

Optimisation of University Examination Timetable Using Hybridised Genetic and Greedy Algorithms: A Case Study of Computer Science Department, University of Ibadan

Sunday J. Agbolade^{a*}, Babatunde I. Ayinla^b, Latifat A. Odeniyi^c, Akinola S. O.^d

^a*Redeemer's University, Ede, Osun-state, Nigeria*

^{b,d}*University of Ibadan, Ibadan, Nigeria*

^c*Koladaisi University, Ibadan, Nigeria*

^a*Email: sjagbolade@gmail.com, ^bEmail: ayinlab@gmail.com*

^c*Email: odeniyilateefah95@gmail.com, ^dEmail: solom202@yahoo.co.uk*

Abstract

Timetable scheduling is an important aspect of decision-making in any organisation, particularly in academia. An examination timetable is expected to coordinate students, invigilators, courses, examination hall allocation, and time slots. However, *the* problem could be viewed as a Nondeterministic Polynomial (NP); NP-hard problem, scheduling problem has plagued humanity since its inception. Due to the complex structure of the problem in terms of hard and soft-constraints, most organisations schedule time inefficiently using manual approach. This study introduced an algorithms hybridisation method of genetic and greedy algorithms to automate the timetable scheduling process efficiently. A genetic algorithm is a heuristic search technique based on Charles Darwin's theory of natural evolution. The fitness of each course, venue, and faculty content is determined by the probabilistic optimisation which is the solution candidate in the initial population of all the objects. Subsequently, the greedy algorithm's activities selector selects the best solution. The output demonstrates that the method effectively handled all the constraints associated with timetable scheduling. Hybridising the two algorithms to build a scheduling system, such as the examination timetable. Therefore, it is a viable option to combine genetic and greedy algorithms to have an optimised examination timetable that is flexible to any situation.

Keywords: Timetable; Examination; Greedy Algorithm; Genetic Algorithm and Scheduling.

Received: 4/10/2024

Accepted: 6/2/2024

Published: 6/15/2024

* Corresponding author.

1. Introduction

The Genetic Algorithms (GAs) have been in use in various scheduling activities such as patients, doctors, and medicinal gadgets, creating each object as an abstract resource and capturing domain knowledge as specific rules [21]. Exam scheduling is a well-known and well researched topic that can be solved using genetic algorithms, as demonstrated by [20,3].

Time table scheduling is very important to enable decision-makers to make right decisions in every organised organisation. Time-table scheduling is an imperative activity when it comes to academic areas. Therefore, school calendars must be properly scheduled. There are problems facing academic time-table scheduling, due to manual construction; example of such problem is inaccurate due to some constraints. This area of research has attracted many researchers, and many researchers have worked on this area [1].

As a result, because the scheduling problem is known to be a Non-Polynomial complete problem [9], employing heuristic techniques and a greedy approach can efficiently and reliably solve the problem. Heuristics are classified into three types: iterative improvement algorithms, probabilistic optimisation algorithms, and constructive heuristics. In the probabilistic optimisation group, Genetic Algorithms-based methods are significant, and they have been thoroughly proposed in this research effort alongside Greedy Algorithm for selection based on situation requirement.

Reference [5] define a timetable as a mechanism for scheduling available time and resources for any operation or activity. Nonetheless, this system faces numerous obstacles as a result of the desire to create one that is both successful and efficient. The amount of research activity in the domain of timetable development has significantly increased. The emergence of a series of articles and international conferences on the theory and application of automated time-table systems demonstrates this. There are numerous limits that must be addressed when developing a University Examination timeline. Practically, there are two major constraints; Soft constraints and Hard constraints.

2. Literature review

2.1 Time-table problems

There are challenges during the creation of the examination timetable such as the allocation of venues to the sizable number of students; there is a need to judiciously use all the available resources during the examination period which means that all the available venues must be known and the size of the students that such venue can accommodate. Number of courses that would be taken during the examination periods and the number of hours that each course will be occupied during the examination period. There will be a consideration of the number of days the examination will occupy. The major constraint that this research work is tackling is the area of preference or priority for some courses in examination periods due to some peculiar demands [10].

There are limited available spaces despite the increase in students in our schools, which has led to numerous challenges in allocating resources throughout the examination timetable creation process. Among these issues

are the waste of space resources during the examination period, invigilators' unawareness of invigilation allocation, poor resource management, and a poor communication system between the scheduling committee and the invigilators. Insecurity of files as a result of manual timetable development, causing difficulty in referring to it, and clashes in examination schedules for different examination courses. Another critical area of the problem is when one or two courses are being taught by the head of the department and in a situation where those courses require preference because the lecturer in charge has some other administrative duties to attend to and those courses can start earlier.

It is worth examining the effect of the hybridisation of the genetic algorithm and greedy programming since both can be used for optimisation problems, especially in time-table scheduling rather than using only the genetic algorithm or the greedy algorithm. In these days of fast computing and high technology information science, this problem of examination time-table can best be handled using a computer [19,2].

2.2 Genetic algorithm (ga)

According to Lukas and his colleagues [17], the Genetic Algorithm (GA) effectively addressed assignment difficulties. GA has frequently been used to create university exam schedules [3]. According to Ghasemi, Handley and Howarth [13] heuristic search can be used to tackle scheduling challenges in any organised institution, such as a university. The job of the genetic algorithm is to decide the sequence of all courses to be arranged in one group, whereas the job of the heuristic search is to discover the time slots used to schedule the courses [22,17].

Genetic algorithms are a black-box optimisation method inspired by biological evolution concepts. While they serve as a general-purpose optimisation tool, their specific instantiations can be heuristic and driven by a small quantity of biological intuition [13]. They routinely reveal globally optimal answers even in the most complex search fields. They work with a population of coded solutions chosen for their quality and then used as the foundation for a new generation of solutions discovered by combining (crossover) or changing (mutating) existing individuals. The search process has traditionally been domain-independent, meaning that the crossover and mutation operators do not know a suitable solution [3].

The genetic algorithm has proven to be a tremendously effective and successful problem-solving approach, displaying the strength of evolutionary principles. In a variety of domains, genetic algorithms have been employed to create solutions to challenges that are as complicated as or more complex than those encountered by human designers. Furthermore, the solutions they develop are frequently more efficient, elegant, or complicated than anything comparable produced by a human engineer. In certain circumstances, genetic algorithms have produced results that have perplexed the developers who developed the algorithms at first instance [23].

2.3 Related works

A graph-coloring-based algorithm for the exam scheduling application was introduced by [18] with the aim of achieving fairness, accuracy, and optimal exam time period. Some few assumptions were considered in terms

of constraints that were related to the exam scheduling problem generally in universities. An autonomous exam-scheduling algorithm was built to ensure fairness while shortening the exam period. The experiment yielded a graph-coloring-based test scheduling algorithm that meets the objectives of fairness, accuracy, and optimal exam time period. Numerous studies have been conducted to address the issue of exam scheduling [3,4,14]. In nutshell, the algorithm consists of two major steps. The first step builds the weight matrix and graph. The second step assigns colors to the nodes of the graph. The method first represents the courses by graph nodes, where 2 nodes are adjacent if at least one student registers the 2 corresponding courses. Then, it is required to assign each course represented by a node a time slot, such that no two adjacent nodes have the same slot, in condition that a set of constraints imposed on the problem are also met. This problem was solved by using node graph coloring technique. The set of assumptions and restrictions taken into account is the main distinction between diverse investigations. For instance, Burke, Elliman, and Weare [3] used graph coloring to adopt a similar strategy. However, they did not apply any restrictions to their method; they just dealt with disputes.

In the same vein, East, [10] worked on genetic algorithm machine learning technique to create course timetables for a university. The algorithm improved timetables with user preferences within the limits of a schedule. The timetables generated devoid of any conflicting scheduled modules for lecturers, students, or venues. The algorithm is given over a web API and executed effectively on an enterprise Java application server running in the cloud. A database housed all the entities that need to be planned into a timetable. The algorithm, a user interface running as a contemporary web app, and the database service with its own API were all put into a service-oriented architecture. The use of genetic algorithms within the framework of the issue domain is used, along with the specifics of this application's development.

By combining a binary crossover at a high crossover rate with a roulette-wheel selection modified to always select several of the top, most-fit candidate solutions per generation, a published example of using a genetic algorithm to schedule truck loading dock usage provided concrete evidence of high convergence speed [8]. According to the objectives of this research and using the Java programming language, Reference [15] provided an example of using a genetic algorithm to schedule college courses with instructors into lecture halls.

According to [1] in the article, A Novel Educational Timetabling Solution through Recursive Genetic Algorithms. The research sought solutions that satisfied the hard constraints while minimising soft constraint breaches. Furthermore, colleges frequently differ in terms of limits and the number of professors, courses, and resources involved, increasing the challenge's scale and complexity. We offer a simple, scalable, and customisable way to solving Timetabling Problems for numerous courses by recursively using Genetic Algorithms, which are efficient search methods used to produce an optimal or near ideal timetable.

In this work, the limitations of educational timetable scheduling were included in the fitness function and can be simply added or removed. The Federal Institute of Cear's environment served as the basis for the definition of the soft and hard restrictions taken into consideration in the work. The soft restrictions are (i) neighboring lectures (more than three classes in a row) and (ii) agent unavailability (agents unavailable owing to other activity). The hard constraints include (i) agent matches when agents are allocated to contemporaneous events and (ii) overlapping lectures from the same course semester, which reduces overlapping lectures of courses with

common students. The methodology addresses course timetabling challenges using Genetic Algorithms recursive executions, yielding a global timetable.

3. Methodology of the system

3.1 Modeling the examination timetable system

Modeling is creating specifications from which a new system will emerge [12]. Its goal is to create specifications allowing the new system to be completed and implemented correctly. Modeling must also be sensitive to changes. These aims and objectives may occur over time in response to adjustments in the demanding environment.

The timetable structure that will be formed is made up of input data, the relationship between the input data, and the application of the genetic algorithm.

The input data contains:

- 1) *Supervisor*: Defines the names and identifying numbers of the lecturers.
- 2) *Course-code*: It depicts the names of current term courses.
- 3) *Venue*: The data describes the room number and capacity of the exam room.
- 4) *Tim-slot*: It specifies the start time as well as the length of a lecture.
- 5) *Day*: It specifies the specific day of the week on which the exam will be administered.

These are the content of chromosome which are genes.

The heuristic search is used in this study, a two-dimensional matrix known as the times matrix is used. This matrix identifies appropriate time slots for scheduling courses [22]. This times matrix employs five sets, as previously stated. They are $C=Cc1, Cc2, \dots, Ccn$, type Venue set $V=v1, v2, \dots, vn$, supervisor code set $S=S1, S2, \dots, Sn$, day set $D=d1, d2, \dots, dn$, and time-slot set $TS=ts1, ts2, \dots, tn$. All index sets begin with one. The Course-code set, Supervisor set, and Time-Slot set all have close relationships. Assume that one of the relationships is $Cc1, S1, TS1$, suggesting that Course-code $Cc1$ is to be completed in $TS1$ and supervised by $S1$.

Furthermore, $D1, V1$ is a relation, which signifies that on day $D1$ on $V1$ is assigned. If $Cc1, S1, TS1$ is associated with $D1, V1$, it signifies that the Course-code $Cc1$ exam will be held in Time-slot $TS1$ and overseen by supervisor $S1$ on Day $D1$ at Venue $V1$. All course code, time slot, and supervisor code pairs must be linked to all day and venue pairings. This connection is recorded in a matrix known as the times matrix. The number of rows required is calculated by multiplying the number of venues by the number of days in a week by the number of time slots in a day. There are two venues, $V1$ and $V2$, that are scheduled on the same day but have separate time slots, $D1, TS1$ and $D1, TS2$ for the course-code $Cc1$ supervised by $S1$ and $Cc2$ supervised by $S2$. Each cell

in the times matrix is subjected to five functions. These functions are fcc, fts, fs, fd, and fv, and they are used to obtain information on the course code, time-slot, supervisor, venue, and day, in that order. fCc(v11)=Cc1, fTS(v11)=TS, fS(v11)=S1, fV(v11)= V1, fD(v11)= D1. All of these functions are combined to form a times matrix.

Assume that the sequence of courses represented by created chromosome [1, 2] reflects Cc1, TS1, S1 and Cc2, TS1, S2; the first course to be scheduled is gene chromosome code Cc1, TS1, S1, and the last is Cc4, TS1, S4. The outcome is shown in Table 3.1. Cc1, TS1, S1 are successfully scheduled on (V1, D1, TS1), (V1, D1,TS2), (V2, D1,TS1), and (V2, D1,TS2). Table 3.1 shows that chromosome [1, 2] can allocate four courses to two venues, V1 and V2, for two time slots, TS1 and TS2 in V1 and TS1 and TS2 in V2.

The value of 4 over 4, which is 1, is referred to as the chromosome's fitness. The fitness is 1 if the chromosome is [1, 2]. The fitness of a chromosome represents the goal of the time table. In this scenario in Fig. 1, the goal is to maximise the number of courses that can be planned for each timetable. If s(k) is the number of courses scheduled for the kth column, p is the maximum column in a times matrix, u(j) is the unit number of jth students to be scheduled for each venue name, and t is the total number of students for each venue name, then fitness of a chromosome is defined as 1.

$$Fitness = \frac{\sum_{K=1}^p s(k)}{t \cdot n(c) \cdot \sum_{j=i} u(j)}$$

Figure 1: Mathematical Expressions for Fitness

There are numerous timeframes for university examinations. Because each department's degree is structured for four to five years of study, each department has at least four-time tables for each semester. Each batch has its own timetable. There is only one-time table in one department, no matter how large the batch is. It has no effect on the processing time. The longer the processing time to generate the time table, the larger the batch. This is due to the fact that the target matrix will have a greater number of rows as well as columns.

3.2 Pseudocode of genetic algorithm

The algorithm below demonstrates the simplicity of genetic algorithm to generates the fit population for the examination timetable schedule.

t = 0
Initialize P (t)

```
Evaluate P (t)
while Termination condition is not satisfied do
  Selection P(t + 1) from P(t) Perform Crossover on P(t + 1) Perform Mutation on
  P(t + 1) Evaluate P(t + 1)
  Replace P(t) with P(t + 1)
  t = t + 1
end while
```

Where P is the population path to each resource (Staff, Students), and t is the resource (Day, Time, Venues).

After scheduling a resource, the population path to that resource is erased to prevent it from being rescheduled using the operators' Selection, Crossover, Mutation, and Evaluation.

Initial population: The formation of an initial population is the first step in the operation of a GA. Each individual in this group encodes a potential solution to a problem. Following the generation of the initial population, each individual is examined and awarded a fitness value based on the fitness function. It has been established that if the initial population of the GA is good, the algorithm will have a better chance of finding a decent solution.

1) Selection: This operator chooses which chromosomes in the population will reproduce. The better the fit, the more likely it is to be picked for reproduction.

2) Crossover: The crossover is a genetic operator used in genetic algorithms to modify the programming of a chromosome or chromosomes from generation to generation. It is similar to reproduction and biological crossover, which are the pillars of genetic algorithms. The practice of integrating numerous parent solutions to form a kid solution is known as crossover. There are methods for choosing chromosomes. This operator selects a locus randomly and swaps the subsequence between two chromosomes before and after that locus to produce two progenies. For example, the strings 11000100 and 11111100 might be crossed after the third locus in each to generate 11011100 and 11100100. The crossover operator is designed to approximate biological recombination between two chromosomes.

3) Mutation: Mutation is a genetic operator that is employed to maintain genetic diversity in a population of genetic algorithm chromosomes from generation to generation. It is analogous to biological mutation. A mutation alters the values of one or more genes on a chromosome from their original state. The solution in mutation may be drastically different from the previous solution. As a result, by using mutation, GA can achieve a better solution. This operator flips some of the bits of a chromosome at random. For instance, the string 00000100 could be modified in its second position to produce 01000100. Mutation can occur with a tiny chance at each bit location in a string.

4) Evaluation: An evaluation is a test that is performed on the operation's outcome in terms of producing the best chromosomes. It is used to assess the fitness of such chromosomes.

3.3 Pseudocode of greedy algorithm

There is a list of activities with starting and ending timings for this algorithm. However, the goal is to choose the greatest number of non-conflicting tasks that a person or machine can execute at the same time, given that the person or machine can only work on one activity at a time. If the starting time of one activity is higher than or equal to the finishing time of the other, the two activities are said to be non-conflicting as illustrated in Fig. 2.

To address this issue, the activities were arranged in ascending order based on their completion time.

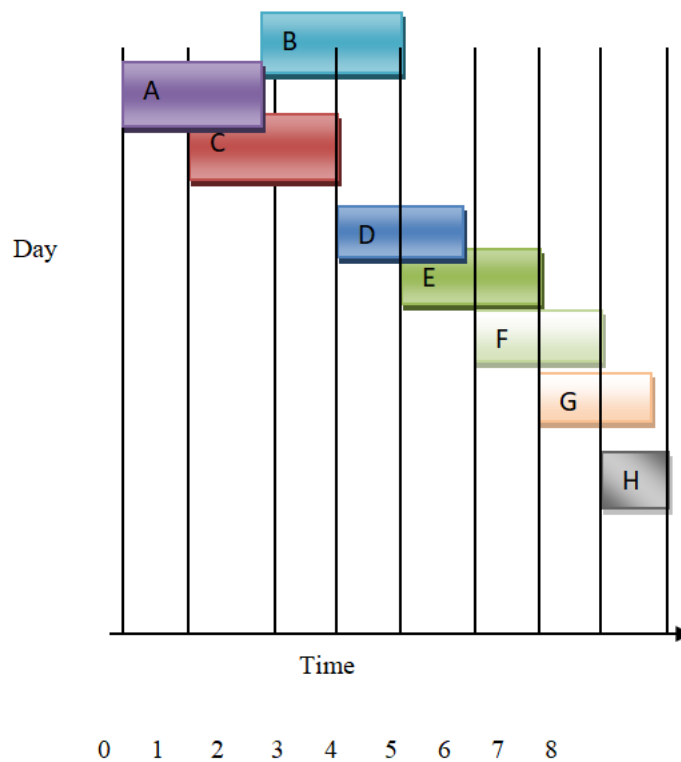


Figure 2: Shows un-schedule Activities in terms of Day and Time.

The above Fig 3.3 shows the given activities regarding the unscheduled time and days. A, B, C, D, E, F, G, and H are the given activities which represent the courses in the examination which need to be scheduled in a day with time. Assuming each activity has three hours before expiration according to the above diagram. We look for a way to schedule these given activities so that each activity will not affect the other within a given venue. Below are un-scheduled activities with time; assuming there are eight activities like in the table below to schedule for a day and in the computer science department in the University of Ibadan, there are only three venues for the examination such as Koladasi Lecture Theatre (KDLT), Computer science Lecture Theatre (CLT) and Post Graduate (PGC) classroom which were tagged with venue 1, venue 2 and venue 3 respectively. Moreover, all these venues have different capacities in terms of fitness. Meanwhile, the genetic algorithm has

already determined fitness in terms of population to a venue capacity.

The steps below illustrate how activities are scheduled with respect to time.

PERFORM STEP 3 UNTIL all activities are checked

Steps in activities selector algorithm

Step 1: arrange the activities in the order of their finishing time.

Step 2: choose the first activity

Step 3: choose the next activity whose start time is greater than or equal to the prior activity's finish time.

Pseudo Code

```
Let i = 0;      //pointing at first element
  for k = 1 to n-1 do //n denotes the length of activities
    if start time of k >= finish time of i then
      Print k
  Let i = k
  endif
endfor
```

This pseudocode was used in order to get the ordered date of the examination, Time-slot (start-time and finished-time), where n is a variable that holds the total number of activities to be scheduled. i is the first activity with the lowest finishing time for the schedule; if the starting time of activity k is greater than or equal to the finishing time of activity i, then activity i will be executed. This action will be repeated for all the activities until all the finishing times of the activities are in ascending order.

It is assumed that the number of students that offer courses A, D and G are vast in population. Therefore, these three courses can be scheduled for the same venue with a large capacity supporting such a population.

Also, activities (courses) like C, E and H are within the range of a population that can be fixed into another venue called Venue 2 or CLT. So, for the remaining activities, B and F could be scheduled into another Venue 3 since the population can fit into the capacity of the venue. All these activities are scheduled for a single day in all three available venues.

Activity selector algorithms are applied for each venue to determine the order in which activities will occur. At this point, the preferred activities could be determined before another by giving it a time slot that matches our desire. Then, from these, it shows how flexible our timetable can be in order to meet the purpose or desired goals.

At the Computer Science Department, University of Ibadan, there are four levels, from 100 to 400 levels, from year 1 to year 4. In this research work, one semester of data was used. Certain numbers of courses, venues, supervisors, days and time-slots to be scheduled for every level are inputted. For example, in the 100 level, which is year one, is assumed they have 7 courses; in 200 level, there are 10 courses, also in the 300 level, there are 12 courses; and in 400 level, there are 10, all the available venue are just three and the period for the exams is just two weeks which means there are only 10 days. Each day has two examination time-slots.

The total number of course-codes are: $7+10+12+10=39$

Total number of venues available = 3

Total number of days = 10 days (i.e Monday to Friday of every week)

Total time slot per day = 3 (i.e Morning, Afternoon and Evening)

Total number of supervisors = 20

All these are the initial population for the genetic algorithms.

Table 1: Relationship among the gene to form chromosome for times matrix.

	<i>Cc1, TS1, V1</i>	<i>Cc2, TS2, V2</i>	<i>Cc3, TS3, V3</i>
WK1,D1	1	1	1
WK1,D2	1	1	1
WK1,D3	1	1	1
WK1,D4	1	1	1
WK1,D5	1	1	1
WK2D1	1	1	1
WK2,D2	1	1	1
WK2,D3	1	1	1
WK2,D4	1	1	1
WK2,D5	1	1	1
WK3,D1	1	0	1

Table1, shows that $n(Cc)=39$, $n(S)=64$, $n(V)=3$, $n(D)=11$, and $sn(TS)=3$. In the genetic algorithm, all of these make up the chromosome's genes. The times matrix has two columns and eleven rows. Following receipt of preliminary data, such as the restricted day and time for a certain venue, what venue can be used, and the capacity of that facility.

The highest best fitness value, 1, can be attained if the system runs with 11 chromosomes in a population and 11

generations are set in the experiment without taking into account the chance of selection, crossover, and mutation. This means that all classes can be scheduled as needed. As a result, the number of populations and generations must be limited. It means that creating a schedule will take less time.

4. Results presentation and discussion

4.1 User interface design

This is a graphical user interface. The section of this program allows the user to input data into the system to create an initial population concerning genetic algorithms. This aspect makes this program very friendly and vividly understood. Its major function is to allow users to enter the initial data required to construct the examination timetable.

4.1.1 User authentication

The user authentication page authorises the system's authentic users to access the system.

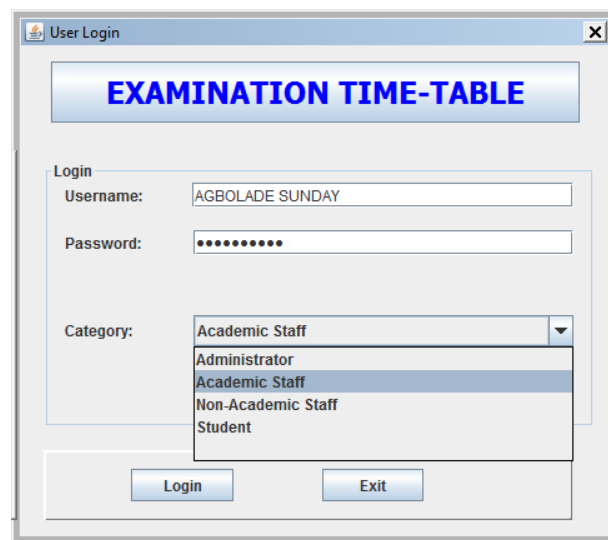


Figure 3: Login portal.

Fig. 3, demonstrates the usage of a username and password to protect the information in the system from unauthorised users. The administrator would create the username and password for the faculty users to access the new examination timetable system.

4.1.2 Course scheduling for the examination timetable

Fig. 4.5 helps the administrator to schedule all registered courses for the session, and this helps to avoid duplication of records in the database, for instance, rescheduling one course twice

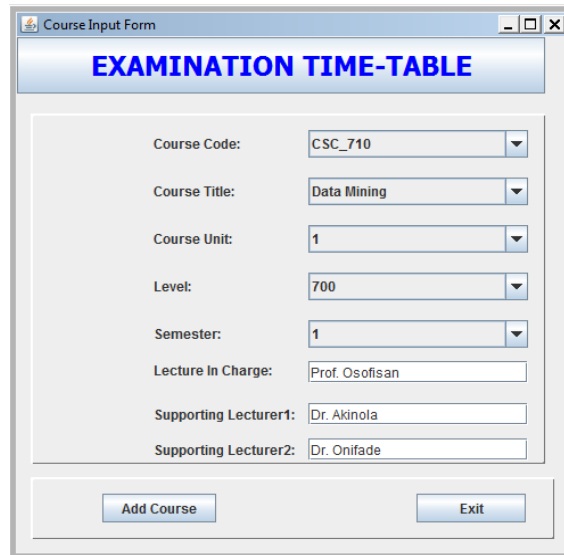


Figure 4: Scheduling courses and supervisors'

Fig. 4, illustrates how supervisors are scheduled for invigilation during the examination period.

4.1.3 Venue scheduling for the examination timetable

Fig. 5, helps the administrator to allocate venues, considering the venue capacity, so as to assign an appropriate course to the venue base on the capacity of the venue.

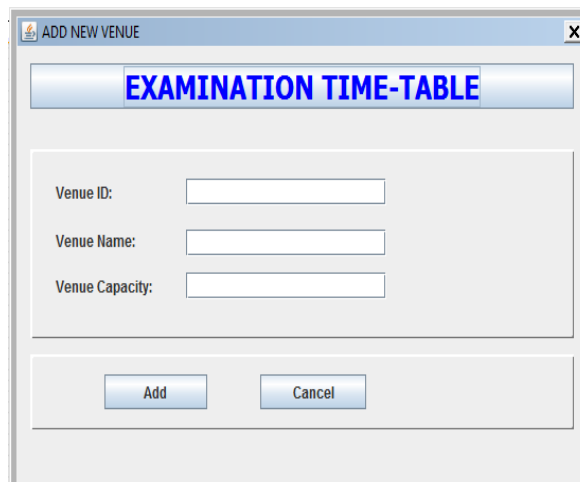


Figure 5: Scheduling venue for examination.

The above activities of the new system show a graphic user interface in which all the necessary data to form chromosomes in genetic algorithms were collected.

4.1.5 Examination timetable results

The output of implementing the hybridised genetic and greedy algorithms for the examination timetable is clearly

illustrated. The programming language used was Java and Microsoft SQL server for implementation. The timetable shows the scheduled time for each course with respect to days and the allocated venue for every course. This result clearly revealed the effectiveness of the greedy algorithm because it selected the courses that belong to the same venue according to the time allocated to every course. The time allocated to each course shows that early time courses are given priority in the timetable construction. Fig. 7 depicts the timetable generated after the resources were scheduled.

COURSE CODE	COURSE TITLE	VENUE NAME	DATE	TIME
CSC_212	Introduction to Com.	CSCLT	2018/01/10	8:00-10:00
CSC_101	Introduction to Com.	KDLT	2018/02/01	8:00-10:00
CSC_411	System Programms	PGLR	2018/02/01	11:00-1:00
CSC_311	Software Engineeri.	CSCLT	2018/02/02	2:00-4:00
CSC_233	Programming and	CSCLT	2018/02/02	11:00-1:00
CSC_433	Microcomputers S	CSCLT	2018/02/03	2:00-4:00
CSC_321	Data Structure	CSCLT	2018/02/03	8:00-10:00
CSC_421	Computer Architect	KDLT	2018/02/03	11:00-1:00
CSC_272	Discrete Structure	KDLT	2018/02/04	8:00-10:00
CSC_332	Computer Hardware	CSCLT	2018/02/04	11:00-1:00
CSC_451	Theory of Computat	KDLT	2018/02/05	8:00-10:00
CSC_291	Elementary Data Pr	PGLR	2018/02/05	2:00-4:00
CSC_333	System Programms	KDLT	2018/02/05	11:00-1:00
CSC_299	Information Manag.	KDLT	2018/02/06	8:00-10:00
CSC_341	Computer Networks	PGCR	2018/02/09	11:00-1:00
CSC_221	Introduction to Com.	PGCL	2018/02/09	8:00-10:00
CSC_234	Assembly Language	PGCR	2018/02/10	8:00-10:00
CSC_301	Software Engineer.	PGLR	2018/02/10	11:00-1:00
CSC_422	Comp. Operating S.	CSCLT	2018/02/11	8:00-10:00
CSC_412	Business Progra.	KDLT	2018/02/11	11:00-1:00
CSC_334	ANALYSIS I	PGCR	2018/02/12	8:00-10:00
CSC_232	Introduction to Oper	PGCR	2018/02/12	11:00-1:00
CSC_312	Software Engineer.	CSCLT	2018/02/23	8:00-10:00

Figure 7: Generated Examination Timetable

4.2 Results discussion

The result of this study reveals that using both the genetic and the greedy algorithms for timetable construction is very efficient in handling all the constraints, aforementioned. This assertion was in line with the East [10] and Alves [1].

The output results show that every examination venue in the department will be used every day. However, each examination venue has three-time slots allocated to it daily. As revealed from the result, only students in the higher level like 200 level, 300 level and 400 level can have carry-over courses from their previous classes. The result of the system's output was able to handle the schedule so that there is no course clash in the timetable.

Again, the result shows that if the 200-level student has a 100-level course carry-over, this system shows that the 200-level student can sit for the carry-over course without interfering with any of the 200-level courses. The result demonstrates that the system schedules the timetable so that any 200-level, 300-level or 400-level student can attend to their carry-over. At the same time, it will not clash with their current class examination. Also, students in the 300-level can have carry-over in the 100-level and 200-level, which means that none of the 100-level, 200-level and their current class examinations should be scheduled for the same time slot on the same day, even in a different examination venue. The same thing will be repeated for the advantage of 400-level students so that they can have the opportunity to attend to their carry-overs.

The graph in Figure 4.1 shows the time taken to complete the fitness test with respect to the number of the initial

population. The graph reinforced the claim of Foy, Benekohal and Goldberg [11] on signal timing determination using genetic algorithms that the higher the number of the initial population for the genetic algorithm to be tested, the higher the time taken to determine the fitness of that population. Malkawi, Hassan and Hassan [18] report also substantiate the findings with respect to the timing of fitness.

Table 2: Average time used to determine the fitness with population size

Serial No	Population sizes	AverageTime(milliseconds)
1	10	1.31
2	15	1.06
3	20	1.903
4	25	2.543
5	30	3.31
6	35	3.385
7	40	3.775

When this result is compared with other researchers' findings, it was discovered that the time taken to construct this examination timetable improved. The involvement of a greedy algorithm in the course schedule contributed to the optimisation. Consequently, the hybridisation enhanced the resolution of the problem identified by Malkawi, Hassan and Hassan [18] regarding future work in improving timetable scheduling processes.

5. Conclusion

This paper presented a hybridised algorithm to tackle timetabling scheduling problems. The model enhanced examination timetable scheduling or any time scheduling activities. The technique optimally reduced the time and space wastage with respect to the student population and venue capacity. The concurrent exam sessions were efficiently handled, and the best fit was selected for every available resource. The model worked based on the time allocated to each course, which looks for the best of the fittest candidate from genetically evaluated results first and schedules it before taking the next fittest for scheduling.

In light of the above, when an examination timetable is to be constructed efficiently, it is better to use both genetic and greedy algorithms. In this case, each of these algorithms performs its duty constructively. The combination of this algorithm is better than using only a genetic algorithm for timetable scheduling. There have been hybridised scheduling systems, in which a modified graph coloring algorithm is used to generate a population of workable but unoptimised timetables, and then a genetic algorithm is used to choose an optimal timetable Burke, Elliman and Weare [3]. This combined algorithm approach is functional, but a pure genetic algorithm approach is nearly as efficient, and a solution can be reached nearly as quickly by starting with randomised, invalid timetables and using the fitness function to select for both more feasible and more optimised individuals.

According to De Jong, 1988 if the required domain information is totally understood and can be used to create a traditional search algorithm, such an approach should be chosen. A classical algorithm can leverage a surplus of information more effectively than a genetic algorithm. A genetic algorithm approach is preferable for organising a timetable compared to the offered alternatives because of the flexibility of the fitness function to distil complicated notions of real-world domain knowledge of the problem down into local comparisons [10].

Reference

- [1]. Alves, S. S., Oliveira, S. A., & Neto, A. R. R. (2015, October). A novel educational timetabling solution through recursive genetic algorithms. In *2015 Latin America Congress on Computational Intelligence (LA-CCI)* (pp. 1-6). IEEE.
- [2]. Bagga, S., Gupta, S., & Sharma, D. K. (2019). Computer-assisted anthropology. In *Internet of Things in Biomedical Engineering* (pp. 21-47). Academic Press.
- [3]. Burke E., Elliman D., and Weare R., "A University Timetabling System Based on Graph Coloring and Constraint Manipulation," *Journal of Research on Computing in Education*, vol. 27, no. 1, pp. 1-18, 1994.
- [4]. Burke E., Elliman D., and Weare R., "Automated Scheduling of University Exams," Department of Computer Science, University of Nottingham, 1993.
- [5]. Burke E.K, Eckersley B.J. McCollum, Petrovic S, and Qiu R., (2004). "Analysing Similarity in Examination Timetabling". *5th International Conference on the Practice and Theory of Automated Timetabling*. Pittsburgh, PA USA.
- [6]. Burke, E. K., & Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE transactions on evolutionary computation*, 3(1), 63-74.
- [7]. Burke, E. K., Newall, J. P., & Weare, R. F. (1996). A memetic algorithm for university exam timetabling. In *Practice and Theory of Automated Timetabling: First International Conference Edinburgh, UK, August 29–September 1, 1995 Selected Papers 1* (pp. 241-250). Springer Berlin Heidelberg.
- [8]. Cekała T., Z. Telec and B. Trawiński, "Truck loading schedule optimisation using genetic algorithm for yard management," in *ACIIDS 2015: Intelligent Information and Database Systems*, Bali, Indonesia, 2015.
- [9]. Cooper, T. B., Kingston, J. H. (1995). The Complexity of Timetable Construction Problems. <http://www.cs.usyd.edu.au/~jeff/>.
- [10]. East, A. R. (2019). Timetable Scheduling via Genetic Algorithm. *National University of Ireland*.
- [11]. Foy, M. D., Benekohal, R. F., & Goldberg, D. E. (1992). Signal timing determination using genetic algorithms. *Transportation Research Record*, (1365), 108.
- [12]. Gajski, D. D., Zhu, J., Dömer, R., Gerstlauer, A., & Zhao, S. (2012). *SpecC: Specification language and methodology*. Springer Science & Business Media.
- [13]. Ghasemi, O., Handley, S., & Howarth, S. (2022). Illusory intuitive inferences: Matching heuristics explain logical intuitions. Available at SSRN 4235957.
- [14]. Husseini, S., Malkawi, M., & Vairavan, K. (1992). Distributed Algorithms for Edge Coloring of Graphs. In *the 5th ISMM International Conference on Parallel and Distributed Computing Systems*.

- [15]. Jacobson L. and Kanber B.(2015). “Genetic Algorithms in Java Basics: Solve Classical Problems like The Travelling Salesman with GA.” (1st edition)). [On-line]. pp. 139-151. Available: <https://link.springer.com/book/10.1007/978-1-4842-0328-6> [May 31, 2024].
- [16]. Lange, R., Schaul, T., Chen, Y., Lu, C., Zahavy, T., Dalibard, V., & Flennerhag, S. (2023, July). “Discovering attention-based genetic algorithms via meta-black-box optimization.” In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 929-937).
- [17]. Lukas, S., Aribowo, A., & Muchri, M. (2012). “Solving timetable problem by genetic algorithm and heuristic search case study: universitas pelitaharapan timetable.” *Real-World Applications of Genetic Algorithms*, 3(78), pp.303-316.
- [18]. Malkawi, M., Hassan, M. A. H., & Hassan, O. A. H. (2008). A New Exam Scheduling Algorithm Using Graph Coloring. *International Arab Journal of Information Technology (IAJIT)*, 5(1).
- [19]. Margaret Rouse (2023), Information and Communication Technology (ICT), <https://www.techopedia.com/definition/24152/information-and-communications-technology-ict> Accessed 28 June, 2023.
- [20]. Nie S. Z.(2014), "A Java EE platform test system based on improved genetic algorithm," *Applied Mechanics and Materials*. 556(562), pp. 2581-2585.
- [21]. Podgorelec V. and P. Kokol(1997). “Genetic algorithm based system for patient scheduling in highly constrained situations.” *Journal of Medical Systems*. 21(6), pp. 417–427.
- [22]. Thanh, N.D (2007). "Solving Timetabling Problem Using Genetic and Heuristic Algorithms". *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. 33(10), pp. 472-477.
- [23]. Zeebaree, D. Q., Haron, H., Abdulazeez, A. M., & Zeebaree, S. R. (2017). “Combination of K-means clustering with Genetic Algorithm:” A review. *International Journal of Applied Engineering Research*. 12(24), pp.14238-14245.