: A set of s students, $s_1, s_2, s_3, \ldots, s_s$, in the campus of the university

L: A set of l teachers, $l_1, l_2, l_3, \ldots, l_l$ in the university.

R: A set of s examination registration lists for all students in the campus, that is, $R_{s_1}, R_{s_2}, R_{s_3}, \ldots, R_{s_s}$

V: A set of all *v* venues in the campus, that is, $v_1, v_2, v_3, \ldots, v_v$

3.1.2. Decision Variables

All decision variables can have a value of 0 or 1.

i. $e_m p_{kd_h}$: is the instance of an examination e_m scheduled in period p_k of day d_h. $e_m p_{kd_h}$= 1 if schedule or 0 otherwise.

ii. $s_j e_m$: is the instance that student s_j enrolled for examination e_m. $s_j e_m$= 1 if student enrolled or 0 otherwise.

iii. $s_i e_m v_y$: is the instance that student s_j who enrolled for examination e_m is scheduled for venue v_y. $s_i e_m v_y$= 1 if scheduled or 0 otherwise.

iv. $s_j e_m p_{kd_h}$: is the instance that student s_j is to sit for examination e_m scheduled for period p_k of day d_h. $s_j e_m p_{kd_h}$= 1 if student s_j is scheduled or 0 otherwise.

v. $l_g v_y p_{kd_h}$: is the instance of a teacher l_g scheduled to be in venue v_y at period p_k of day d_h. $l_g v_y p_{kd_h} = 1$ if scheduled or 0 otherwise.

3.1.3. Notations Used

The notation $n(e_1)$ denote the number of students that enrolled for examination e_1, $cap(v_y)$ denote the capacity of venue v_y, and $duration(p_k)$ denote the number of hours in period p_k.

3.1.4. Assumptions

The following are the assumption made in carrying out the research experimentation:

i. There are only three (3) periods in a day, that is,:
$\forall d_h \in D, \exists p_{k+i} \in P: i = 1,2,3.$

Where $p_k$is the last period of the previous day and $k \in N_0$.

ii. Each period is of a fixed 3-hour duration, that is,
$\forall t_b \in T, duration(t_b) = 3.$

iii. All venues dedicated for use during examination are available during the entire examination period.

3.1.5. Constraint and Objective Function Modelling

The using the decision variables, the constraints (see Table 2) are modelled as follows:

HC1: No student should write more than one examination at a time (period) in any given day.

$$HC1 = \sum_{j=1}^{e} s_i e_j p_{kd_h} \leq 1 \qquad (1)$$

HC2: No Teacher should be scheduled to be in more than one venue at the same time (period) in any given day.

$$HC2 = \sum_{k=1}^{p} l_g v_y p_{kd_h} \leq 1 \qquad (2)$$

HC3: All scheduled venues must have adequate capacity to contain the students that enrolled for the examinations scheduled in them. If x represent the number of students that enrolled for the examination e_m, then:

$$HC3 = \sum_{i=1}^{x} s_i e_m v_y = n_s(e_m): n_s(e_m) \leq cap(v_y) \qquad (3)$$

SC1: No student should be scheduled to sit for two non-consecutive (or more than one) examination in a day.

$$SC1 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} \leq 1 \qquad (4)$$

SC2: No student should be scheduled to sit for two consecutive examinations in a given day.

$$SC2 = \sum_{k=a}^{k=a+1} s_i e_m p_{kd_h} < 2 \qquad (5)$$

where $a = 1$ *or* $2 : a + 1 \leq 3$.

SC3: No student should be scheduled to sit for three consecutive examinations in a day.

$$SC3 = \sum_{k=1}^{k=3} s_i e_m p_{kd_h} < 3 \qquad (6)$$

SC4: No teacher should be scheduled to invigilate two non-consecutive exams in a day.

$$SC4 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} \leq 1 \qquad (7)$$

SC5: No teacher should be scheduled to invigilate in two consecutive periods.

$$SC5 = \sum_{k=a}^{k=a+1} l_g e_m p_{kd_h} < 2 \qquad (8)$$

where $a = 1$ *or* $2 : a + 1 \leq 3$.

SC6: No teacher should be scheduled to invigilate in three consecutive periods.

$$SC6 = \sum_{k=1}^{k=3} l_g e_m p_{kd_h} < 3 \qquad (9)$$

3.1.6. The ETP Objective Function

In this research, the objective f_o is defined in terms of the penalty function f_p as:

$$f_o = f_p = f_{p(hard)} + f_{p(Students)} + f_{p(invigilators)} \qquad (10)$$

where $f_{p(hard)}, f_{p(Students)} \wedge f_{p(invigilators)}$ represent the three components of the penalty function as can be deduced from Table 2.

Equation 2.11 can be written as:

$$f_o = f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j=1}^{j=3} w_j SC_j + \sum_{k=1}^{k=3} w_k SC_k \qquad (11)$$

Where $w_j SC_j$ and $w_k SC_k$ represent the students and invigilator-related constraints respectively. If the number of student-related constraints is pairwise comparable with that of the invigilator constraints and the assigned weights (penalty) are same for each pair as in the case in this work (see Table 2), that is,

$$\sum_{j=1}^{q} w_j SC_j \equiv w_k \sum_{k=1}^{z} SC_k : q = z \wedge w_j = w_k;$$

With the objective function stated in terms of the penalty function as a minimization problem, equation 2.13 can be simplified to:

$$f_p = w_h \sum_{i=1}^{i=4} HC_i + \sum_{j,k=1}^{q=z=3} w_j \left(SC_j + SC_k \right) \qquad (12)$$

Considering that HC_i, SC_j, and SC_k are hard and soft constraints with varying violation consequences, the weights $w_h, w_{j=1}, w_{j=2} \wedge w_{j=3}$ are chosen such that $w_h \gg w_{j=1} \gg w_{j=2} \gg w_{j=3}$.

From the foregoing, the ETP can now be stated as an optimization problem as follows:
Minimize equation (12), subject to the constraints in equations (1) to equation (9).

3.2. Object-Oriented Timetable Representation

In this research, the GA chromosome (timetable) was represented as an object, modelled using the following classes: Staff, Student, Examination, Registration, Venue, Period and Timetable. The GA chromosome contains an arraylist of alleles of time-slot or period objects. Period objects in turn contains three arraylists for examinations, venues and staff (invigilator) objects. Each examination object contains an arraylist of matriculation numbers of students in enrolment. Figures 1 illustrates the different components of the GA chromosome.

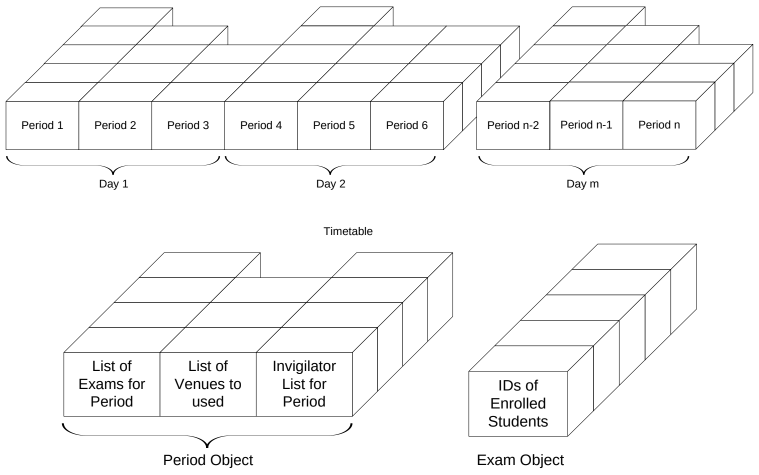

Figure 1: Different components contained in the timetable object.

3.3. Implementation of the GA Algorithm Using an OOP Approach

To facilitate experimentation and comparison with existing result generated using Bells University of Technology dataset, the Genetic Algorithm was implemented or incorporated into an existing object oriented timetabling application previously developed and used in [37]. The application already implemented the classes used in the design and representation of the GA chromosome. It also has the algorithm for the standard Tabu Search implemented. The classes used and the interrelationship between them in the application mentioned are captured in Figure 2. This application also implemented the algorithm for generating the initial timetable solution and for scheduling the invigilators to actual venues. With this application, the timetable generation is in four phases: (i) generation of the initial population (timetables) (ii) Optimizing the population using GA (iii) Assignment of students and invigilators to actual venues (iv) Reporting (displaying) the best individual (timetable) of the population.

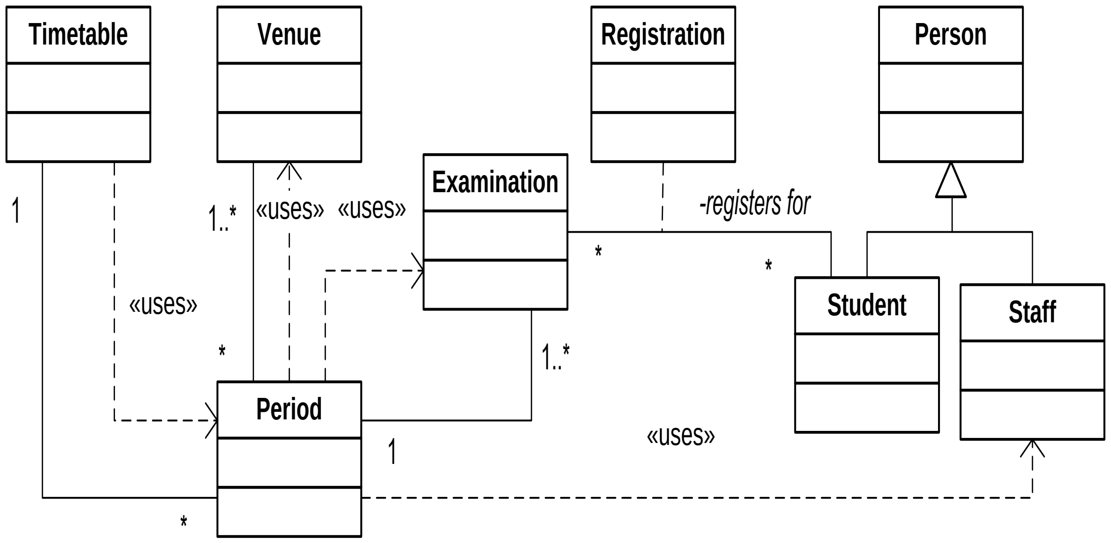

Figure 2: The class relationship diagram of the application of [37].

The pseudo-code for the application is captured in Table 3. Line 7 indicates the points where the GA was called.

Table 3: The Pseudo-code of the timetabling application used in [37]

	Pseudo Code for the Timetabling Application
1	Start
2	Declare and Initialize working variables
3	Load Data from Database (Venues, Courses, Registrations, Students, Staff)
4	Extract Course Registration List for each student in semester
5	Extract student's list for each enrolled course
6	Create Initial Solution
7	Optimize Population suing GA
8	Schedule Invigilators
9	Display Timetable
10	End.

Using the following variable, a formal description of the GA is given in Table 4:

H = List of n individual forming initial GA population
H' = List of n individual forming final pupation
P = List of two selected individuals (parent)
T = List of individuals selected from H for tournament
t = an individual (that is, a single timetable solution)
StudSemRegL = the list of all students' semester's examination registration

Table 4: A formal description of the implemented GA.

Algorithm GA

```
     Input: H
     Output: H'
1    T ← ∅    // Tournament Individuals List
2    P ← ∅
3    t ← $null$
4    x ← $f(H)$  // the number of competing individuals, a function of H
5    for(i = 1 to n, do)
6        evaluateTitness($H_i$)
7    endfor
8    while(not stopCondition)
9        T ← selectIndividualsForTournament(H,x)
10       P ← peformTornamentSelection(T)
11       t ← peformCrossOverWithMutation(P, studSemRegL)
12       evaluateFitness(t)
13       addToPopulation(t)
14       normalisePopulation(H, t)
15   endwhile
16   H' ← sort(H)
17   return H'
```

3.3.1. The Crossover Operator

The GA implements a heuristic crossover operator similar to that shown in Figure 3 as described in [2], however, in an object oriented manner. While ensureing feasibility, the process elicit all common exams in both periods and introduces some other exams in the pool or from that left over in previous crossing.

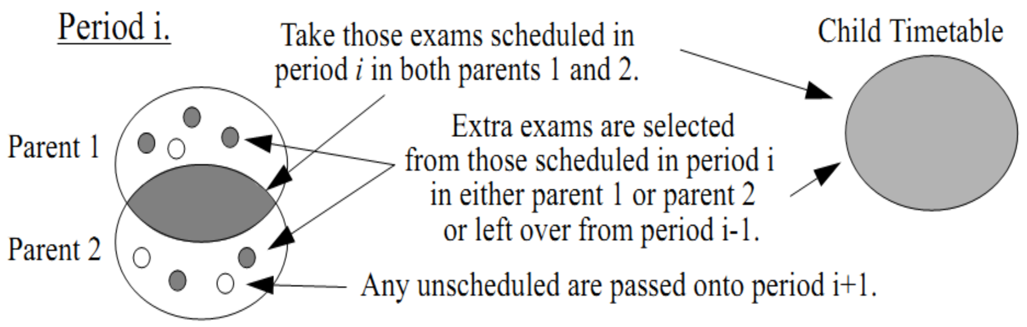
Period i.
Take those exams scheduled in period *i* in both parents 1 and 2.
Child Timetable
Parent 1
Extra exams are selected from those scheduled in period i in either parent 1 or parent 2 or left over from period i-1.
Parent 2
Any unscheduled are passed onto period i+1.

Figure 3. A hybrid heuristic Crossover operator described in [2].

3.3.2. The Mutation Operator

The mutation operator was incorporated into the crossover function as done in [2], by adding examination into the current search that would otherwise not be considered until a later period. This was necessary because mutation by randomly picking two examination from different period and exchanging them may result in an infeasible examination timetables. Table 5 captures the pseudo code of the peformCrossOverWithMutation(P, studSemRegL) method, (line 11 of Table 4).

Table 5: Pseudo code for the peformCrossOverWithMutation( )

Pseudo Code for PeformCrossOverWithMutation() // Multi-point

```
1   Declare and Initialize working variable
2   Create child  chromosome object
3   While (not end of allele in parents, starting from 1st to last)
4       Create Period Objects for child chromosome
5       Extract examinations in opposite alleles of both parents into scheduling list
6           Identify and extract intersecting examinations into list
7           Add remaining unique examinations from both parents' allele to list
8           Add few additional unique examinations from exam pool to list // helps mutation
9       Schedule extracted exams in list to child's period (random selection)
10  Endwhile
11  While (existing unscheduled exams in list)
12      Add additional period(s) to child chromosome // implies longer period length
13      Schedule existings examinations in list to period
14  Endwhile
15  Compute Timetable Conflicts and fitness
16  Return Child timetable.
```

3.3.3. The GA Configuration Used

The GA chromosome was encoded as an object. The initial population of solution were generated by randomly picking examination objects from a pool; this process ensured population diversity, without which the population will converge prematurely over a few generations [38]. The diversity introduced by the randomly generated solution have been observed to be responsible for driving GAs population to optimality over the generations [38]. Through experimentation, a population size of 100 individuals was considered suitable for this research, as too small a population size may result in poor mating pool while too large a size may slow down the GAs operational process [19] .

A tournament size was adapted to the population used (10% for a population size of 100) and used for mating parents. The heuristic crossover which implements an embedded mutation was set to occur 100% of the time in each generation. Using a steady-state GA population model, a fitness-base survivor selection approach was used; this implies the children produced in a generation replaces the least fit individuals in the population. The implemented GA terminates when the set duration is reach. Table 6 summarizes the GA configuration used in this work.

Table 6: A Summary Description of the GA Configuration Use

Representation Type	**Direct (Object-Oriented)**
Initialization	*Random*
Population Size	*100 (also varies)*
Population Model	*Steady-state (replace worst)*
Parents Selection	*Tournament (best two out of random 10% selected*
Number of Generations	*Time bound*
Crossover type	*Heuristic*
Mutation embeded	*Embeded in Crossover*
Tournament Selection	*Adaptive to population*
Crossover probability (p_c)	*1 (i.e 100%)*
Mutation probability (p_m)	-
Number of Children	*1*

3.4. Data Gathering and Generation

The dataset used for this research was gathered from Bells University of Technology, Ota, at the end of 1st Semester 2012/2013 Sessions. A summary of this is given as follows:

Total number of students	1896
Total number of Registrations	16866
Total Number of Examinations	443
Total Number of Venues	25 (total capacity: 1436)
Total number of Invigilating staff	170

In order to investigate the impact of increasing student's population on the GA algorithms, dataset of 25,000, 50,000, 75,000 and 100,000 students population, with 443 available examinations was generated using Bells University of Technology dataset as seed. Staff to student's ratio was set at 1 to 30 and the total available venue capacity was set at 60% of the student population.

3.5 Experimental Environment

The algorithm was implemented in Java (JDK8u54) on Windows 7 64-bit Operating system running on a Dell Inspiron N5110 Laptop with Intel core i5 (quad core) processor, 6 GB RAM, 700 GB HDD (Hitachi) at 5400 RPM. NetBeans IDE 8.02 was used as the IDE for the algorithm integration. The application has XAMPP version 3.2 installed, which incorporates MySQL database.

3.6 Experimental Procedures

Dataset for the ETP problem was loaded from the database and used in creating the initial timetable solutions. Examination were randomly picked from a pool and scheduled into the available venue spaces, while avoiding violation of the HC constraints (see Table 2). The number of periods in the initial solutions varied from about 21 to 28. For uniformity, initial solutions where normalized to 30 periods by adding additional periods and spreading scheduled examinations. The 30 period used implies examination duration of 10 days, or two weeks of five working days each for a 3-periods-in-a-day examination schedule. The generated initial solutions were then optimized using the GA, after which invigilators' scheduling was done. Two sets of experimental runs were conducted; a set using the Bells University dataset was conducted for 10 times and the other set using generated data for student population of 25, 000, 50,000, 75,000 and 100,000 was conducted for three times. Since the GA used a population of individual (i.e. timetables), a best solution, GA(best), that is one with lowest penalty cost, was captured for each run, the average of the best solutions for the two sets of experimental runs (i.e. GA(avgB)) were also determined as well as the average penalty cost for the population for all the runs (GA(AvgAR) for each of the set. The results of the experimental runs were harvested for analytical purposes and reported in the next section.

4. RESULTS AND DISCUSSION

Appendix A is an extracted page of one of the generated timetable using the actual dataset from Bells University of Technology.

4.1 1st Experimental Run: The Quality of Timetables Generated by GA

Table 7 summarizes the results obtained from the 1st experimental run and captures the quality of timetable generated in terms of violated conflicts. No hard (HC) constraints were violated because this was prevented in the scheduling process. However, the GA did not succeed in eliminating the 2nd OC constraint violation.

Table 7: Summary of Results from 1st set of Experimental Runs
Parameters: Student Pop = 1896, GA Pop = 100, Simulation time = 300s (5 min)).

		GA	**GA (avg)**	**GA(avgAR)**
Constraint Violation		**Best of 10 Runs**	**Avg. of Best**	**Avg. of All Runs**
1st Order Conflict Count (OCC)		0	0.0	0.0
2nd OCC		356	454.78	-
3rd OCC		2402	2584.89	-
% improvement		41.0	13.2	28.6

.4.2 The Effect of increasing Processing Time GA on Timetable Quality

Using the database of 25000 students, Figure 4 showed the effect of varying the processing time from 0 to 5,400 seconds (i.e. 1½ hours) for the GA. The timetable quality improves with time, but at a slow rate.

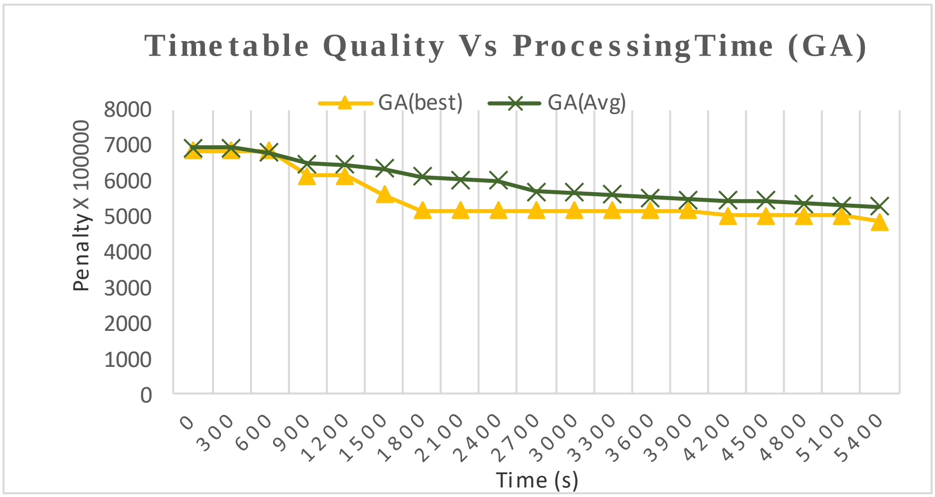

Figure 4: The effect of increasing the processing time on Timetable Quality

4.3 2nd Experimental Run: The Quality of Timetables Generated

Table 8 summarizes the results on the quality of timetables generated from the second set of experimental runs using the generated dataset of 25,000 student population. Again, the GA did not eliminate instances of 2nd OC (the SC3) violation; thought the best individual improved by 29.2%, the overall population recorded a 49.2% improvement during the 1½ hour experimental run.

Table 8: Summary of Results from 2nd Set of Experimental Runs
Parameters: Student Pop = 25,000, GA Pop = 100, Simulation time = 5400s (1½ hrs).

		GA	**GA (avg)**	**GA(avgAR)**
Constraint Violation		**Best of 10 runs**	**Average of 10**	
1st OCC		0	0.0	0.0
2nd OCC		4801	5228.5	-
3rd OCC		39624	33375.0	-
% Improvement		29.2	23.9	49.2

GA's inability to eliminate the SC3 constraint violation is certainly due to the limitation of the crossover operator, the key search operator in GA, employed in this project. From this observation, It is needful to design or investigate other heuristic crossover operators on the GA algorithms performance on the ETP.

4.4 Effect of varying GA Population on Timetable Quality:
Figure 5 shows the effect of varying GA population on Timetable quality from 25 to 225 in steps of 50, while holding the GA generation as 3000. The Timetable quality appears to be decreasing with increasing GA population. One would have expected the reverse, since more population will imply better coverage of the search space. This unexpected outcome may however be attributed to the nature or complexity of the problem and to the stochastic GA population generation process used in this experiment. It can also be related to the apprarent ineffectiveness of the implemented crossover operator as Figure 5 reflects.

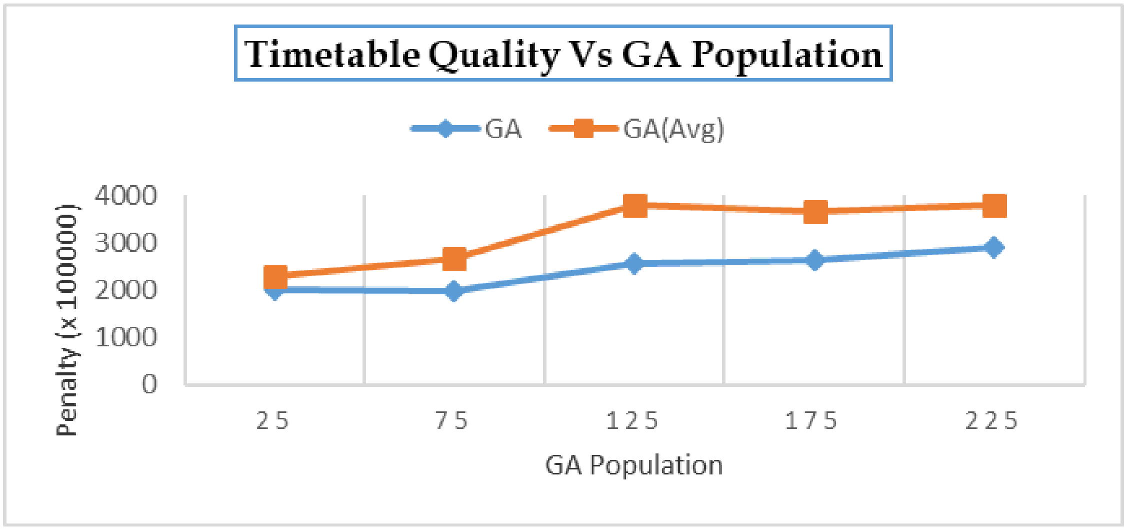

Figure 5: The Effect of varying GA population on Timetable Quality

4.5 Effect of varying Student Population on Timetable Quality
Using the generated databases, the effect of varying student's population from 25,000 to 100,000, in steps of 25,000 is illustrated in Figure 6. The timetable quality generated by the GA algorithm degraded in a fairly linear manner with increasing students' population. It can be concluded that with increasing problem complexity, the performance of GA drops.

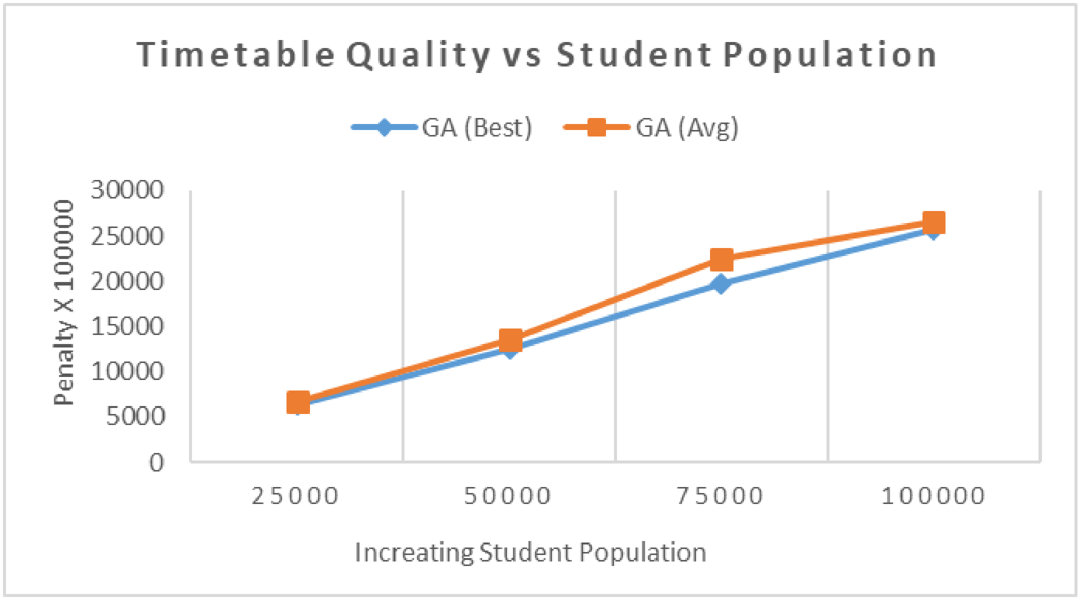

Figure 6: The effect of varying student' population on the Timetable quality.

4.6 Evaluation based on Space Complexity

The generated databases of students' population varying from 25,000 to 100, 000 in steps of 25,000 was used to determine the empirical Space Complexity of the GA Algorithm; its memory consumption differs before and after garbage collection (GC). The empirical space complexity of the GA is concluded to be of order $O(n)$. Figure 7 illustrates this.

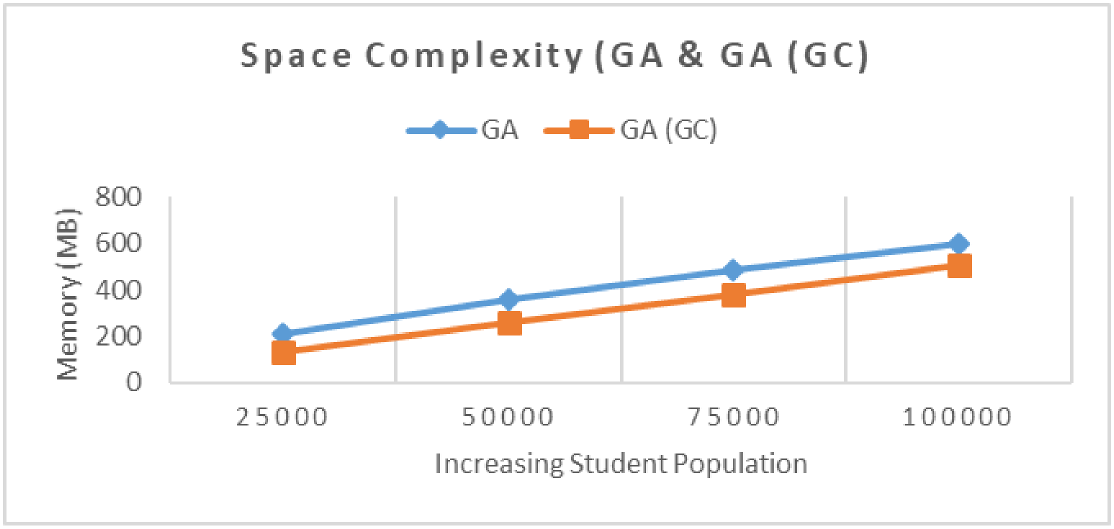

Figure 7: The GA Space Complexity of GA (with and without GC).

5. CONCLUSION

This research implements a Genetic Algorithm that used an adapted heuristic crossover operator on an object-oriented chromosome representation in solving the Examination Timetabling Problem. The implemented GA was experimented with using the dataset from Bells University of Technology Ota. The performance of the GA with the adapted heuristic crossover operator was evaluated using standard timetabling metrics. Results obtained showed that the GA's performance was poor in comparison with previously obtained results on the same dataset. The poor performance of the GA was attributed to the ineffectiveness of the crossover operator used, the GA's key search operator and not to the object-oriented representation used for its chromosome, It is expected that with a better or more effective crossover operator, the GA's performance on this dataset will improve. A future research work could look into different approaches for implementing cross-over operators for GAs, particularly when solving highly constrained problems like the ETP.

REFERENCES

1. Burke, E.K. and S. Petrovic, *Recent Research Directions in Automated Timetabling.* European Journal of Operational Research - EJOR, 2002. **140**(2): p. 266-280.
2. Burke, E.K., D.G. Elliman, and R. Weare, *The automated timetabling of university exams using a hybrid genetic algorithm.*, in *AISB Workshop on Evolutionary Computing. 1995*1995, Society for the Study of Artificial Intelligence and Simulation of Behaviour (SSAISB), 1995: University of Leeds, UK, 3-7 April 1995.
3. Fowler, J., G. Kendall, and B. McCollum, eds. *Proceedings of the 5th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2011).* Vol. 2014. 2011: Phoenix, Arizona, USA.
4. Komar, M., D. Grbic, and M. Cupic. *Solving Exam Timetabling Using Distributed Evolutionary Computation*. in *The ITI 2011 33rd Int. Conference on Information Technology Interfaces*. 2011. Cavtat, Croatia, June 27-30.
5. Pinedo, M.L., *Scheduling: Theory, Algorithms, and Systems*. 3 ed. 2008, 233 Spring Street, NewYork, NY 10013, USA: Springer Science+Business Media, LLC. 664.
6. Qu, R., et al., *A Survey of Search Methodologies and Automated System Development for Examination Timetabling.* Journal of Scheduling, 2009. **12**(1): p. 55 - 89.
7. Schaerf, A., *A survey of automated timetabling. .* Artificial Intelligence Review., 1999. **13**(2): p. 87-127.
8. Corne, D., P. Ross, and H. Fang, *Evolutionary timetabling: Practice, prospects and work in progress.* , in *Proceedings of UK Planning and Scheduling SIG Workshop.*, P. Prosser, Editor. 1994.
9. Corne, D., P. Ross, and H. Fang, *Fast Practical Evolutionary Timetabling*, in *AISB Workshop*1994: Leeds, U.K.
10. Burke, E.K., D.G. Elliman, and R. Weare, *Specialised Recombinative Operators for Timetabling Problems*, in *AISB Workshop on Evolutionary Computing*1995: University of Leeds, Sheffield, UK, 3-7 April 1995.
11. Ross, P., E. Hart, and D. Corne, eds. *Some observations about GA-based exam timetabling.* LNCS 1408, Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference. II : Second International Conference, PATAT 1997, ed. E.K. Burke and M.W. Carter. Vol. 1408. 1997, Springer-Verlag: Toronto, Canada. 115-129.
12. Garey, M.R. and D.S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences.* 1979, San Francisco, Calif: WH Freeman and Company.
13. Cooper, T.B. and J.H. Kingston, *The complexity of timetable construction problems. In (Burke & Ross, 1996).* 1995: p. 283-295.
14. Burke, E.K., et al., *Examination timetabling in British universities: A survey.*, in *Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. LNCS 1153.* , B. E.K. and R. P., Editors. 1996, Springer-Verlag: Berlin, Heidelberg. p. 76-90.
15. Boizumault, P., Y. Delon, and L. Peridy, *Constraint logic programming for examination timetabling.* The Journal Of Logic Programming, 1996.
16. Carter, M.W., G. Laporte, and J.W. Chinneck, *A general examination scheduling system.* Interfaces, 1994. **24**: p. 109-120.
17. Goldberg, D.E. and D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. 1989: Addison-Wesley Publishing Company, 1989.
18. Whitley, D. *A Genetic Algorithm Tutorial*. Statistics and Computing 1994. **4**, 65-85 DOI: https://doi.org/10.1007/BF00175354.
19. Sastry, K., D.E. Goldberg, and G. Graham Kendall, *Genetic Algorithms*, in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, E.K. Burke and G. Kendall, Editors. 2014, Springer New York Heidelberg Dordrecht London.
20. Noor, N.B.M., *Study on Genetic Algorithm*, 2004, Faculty of Information Technology and Quantitative Science, University Technology Mara.

21. Haupt, R.L. and S.E. Haupt, *Practical Genetic Algorithms* 2ed. 2004, Hoboken, New Jersey: John Wiley & Sons, Inc Published by John Wiley & Sons, Inc.
22. Niazia, A. and R. Leardib, *Genetic algorithms in chemometrics.* Journal of Chemometrics, 2012. **26**(6): p. 345-351.
23. Cantu-Paz, E., *A survey of Paral lel Genetic Algorithms (IlliGAL Report 97003)*, 1997: Department of General Engineering, UIUC.
24. Fonseca, C.M. and P.J. Fleming, *An Overview of Evolutionary Algorithms in Multiob jective Optimization.* Evolutionary Computation,, 1995. **3**(1): p. 1-16.
25. Wang, J., et al., *An Improved Real-Coded Genetic Algorithm Using the Heuristical Normal Distribution and Direction-Based Crossover.* Computational Intelligence and Neuroscience, 2019. **2019**.
26. Shorman, S.M. and S.A. Pitchay, *Significance of Parameters in Genetic Algorithm, the Strengths, its Limitations and Challenges in Image Recovery.* ARPN Journal of Engineering and Applied Sciences, 2015. **10**(2).
27. Lucasius, C.B. and G. Kateman, *Understanding and using genetic algorithms Part 1. Concepts, properties and context.* Chemometrics and Intelligent Laboratory Systems, 1993. **19**(1): p. 1-33.
28. Bhattacharya, M., *Expensive Optimisation: A Metaheuristics Perspective.* (IJACSA) International Journal of Advanced Computer Science and Applications,, 2013. **4**(2): p. 203-209.
29. Hassani, A. and J. Treijs, *An Overview of Standard and Parallel Genetic Algorithms*, in *IDT Workshop on Interesting Results in Computer Science and Engineering (IRCSE '09)*2009: Mälardalen University, Sweden
30. Gordon, V., A. Bohm, and D. Whitley, *A note on the performance of genetic algorithm on zero-one knapsack problems*, in *Technical Report CS-93-108*1993, Colorado State University, Dept of Computer Science.
31. Raghavjee, R. and N. Pillay, *The effect of construction heuristics on the performance of a genetic algorithm for the school timetabling problem*, in *SAICSIT '11 Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*2011, ACM: South Africa. p. 187-194
32. Salcedo-Sanz, S., *A survey of repair methods used as used as constraint handling techniques in evolutionary algorithms.* Computer Science Review, 2009. **3**(3): p. 175-192.
33. Jha, S.K. *Exam Timetabling Problem Using Genetic Algorithm*. IJRET: International Journal of Research in Engineering and Technology 2014. **3**.
34. Terashima-Marín, H., *A Comparison of GA-based Method and Graph-colouring methods for solving the timetabling problem*, in *Department of AI*1994, University of Edinburgh: Edinburgh.
35. Kendall, G. and N.M. Hussin, *Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at University of Technology MARA]*, in *PATAT 2004, Selected Papers from the 5th International Conference. Lecture Notes in Computer (LNCS) 3616,* , E.K. Burke and M. Trick, Editors. 2005, Springer-Verlag Berlin Heidelberg (2005). p. 199-218.
36. White, G.M. and B.S. Xie, *Examination Timetables and Tabu Search with Longer-Term Memory*, in *PATAT 2000, Lecture Notes in Computer Sciences (LNCS) 2079*, E. Burke and W. Erben, Editors. 2001 Springer-Verlag Berlin Heidelberg. p. 85-103.
37. Ezike, J.O.J., et al., *Generating Feasible and Optimized Timetables using Tabu Search: The Case of Bells University of Technology, Ota.* Computing, Information Systems & Development Informatics Journal. , 2018. **Vol 9,**(1): p. 84-104.
38. Gupta, D. and S. Ghafir, *An Overview of methods maintaining Diversity in Genetic Algorithms.* International Journal of Emerging Technology and Advanced Engineering, 2012. **2**(5).

APPENDIX A: SAMPLE GENERATED TIMETABLE

```
                   BELLS UNIVERSITY OF TECHNOLOGY, OTA
                         EXAMINATION TIMETABLE

--------------------------------------------------------------------------------------
----------------------------------
|       Days     |   Courses/No of Students                |  Venues(Count)  |
Invigilators' IDs                        |
--------------------------------------------------------------------------------------
----------------------------------
|     Mon        | GES101(589)-MPH(240);HallD(230);B11(56);B10(56);BioCLab(7)| MPH(240)
| 221; 185; 121; 104; 103                    |
| 9.00am-12.00pm| MEE203(172)-B9(56);Rm6(56);Rm5(56);BioCLab(4)| HallD(230)     | 578;
159; 555; 581; 582                    |
|               | ITP401(71)-Rm1(56);BioLab3(15)           | B11(56)         | 274; 135
|
|               | HRM303(19)-BioLab1(19)                   | B10(56)         | 302; 200
|
|               | MEE403(15)-ChemLab2(15)                  | B9(56)          | 296; 272
|
|               | BIO203(15)-ChemLab2(15)                  | Rm6(56)         | 171; 276
|
|               | GES313(1)-BioLab3(1)                     | Rm5(56)         | 99; 586
|
|               |                                          | Rm1(56)         | 89; 236
|
|               |                                          | ChemLab2(30)    | 38
|
|               |                                          | BioLab1(19)     | 254
|
|               |                                          | BioLab3(16)     | 253
|
|               |                                          | BioCLab(11)     | 178
|
--------------------------------------------------------------------------------------
----------------------------------
|     Mon       | ACC303(55)-Rm1(55)                       | HallD(133)      | 39; 329;
332                                   |
| 12:30pm-3.00pm| AMS427(1)-Rm1(1)                         | Rm1(56)         | 381; 351
|
|               | BDT309(4)-BioCLab(4)                     | Rm5(56)         | 365; 238
|
|               | ARC405(13)-BioCLab(13)                   | B6(40)          | 362
|
|               | MKT303(6)-BioLab3(6)                     | PhyLab1(30)     | 123
|
|               | NUD311(2)-BioCLab(2)                     | BioLab1(30)     | 240
|
|               | TML501(1)-BioCLab(1)                     | BioLab2(30)     | 508
|
|               | FSB505(2)-BioLab3(2)                     | PhyLab2(30)     | 552
|
|               | ITP303(51)-Rm5(51)                       | ChemLab2(30)    | 506
|
|               | PMT307(4)-Rm5(4)                         | ChemLab1(30)    | 473
|
|               | PHY309(5)-BioLab3(5)                     | BioCLab(20)     | 505
|
|               | FDT201(5)-BioLab3(5)                     | BioLab3(20)     | 551
|
|                | ECO303(39)-B6(39)                    | B5(14)          | 550
|
|                | BDT413(2)-BioLab3(2)                 |                 |
|
|                | BTE403(2)-BioLab1(2)                 |                 |
|
|                | MIC403(10)-BioLab1(10)               |                 |
|
|                | MIC303(14)-BioLab1(14)               |                 |
|
|                | BME407(1)-Rm5(1)                     |                 |
|
|                | ARC305(25)-ChemLab2(25)              |                 |
|
|                | BME305(6)-BioLab2(6)                 |                 |
|
|                | EEE413(28)-PhyLab1(28)               |                 |
|
|                | BUS101(133)-HallD(133)               |                 |
|
|                | URP409(2)-PhyLab1(2)                 |                 |
|
```