Intelligent Automation And Soft Computing, 2020
Vol. 26, no. 3, 385–396
DOI: 10.32604/iasc.2020.013915

Genetic Algorithm and Tabu Search Memory with Course Sandwiching (GATS_CS) for University Examination Timetabling

Abayomi-Alli A.[1], Misra S.[2,3], Fernández-Sanz L.[4], Abayomi-Alli O.[2,*], and Edun A. R.[1]

[1]Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria.
[2]Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria.
[3]Department of Computer Engineering, Atilim University, Ankara, Turkey.
[4]Department of Computer Science, University of Alcalá, Spain.

ABSTRACT
University timetable scheduling is a complicated constraint problem because educational institutions use timetables to maximize and optimize scarce resources, such as time and space. In this paper, an examination timetable system using Genetic Algorithm and Tabu Search memory with course sandwiching (GAT_CS), was developed for a large public University. The concept of Genetic Algorithm with Selection and Evaluation was implemented while the memory properties of Tabu Search and course sandwiching replaced Crossover and Mutation. The result showed that GAT_CS had hall allocation accuracies of 96.07% and 99.02%, unallocated score of 3.93% and 0.98% for first and second semesters, respectively. It also automatically sandwiched (scheduled) multiple examinations into single halls with a simulation time in the range of 20-29.5 seconds. The GAT_CS outperformed previous related works on the same timetable dataset. It could, however, be improved to reduce clashes, duplications, multiple examinations and to accommodate more system-defined constraints.

KEY WORDS: Genetic Algorithm (GA), Tabu search, Timetabling, Sandwiching, Optimization.

1 INTRODUCTION

TIMETABLING problems are complicated constraint satisfaction problems said to be Non-Deterministic Polynomial-time (NP) hard. Many Institutions of higher learning use timetables to schedule classes, lectures, and examinations to assign appropriate venues and timeslots for such events in order to optimize available resources. Institutions of higher learning like Universities are generally concerned with a large number of examination courses to be allocated to fewer examination halls/venues within a short period of the exam. This often leads to poorly designed timetables that are expensive considering the time and other resources expended in developing them.

"Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space and time, to satisfy as nearly as possible a set of desirable objectives" (Wren, 1996). The main objective when designing an examination timetable is to develop an optimized most appropriate schedule for a set of examinations that satisfies pre-defined constraints (Arogundade, Akinwale, and Aweda, 2010). Hence, the construction of a University examination timetable is a complex problem due to the high number of constraints required, large class sizes, venue capacity, and their availability, etc. Burke, Kingston and De Werra (2004) defined the timetabling problem as "A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints". It has created the challenge of assigning venues and time slots to events to satisfy the constraints maximally. This has been an interesting research area in operation research and artificial intelligence for the last decade (Bargetto, Della and Salassa, 2016).

University timetabling is the most widely studied among other timetabling problems since it is a required and time-consuming task which occurs regularly per semester, be it lectures, examinations, seminar or faculty scheduling. Timetabling problems in academic institutions is however broadly divided

CONTACT Abayomi-Alli Adebayo abayomiallia@funaab.edu.ng

into two: (i) examination and (ii) course scheduling. While the examination scheduling problem is concerned with assigning exams to venues over a certain period, the time is fixed within a week for course scheduling.

The constraints to be satisfied in timetabling are usually of two types: Hard and Soft Constraints. Hard constraints must be met; they are also called unavoidable constraints while soft constraints may not be satisfied but it is preferably not to be violated. However, meeting them is essential to having a good and feasible timetable (Van Staereling, 2012). Constraints' affecting University's examination timetable include:

1. The population of students (class domicile);
2. Incapacitated exam venues;
3. The limited time frame for examination per day (timeslots);
4. Not assigning exams with common resources simultaneously.

This research study is concerned with the allocation of semester examinations taken by student populations to limited available examination venues within specific time slots. This is a problem that no efficient algorithm has been able to solve optimally (Esraa and Ghada, 2016) and it occurs because:

1. There are challenges of insufficient resources such as exam venues (number and capacity) and timeslots to carter for the entire student population;
2. Satisfying all the constraints is becoming increasingly difficult;
3. There are very few venues/halls available during the short period of examination.

THIS study aims to develop an automatic examination timetable system for a University that will enhance course allocation to examination halls/venues and minimizes course un-allocation, clashes, duplication and multiple examinations in the University timetable. This system is expected to meet all the timetable constraints (soft and hard) optimally through a combination of Genetic Algorithm (GA) and Tabu Search (TS) memory that is enhanced with course sandwiching. The motivation for the study is the present method of manual scheduling of examination timetabling which is becoming more cumbersome and expensive as the University is growing.

THE rest of the paper is organized as follows: Section 2 presents a brief literature review on the subject area and previous related study, while Section 3 represents the research approach and methodology employed in the development of the automatic University examination timetabling system. Section 4 discusses the results obtained while the paper concludes with directions for future studies in Section 5.

2 LITERATURE REVIEW

A brief literature review and closely related works are presented in this section.

2.1 Timetabling Problems

Timetabling problems are complicated Scheduling problems which belong to a broad class of combinational optimization problems aimed at finding an optimal matching of tasks to different sets of resources (Anisha, Ganapathy, Harshita and Rishabh, 2015); (Mousavi and Zandieh, 2016); (Sadhasivam and Thangaraj, 2016) and (Srivastav and Agrawal, 2017). Merlot, Boland, Hughes, and Stuckey (2003) developed a "hybrid approach for solving the final examination timetabling problem that generates an initial feasible timetable using constraint programming and then applied simulated annealing with hill climbing to obtain a better solution".

TS approach was explored by Gaspero and Schaerf (2000) using graph coloring-based heuristics. Wilke and Ostler (2008) considered four algorithms, namely: TS, Simulated Annealing, GA, and Branch & Bound for solving a typical school timetable problem and evaluated their performance. Results showed that Simulated Annealing gave the best trade-off between accuracy and execution time while TS, GA, Branch & Bound were next in performance.

Ruey-Maw and Hsiao-Fang (2013) employed constriction Particle Swarm Optimization (PSO) with a local search for the timetable problem. Burke and Bykov (2016) developed an adaptive method that was general and fast to arrange heuristic for ordering the next scheduled examinations. However, methodologies such as GAs, evolutionary techniques, and others have been implemented with mixed success (Anuja, Priyanka, Shruti, Sonali, Rupal, and Dinesh, 2014). Unfortunately, many of the studies in the research area have been carried out with artificial data sets or on highly simplified real-world case studies (Mccollum, 2006).

In developing countries especially sub-Saharan African countries, first-year students (100 level) may be up to 1500 students in a course, thus making the timetabling problem even more challenging.

2.2 The Current Examination Timetabling System in the University under Study

A public University in Nigeria is used as the case study for this paper as the present method of examination timetabling as described by Arogundade, Akinwale and Aweda (2010) as a grid with days of the week, venues and time periods on different axes. This approach facilitates visual checking for conflicts in the timetable schedule and allows for modification until the timetable is adjudged as satisfactory. However, it is a very cumbersome and time-consuming exercise which often leaves several constraints unsatisfied. As the University is growing larger, the final exam

timetable is preceded by three to four drafts that are released weeks before the start of the semester examination. Students and Departments are required to check the draft timetables for issues like non-allocation, clashes, duplication and/or multiple courses on the timetable. This takes about two to three weeks and after every correction, an updated version is sent to the Departments for review until the start of the semester examination. Hence, the present approach is no longer sustainable as it can only suffice for small inputs. In this study, an automatic University examination timetabling system is proposed based on Genetic Algorithm and Tabu Search Memory with Course Sandwiching (GATS_CS).

2.3 Genetic Algorithm

Hamed, Kuan, and Adnan (2016) define a Genetic Algorithm (GA) as a powerful search technique that imitates the biological evolution and natural selection process. It has been applied to solve complex problems with large spaces in optimization (Bull, Martin and Beasley, 1993; Dipesh, Hiral, Mohammed, and Renuka, 2015). GA is a procedure used to discover the best solutions to search problems using the principles of evolutionary biology such as natural selection, mutation, genetic inheritance, and sexual reproduction. GA structure is governed by import laws of the theory of evolution of species and concreteness in two fundamental concepts:

1. Selection;
2. Reproduction (Jose, 2008).

The major problem in optimization is how to satisfy constraints, in a problem that is defined by several solutions. The GA determines the overall solution or one that is acceptable within the time defined for the algorithm (Jose, 2008). GA is implemented basically with five components as stated by Gen, Cheng, and Lin (2008):

1. A genetic representation of potential solutions to the problem;
2. A way to create a population (an initial set of potential solutions);
3. An evaluation function rating solution in terms of their fitness;
4. Genetic operators that alter the genetic composition of offspring (crossover, mutation, selection, etc.);
5. Parameter values that genetic algorithm uses (population size, probabilities of applying genetic operators, etc.).

2.4 TABU Search

Glover (1986) proposed an approach called Tabu Search (TS). TS was proposed as an alternative local search algorithm to solve several combinatorial optimization problems. The solution set must be finite or countable infinite (Michiels, Aarts and Korst, 2007) as many of such problems are expressed as a search for a specific permutation (Russell, Chiang, and Zepeda, 2008). Solution space of combinatorial optimization problems can typically be represented by sequences, permutations, graphs and partitions (Michiels, Aarts and Korst, 2007). Gendreau (2002) stated that the basic principle of TS "Is to pursue local search whenever it encounters a local optimum by allowing non-improving moves. Moving back to previously visited solutions is prevented using memories called Tabu lists that record the recent history of the search". Thus, it is prevented from revisiting the previous solution by tracking them in memory. TS has several memory types such as Short-term and Long-term (Harun, Engin, and Burak, 2008), Recency-based (Glover and Laguna, 1997), Frequency-based (Dréo, Petrowski, Siarry, and Taillard, 2006) memories, etc. It uses a Tabu list to track Tabu moves or attributes of moves. Short Tabu lists may not prevent cycling, thereby resulting in information loss while long Tabu lists may excessively prevent a neighborhood such that moves are limited to some extent (Harun, Engin, and Burak, 2008).

3 RESEARCH METHODOLOGY

THIS section represents the research methodology employed in the development of GAT_CS, which includes: The Design objectives, Requirements gathering, System constraints, Design approach, System Parameters, and the primary data source.

3.1 The Design Objective: Exam Timetable

The objective of this study is to design and implement an examination timetable system for a public University in Nigeria. All courses are assigned to a hall with enough seats to accommodate the course registered students; this will address the hall congestion issue. Course allocation and hall scheduling are optimized based on the constraints presented to the system.

The following are the components of the timetable:

1. The course code;
2. Total number of registered students;
3. The hall assigned to each course;
4. The examination session (morning or afternoon);
5. The day scheduled for each exam.

3.2 Requirements Gathering

During the system design, the Timetable and Examination Committee of the University was approached for necessary information and answers were promptly provided with supporting documents. Some of the details required were:

1. The number of halls in the University;
2. The capacity of each hall in the University;
3. The total number of courses to be examined in the semester exam;
4. The number of sessions per day;

5. The number of weeks for examination.

3.3 Constraints

As mentioned in the preceding section, there is a need to consider some factors that must be fulfilled at all cost. These factors are known as hard constraints. Hard constraints play a major role in arriving at an optimized timetable. The other constraints to be considered are soft constraints which do not necessarily affect the quality of the timetable but has considerable influence on the output of the system. They are also known as optional constraints.

3.3.1 Hard Constraints (Compulsory)

The following hard constraints were modelled by GAT_CS:

1. Exam Population (registered students for the exam) must be less than the hall capacity;
2. Practical should be scheduled to hold in its corresponding hall;
3. One hall must not be used more than twice in a day.
4. Students in one level should not have more than one exam at a time (exam session).

3.3.2 Soft Constraints (Optional)

For the soft constraint students in one level should not have more than two (2) theory-based examinations in one day.

3.4 Design Approach

The proposed University timetable system is a combination of modified Genetic Algorithm (GA), Tabu Search (TS) memory and course sandwiching to enhance hall optimization and course allocation accuracy. The developed timetable examination system is named GATS_CS. GATS_CS is expected to improve the performance and time required to discover the optimum solution in an automatic University timetable.

Direct encoding was used with each chromosome representing a candidate solution (i.e. a Timetable). The chromosome is a list that represents the number of courses to be scheduled (e.g. C); while an element in the list corresponds to the course examination to be scheduled (e.g. between 1 and C). For each value (gene) in the chromosome, the index of that value corresponds to the hall encoded with the number. Thus, for a chromosome [4, 8, 9, 7, 12, 6, 15], exam 4 will take place at hall 1, exam 8 at hall 2 and exam 12 at hall 5, respectively.

The Concepts of Genetic Algorithm i.e. Selection and Evaluation was implemented while the memory properties of TS (Taburization) and course sandwiching replaced Crossover and Mutation Operators. The algorithm required for the developed system is presented in the sub-sections below.

3.4.1 Selection Phase

A specific number of timetable solutions are randomly generated as specified at the beginning of the execution. After the initialization of the population, the fitness of each timetable is evaluated using a simple objective or fitness function to ascertain the viability of the candidate timetable. A simple fitness function based majorly on the hard constraints is evaluated to prevent complex calculations and reduce execution time. The timetable with the least contradiction (i.e. that best satisfies the constraints) is selected as the best fit and is moved to the next phase of the algorithm. Fitness function $f(X_1)$ is defined in Equation 1 for i=1… C.

$$f(X_1) = \frac{Exam\ Population\ (n_1)}{\text{Hall capacity}\ (h_a)} * 100 \qquad (1)$$

where X_1: First constraint; n_i: the population of the exam at index i and h_a: The capacity of the hall assigned to an exam at index i.

$$f(X_2) = (P_j: \text{Laba}) \quad j = 1 \ldots L \qquad (2)$$

where X_2: Second constraint; P_j: Practical exam at index j; Lab_a: Corresponding Laboratory and L: The total number of Labs.

After the best-fit chromosome (Timetable) has been selected, it is then split into two memories: Short Tabu and Long Tabu (Tabu Lists) as described in (Harun, Engin, and Burak, 2008). The 'Tabus' are also used to a perform memory function. This splitting is done by calculating the degree of fitness of individual allocation (i.e. EXAM: HALL); the ones that perfectly fit (i.e. 100% fitness) are placed in the 'Long Tabu' restricting them from further changes while the ones that do not fit perfectly are placed in the 'Short Tabu' for further evaluation and optimization. The nature of both 'Tabus' at the end of the selection phase is shown below:

TabuListShort = [exam 1, exam 2 … exam t]

TabuListLong = [EXAM 1: HALL A, EXAM 2: HALL B … EXAM 3: HALL Z]

The following are the steps required in the selection phase:

Step 1: Generate a random population of courses (Chromosomes);

Step 2: Evaluate the Total fitness of the population based on the individual fitness of each course;

$$\text{Fitness of each exam} = \frac{Hall\ Capacity}{\text{Exam Population}} * 100 \qquad (3)$$

Condition: Fitness =100%

Step 3: Select the population based on its Total fitness.

Find the selection algorithm below:

Input: i, k, x, e, E, H

Output: popFitness, BestFitPop

Start

1. Select $x \leftarrow I^{[1\ldots n]^C} \in \mathbb{Z}^+$
2. for$(1 \leq i \leq N)$
3. Compute $e_i \in L: i \in [E, H]$

4. for $(1 \leq k \leq H)$
5. Compute $f(x) \leftarrow i{:}k \ \forall \ (i,k) \in \ [E,H]$
6. While $(i = k)$
7. Compute $f(x) \leftarrow E[i] + H[k]$
8. Compute $E[i] \leftarrow H[k]$
9. Compute $E[i] \leftarrow f(x)$
10. Compute $f(x) \leftarrow H[k]$
11. for $(1 \leq e_i \leq E(n))$
12. Compute popFitness $\leftarrow E{:}\, H \,\forall\, i \in [E,H]$
13. If (Fitness= 100%) $\ni E[i]{:}H[i] \forall\, i \in [E,H]$
14. popFitness $\leftarrow$ popFitness
15. Else if
16. popFitness $\leftarrow$ popFitness+1 $\ni E[i] < H[i] \ \forall\, i \in [E,H]$
17. Else if
18. popFitness$>$ 100% $\ni E[i] > H[i] \ \forall\, i \in [E,H]$
19. popFitness$\leftarrow$ BestFitpop
20. Else
21. BestFitpop$\leftarrow$ BestFitpop $\ni E[i] \leftarrow H[i] \ \forall\, i \in [E,H]$

End

3.4.2 Optimization Phase

The best solution (that has been split into short and long Tabu) from the selection phase is adjusted (mutated) to ensure the solution produced is the most optimal and the values in the short Tabu form the next population. The halls are re-allocated such that the exam population fits perfectly or slightly above perfect as specified at the beginning of the execution (i.e. 100% or 120%). This phase is divided into two categories: (1) Taburization and (2) Optimization.

Taburization: This is the process involved in placing the allocations in its corresponding 'Tabu'. The following are the steps required in the Taburization phase:

Step 1: After selecting the population based on total fitness, the individual fitness of the allocated exams is used to place each chromosome (Courses) into its Corresponding 'Tabu'.

Step 2: The chromosomes that are optimal (i.e. meet up with the fitness condition) are placed in the Long Tabu List while those that are otherwise are put in the Short Tabu List. The Long Tabu List has the capacity to hold the courses for a long time while the Short Tabu is a memory structure to keep values stored in it for a temporary period.

Find the Taburization algorithm below:

Input: $E[i]{:}H[i] \ \forall\, i \in [E,H]$

Output $E[i]{:}H[i] \in$ LongTabu $\forall\, i \ = 1$ to L
$E[j]{:}H[j] \in$ ShortTabu $\forall\, j \ = 1$ to S

Initialize: LongTabu [ ] $\leftarrow$ BestFitpop $\leftarrow$ BestFitpop

While $n = L + S \equiv E[i+j]{:}H[i+j] \in (L \cup S) \subset n \forall\, (L,S) \in n$

1. for $(1 \leq i \leq E(n))$
2. Compute popFitness $\leftarrow E{:}\, H \,\forall\, j \in [E,H]$
3. If $(100\% \leq$ popFitness $\leq 120\%) \ni E[i]{:}H[i] \forall\, i \in [E,H]$
4. LongTabu [ ] $\leftarrow E[i] \forall\, i \ = 1$ to L
5. Else
6. ShortTabu [ ] $\leftarrow E[j] \ \forall\, j \ = 1$ to S

End While;

Optimization: This is introduced by adjusting the fitness value such that much more allocations are done in the next generation. The following are the steps required in the selection phase:

Step 1: Adjust the fitness condition to allow more optimal allocations (increase or decrease it);

Step 2: Randomly allocate halls to exams;

Step 3: Check if allocation meets up with the Optimal Condition (Satisfies the fitness condition);

Step 4: Augment the Long Tabu List with Optimal solutions from Step 3.

Only the Chromosomes in the Short Tabu List are used at this stage because they have not met with the fitness conditions. The optimization algorithm is below:

Where U is the number of times the hall can be used.

Input: $E[j]{:}H[j] \in$ ShortTabu $\forall\, j \ = 1$ to S

Output: $E[i]{:}H[i] \in$ LongTabu $\forall\, i \ = 1$ to L, U

1. While $E[i+j]{:}H[i+j] \in (L \cup S) \subset H[k]{:}U$
2. for$(1 \leq e_j \leq E(n)$
3. Compute popFitness $\leftarrow E{:}\, H \,\forall\, j \in [E,H]$
4. If (popFitness $> 120\%$)
5. LongTabu [ ] $\leftarrow E[j]$
6. Else
7. ShortTabu [ ] $\leftarrow E[j]$
8. End While;

3.4.3 Fine tuning Phase

The solution produced in the optimization is now separated into the number of days required, as specified as one of the parameters at the beginning of execution. Some other examinations that need to be sandwiched (i.e. a situation where more than one hall is allocated for an examination when exam population size is far higher than the capacity of the largest hall) are also considered. This phase is also divided into two parts, Sandwiching (Optional) and Separating into Days.

Sandwiching: This is the term used to describe the merging of two or more examinations to a venue such that a class population perfectly fits into the hall capacity. This process was introduced because some courses weren't assigned in the timetable since the system couldn't find a venue that fits the class size perfectly. Sandwiching is a major contribution of this paper, as observed from previous reviews. The ability of the timetable system to sandwich several courses

into a single examination hall increases the course allocation accuracy. The following are the steps required in the sandwiching phase:

Step 1: If there are still courses in the Short Tabu List, merge more than one exam for unused halls;

Step 2: Repeat Step1 until all courses in Short Tabu has been allocated to a Hall. The sandwiching algorithm is below:

Input: $E[j]:H[j] \in \text{ShortTabu} \; \forall \; j = 1 \text{ to } S$

Output: $E[i]:H[i] \in \text{LongTabu} \; \forall \; i = 1 \text{ to } L$

1. $E[i]:H[i] \in \text{LongTabu} \leftarrow L \forall \; i \in [E, H]$
2. $E[j]:H[j] \in \text{ShortTabu} \leftarrow S \; \forall \; j \in [E, H]$
3. While, unused hall, $U \leftarrow H[k]$ and $CapH[k] \subset U \ni E[j] \equiv U - CapH[k] \; \forall \; k \in [H]$
4. LongTabu [ ] $\leftarrow E[j]$
5. End While;

Separating course allocations into days: This is done by separating each allocated exam into days (Mondays-Fridays) with morning and afternoon sessions for three weeks (15 days) considering our case study as implemented below:

1. Let $D_T \leftarrow \sum_i^n D_i$ for $i \leq 5$
2. Suppose $\exists \; D_1$ to $D_5 \in D_T$
3. Then [D_1: Monday, D_2: Tuesday, D_3: Wednesday, D_4: Thursday, D_5: Friday]
4. Compute $C_T \leftarrow \sum_i^n C_i$
5. Since $A_v \leftarrow C_T / D_T$ {Where C_i represents each course and C_T represents the total number of courses}
6. Then $[C_i: D_i] \equiv [C_1: D_1, C_2: D_2, \; \ldots, C_n: D_n]$ for $i \leq n$

3.5 Parameters and Steps for Proposed System

The system will require certain inputs at the beginning of execution which will guide its operations. These inputs are the parameters of the system, and they include:

1. Total Number of Examination Papers;
2. Total Number of Halls;
3. The number of Days for an exam schedule.

The steps for the proposed design are as follows:

Step 1: Generate initial population;

Step 2: Evaluate the fitness of chromosomes in the population (based on hall allocations);

Step 3: Select the Best-Fit chromosome for Taburization and Optimization;

Step 3a: Taburization: split Best-Fit chromosome and store in two TabuLists: Short and Long TabuList;

Step 3b: Optimization: for values in the Short TabuList allocate hall that optimally fits the Exam, and augments Long TabuList with the latest allocation;

Step 4: For Exam population that far exceeds the maximum capacity of the hall, Sandwich Exams and augment Long TabuList with the latest allocation;

Step 5: End when the Termination condition is met.

The flowchart for University timetable system based on Genetic algorithm, Tabu search and Course sandwiching (GATS_CS) is presented in Figure 1.

3.6 Data Source

The dataset for the examination timetabling was obtained from a large public University in Nigeria with 10 Colleges, 41 academic Departments, and about 16,000 full-time students offering several undergraduate academic programs. Examinations are written for three weeks (Mondays-Fridays), each day having morning and afternoon sessions while the timetable is done manually using a word processor. The dataset presented here is the undergraduate written examinations for the 2015/2016 academic session and it excludes the computer-based (CBT) exams. The Timetable and Examination Committee of the University provided the dataset and the general parameters are shown in Table 1. It should be noted that the sitting capacities represent the number of students that can write an exam in the hall under ideal examination conditions, it does not represent the actual sitting capacity of the hall.

Table 1. The University Examination Timetable Parameters.

Parameter	First Semester 2015/2016	Second Semester 2015/2016
No. of Exam Halls	39	39
Total Exam Sitting Capacity of Halls	4,988	4,988
No of Academic Departments	41	41
Total No. of Courses Examined	483	408
No. of Students Writing the Courses	126,983	124,880

4 IMPLEMENTATION AND VALIDATION OF GATS_CS

IN this section, the GAT_CS timetabling system was implemented based on the proposed methods and algorithms while the system performance was validated and benchmarked with previous studies.

4.1 Implementation

This section provides the results obtained after implementing GATS_CS. As specified above the parameters of the algorithm are:

1. The Total Number of Examination Papers;
2. The Total Number of Halls;
3. The number of Days for an exam schedule.

Parameters 1 and 2 are entered into the system automatically through a file created for the program and read at the start of execution, this is known as "Initialize Population from file". After the second generation, it was realized that some courses have not been allocated because their fitness does not meet the value specified in the program. The concept of

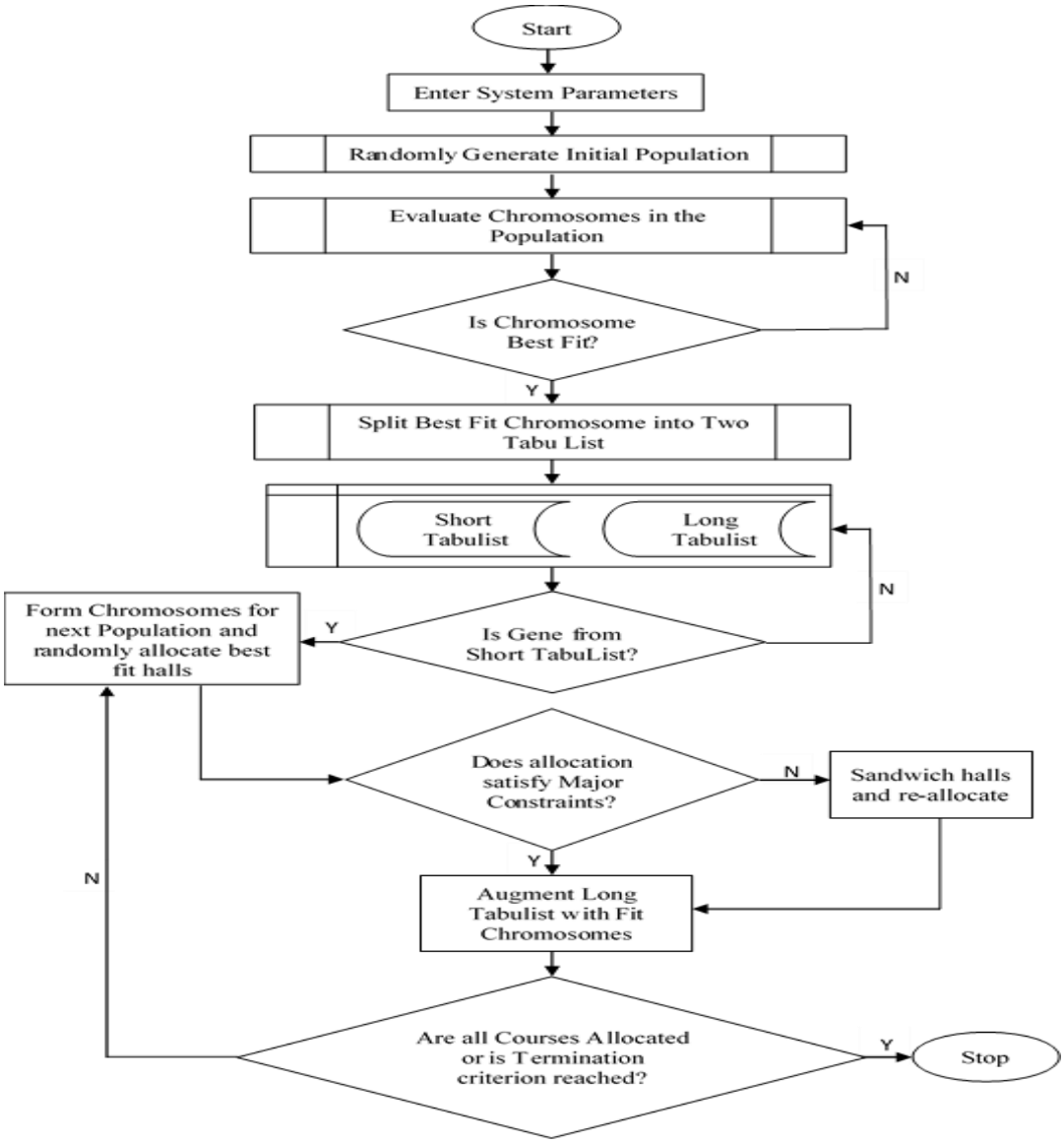

Figure 1. Flowchart for University timetable system based on Genetic algorithm, Tabu search and Course sandwiching (GATS_CS).

sandwiching is then applied which hasn't been realized before in previous studies, hence the low performance of such systems in allocating courses to halls.

Sandwiching involves scheduling two or more examinations to hold in one hall to meet up with the fitness condition specified. This is necessary for situations when an examination has been allocated to a hall, but the space is big enough to accommodate other examinations. Thus, more than one examination will be sandwiched (allocated) to hold in a single hall. For example, an examination hall (MP01) with exam capacity of 400 will accommodate five examinations concurrently namely: MCE510, ECO412, CVE308, FRM508 and FSM410 with 40, 100, 40, 60 and 90 students, respectively.

After sandwiching has been implemented, the course allocations (which are now in the Long Tabu List) are now classified into fifteen pots (corresponding to the number of days provided as input at the beginning of execution) for onward separation into days. The classification is done to simplify separation into days.

4.2 Validation

The following metrics were used to validate the performance of GATS_CS:

Allocation Accuracy (Acc): Total number of courses successfully allocated to an examination venue.

$$Acc = \frac{\text{Total number of allocations}}{\text{Total number of courses}} * 100 \quad (4)$$

Un-allocation (UA): Total number of courses not successfully allocated to an examination venue.

$$UA = \frac{\text{Total number of Un-allocations}}{\text{Total number of courses}} * 100 \quad (5)$$

Clash Courses (CC): A condition whereby two courses are allocated to the same venue at the same time.

$$CC = \frac{\text{Total number of Clashes}}{\text{Total number of courses}} * 100 \quad (6)$$

Duplicated Courses (DC): A condition whereby a course appears more than once on the timetable.

$$DC = \frac{\text{Total number of Duplicated Courses}}{\text{Total number of courses}} * 100 \quad (7)$$

Multiple exams (ME): A condition whereby a class has more than one exam at the same time in different venues.

$$ME = \frac{\text{Total number of multiple courses}}{\text{Total number of courses}} * 100 \quad (8)$$

Table 2 shows simulation experiments for GATS_CS with an average of 10 independent runs and different iterations. The time of the first solution represents time taken by GATS_CS to represent the first solution, while the time of best solution is the time taken to obtain the optimal solution. Average time in seconds is the mean of both first and best time, which for 10 runs was between 22.5-29.5 and 20-26 seconds in first and second semesters, respectively.

From Table 3, results show that GATS_CS provided optimal solutions which were considerably better than those obtained in the worst solutions. Allocation accuracy increased while un-allocation, clashes, duplication, and multiple exams were reduced after subsequent runs. However, observation shows that the first solutions were not the worst solution in some cases. Figures 2 and 3 show the deviation due to improvements between worst and best solutions during simulation experiments. It was observed that apart from the increase in course allocation, reduction of clashing courses was more pronounced than others with -11.80 and -11.52 in first and second semesters, respectively.

Table 2. GATS_CS Simulation Time with First, Best and Average Time in 2015/2016 Examinations.

Runs	First Semester Simulations				Second Semester Simulations			
	Iterations	Time of first solution (secs)	Time of best solution (secs)	Avg. Time (secs)	Iterations	Time of first solution (secs)	Time of best solution (secs)	Avg. Time (secs)
1	22530	14	45	29.5	26459	12	40	26
2	26759	8	45	26.5	25963	13	38	25.5
3	28241	5	41	23	26344	9	34	21.5
4	22857	12	38	25	18321	7	33	20
5	27266	11	40	25.5	22418	7	38	22.5
6	22449	9	36	22.5	20011	11	39	25
7	19520	6	46	26	18375	11	29	20
8	21580	15	31	23	18909	7	37	22
9	25202	9	42	25.5	27439	6	36	21
10	28345	8	45	26.5	22863	8	35	21.5

Table 3. GATS_CS Worst and Best Solutions in 2015/2016 Examination Timetable.

Conditions	First Semester				Second Semester			
	Worst solution		Best solution		Worst solution		Best solution	
	#	%	#	%	#	%	#	%
Allocation	399	82.61	464	96.07	365	89.46	404	99.02
Un-allocation	45	9.32	19	3.93	26	6.37	4	0.98
Clashes	67	13.87	10	2.07	53	12.99	6	1.47
Duplication	21	4.35	13	2.69	16	3.92	8	1.96
Multiple Exams	16	3.31	11	2.28	9	2.21	4	0.98

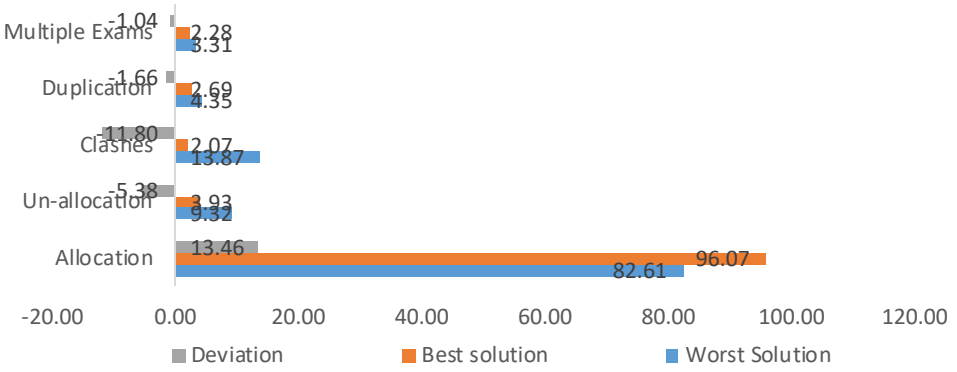

Figure 2. GATS_CS Worst and Best Solutions with the Improvement Obtained (Deviation) in First Semester Timetable.

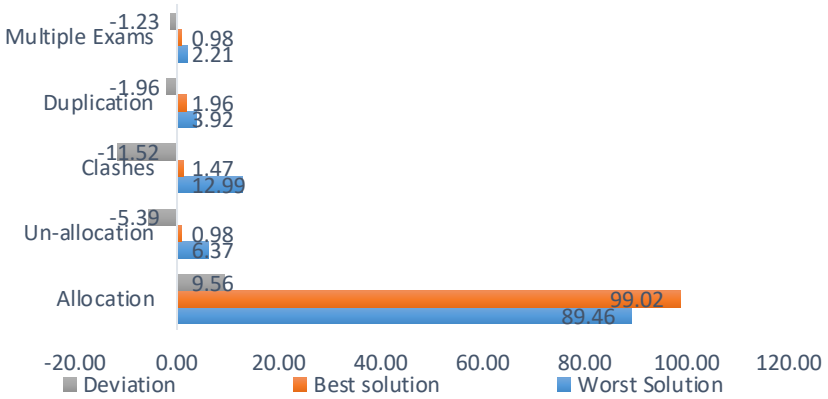

Figure 3. GATS_CS Worst and Best Solutions with the Improvement Obtained (Deviation) in Second Semester Timetable .

From the results, it was observed that GATS_CS successfully allocated 96.07% and 99.02% of the total courses while 3.93% and 0.98% courses were un-allocated in first and second semesters, respectively as shown in Table 4. Table 5 represents the performance of the present manual timetable system in the University with the results close to those obtained from GAT_CS especially with respect to course allocation and un-allocation. However, GATS_CS clearly outperforms the manual timetables in reducing course clashes, duplications, and multiple examinations considerably. Worth mentioning is the result of the manual timetable in Table 5 which is the final draft by the Timetable and Examination Committee after series of reviews from students and academic Departments three to four weeks before the commencement of examination.

Therefore, the results in Table 5 is not particularly impressive considering the cost in terms of time and the human resources expended to achieve it.

The performance of GATS_CS was also benchmarked with three other timetable systems previously implemented. They are Abayomi-Alli (2019) Particle Swarm Optimization with Local Search (PSO_LS), Genetic Algorithm (GA) closely related to Arogundade, et al. (2010) and Lawal et al. (2014) University Examination Timetabling Using Tabu Search. Results obtained in Table 6 showed that GATS_CS outperformed PSO_LS and GA in Allocation, Un-allocation, and Clash rates; while PSO_LS outperformed GATS_CS and GA in Duplication and Multiple examination rates, respectively. Lawal, et al. (2014) outperformed GATS_CS in course clash (hall clash) and multiple exams (exam clash) with 0% to 2.07% and 0.7% to 2.28%, respectively. However, the study didn't consider the percentage of allocated courses, unallocated courses and course duplications, which are the crux of the timetabling system evaluation. Results also showed that GAT_CS had faster simulation time in the range of 20-29.5 seconds as compared to an average of 252 seconds (4.2 minutes) reported in Lawal, et al. (2014).

Table 4. Summary of GATS_CS Performance on2015/2016 Examination Timetable.

S/N	Condition	1st semester Total number of courses= 483		2nd semester Total number of courses= 408	
		No of courses	%	No of courses	%
1	Allocation	464	96.07	404	99.02
2	Un-allocation	19	3.93	4	0.98
3	Clash	10	2.07	6	1.47
4	Duplication	13	2.69	8	1.96
5	Multiple exams	11	2.28	4	0.98

Table 5. Performance of the Final Drafts of the 2015/2016 Manual Examination Timetables.

S/N	Condition	1st semester		2nd Semester	
		No of Courses	Accuracy (%)	No of Courses	Accuracy (%)
1	Allocation	465	96.27	397	97.3
2	Un-allocation	18	3.73	11	2.70
3	Clash	40	8.28	23	5.64
4	Duplication	30	6.21	19	4.66
5	Multiple exams	14	2.90	6	1.47

Table 6. Comparing GATS_CS with PSO_LS and GA Based Timetable Systems.

S/N	Condition	1st Semester			2nd Semester		
		GATS_CS	PSO_LS	GA	GATS_CS	PSO_LS	GA
		Score (%)	Score (%)	Score (%)	Score (%)	Score (%)	Score (%)
1	Allocation	96.07	84.1	68.3	99.02	76.2	72
2	Un-allocation	3.93	15.9	31.7	0.98	23.3	28
3	Clash	2.07	10.2	15.3	1.47	2.5	17.4
4	Duplication	2.69	0	8.2	1.96	0	9.6
5	Multiple exams	2.28	0	5.1	0.98	0	3.3

5 CONCLUSION

IN this study, the problem tackled is a complex one considering the number of students writing the examination within the three weeks, the number of examination halls/venues available and their exam sitting capacities. The aim of GATS_CS timetable system was to solve the Universities examination timetabling problem automatically and efficiently using an enhanced combination of Genetic Algorithm and Tabu Search memory with a newly proposed and developed course sandwiching algorithm. Results obtained were quite promising in terms of course allocation to exam halls, optimizing large halls by allocating multiple examinations into them without exceeding the hall capacity with minimal errors. However, the system isn't 100% accurate because of issues like duplication of courses, course clashes and multiple allocations of courses but the results obtained were promising, way faster and less cumbersome than the present manual timetable approach. For future research directions, some constraints worth considering are issues with:

1. Courses electives;
2. Students re-writing failed courses;
3. Students offering courses at lower levels;

All these would be considered as part of future research directions in University examination timetabling system while trying to increase the course allocation rates and reducing clashes, duplications, and multiple examinations. Future University timetabling systems could also consider research into (1) Computer-based examinations (e-exams) and (2) Lecturers (invigilator) scheduling.

6 ACKNOWLEDGEMENT

THE authors will like to thank the Timetable and Examination Committee of the case study University for providing the relevant data for the study. We also appreciate all the anonymous reviewers for providing constructive and generous feedback all throughout the review process.

7 REFERENCES

Abayomi-Alli O., Abayomi-Alli A., Misra S., Damasevicius R. and Maskeliunas R. (2019). "Automatic Examination Timetable Scheduling Using Particle Swarm Optimization and Local Search Algorithm", Shukla R. K. et al. (eds.), *Data, Engineering and Applications*, Springer Nature, Singapore Pte Ltd, DOI.:10.1007/978-981-13-6347-4_11

Ahmed, K. and Keedwell, E. (2016). "A Hidden Markov Approach to the Problem of Heuristic Selection in Hyper-Heuristics with a Case Study

in High School Timetabling Problems", *Evolutionary Computation*: pp. 1-29.

Anisha, J., Ganapathy, A., Harshita, G. and Rishabh, B. (2015). "A Literature Review on Timetable Generation Algorithms Based on Genetic Algorithm and Heuristic Approach", *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):159-163.

Anuja, C., Priyanka, K., Shruti, D., Sonali, I., Rupal, R. and Dinesh, G. (2014). "Timetable Generation System", *International Journal of Computer Science and Mobile Computing*, 3(2):410-414.

Arindam, C. and Kajal, D. (2010). "Fuzzy Genetic Heuristic for University Course Timetable Problem", *International Journal of Advances in Soft Computing and its Applications*, 2(1):101-150.

Arogundade, O., Akinwale A., Aweda, O. (2010). "A Genetic Algorithm Approach for a Real-World University Examination Timetabling Problem", *International Journal of Computer Applications*, 12(5):1-4.

Bargetto, R., Della, C., and Salassa (2016). "A Metaheuristic Approach for an Examination Scheduling Problem", *International Conference on the Practice and Theory of Automated Timetabling*, Udine Italy: pp. 23-26.

Beasley, D., Bull, D., and Martin, R. (1993). "An Overview of Genetic Algorithms: Fundamentals", *University Computing*, 15(2):58-69.

Burke, E. K., and Bykov, Y. (2008). "An Adaptive Fle-Deluge Approach to University Exam Timetabling", *Informs Journal on Computing*, 28(4):781- 794.

Darwin, C. R. (1859). The Origin of Species. Vol. XI. The Harvard Classics. New York: P. F. Collier and Son, 1909–14; www.bartleby.com/11/. [10th May 2017].

Daskalaki, S., Birbas, E., and Housos, E. (2004). "An Integer Programming Formulation for a Case Study in University Timetabling", *European Journal of Operational Research*, 153(1):117–135.

Dipesh, M., Hiral, D., Mohammed, S. and Renuka, N. (2015). "Automatic Timetable Generation using Genetic Algorithm", *International Journal of Advanced Research in Computer and Communication Engineering*, 4(2):245-248.

Dréo, J., Petrowski, A., Siarry, P. and Taillard, E. (2006). "Metaheuristics or Hard Optimization: Methods and Case Studies", *Springer-Verlag Berlin*, Heidelberg, Germany.

Esraa, A. and Ghada, A. (2016). "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem", *Alexandria Engineering Journals*, 55(2):1395-1409.

Gaspero, L. and Schaerf, A. (2000). "Tabu Search Techniques for Examination Timetabling", *LCNS Archive Selected Papers from The Third International Conference on Practice and Theory of Automated Time Tabling*, Springer-Berlin Heidelberg, Konstanz Germany, (2079):104-117.

Gen, M., Cheng, R. and Lin, L. (2008). "Network Models and Optimization: Multi-objective Genetic Algorithm Approach", *Springer London*:1- 47.

Gen, M. and Cheng, R. (1997). "Genetic Algorithms and Engineering Design", *Wiley Interscience Publication*: 1-2.

Gendreau, M. (2002). "An Introduction to Tabu Search", *International Series in Operations Research and Management Sciences*, (57):37-54.

Glover, F. and Laguna, M. (1997). "Tabu Search", *Springer US*. (1):19-382.

Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operation Research*, (13):533-549.

Hamed, P., Kuan, Y. and Adnan, H. (2016). "A Hybrid Genetic Algorithm with a Knowledge-Based Operator for Solving the Job Shop Scheduling Problems", *Journal of Optimization*, (2016):1-13.

Harun, P., Engin, B. and Burak, E. (2008). "Tabu Search: A Comparative Study", *Tabu Search*, I-Tech, Vienna Austria:1-29.

Jose, J.M. (2008). "A System of Automatic Construction of Exam Timetable using Genetic Algorithms", *I-CSRS*: 319-336.

Joseph, M. and Jonathan, A. (2012). "Implementation of a Time Table Generator using Visual Basic.Net", *ARPN Journal of Engineering and Applied Sciences*, 7(5):548-552.

Lawal, H. D., Adeyanju I. A., Omidiora, E. O., Arulogun O. T., and Omotosho O. I. (2014). "University Examination Timetabling Using Tabu Search", *International Journal of Scientific & Engineering Research*, 5(10):785-788

Mccollum (2006). "University Timetabling: Bridging the Gap", *Proceeding of the 6th International Conference Practice and Theory of Automated Timetabling*: pp. 15-35.

Merlot, L., Boland, N., Hughes, B. and Stuckey, P. (2003). "An Hybrid Algorithm for the Examination Timetabling Problem", *LNCS*, (2740):207-231.

Michiels, W., Aarts, E., and Korst, J. (2007). "Theoretical Aspects of Local Search, Monographs in Theoretical Computer Science", *EATCS Series*, Springer Berlin Heidelberg:1-238.

Mousavi S. M. and Zandieh M. (2018). "An Efficient Hybrid Algorithm for a Bi-objectives Hybrid Flow Shop Scheduling", *Intelligent Automation and Soft Computing*, 24(1):9-16.

Radomir, P. and Jaroslav, R. (2013). "Self-Learning Genetic Algorithm for a Timetabling Problem with Fuzzy Constraints", *International Journal of*

Innovative Computing, Information and Control, 9(11):4565-4582.

Rawats, S. R. (2010). "A Timetable Prediction for Technical Education System using Genetic Algorithm", *Journal of Theoretical and Applied Information Technology*, 13(1):59-64.

Ruey-Maw, C. and Hsiao-Fang, S., (2013). "Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search Algorithms", *MPDI*, (6):227-244

Russell, R., Chiang, W. and Zepeda, D. (2008). "Integrating Multi-Product Production and Distribution in Newspaper Logistics", *Computers and Operations Research*, 35(5):1576-1588.

Sadhasivam, N. and Thangaraj, P. (2016). "Design of an improved PSO algorithm for workflow scheduling in cloud computing environment", *Intelligent Automation and Soft Computing*, 23(3):493-500.

Srivastav, A. and Agrawal S. (2017). "Multi-Objective Optimization of Slow Moving Inventory System Using Cuckoo Search", *Intelligent Automation and Soft Computing*, 24(2):343-350.

Van Staereling, I.V. H. (2012). "School Timetabling in Theory and Practice", *Faculty of Sciences, VU University*, Amsterdam:1-50.

Wilke, P. and Ostler, J. (2008). "Solving the School Time Tabling Problem using Tabu Search, Simulated Annealing, Genetic and Branch and Bound Algorithms", *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 23(1):14-16

Wren, A. (1996). "Scheduling, Timetabling and Rostering – A Special Relationship", *In LNCS: Practice and Theory of Automated Timetabling*, (1153):46-75.

8 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

9 NOTES ON CONTRIBUTORS

Abayomi-Alli Adebayo graduated B.Tech. and Ph.D Degrees from Ladoke Akintola University of Technology, Ogbomoso, Nigeria in Computer Engineering and Computer Science, respectively while his M.Sc. Degree in Computer Science was from University of Ibadan, Ibadan, Nigeria. He is a Senior Lecturer at the Federal University of Agriculture, Abeokuta, Nigeria. His research interest are Pattern recognition, Machine learning, and Soft computing.

Sanjay Misra is a Professor in Computer (Software) Engineering at Covenant University, Nigeria. He has authored more than 250 research papers and received several awards. He has delivered more than 60 keynote/invited speeches at international conferences and institutions spanning across 45 countries. His research interest covers Software Engineering, Project management, Quality assurance, HCI, AI, Cognitive Informatics and Web Engineering.

Luis Fernández-Sanz is an Associate Professor at the Department of Computer Science of the University of Alcalá (UAH). He earned a Degree in Computing at Polytechnic University of Madrid (UPM) and Ph.D. in Computing with a special award at the University of the Basque Country. He has over 20 years of research and teaching experience at UPM, Universidad Europea de Madrid and UAH. His general research interests are software quality and engineering, accessibility, e-learning, and ICT professionalism and education

Abayomi-Alli Olusola obtained a B.Sc. Degree in Electronics and Computer Engineering from the Lagos State University, Nigeria and M.Sc. Degree in Computer Science from the Federal University of Agriculture, Abeokuta, Nigeria. She is presently on her Ph.D. and also a faculty in the Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria. Her research interest includes information security, text mining and Soft computing.

Edun Rebecca Abisola obtained a B.Sc. Degree in Computer Science from the Federal University of Agriculture, Abeokuta, Nigeria in 2017. Her research interest includes optimization and application of nature-inspired algorithms.