Nigerian Journal of Technology (NIJOTECH)
Vol. 39, No. 1, January 2020, pp. **1 – 14**

Print ISSN: 0331-8443, Electronic ISSN: 2467-8821
www.nijotech.com
http://dx.doi.org/10.4314/njt.v39i1.1

AUTOMATED UNIVERSITY LECTURE TIMETABLE USING HEURISTIC APPROACH

A. M. Hambali [1,*], Y. A. Olasupo [2] and M. Dalhatu [3]
[1, 2, 3,] COMPUTER SCIENCE DEPARTMENT, FEDERAL UNIVERSITY WUKARI, WUKARI, TARABA STATE, NIGERIA
E-mail addresses: [1] *hambali@fuwukari.edu.ng,* [2] *ademolay15@gmail.com,*
[3] *dalhatu@fuwukari.edu.ng*

ABSTRACT
There are different approaches used in automating course timetabling problem in tertiary institution. This paper present a combination of genetic algorithm (GA) and simulated annealing (SA) to have a heuristic approach (HA) for solving course timetabling problem in Federal University Wukari (FUW). The heuristic approach was implemented considering the soft and hard constraints and the survival for the fittest. The period and space complexity was observed. This helps in matching the number of rooms with the number of courses.

Keywords: *Heuristic approach (HA), Genetic algorithm (GA), Course Timetabling, Space Complexity.*

1. INTRODUCTION

Scheduling is necessary in many sectors for effective and smooth operations. It can be found in public transport, at hospital, education sectors etc. Educational environs, especially the higher institution of learning, have a lot of things to be put into consideration which makes scheduling a great task to accomplish [1]. Some of the common considerations include the availability of lecturers, number of classes and courses, and budgeting.

Planning schedule manually is effort and time consuming compare to automated scheduling. In scheduling, several constraints have to be fulfilled. The Common constraints that need to be considered are; only one teacher can teach one class at one specific time, a room can only be occupied by one class at a time and students should not have more than one class each time period. These constraints are often divided into hard and soft constraints [2]. The hard constraints are not allowed to be violated, while the soft constraints may be violated, but with the setback of a less optimal scheduling. Due to the huge amount of time and money spent on scheduling manually, there have been numerous attempts to automate this task with the help of computers. Research has shown that this problem is most commonly NP-complete (Non polynomial Complete) [3]. However, this of course depends on how many and how complex the constraints are. Due to the difficulty of problem and the many different constraints, there is no general algorithm which will find the optimal solution for every timetable problem. To get around with this problem, several optimization algorithms have been implemented. These algorithms are mostly meta-heuristic and range from local search algorithms like Tabu search [4] and simulated annealing [5] to evolutionary algorithms like particle swarm optimization [6] and genetic algorithms [7]. Jonas and Rasmus [8] revealed that the reason for the many different algorithms being implemented is because of the complex nature of the problem. Almost every school has different constraints and pre-conditions which need to be fulfilled. The evolutionary algorithms mostly perform better in the early stages of the process whereas the local search algorithms perform better in the late stages. This has led to the creation of many hybrid algorithms [9] which use evolutionary algorithms to narrow down the search space and local search algorithms to find the best solution in that space.

Providing an effective timetable that affects the life of students and lecturers in the university education; it is very necessary to develop an automated system

* Corresponding author, tel: +234 7065868393

to tackle both simple and complex timetabling problem. It leads to substantially better timetables compared to those made manually, and it significantly lessens the work of the university's administration.

In general, a university course timetabling problem usually refers to finding the exact allocated time within a limited time period. For example, in a week, a number of events (courses-lectures) schedule and assign of the events to a number of resources (lecturers-rooms) in such a way that a number of constraints are satisfied. According to Petrovic [10], on " Novel Similarity Measure for Heuristic Selection in Examination Timetabling", as cited by [2] has defined the timetabling as the allocation of a set of subjects into a classroom over a limited number of time periods to avoid the occurrence of conflicts of interests between two subjects or lecturers. A good scheduling technique that can lead to optimization is important to ensure it is able to produce all timetable for students and lecturers. The main problem in the university timetable generation is to provide lecturers and lecture activities by matching all lectures with allotted time as well as the person responsible for it. The information required for the course schedule including room availability, time slots and several specific policy options. For example, information on room availability can be specified to the room capacity for certain events. In the domain of university timetable, it is often used to refer to the construction of schedule (with time slots) through the system by considering several numbers of constraints [2].This proposed system is developed for FUW to generate courses timetable. The system can be adopted by any Nigerian university with similar constraints and characteristics. Heuristics approach is used to allocate courses, period and room resources in different phases so as to obtain a feasible solution that will satisfy the users' requirements (lecturers and students).

2. REVIEW OF RELATED WORK

Zahra [11] works on four main metaheuristic algorithms to solve the course timetabling problem on ten dataset. The four algorithms were compared. It was found at the final result that Ant colony System (ACS) works better of all, follow by the Tabu Search (TS). ACS gives initial better solution than the TS and thus gives a less cost reduction.

Metaheuristic can be used to solve timetabling problem as suggested by [12]. In this case, the metaheuristic algorithm noted to have been divided into three categories; two stages of optimization algorithm and algorithm that allow relaxation.

Samuel, Arnold and Milyandreana[1] suggested heuristic and genetic algorithm for course timetabling problem. The target matrix does not include room. The system merges the room to the cell in the timetable. They planned the course available in the parallel matrix to all course in the timetable, and if there is a need for courses to be split, it has to be differentiated.

Asaju et al. [13] provide an effective way of solving course Timetabling problem using artificial bee colony algorithm. The tool carefully traverses the UTP search space using the vicinity structure repetitively, surrounded by the onlooker and employed bee operators. The ABC algorithm was assessed using curriculum-based course timetabling (CB-CTT) together with the Uncapacitated Examination Timetabling Problem (UETP).

In [14], the room scheduling of lectures in Utrecht University was planned to be automated. The number of lectures examined with the density of the Problem (NP-Complete) account for difficulty of the current system. The current system uses the greedy algorithm. A local search was effectively employed which resolve the main problem together with further restrictions preferred by the scheduler at the Utrecht University. It was found that the algorithm of syllabus plus was far better than a greedy algorithm at the expense of considering several algorithm.

Awadallah et al. [15] suggested a hybridization of Harmony search Algorithm for Nurse Sheduling problem. The Harmony search Algorithm is hybridize with a greedy shuffle. The algorithm was tested on four dataset which is outline in the first International Rostering Competition.

Solving timetabling problem prove to be cumbersome and in most cases contain series of conflicts [16]. He stated that the timetabling problem could be solved by encoding them into maximum satifiability (Max-SAT). The course timetabling problem of the department of Mathematics, Cairo University was tackle by encoding them into Max-SAT. The constraints were modeled as maximum satifiability (Max-SAT) instance.

Hamed, Jaber and Amin [17] analyze available approaches for solving university timetabling problem which include the metaheuristic methods, operational researches, intelligent novel method and distributed multi agent system based approach known as Cooperative Search method. Similarly, the investigation of distributed multi agent system approach enables the timetabling of similar events between the departments. Moreover, Hamed, Jaber and Amin [16] stated that the hard constraints should not, in any circumstances, be dishonored and the soft constraint should be less violated.

Ruggero et al. [18] worked on simulated annealing and tunning search approach to solve curriculum-based course timetabling problem(CB-CTT). It was reported the outcome of the methodology employed account for the modelling of similarities between the parametric search method and instances features which allows the setup of parameters unseen instances base on the assessment of instances itself.

De Werra, Asratian and Durand [19] modelled the timetabling in similitude to the edge of boarder of a bipartite multigraph. They made an extension by dividing the set of classes into groups. It contains a given lecturer to a given class and a set of lecturers given lectures to group of classes. Moreover, they stated that NP (Non Polynomial) – complete is only when the a lecturer is to three groups of classes.

Recent article reveals that a new approach of combining the invocation standard of Mixed – Integer programming (MIP) solver with problem specific modulo network simplex heuristic method (ModSim). It was later reported that the iterated approach allowed the ModSim to prevail over the local minima proficiently and enable the MIP solver to be a better start up solution. The experiment was based on sampling 16 railway instances of PESPlib which is the available periodic event scheduling problem instance at that moment. With the iterative combine method used, it is said to account for the reduction of the purpose of earlier known solution to at least 10% and up to about 23% approximately [20].

Gerhard et al. [21] presented a paper on the 3rd international Timetabling Competition 2011. The paper was reported with the aim to raise the profile of automating timetabling problem in higher institution. 35 instances were considered in 10 countries. It explains the data model used, XML data format, in which the ambiguous instances and solution can be simplified precisely.

Naderi [22] proposed three algorithms for solving the problem of university course timetabling in form of linear integer programming model. The three algorithms proposed are the imperialist competitive algorithm, variable neighborhood search and simulated annealing. The outcome shows that the imperialist competitive algorithm outweighed the other algorithm in term of performance.

Tomáš et al. [23] initiate a Passenger Centric Train Timetabling Problem. This accounts the contentment in the design of the scheduling and considered the cyclic and non-cyclic timetables. The non-cyclic timetable shows high density demand in comparison to cyclic timetable.

In considering a student-centric point of view during scheduling to solve the Examination Timetabling Problem, two columns Generation Algorithm could be used [24]. The article explained how it is used to solve the examination timetable at KU Leuven campus Brussels in Belgium.

2.1. Constraints

The problem is considered solved when the following criteria are satisfied:

Every event in every course is assigned a time slot.

All events are in the right kind of room.

No student group has two events at the same time.

No lecturer has two events at the same time.

No two events are scheduled in the same room at the same time.

No event is in a room with less capacity than the number of students at the event.

These constraints are referred to as hard constraints, which mean that they are absolutely necessary for the solution to be valid [8].

3. METHODOLOGY

In this research work, in order to achieve an optimum output design, the researchers adopt Charles's Darwin theory on survival of the fittest (genetic algorithm), simulated annealing along with graph coloring heuristic to generate a multidimensional array as space for referencing the entire courses in any given semester.

Requirement Analysis

What is required of the system would be:

A set of courses C with n the number of courses

A set of lecturers T with t the number of lecturers

A set of rooms R with r the number of rooms

A set of timeslot P with d the number of days and h the number of hours

A set of courses K with k number of departments offering at least one course
A set of hard constraints X with x number of constraints
A set of soft constraint S with s number of soft constraints
A set of faculty W with w number of faculties
A set of department Q with q number of departments
A set of levels M with m number of level
In addition to the above expectations, all the credit units of courses on a set C must be spelt out. The set of lecturers is an optional requirement at the general perspective of the university's timetable, however, this should be considered at the faculty or departmental level of timetabling. So all combinatorial optimization challenges must have the goal of finding a solution, which is a combination of a set of discrete variables that could respect all hard constraints and minimizes or maximizes the value of the objective function (soft constraints). This is best attainable by the administrative policy of the university.

3.1. Method of Data Collection

Data for this research work was collected from Faculty of Pure and Applied Sciences, Federal University Wukari, Taraba State.

3.2. Proposed System Model

The following assumptions may be taken into consideration for effective performance of the design. These are highlighted as follows:

i. *No student should attend more than one event (lecture) at the same time.*
ii. *The room must be big enough for all the attending students and should satisfy all the features required by the event.*
iii. *Only one event is put into each room in any timeslot.*
iv. *Events are only assigned to timeslots that are pre-defined as available for those events.*
v. *General course should be available for all faculty students.*
vi. *The timetable has to deal with total courses registered in a particular Semester.*
vii. *Each room is to be of defined timeslots every week.*
viii. *Wednesday session stops by 3pm and Friday session stops by 1pm till 3pm.*
ix. *Some courses that need more than one session per week may not necessarily be considered.*
x. *Two sessions of one course cannot be set on one day or two days consecutively.*

3.3. Framework of the Proposed System

The layered structures of this problem indicate what program should be built and how they will interrelate. The system interface offers programming tool used for designing the system. The framework involves the flowchart of the proposed system. The proposed framework clarifies the steps and conditions that could be adopted while developing the system. The framework is based on four perspective area of interest for efficient running of the system:

1. Schedule courses on regular lecture rooms while the number of students enrolled for the course is not greater than the room capacity.
2. Schedule practical courses on specified laboratory while the number of students enrolled for the course is not greater than the room capacity
3. Schedule inter departmental courses (Crs) considering the room capacity and the common room that can accommodate the departments involved such that the timeslots of their respective departmental and GST Courses (Crs) is not clashed or overlapped.
4. Schedule GST courses on regular lecture room in the highest room capacity considering the timeslots for departmental and practical courses of each department that register the courses.

From this proposed framework there are several steps that must be carried out to solving the problem. Fig. 1 and 2 depicts the architecture of the timetabling problem for Federal University Wukari (FUW).
The processing of input data requires that departmental and practical courses must be scheduled before interdepartmental & GST Courses. The polygon represents a fresh step which is a subset of step in the system; for example, all departmental courses from one department to the next department in a faculty are scheduled before jumping to other departments in another faculty. This can be accomplished sequentially while the courses and the room number are in ascending or descending order.

3.4. Architectural of Proposed System Flow Chart

Allotting of time slots, the periods, days & rooms for each course must start from hundred (100) levels to five hundred (500) level beginning from the first department in the first faculty to the last department of the last faculty. This means scheduling in sequential order except where the algorithm changes in case of the complexity found in timeslots for interdepartmental and GST courses. The derivative can be viewed as follows:

$a[W_rQ_kM_yC_x]$ stands for the array of total courses of the institution per semester such that faculties, departments, levels, courses are arranged in sequential order. $W_r = \{w_i \ldots w_l\}$ is a set of all faculties where w_i is the first faculty in the set and w_l is the last faculty, the same applicable to the departments, levels and courses.

4. META-HEURISTIC ALGORITHMS

Meta-heuristic Algorithm utilizes the evolution system of algorithm to solve the timetabling problem. The most commonly meta-heuristic algorithms using the power of evolution are the genetic algorithm or local search such as simulated annealing. These kinds of algorithms calculate an approximate solution rather than the optimal one. This is to severely decrease the run time of the program and still get an acceptable solution.

4.1. Genetic Algorithm

A genetic algorithm starts with a set of random solutions to the problem [8]. This algorithm works with number of solutions and each solution is called a chromosome. The chromosome consists of several genes which are values corresponding to certain properties in the solution. The genes can then be used to control the fitness of the chromosome. Based on the chromosomes' fitness, a new off string is created by crossing. These off springs are then randomly mutated to create a bigger search space. When an offspring matches a specified fitness condition, this means an acceptable solution has been found and the algorithm terminates. There are two main stages in the genetic algorithm; the selection and the crossover.

4.1.1. Selection

There are few ways in selecting which chromosomes to be crossed. Some of these are elitism selection, roulette-wheel selection and tournament selection [7].

4.1.2 Crossover

It may vary which genes are carried over when two chromosomes are being crossed. To decide this, there are few different methods: Some of them are single point crossover, two point crossover and uniform crossover [7].

4.2. Simulated Annealing

Simulated annealing is based on neighborhood search with the special property of sometimes accepting a worse solution to avoid getting caught in a local optimum and instead finding the global one. The idea of simulated annealing is inspired by the annealing process in metal work. The colder a metal is the more stable its shape is. To change the shape of the metal it is heated up and then processed while it is cooling down, ultimately freezing its shape until reheated. Simulated annealing works in a similar way, where it has a temperature variable controlling the heating process [8].

The temperature variable is initially set to a high value and is then slowly decreased while the algorithm runs. The higher the temperature is, the more probable the algorithm is to choose a worse solution than the current one. This gives the algorithm the chance of avoiding getting stuck in a local optimum early on. As the temperature decreases, so does the chances of the algorithm choosing a worse solution, which in the end leads to a local search in a much more narrow search space and hopefully finding a close to optimal solution. Algorithms using only downhill search have a very large chance of getting stuck in a local optimum, whereas a better global optimum might be found just a few neighbors away. As reported in [8], the graduated cooling process terminates this problem effectively and makes it much better than the downhill algorithms on large search space with numerous local optima [5].

4.3. Time Complexity

The timetable contains a fixed number of time slots available to assign events to timeslots. These timeslots can contain several events in parallel as long as none of these events have the same resources. There are also other constraints that specify which events that can be contained in the

same time slot for interdepartmental courses. The times usually make up one week, but this is up to the university's managerial decision. There may be instance of where two weeks make the timetable instead of one week; this issue relies upon NP-complete problems. However, it would be more efficient to fix all the necessary resources (hard constraints) in one week on the timetable.

Another challenge that needs to be resolved is allocating appropriate hour(s) for every course. This may be influenced by the university administration and management based on the extent of current resources as well the growing rate of the institution. From this scenario, the lecture hour for each course will vary according to credit units Table 1: Lecture Length (Hour).

4.4. Computational Complexity

It is provable that the decision problem underlying FUW is NP-complete. The following proof considers (as could be seen in the next section) one sectional computation in which all courses are assigned on each semester. This means that the sub problem of selecting the appropriate courses and scheduling them feasibly.

Table 1: Lecture Length Per Credit Unit

1	1credit unit	2credit units	3credit units	Proportional to credit units
2	1 credit unit	2 credit units	3 credit units	If credit unit≤2, lect.hr = 2hrs else lect.hr= 3hrs
3	1credit unit	2credit units	3credit units	If credit unit≤3, lect.hr=2hrs else it should be 3hrs

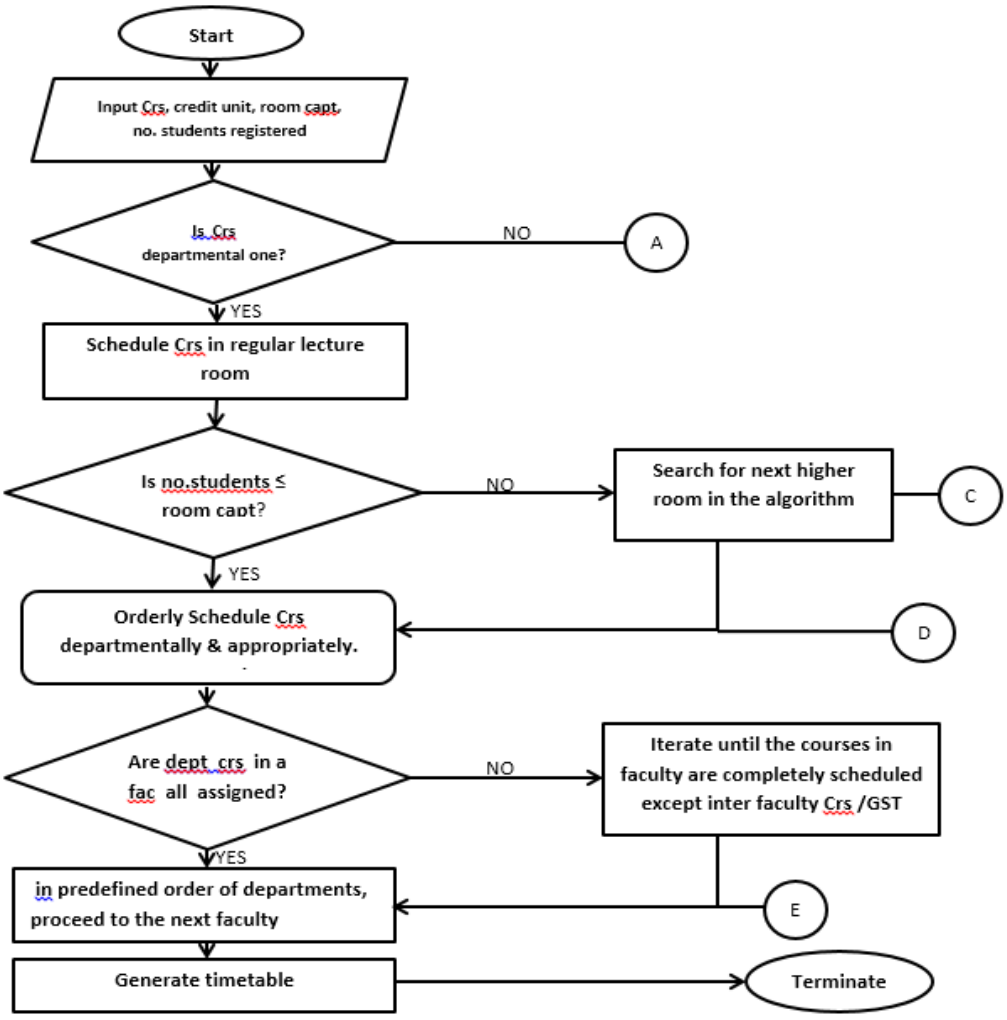

Fig. 1: Proposed System Framework (Part A)

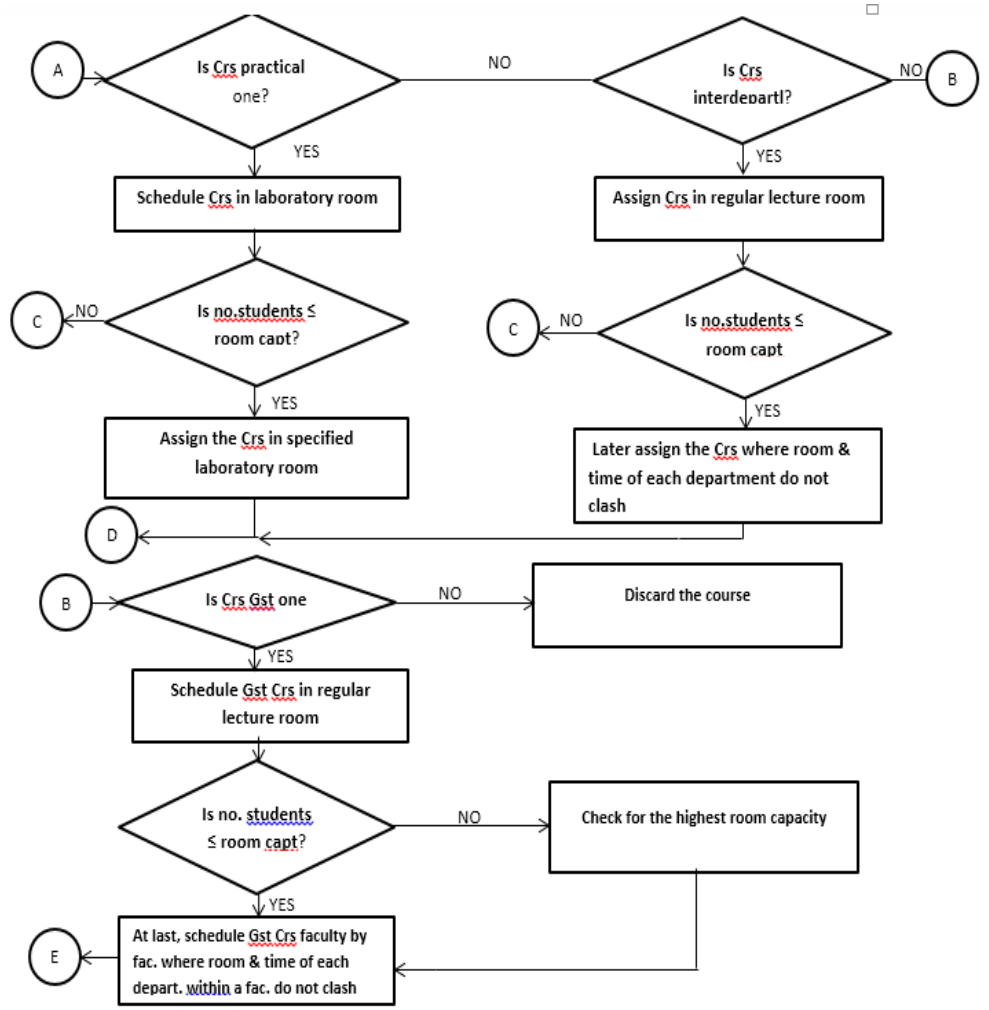

Fig. 2: Proposed System Framework (Part B)

4.5. Space Complexity

To generate the timetable, as had been emphasized earlier in the previous sections, is to allocate each course on a set of rooms at specific lecture period without overlapping. The lesson period for all the courses must correspond to a memory location respectively. However, general courses at departmental or at faculty level may share the same lecture hall and time (memory space) depending on the schedule. The space for departmental course must be distinct altogether within the matrix (array). Really, searching for space in order to schedule interdepartmental, GST and practical courses having scheduled departmental courses involve thorough swapping. Furthermore, it would be better to consider departmental course first in allocation to ease the complexity.

4.6. Memory Space Generation

The total number of lecture hall for a semester should be grouped sequentially in alphabetical order such that the cardinality of each group must exactly be equal except where it is necessary. Even though, number of days of lecture on each room and lesson period may not be the same, it will be quite fitting to initialize all the resources with the room having the highest number of days per week and periods a day. This is to enable the generation of perfect dynamic array; but the number of events on each room as initialized by the user will remain constant. Therefore, the memory space for such rooms will be considered imaginary; that is array index for such rooms would be out of bound.

Let the total room number be R_{total}= N Where N is the sum of all the lecture halls which is

$N = \sum_{i=1}^{L} R_i = R_1 + R_2 + \ldots + R_L.$ (1)

Therefore, $^{R_{total}}/_{R_{max}} = G_{total}.$ (2)

$R_{1max} = R_{2max} = \ldots = R_{nmax} \| R_{1max} \neq R_{2max} \neq \ldots \neq R_{nmax}.$ (3)

That is the number of days for lecture to be scheduled in each room in a week may or may not be the same due to university policy; for this reason, the hall(s) with the highest number of days for scheduling could be taken as the R_{max}.

The Room Grouping:

Initially, $G_{total} = {}^{R_{total}}/_{R_{max}} = G_1 = G_2 = G_3 = \ldots = G_n$ (4)

iffR_{total}/R_{max} is an integer else $G_1 = G_2 = \ldots \neq G_n$

where G_n is the last group. Notwithstanding, G_n must be scheduled whether it is equal to G_1 or not but each number referencing the value of array variables for last group will be imaginary. However in the series, such imaginary cell must be considered. Ideally, for less strenuous derivation of the algorithm for efficient allocation of resources, rooms and lecture hours need to be grouped to form an array.

Conditional Room Grouping:

If $R_{total}/R_{max} = G_{total}$ such that G_{total} is an integer value then

$G_1 = \{R_i \ldots R_x\}$, $G_2 = \{R_{(x+1)\ldots}R_{2x}\}$, $G_3 = \{R_{(2x+1)\ \ldots}\ R_{3x}\}$, $G_n = \{R_{((n-1)x+1)}\ldots R_{nx}\}$ (5)

where $G_1(|R|) = G_2(|R|) = \ldots = G_n(|R|)$ for the whole group; that is the number of venues for each group must be equal. But, if the G_{total} is a double value, then $G_n(|R|) \neq G_1(|R|) = G_2(|R|) = \ldots$

Exceptionally, $G_n(|R|) = R_{total} - R_{max}$ ($R_{total}\%R_{max}$).

The last group G_n with least number of rooms would be declared as having the same number of rooms as G_1; the memory locations for the additional rooms for the last group may be restricted from scheduling any event (course). For this reason,

$G_1 = \{R_i \ldots R_x\}$, $G_2 = \{R_{(x+1)\ldots}R_{2x}\}$, $G_3 = \{R_{(2x+1)\ldots}R_{3x}\}$, $G_n = \{R_a \ldots R_L\}$ (6)

where R_a is the room number after the last room number of the preceding group and R_L is the last room number of the entire group. Note that room numbers must be in sequential order or otherwise by using the room names or the room's code. The Array: before delving into the actual algorithm for the array, some terms ought to be defined as follow:

a $[p_c r_c d_c]G_1 \ldots G_n$ is an array of memory locations for group one where the intersection of these parameters (period, room & day) corresponds to each lecture time for any course; where P_c is a set of lecture period P, $P_c = \{P_i \ldots P_n\}$ at a given day per room such that $P_i \geq 1$ where P_i is an index for the first period and $i \geq 1 \ldots n$, P_n the last period in the set.

R_c is a set of room r, such that $R_c = \{r_i \ldots r_n\}$ having the same period at a given day where r_i is an index number for the first room, r_n the last room in the set.

D_c is a set of days d, such that $D_c = \{d_i \ldots d_n\}$ for a given period per room in a group where d_i is the first day of a given room and d_n the last day such that the cardinality of D_c, $|\ d_i \ldots d_n\ | \leq 6$ and $i \geq 1$(for 6 is the maximum number of required days per week)

r_x= last room of $G_1 = G_2 = \ldots = G_n$

d_x= last day of $G_1 = G_2 = \ldots = G_n$ of r_x

p_x = stopping function

P_{x1}= first stopping function of G_1 = last periods of G_1 in $r_x = P_c r_x d_1$

P_{x2} = second stopping function of G_1 = last periods of G_1 in $d_x = P_c d_x r_1 . P_x(G_1) = P_x(G_2) = \ldots = P_x(G_n)$ in r_x and d_x

r_L is the last room of any period in d_n

d_L is the last day of any period in r_n

$P_c d_x r_1 = P_c d_n r_1$, $P_c r_x d_1 = P_c r_n d_1$

R_a is the room number after the last room number of the preceding group.

R_L is the last room number of the entire group.

The table 2 depicts the structural model of the dynamic array.

4.7. The Proposed Algorithm

1. Function allocateCourseToTimeslot()
2. do {
3. Process current fac
4. Get the next fac
5. Process the next fac
6. while there is fac to process
7 do{
8. Process current dept
9. Get the next dept
10. Process the next dept
11. while there is dept to process
12. do {
13. Process current lev
14. Get the next lev
15. Process the next lev

```
16.    While there is lev to process
17.    FunctionscheduleLevelCourse()
18.          FunctionscheduleDepartmentalCourse()
19.    while there is DeptCoursedo {
20.         Process current DeptCourse
21.          Get the next DeptCourse&& process
22.   if cell is empty && the NumberOfStud                    enrolled <= roomCapt
23.         RoomType is normal class
24.              Timeslot for DeptCourse do not overlapped // clashed then {
25.   IndexRefernceValue←  Deptcourse
26.                                          }
27.  else {
28.       Search for next fitted timeslot
29.                                }
30.                              }
31.                        end if
32.                    end while
33.               end function
34.  FunctionschedulePracticalCourse()
35.    While there is PractCoursedo {
36.       Process current PractCourse
37.       Get the next PractCourse&& process
38.       if cell is empty && the NumberOfStud enrolled <= roomCapt
39.         RoomType is Practical class
40.      Timeslot for PractCourse do not overlapped // clashed then {
41.        IndexRefernceValue←  Practcourse
42.                                    }
43.         else {
44.                 Search for next fitted timeslot
45.                                    }
46.                                  }
47.                        end if
48.                    end while
49.               end function
50.  FunctionscheduleInterdepartmentalCourse()
51.     while there is InterDeptCoursedo {
52.       Process current InterDeptCourse
53.         Get the next InterDeptCourse&& process
54.       if cell is empty && the NumberOfStud enrolled <= roomCapt
55.         RoomType is normal class
56.       Timeslot for DeptCourse do not overlapped // clashed then {
57.        IndexRefernceValue ← InterDeptcourse
58.       }
59.     else {
60.          Search for next fitted timeslot
61.          }
 62.                                  }
63.      end if
64.   end while
65.  end function
66. FunctionscheduleGSTCourse()
```

```
67. while there is GSTCoursedo {
68.    Process current GSTCourse
69.     Get the next GSTCourse&& process
70.        if cell is empty && the NumberOfStud enrolled <= HighestRoomCapt
71.        RoomType is normal class
72.        Timeslot for GSTCourse do not overlapped // clashed then {
73.           IndexRefernceValue   ← GSTcourse
74.                          }
75.      else {
76.            Search for next fitted timeslot
77.           }
78.                          }
79.     end if
80.  end while
81. end function
82.                 }
83.           }
84.      }
```

In the algorithm above, the allocation of courses to each timeslots is considered first at the faculty level procedurally, until all the faculties are scheduled. The department under each faculty must also be in series, the same thing with levels and courses in every department. The program will search repeatedly for all course type in each level before moving to another level of the same department, until all levels are exhausted. Likewise, until all departments are exhausted before allocating resources in appropriate timeslot for another faculty. But there is an exception in the case of interdepartmental course and GST course where set theory must be applied in the program. This mean that the program will check for common timeslot for GST and interdepartmental courses for the department enrolled for. The system will skip iteration of such courses (GST and InterDeptCourse) provided they were allocated in some previous department such that clashing and overlapping are all exempted

5. RESULT AND DISCUSSION

The main menu of the system is being displayed on the Dashboard page and up to five sub-menus, depending on the permissions of the user who logs in. These sub-menus are the Add Faculty, Department and Courses, Generate Time Table, Print Time Table, Add System user and Exit. The menus shown in figure 3 are for an administrator user, who has access to all sub-menu options.

5.1. Subsystem Implementation

The subsystems or modules which represent separate units of functionality of the system are the Add Faculty, department and courses, Generate Time Table, Print Time Table Add System user and Exit.

5.2. Add Faculty, Department and Courses

The Add Faculty, Department and Courses page (Fig 4) allows the admin of the system to add new faculties, departments and even courses to the system. This page makes the system more flexible in case of addition of some of these parameters.

5.3. Generate Time Table

This is the main timetable page (Fig 5) that generates the timetable for each department. The timetable is been generated based on the number of courses register previously in a particular department.

5.4. Add System User

This page (Fig 6) enables the system administrator to add more system users that can also have access to the system.

5.5. Database Implementation

This (Fig 7) shows the database of the system with all the tables and their relationships.

Table 2: Model of Multidimensional Array.

Time/Day	Mon	Tue	Wed	Thur	Fri		Mon	Tue	Wed	Thur	Fri	Sat	Time/Day
Am/pm	R1	R1	R1	R1	R1		R1	R1	R1	R1	R1	R1	Am/pm
8 - 10	1	71	89	103	114		1	70	95	116	130	141	8 – 9
10 – 1	6	74	92	105	116		6	75	99	118	132	143	9 - 11
1 – 3	11	78	95	107	118		11	80	104	121	134	145	11 - 1
3 - 5	16	82	98	109	120		16	85	108	124	136	147	1 – 3
Am/pm	R2	R2	R2	R2	R2		21	90	112	127	138	149	3 - 5
7 - 8	26	2	72	90	104		R2	R2	R2	R2	R2	R1	Am/pm
8 – 11	30	7	75	93	106		26	2	71	96	117	131	8 - 10
11 – 2	34	12	79	96	108		30	7	76	100	119	133	10 - 12
2 - 4	38	17	83	99	110		34	12	81	105	122	135	12 - 2
4 - 6	42	21	86	102	112		38	17	86	109	125	137	2 - 4
8 - 10	R3	R3	R3	R3	R3		42	22	91	113	128	139	4 - 6
10 - 12	46	27	3	73	91		R3	R3	R3	R3	R3	R3	Am/pm
12 – 3	49	31	8	76	94		46	27	3	72	97	117	7 - 8
3 - 5	52	35	13	80	97		49	31	8	77	102	120	8 – 10
5 - 6	55	39	18	84	100		52	35	13	82	106	123	10 – 1
Am/pm	R4	R4	R4	R4	R4		55	39	18	87	110	126	1 - 3
7 - 8	61	47	28	4	74		58	43	23	92	114	129	3 - 5
8 - 11	*63*	*50*	*32*	*9*	*77*		*R4*	*R4*	*R4*	*R4*	*R4*	*R4*	*Am/pm*
11 - 1	65	53	36	14	81		61	47	28	4	73	98	8 - 10
1 – 3	67	56	40	19	85		63	50	32	9	78	103	10 - 1
3 – 6	69	59	44	23	88		65	53	36	14	83	107	1 - 3
Am/pm	R5	R5	R5	R5	R5		67	56	40	19	88	111	3 - 5
8 – 10	113	62	48	29	5		69	59	44	24	93	115	5 - 6
1 - 12	115	64	51	33	10		R5	R5	R5	R5	R5	R5	Am/pm
12 - 2	117	66	54	37	15		140	62	48	29	5	74	7 - 9
2 - 4	119	68	57	41	20		142	64	51	33	10	79	9 - 12
4 - 6	121	70	60	45	24		144	66	54	37	15	84	12 – 2
Am/pm	Rn	Rn	Rn	Rn	Rn		146	67	57	41	20	89	2 - 4
							148	68	60	45	25	94	4 - 6
							Rn	Rn	Rn	Rn	Rn	Rn	Am/pm

Fig. 3: The Main Menu

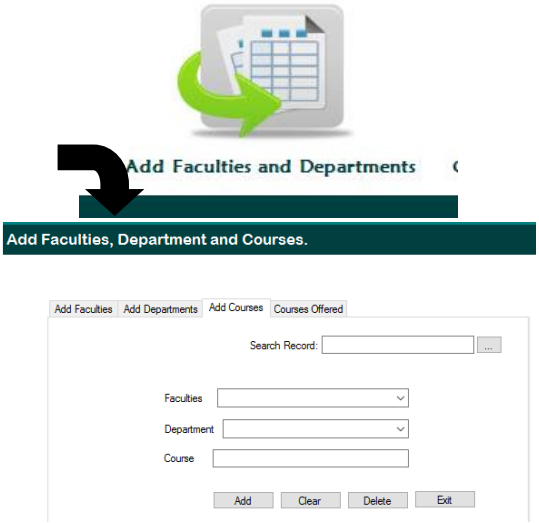

Fig. 4: Add Faculty Module

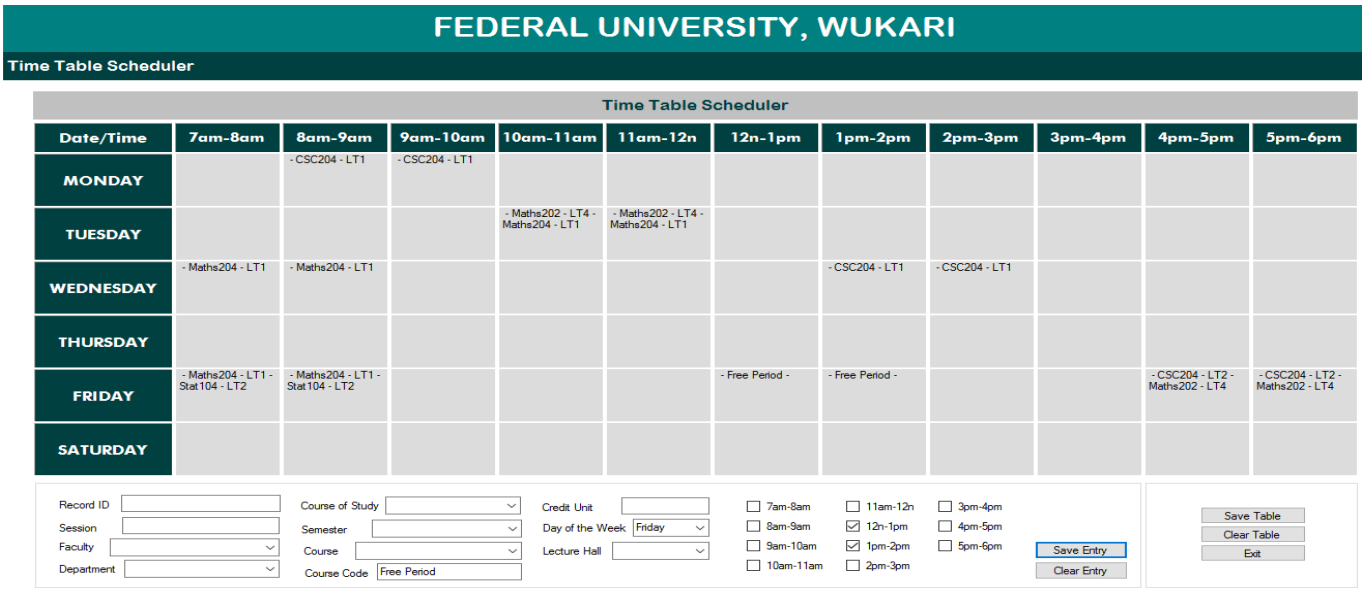

Fig. 5: Generate Timetable

Fig. 6: Add User

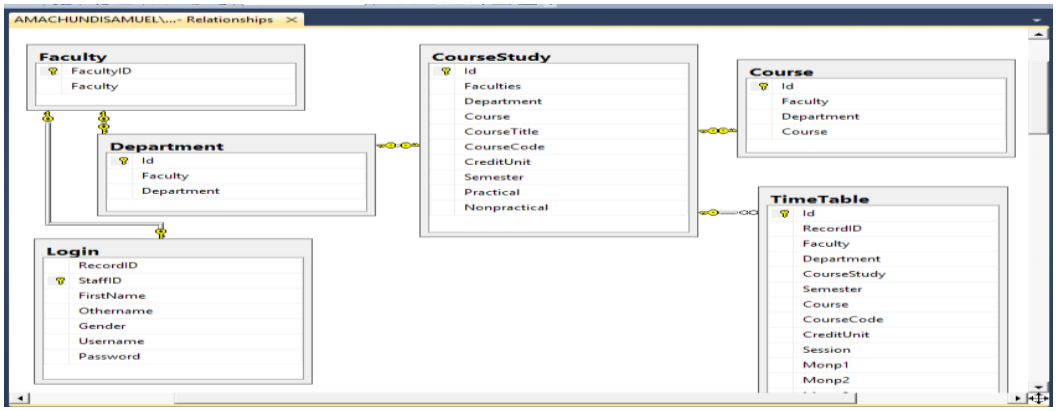

Fig. 7: Entity Relational Diagram

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

The teaching staff usually spend a lot of time in timetable generation and timetable management. The aim of this work is to enable the process of timetable generation to be done faster and more efficient using the computerized method. The Timetable Scheduler as an application for generating lecture timetables has been effectively and successfully deployed. The efficiency of this easy to use software is shown to generate a zero clash timetable in only eight iterations. The system is more flexible to work perfectly in other institutions that have similar constraints with FUW. Also, other institutions that have different constrains can define them before generating the timetable, for example, available lecture rooms space, define free period and so on. The data used in generating the timetable such as courses can also be used for other purposes such as managing students in their respective course registration processes.

6.2. Future work

The automated timetable scheduler is a flexible driven desktop application that enhances the generation and management of the time table. On the area of further works, the researcher suggests the implementation of the online or web based application that is accessible to all students and lecturers within the institution and the world at large.

7. REFERENCES

[1]. L. Samuel, A. Arnold, & M. Milyandreana, "Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study," Universitas Pelita Harapan Timetable. *Universitas Pelita Harapan Computer Science Journal* , pp. 87-93, 2012.

[2]. E. Burke, K. Jackson, J. H. Kingston, & R. Weare, "Automated University Timetabling: The State of the Art," *The computer Journal*, *40*(9), pp. 565-571, 1997.

[3]. Y. Awad, A. Dawood, & A. Badr, "An Evolutionary Immune Approach for University Course Timetabling," *IJCSNS International Journal of Computer Science and Network Security*, *11*, pp. 127-135, 2011.

[4]. C. Renman, & H. Fristedt, "A comparative analysis of a Tabu Search and a Genetic Algorithm for solving a University Course Timetabling Problem," KTH, School of

computer, Degree Project, in Computer Science, First Level Stockholm, Sweden 2015

[5]. S. Kirkpatrick, C. D. Gelatt, & M. P. Vecchi, "*Optimization by Simulated Annealing," Science*, *220* (4598), pp. 671-680, 1983.

[6]. R. M. Chen, & H. F. Shih, "Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization With Local Search," *Algorithms*, *6*(2), pp. 227-244, 2013.

[7]. H. V. Yamazaki, & J. Pertoft. "Scalability of a Genetic Algorithm that Solves a University Course Scheduling Problem Inspired by KTH," 2014.

[8]. R. Fredrikson, & D. Jonas, "A Comparative Study between a Simulated Annealing and a Genetic Algorithm for Solving a University Timetabling Problem," *KTH Royal Institute of Technology,* Degree Project in *School of Computer Science and Communication,* 2016.

[9]. Z. Lü, & J. K. Hao. "Solving the Course Timetabling Problem with a Hybrid Heuristic Algorithm," In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pp. 262-273, Springer, Berlin, Heidelberg, 2008.

[10]. Y. Yang, and S, Petrovic. "A Novel Similarity Measure for Heuristic Selection in Examination Timetabling," *In International Conference on the Practice and Theory of Automated Timetabling*, pp. 247-269, Springer, Berlin, Heidelberg, 2004.

[11]. N. A. Zahra, "Comparison of Metaheuristic Algorithms for Examination Timetabling Problem," *J. Appl. Math. & Computing , 16*, pp. 337 – 354, 2004.

[12]. L. Rhydian, "A Survey of Metaheuristics-Based Technique for University Timetabling Problem," *Wales* , pp. 1-26, 2007.

[13]. L. B. Asaju, T. K. Ahamad, A. A.-B. Mohammed, & A. A. Mohammed, "Artificial Bee Colony Algorithm for Solving Educational Timetabling Problems," *International Journal of Natural Computing Research , 3* (2), pp. 1-21, 2012.

[14]. H. C. Kampman, "Timetabling at Utrecht University," Master's thesis, 2013.

[15]. M. A. Awadallah, A. T. Khaer, M. A. Al-Betar & A.L.A. Bolaji, "Harmony Search with Greedy Shuffle for Nurse Rostering," *International Journal of Natural Computing Research (IJNCR)*, 3(2), pp. 22-42, 2012.

[16]. E. H. Mohamed, "Solving the Course-Timetabling Problem of Cairo University Using Max-SAT." *arXiv preprint arXiv:1803.05027*, 2018.

[17]. H. Babaei, K. Jaber & H. Amin, "A Survey of Approaches for University Course Timetabling Problem." *Computers & Industrial Engineering* 86, pp. 43-59, 2015.

[18]. R. Bellio, C. Sara, D. G. Luca, S. Andrea, & U. Tommaso, "Feature-Based Tuning of Simulated Annealing Applied to the Curriculum-Based Course Timetabling Problem," *Computers & Operations Research,* 65, pp. 83-92, 2016.

[19]. D. De Werra, A. S. Asratian, & S. Durand, "Complexity of Some Special Types of Timetabling Problems," *Journal of Scheduling*, *5*(2), pp. 171-183, 2002.

[20]. G. Marc, & L. Christian, "An Improved Algorithm for the Periodic Timetabling Problem," *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, 12; pp. 1–14, Germany, 2017

[21]. G. Post, D. G. Luca, H. K. Jeffrey, M. Barry, & S. Andrea, "The Third International Timetabling Competition," *Annals of Operations Research* 239, (1), pp. 69-75, 2016.

[22]. B. Naderi, "Modeling and Scheduling University Course Timetabling Problems," *International Journal of Research in Industrial, 5* (1-4), pp. 1-15, 2016.

[23]. T. Robenek, M. Yousef, S. A. Shadi, C. Jianghang, & B. Michel, "Passenger Centric Train Timetabling Problem." *Transportation Research Part B: Methodological* 89, pp. 107-126, 2016.

[24]. G. Woumans, D. B. Liesje, B. Jeroen, & C. Stefan, "A Column Generation Approach for Solving the Examination-Timetabling Problem," *European Journal of Operational Research* 253 (1), pp. 178-194, 2016.