First Fit Algorithm;A Graph Coloring Approach to Conflict-Free University Course Timetabling

ABSTRACT

Aims: Tackling scheduling issues with the most optimal graph coloring algorithm has consistently posed significant difficulties. The university schedule scheduling problem can be expressed as a graph coloring problem, where courses are depicted as vertices and the connections between courses that have common students or teachers are represented as edges. Subsequently, the task at hand is to assign the vertices with the minimum number of colors. In order to accomplish this task, this paper present a graph coloring technique to conflict free university course timetabling using first fit algorithms.

Methodology: The conflict graph is partitioned into a set of independent color classes to be assigned time slots and transformed into a conflict-free timetable.The Ladoke Akintola University of Technology (LAUTECH) University Course Timetabling Data was adopted. The allocation of venue based on the allotted time slots is done using first fit packing algorithm. The proposed model is implemented using Python programming language. The developed model had courses being represented as vertices and edges. The course conflict graph was created based on the acquired dataset using vertices-edges relationship diagram. The implemented model is evaluated in terms of Halstead complexity metrics: Program Volume (PV), Program Length (PL), Program Effort (PE), Program Difficulty (PD) and Execution Time (ET).The PV, PL, PE, PD and ET values obtained for the implemented model are 18.45kbits, 0.51, 1037684, 1.97 and 20.45 secs, respectively.The PV, PL, PE, PD and ET values obtained for the implemented model are 18.45kbits, 0.51, 1037684, 1.97 and 20.45 secs, respectively.

Conclusion: The proposed model shows a significant improvement over the existing models by producing conflict-free course timetabling problem with better evaluation results. This work will be highly useful in solving various scheduling, optimization and NP-hard related computational problems.

Keywords:*First fit algorithm, Timetabling problem, Computational problem, Optimization, Graph Coloring, vertices, Conflict graph,*

1. INTRODUCTION

institution course scheduling is the allocation of specific time slots to the courses offered by an institution within a specific semester [1]. The time slots must be allocated in a way that ensures no two courses with the same students and teachers are assigned the same time slot. Many colleges often employ a manual implementation procedure, where teaching staff or other designated individuals are responsible for scheduling their courses [2]. The manual process driven by humans may be efficient for a system with a limited number of courses, students, and teachers. However, as the number of courses, students, and teachers increases, this process can result in significant wastage of human effort and time, as well as potential conflicts in finding solutions. Since [3], there has been significant research interest in automating course scheduling as a means to tackle this issue.

Graph theory encompasses three primary variations of the graph coloring problem. The three types of coloring used in graph theory are vertex coloring, edge coloring, and face coloring. Vertex coloring is the act of assigning colors to the vertices of a graph in a way that ensures no two adjacent vertices have the same color. The scheduling of a university timetable can be represented as a vertex coloring issue. Vertex coloring, also known as graph coloring, is a suitable approach as other graph coloring problems can be transformed into vertex coloring problems. This practice has been consistently applied in this study. In the context of a graph G, k-coloring is the act of assigning k distinct colors to the vertices of G, so that no two adjacent vertices have the same color. The colors are represented by non-negative integers ranging from 1 to k. The chromatic number of graph G is the smallest value of k for which there exists a k-coloring of G. The chromatic number of a graph G is denoted as X(G) and is within range [4, 5].

Several algorithms now exist for addressing the issue of university schedule scheduling, including constraint-based algorithms, genetic algorithms, local search algorithms, and backtracking algorithms. The current algorithms have some shortcomings. They may lack efficiency, fail to discover the ideal solution, struggle with complex restrictions, have difficulty scaling to large-scale issues, and be computationally expensive.

Creating a comprehensive course scheduling report that satisfies all the limitations manually is an arduous and time-consuming effort for the teaching staff at Ladoke Akintola University of Technology. Implementing even little changes can disrupt the entire schedule, making it challenging to modify the course plan and implement a more effective strategic approach. The primary aim of this work is to substitute this laborious procedure with efficient graph coloring methods, specifically First Fit, for resolving the problem. This work provides a graph coloring technique for conflict-free university course timetabling utilizing first-fit algorithms.

2. RELATED WORK

Several studies have been conducted over the years on graph coloring difficulties. Within this area, we shall examine the graph coloring alternatives as a means to tackle the issue of course scheduling. In their work, the authors in [6] conducted a comparative analysis of graph coloring algorithms, evaluating their solution correctness and execution times. The authors introduced a course timetable scheduling system in [7, 8] that utilizes a graph coloring technique. Their primary focus is on college course scheduling, which involve specific and rigid limits as well as more flexible constraints. In addition, they examined a problem involving the scheduling of teachers and subjects. They utilized two different graph coloring techniques and successfully supplied a comprehensive solution. In [9], the researcher expressed a worry over the issue of scheduling course timetables. The adoption of diverse graph coloring algorithms includes the Saturation algorithm, Degree of Saturation Algorithm, Simulated Annealing algorithm, and Greedy algorithm. The graph coloring algorithm is meant to minimize clashing schedules.

The authors in [10] introduced a university examination scheduling system that utilizes a graph coloring algorithm grounded in RFID technology. The present study investigates the exam timetabling problem through the application of several artificial intelligence techniques. In their study, the researcher in [11] introduced an innovative method of graph coloring to address the challenges of university course timetabling problems. The Welch Powell algorithm was utilized to construct graph coloring, which enables the generation of class schedules. The author in [12] suggested a method for scheduling timetables using graph coloring. They create a comprehensive system that can handle the constantly evolving

needs of huge educational institutions and offers effective course scheduling. The authors in [13] utilized a coloring scheduling technique to create college course timetables in an automated manner. They suggested an algorithm that will take into account all necessary constraints from the perspectives of both the students and teachers. The authors in [14] introduced automata-based approximation techniques to address the least vertex coloring problem. These algorithms were compared with established coloring algorithms.

The alternative graph coloring approach was introduced in [15] for the purpose of timetabling courses at a university. This strategy involves modeling and solving the timetabling problem using graph coloring techniques. In addition, they provide a university schedule that includes room allocations while utilizing an alternative graph coloring technique. The authors in [16] presented a graph coloring approach to address the University Course Timetabling Problem. They suggested a novel method to address the recurring course scheduling challenges faced by educational institutions.

The author introduced a graph coloring technique in [17] to enhance the optimization of solutions for the timetabling problem. A university timetabling system was created in [18] using graph coloring, which is capable of managing several typical timetabling functionalities. In addition, they also create standard scheduling functionalities that may be managed within the system. The paper introduces a novel approach by combining a memetic teaching learning-based optimization algorithm with a robust tabu search algorithm to address the graph coloring problem.

For this study, which aims to compare the effectiveness of various graph coloring algorithms in solving the university timetable scheduling problem, a comprehensive literature review has been conducted to understand the different methods and techniques that have been previously suggested. This review facilitated the identification of deficiencies in the current understanding and aided in the formulation of research objectives and questions that directed the investigation. Moreover, it aided in establishing the theoretical underpinnings of the study and served as a platform for selecting the most suitable approaches and procedures for the research. The literature evaluation was crucial in influencing the research technique and facilitating a thorough comprehension of the area, hence allowing us to successfully accomplish the research objectives and derive significant conclusions from the study.

3. METHODOLOGY

The method was able to find adaptable, conflict-free solutions to course scheduling issues by using the graph colouring technique. Students should not take more than five classes per day, only one class should be in session at any given time, and students should register for classes in advance before making their schedules; these were the assumptions used to build the model. The first fit was used in the development of the system as shown in algorithm 1. Fig 1 shows the flow diagram for the next stages to solve the course scheduling problem.

i. Acquisition of course data: the course data of LAUTECH for 2015/ 2016 session (rain and harmattan semester) were assembled that serve as input to our problem instance. Each course to be scheduled constitutes a data entry, containing required or optional information.
ii. Modeling of course conflict graph from university course data: the course data from the university were used to create conflict graph such that each course represents the vertices and the edges denote connection or relationship between each pair of conflicting courses.

iii. Coloring of the course conflict graph: the vertices and edges representing the courses and their relationships were colored with the different color set if each pair of vertices shares the same edges and same color if adjacent to produce conflict-free university course timetable.
iv. Transforming of coloring into conflict free timetable.
v. Venue allocation for the course: the venues were assigned to each course by considering the number of students' enrolment and the room capacity.

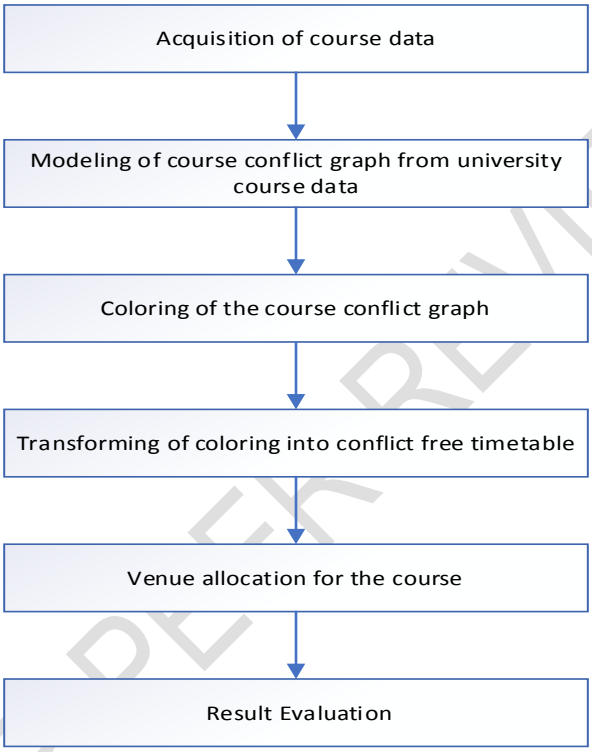

Fig 1. Flow diagram of the develop system

Algorithm 1 First Fit

```
for i ← 1 to n do
    color[i]= −1 used[i]= false
end
for u ← 1 to n do
    for v ←1 to n do
        if adj[u][v] equals 0 then
        end
            continue
        c = color[v]
        if c equals -1 then
        end
            continue
        used[c] = true
    end
    for c ←1 to n do
        if used[c] equals 0 then
            color[u] = c break
        end
    end
    for i ←1 to n do
        used[i] = false
    end
end
```

3.1 First Fit Algorithm

This algorithm utilizes a straightforward heuristic method by assigning the color that is least accessible to a vertex. It begins with the first vertex in the graph and continues in a sequential manner. The First Fit Algorithm (FF) is both simpler and more efficient than previous greedy algorithms [20, 21]. It is employed as an online algorithm for the dynamic coloring of graphs [22,23]. This refers to a situation where requests are received in a dynamic and unpredictable manner [24,25]. The procedure begins by creating an array called "colors" to record the colors assigned to each vertex. For every vertex, v, in the graph, the procedure assigns the smallest color that is currently not being used to the vertex and marks the color as used. Algorithm 1 provides a comprehensive and sequential breakdown of the algorithmic process.

.

3.2 Data Acquisition

During the acquisition of these data, it was generally assumed that student will not take more than five courses per day, one course will take place in a particular venue at a time and the number of students registered for particular courses are known prior to preparing the university course timetable.

i. *CourseID*: The *CourseID* identified the name of the course along with course code, for instance (CSE 201) as guided by either the faculty or university. The *CourseID*must be unique.
ii. *LecturerName:* The *LecturerName* contain the name of the lecturers in charge of particular course identified by their last name, first name and middle name. For instance, Oyeleye, A.K.
iii. *NUM-DAYS*: The *NUM-DAYS* indicate the number of days that each course was offered per week. The *NUM-DAYS* may be "3" which indicate that particular course will be offered three times in a week or "2" which indicate that particular course will be offered two times in a week or "1" which indicate that particular course will be offered only once in a week and 4 is no preference.
iv. *DAYS:* The *DAYS* will contain preference for the days of the week that the course was offered. *DAYS* likes "Monday =*M*, Tuesday =*T*, Wednesday =*W*, Thursday = *Y*, Friday = *F*". No lectures on Saturday and Sunday. It is assumed that courses taking place three times per week will be on *M, W* and *F*. Courses taking place two times per week will be *T* and *Y* while course taking place one time will be on *M, T, W, Y or F.*
v. *RM-TYPE*: The *RM-TYPE* contains preference for the type of the venue that the lecture is due to take place, as directed by the type of the lecture.
vi. *RM-SIZE*: The *RM-SIZE* specify the expected enrollment of the course, that is, the number of students that register for particular course determined the size of the venue that will be schedule for such lecturer. The *RM-SIZE* should have capacity enough to occupy the number of student register for the course.
vii. *TIM-DAY*: The *TIM-DAY* contains the time of day the lecture is due to take place whether in the Morning = 1, Afternoon = 2, Evening = 3 and No preference = 4.

3.2.1 Creation of course conflict graph using vertices – edges relationship

The course conflict timetabling graph G was created using the provided university course data for the specific academic session. The course conflict graph was generated utilizing university course data as input. The course conflict graph consists of vertices that represent individual courses, and edges that connect pairs of conflicting courses represented by the vertices. These edges illustrate the relationship between each course. The model for solving the course timetabling problem takes into account both hard constraints and soft constraints. The ideal solution is shown in Tables 3.1a-b. Typically, it is expected that students will not enroll in more than five courses per day. The amount of time allotted to each course is determined by the number of units and the number of students who have registered for the courses before the course schedule is created.

Given set of *n* courses {C_1, C_2, C_3,……., C_n} to be scheduled. Eachcourse *Ci* represented by exactly one vertex*Vi* in *G*. Therefore, *G* contains *n* vertices, and*V* (*G*) = {V_1, V_2, . . ., V_n}. Each vertex in *G* is firstclassified as belonging to exactly one of ten "groups",depending on its corresponding course's *NUMDAYS*,*DAYS*, and *TIM-DAY* entry:

i. *Group 1*: Courses with *NUM-DAY=3, DAYS= MWF* and *TIM-DAY =1*
ii. *Group 2*: Courses with *NUM-DAY=3, DAYS= MWF* and *TIM-DAY =2*
iii. *Group 3*: Courses with *NUM-DAY=3, DAYS= MWF* and *TIM-DAY =3*
iv. *Group 4*: Courses with *NUM-DAY=3, DAYS= MWF* and *TIM-DAY =4*
v. *Group 5*: Courses with *NUM-DAY=2, DAYS= TY* and *TIM-DAY =1*
vi. *Group 6*: Courses with *NUM-DAY=2, DAYS= TY* and *TIM-DAY =2*
vii. *Group 7*: Courses with *NUM-DAY=2, DAYS= TY* and *TIM-DAY =3*
viii. *Group 8*: Courses with *NUM-DAY=2*, *DAYS= TY* and *TIM-DAY =4*
ix. *Group 9*: Courses with *NUM-DAY=4*

x. *Group 10*: Courses with *NUM-DAY=1.*

The course data depicting vertices in the conflict graph were categorized based on number of days, time of day and days of the week. The grouping was done as presented in Fig 2.

The edges indicated the pairs of courses that do not want to be schedule at the same time. The edges were added to the conflict graph based on the hard and soft constraints such that if *Lecturer Name* is the same for given pair of courses (course1 and course2), such pair of courses cannot be schedule at the same time because the *Lecturer Name* cannot teach the two courses at the same time. Also, if the *RM-TYPE* request for pair of courses is the same, such pair of courses cannot be schedule at the same time, because it is assumed that only one course will occupy the room at any given time.

Considerations were also given to the *NUM-DAY*, *TIM-DAY* and *DAYS* preference when adding edges to the course conflict graph. Edges can be added to each of the group to reflect the preference for *NUM-DAY*, *TIM-DAY* and *DAYS* as follows. From Fig 2, Groups 1-4 were separated from Groups 5-8 while Group 9 and 10 were outside the circle.

Table 1: Hard constraints

HC	Description
HC_1	Two courses taught by same lecturer cannot take place same time
HC_2	Two courses with the same venue should not be scheduled at the same time
HC_3	Courses that enrol same set of students cannot be scheduled at the same time.
HC_4	Lecturer must be available at the time the course is scheduled.

Table 2: Soft constraints

SC	Description
SC_1	Lecturer must have time preference for the scheduled course.
SC_2	Lecturer must have specific room request
SC_3	Room must be large enough to contain students for particular course.

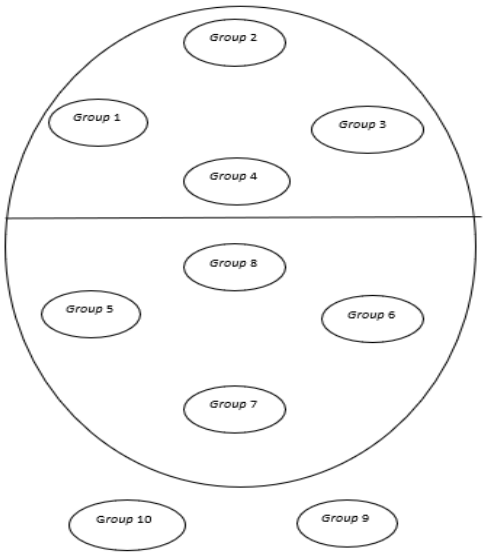

Fig 2: Partitioning of Courses of Conflict Graph into Groups

i. each vertex in Group 1 and each vertex in Group 5
ii. each vertex in Group 1 and each vertex in Group 6
iii. each vertex in Group 1 and each vertex in Group 7
iv. each vertex in Group 1 and each vertex in Group 8
v. each vertex in Group 2 and each vertex in Group 5
vi. each vertex in Group 2 and each vertex in Group 6
vii. each vertex in Group 2 and each vertex in Group 7
viii. each vertex in Group 2 and each vertex in Group 8
ix. each vertex in Group 3 and each vertex in Group 5
x. each vertex in Group 3 and each vertex in Group 6
xi. each vertex in Group 3 and each vertex in Group 7
xii. each vertex in Group 3 and each vertex in Group 8
xiii. each vertex in Group 4 and each vertex in Group 5
xiv. each vertex in Group 4 and each vertex in Group 6
xv. each vertex in Group 4 and each vertex in Group 7
xvi. each vertex in Group 4 and each vertex in Group 8

Edges from groups 1-4 can be added to the group 5-8 to reflect *NUM-DAYS* and *DAYS* as depicted in the Fig .3 to indicate those constraints that cannot be schedule at the same time.

Also, edges from Group 1,2,3,5,6,7,8 and 9 can be added to reflect preference for *TIM-DAY* as illustrated in Fig 4;

xvii. vertex in Group 1 and each vertex in Group 2
xviii. vertex in Group 1 and each vertex in Group 3
xix. vertex in Group 2 and each vertex in Group 3
xx. vertex in Group 5 and each vertex in Group 6
xxi. vertex in Group 5 and each vertex in Group 7
xxii. vertex in Group 6 and each vertex in Group 7

Edges can be added to reflect for *TIM-DAY* where preference is “4”.

i. each vertex in Group 3 and each vertex v_i in Group 4, if the TIME-OF-DAY preference of the corresponding course *Ci* = "4".
ii. each vertex in Group 7 and each vertex v_i in Group 8, if the TIME-OF-DAY preference of the corresponding course *Ci* = "4".

As a result of adding such edges, courses that request to meet "not in the evening" would ultimately not be scheduled for evening time slots. Vertices in Group 9 correspond to courses with NUM DAYS = "-".\

Group 9 and 10 which are outside the partition set of the conflict graph. To reflect any TIME-OF-DAY preferences for courses represented by vertices in Group 9, edges can be added to the conflict graph G in the following manner. For each vertex $V_i$in Group 9 corresponding to a course C_i, edges were added as depicted in Fig 5:

i. each vertex in Group 9 and each vertex in Groups 2, 3, 6, and 7, if the TIME-OF-DAY preference of *Ci* = "1" (i.e., "morning"), or
ii. each vertex in Group 9 and each vertex in Groups 1, 3, 5, and 7, if the TIME-OF-DAY preference of *Ci* = "2" (i.e., "afternoon"), or
iii. ach vertex in Group 9 and each vertex in Groups 1, 2, 5, and 6, if the TIME-OF-DAY preference of *Ci* = "3" (i.e., "evening"), or
iv. each vertex in *Group 9* and each vertex in *Groups 3 a*nd *7*, if the *TIME-OF-DAY* preference of *Ci* = "4" (i.e., "not in the evening").

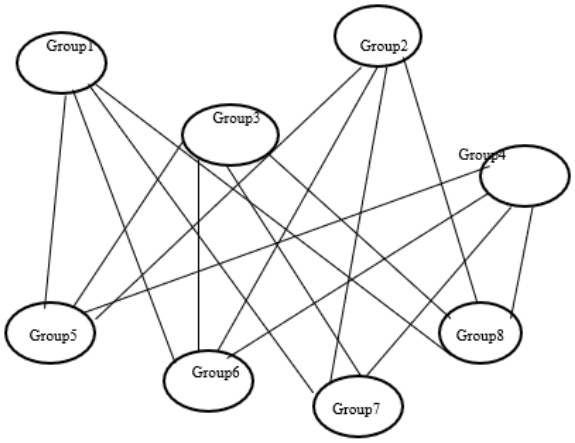

Fig .3: Vertices-Edge Relationship Diagram based on Num-Days and Days

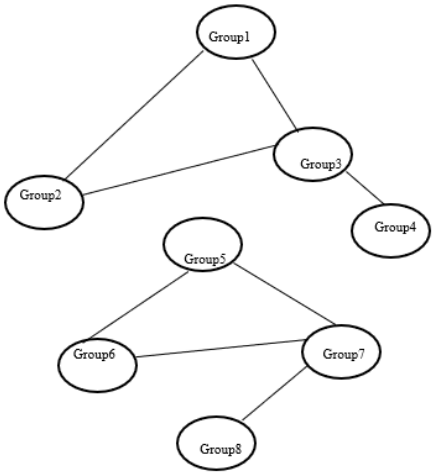

Fig4: Vertices-Edges relationship diagrams based on Tim-Day

Vertices in *Group 10* correspond to courses with *NUM DAYS* = "1". Such courses will be scheduled to meet for 1 day per week Monday, Tuesday, Wednesday, Thursday or Friday. The edges can be added to the Group as follows as indicated in Fig .6:

v. each vertex in *Group 10* and each vertex in *Groups 5, 6, 7*, and *8*, if the DAY preference of C_i= "M", "W", or "F", or
vi. each vertex in *Group 10* and each vertex in *Groups 1, 2, 3*, and *4*, if the *DAY* preference of C_i= "T" or "Y", and
vii. each vertex in *Group 10* and each vertex in *Groups 2, 3, 6,* and *7*, if the *TIME-OF-DAY* preference of C_i = "1", or
viii. each vertex in *Group 10* and each vertex in *Groups 1, 3, 5,* and *7*, if the *TIME-OF-DAY* preference of C_i = "2", or
ix. each vertex in *Group 10* and each vertex in *Groups 1, 2, 5,* and *6,* if the *TIME-OF-DAY* preference of C_i = "3", or
x. each vertex in *Group 10* and each vertex in *Groups 3* and *7*, if the *TIME-OF-DAY* preference of C_i = "4".

3.2.2 Coloring of the course conflict graph using graph coloring technique

The proper coloring of the vertices of the conflict graph to construct a conflict-free timetable was done using graph coloring. The technique ranks the color classes by the number of vertices currently in color set and search the ranked list in ascending order. SFSG provide the most balance and evenly distributed color classes. The techniques assigned each vertex of the graph to a color class and each vertex represents course to be scheduled. The vertex of the conflict graph examines one vertex at a time in ascending order and each vertex was colored with one of the existing colors, if it is not possible to color the vertex with existing color, new color class set was created and vertex assigned to the new color class.

Each of the vertices of conflict graph corresponding to course to be schedule has been grouped into ten (10) groups and was color in group order; first by coloring all the vertices in *Group1*, *Group2, Group3, Group4, Group5, Group6, Group7, Group8, Group9* and *Group10.* Based on the order of *NUM-DAY, TIME-DAY* and *DAYS* using vertices-arcs relationship for each of the group.Each vertex corresponds to exactly one course that was

scheduled, so ordering the vertex in each of the group corresponds to ordering the courses corresponding to the vertices in that group. The courses or vertex within each group was ordered either by using the *CourseID*, L*ecturer Name* and *RM-SIZE.*

3.2.3 Transformation of the coloring into conflict-free timetable

The proper coloring of conflict graph was transformed to a conflict-free timetable, which is denoted by T (G) is a very vital step. In the coloring of conflict graph, each vertex in the conflict graph corresponds to the courses that were scheduled and each edge correspond to the pair of courses that conflict. The proper coloring of the conflict graph, where each color class represent non-overlapping time period, as a result, conflict graph was transformed into conflict-free timetable. Each course will be assigned to time period designated by the color of its corresponding vertex in conflict graph.

The goal is to properly color the vertices of our conflict graph using one of the above additions of edges to the vertices using vertices-arcs relationship based on *NUM-DAY, TIM-DAY* and *DAYS* variation and thereby partition our vertices into independent color classes. These color classes designate sets of courses which can safely be assigned to the same time slot without conflicts. Vertices corresponding to courses which cannot be assigned to the same time slot due to potential conflicts will be colored with different colors in our model.

3.2.4 Venue allocation for courses

The graph coloring technique partitioned courses into a set of independent color classes from which time slots can be assigned. The algorithm does not also assign each course to an appropriate classroom. To assign already timetabled courses to classroom, a concept based on bin packing algorithm was used for the allocation of venues to courses. The bin packing problem is stated as follows: Given a finite set $U = \{u_1, u_2, \ldots, u_n\}$ of "items", a positive integer "size" $S(u) \in Z^+$ for each item $u \in U$, a positive integer bin capacity B, and a positive integer k, find a partition of U into k disjoint subsets $U_1, U_2, \ldots, U_k$such that the sum of the sizes of the items in each $U_i$is no more than B. Each subset U_i is viewed as specifying a set of items to be placed in a single "bin" of capacity B.

The "items" to pack are the n courses $u_1, u_2, \ldots, u_n$, and the "bins" are the m available classrooms $r_1, r_2, \ldots, r_m$. The "size" $S(c_i)$ of course c_i is the expected enrollment or maximum enrollment of c_i for instance, in the course data field (*CLASS SIZE or CLASSMAX SIZE*). Each classroom can hold at most one course, so there will be at most one "item" in each "bin". Furthermore, each "bin" that is, each room r_j has its own size or capacity, denoted $S(r_j)$.

The first-fit packing algorithm places the course in the venue or lecture room, one at a time in the order of increasing index, that is, it places the next course into the lowest-indexed lecture room for which the expected or maximum enrolment in that lecture room is not greater than course place in that lecture room. This produces full solution to the course timetabling problem in which all the hard constraints and slightly alter to maximize the number of soft constraints satisfied.

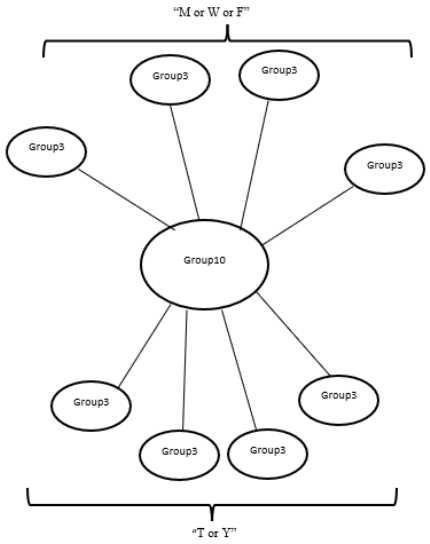

Fig 5: Vertices-Edges relationship diagram for Group 9 based on Tim-Day

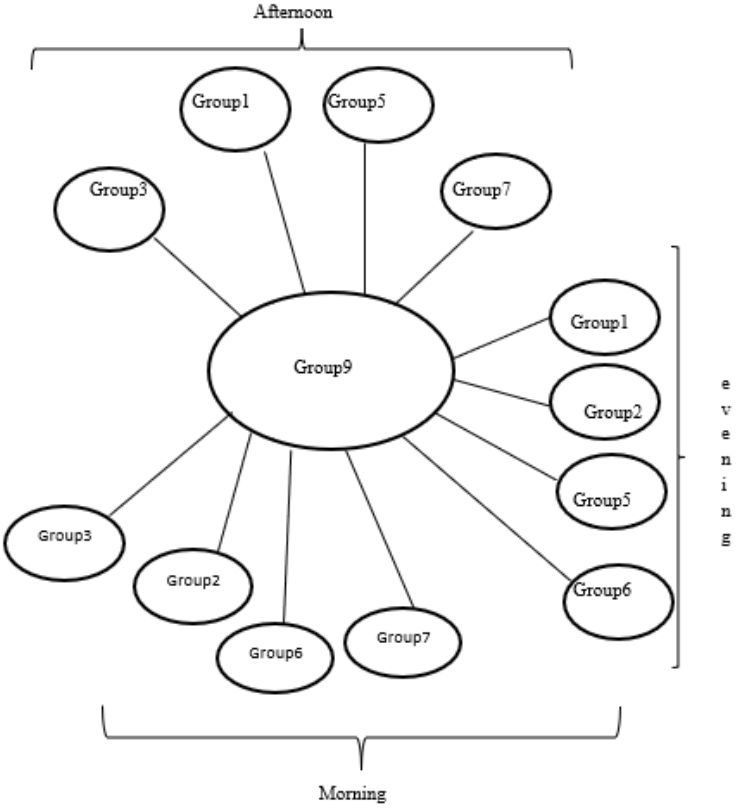

Fig 6: Vertices-Edges relationship diagram for Group 10 based on Num Days

3.3 Halstead Software Complexity Metrics

Halstead software complexity metrics are used to evaluate the performance of the developed model. There are number of distinct operators and operands denoted as $n_1$and n_2 respectively. Also, total number of operators and operands are denoted as N_1 and N_2. The following are the parameters used to evaluate the performance of the developed model which includes Program vocabulary (*n*), Potential minimum volume (V*), Program difficulty (D), Program level (L), Program volume (V), Program effort (E) and Estimated Program length (N). Table 3. denoted all the considered Halstead metrics with their respectively defined parameters.

Table 3: Halstead software complexity metrics and parameters

Software Metrics	Formulae
Number of distinct operators	n_1
Number of distinct operands	n_2
Total number of operators	N_1
Total number of operands	N_2
Estimated Program length	$N= N_1 + N_2$
Program Vocabulary	$n = n_1 + n_2$
Program Volume	$V = N\log_2 n$
Potential minimum volume	$V^* = (2 + n_2) \log_2 (2+ n_2)$
Program length	$L = V^* / V$
Program effort	$E = V/ L$

4. RESULTS AND DISCUSSION

4.1 Experimental Results

The dataset used for this experiment was obtained from course timetable of LAUTECH, Rain and Harmattan 2015/ 2016 with 740 courses and 72 venues. All computations were performed using Python 3.6, Django web framework version 2.2 which runs on Window 10 OS with Processor core i7 2.8GHz, secondary storage of 256 GB SSD, RAM 8GB. Experiment was conducted using smallest-first-search-greedy graph coloring technique on the dataset to produce course conflict graph G with 740 vertices (one vertex for each of the 740 courses to be scheduled) and 208089 edges (reflecting potential timetabling conflicts due to HC and SC).

4. 2 Graph Coloring Results based on Halstead Complexity Software Complexity Measures

The parameters used in measuring the Halstead software complexity metrics are; No of distinct operators (n_1), No of distinct operands (n_2), Total number of operators (N_1), Total number of operands (N_2), where $N = (N_1+N_2)$ and $n = (n_1+n_2)$. The results for measuring the complexity of graph coloring technique is presented in Table 4..

Program Volume (V)

The unit of measurement of program volume is the standard unit for “bits”. It is the actual size of a program if a uniform binary encoding for the vocabulary is used. The program volume V is the information contents of the program measured in bits needed to encode the program. Table 5 revealed that Graph coloring technique has V of 18.45 kbits

Table 4: Data obtained for measuring the complexity of graph coloring technique

Parameters for Halstead Complexity	Graph coloring Technique
No of distinct operators (n_1)	15
No of distinct operands (n_2)	20
Total number of operators (N_1)	3447
Total number of operands (N_2)	150
N i.e. (N_1+N_2)	3597
n i.e. (n_1+ n_2)	35

Table 5 Data obtained during and after execution of graph coloring technique

Parameters	Graph Coloring
Potential minimum Vol.(V*)	9351
Program Difficulty (D)	1.9728
Program Level (L)	0.5069
Program Volume (V)	18.45 kbits
Program Effort (E)	1037684
Estimated Prog.Length (N)	174.7

Program Level (L)

The value of program level ranges between zero and one, with L= 1 representing a program written at the highest possible level, that is, with minimum size. The program level is the measure of a person's ability to understand a program. Table 5 shows that graph coloring technique has program level of 0.506. The result revealed that graph coloring takes less effort to developed and not difficult to understand.

Program Effort (E)

The unit of measurement of programming effort is elementary mental discriminations required to implement the program and also the effort required to read and understand the program. Table 5 indicates that 1037684 elementary mental discrimination is required to construct the program, which takes a longer time to produce.

Program Difficulty (D)

The difficulty level of the program is proportional to the number of the unique operator in the program. Table 5 revealed that graph coloring technique has difficulty level (D) of 1.9728. It implies that it is not difficult to understand.

Program size

The program size is the amount of disk space occupied by the program and it is measured in bits or bytes. Graph coloring technique has a program size of 5.87 mb.
The various Halstead software complexity parameters used for the graph coloring is represented by Fig 1. The program effort (E) with highest value indicates that mental discrimination to construct graph coloring program is huge. The program level (L) with the minimum value indicates that it takes less effort to develop. In terms of program difficulty (D), implies that the program is not much difficult to understand because the number of unique operators is proportional to the program difficulty.

4.2 Performance Evaluation of Graph Coloring Results based on Execution time

The graph coloring technique was evaluated in terms of computational time taken to perform and verify the whole coloring from the course conflict free graph. The chromatic number $X(_G)$of the graph, which is the minimum number of colors to properly colored the graph (G) is equal to11. All computations were performed using Python 3.6 version, Django web framework version 2.2 which runs on Window 10 OS with Processor core i7 2.8GHz, secondary storage of 256 GB SSD, RAM 8GB. Fig5represent the color classes of the various courses with their respective sizes. The C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10} *and* C_{11} represented color classes for each of the course's distribution with the sizes of 55, 232, 165, 120, 60, 20, 38, 10, 12, 15 and 13 respectively.

From the computations performed, the coloring time was 20.56 secs to color the whole dataset that was used for the experiment. The color class (C_2) with the highest size (232)takes much coloring time while (C_7) with the lowest size (10) was executed very faster. The color classes with smaller number of sizes were given low priority while developing the university course timetabling while the color classes with higher number of size were given high priority.

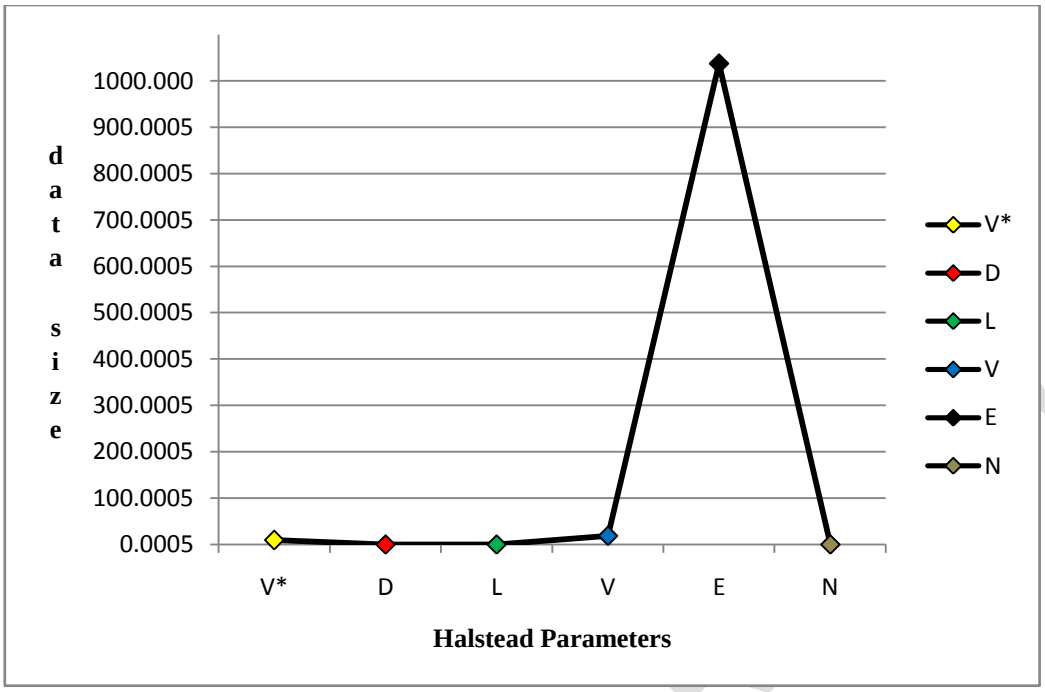

Fig7: Halstead Software Complexity Measures for Graph Coloring Technique

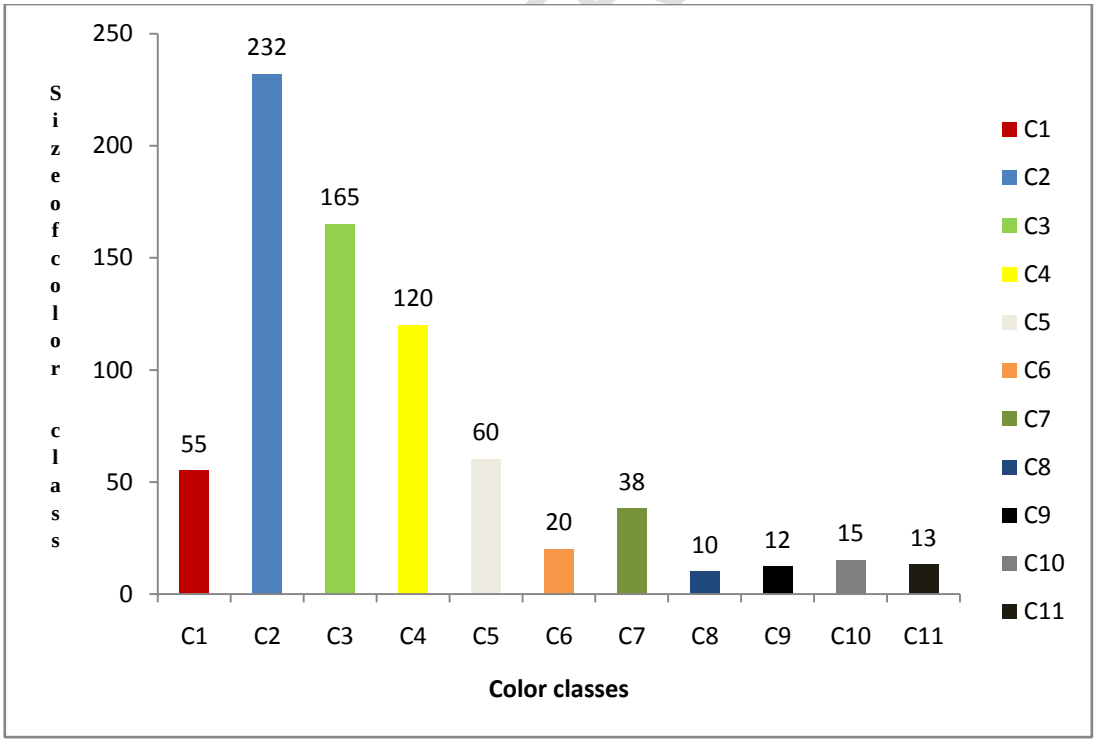

Fig8: Bar chart showing performance of Graph coloring based on Execution time

5. CONCLUSIONS

This study thoroughly investigated the university course timetabling, which is a problem classified as NP-hard, by employing the First Fit Algorithm. The dataset utilized in this study was obtained from Ladoke Akintola University of Technology (LAUTECH), located in Nigeria. The obtained dataset is used as an input for creating a computational model to solve university timetabling problems. This model utilizes the technique of graph coloring to ensure that both the hard and soft constraints are satisfied optimally. The result is a timely and conflict-free timetable that is acceptable to both students and school management.

The study showcased computational findings and additional observations pertaining to the utilization of the new graph coloring model for course timetabling, incorporating real data gathered from LAUTECH. The results highlighted the efficacy of the enhanced graph coloring techniques utilizing greedy search, particularly those that incorporated intelligent color searching and vertex ordering to rapidly generate a timetabling solution that met both the rigid and flexible criteria. A more efficient graph coloring technique utilizing a greedy search algorithm demonstrated notable enhancements in terms of substantial time reduction and reduced utilization of computational resources. The course timetabling problem's hard and soft restrictions were fully resolved by employing an enhanced graph coloring method with a greedy search algorithm. Utilizing graph coloring as a preprocessing technique in the bin algorithm system yields an optimal and satisfactory solution for resolving the course timetabling problem. The study is available for more investigation. It is strongly advised to combine meta-heuristic algorithms, such as the firefly algorithm, bat algorithm, and artificial bee colony, with the graph coloring technique in order to enhance performance.

6. REFERENCES

[1] Sadaf Naseem Jat and Shengxiang Yang. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. Journal of Scheduling, 14(6):617–637, 2011.

[2] Wang Wen-jing. Improved adaptive genetic algorithm for course scheduling in colleges and universities. International Journal of Emerging Technologies in Learning, 13(6), 2018.

[3] Raneem Gashgari, Lamees Alhashimi, Raed Obaid, Thangam Palaniswamy, Lujain Aljawi, and Abrar Alamoudi. A survey on exam scheduling techniques. In 2018 1st International Conference on Computer Applications Information Security (ICCAIS), pages 1–5. IEEE, 2018.

[4] D aniel Marx. Graph colouring problems and their applications in scheduling. PeriodicaPolytechnica Electrical Engineering (Archives), 48(1-2):11–16, 2004.

[5] Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 10(1):85–86, 1967.

[6] Murat Aslan and Nurdan Akhan Baykan. A performance comparison of graph coloring algorithms. International Journal of Intel- ligent Systems and Applications in Engineering, pages 1–7, 2016.

[7] Narayan Poddar and Basudeb Mondal. An instruction on course timetable schedul- ing applying graph coloring approach. Inter- national Journal of Recent Scientific Research, 9(2):23939–23945, 2018.

[8] Runa Ganguli and Siddhartha Roy. A study on course timetable scheduling us- ing graph coloring approach. international journal of computational and applied mathematics, 12(2):469–485, 2017. Graph coloring in university timetable scheduling: A compar- ative study 23

[9] Walter Klotz. Graph coloring algorithms. Verlag nicht ermittelbar, 2002.

[10] Akhan Akbulut and G uray Yilmaz. University exam scheduling system using graphcoloring algorithm and rfid technology. International Journal of Innovation, Management and Technology, 4(1):66, 2013.
[11] Kristoforus Jawa Bendi, Theresia Sunarni, and Achmad Alfian. Using graph color- ing for university timetable problem. Interna- tional Journal of Science and Research (IJSR), 7:1692–1697, 11 2018.
[12] NK Cauvery. Timetable scheduling using graph coloring. International Journal of P2P Network Trends and Technology, 1(2):57–62, 2011.
[13] P Nandal, Ankit Satyawali, Dhananjay Sachdeva, and Abhinav Singh Tomar. Graph coloring based scheduling algorithm to automatically generate college course timetable. In 2021 11th International Conference on Cloud Computing, Data Sci- ence Engineering (Confluence), pages 210–214. IEEE, 2021.
[14] J Akbari Torkestani and MR Meybodi. Graph coloring problem based on learning automata. In 2009 International Conference on Information Management and Engineering, pages 718–722. IEEE, 2009.
[15] Timothy A Redl. University timetabling via graph coloring: An alternative ap- proach. Congressus Numerantium, 187:174, 2007.
[16] Zafer Bozyer, M Sinan Ba sar, and Alper Aytekin. A novel approach of graph coloring for solving university course timetabling problem. Proceeding Number, 300:23, 2011.
[17] Sara Miner, Saleh Elmohamed, and Hon W Yau. Optimizing timetabling solutions using graph coloring. NPAC, Syracuse Uni- versity, pages 99–106, 1995.
[18] EK Burke, DG Elliman, and R Weare. A university timetabling system based on graph colouring and constraint manipulation. Journal of research on computing in education, 27(1):1–18, 1994.
[19] Tansel Dokeroglu and Ender Sevinc. Memetic teaching–learning-based optimiza- tion algorithms for large graph coloring prob- lems. Engineering Applications of Artificial Intelligence, 102:104282, 2021.
[20] Hussein Al-Omari and Khair Eddin Sabri. New graph coloring algorithms. Amer- ican Journal of Mathematics and Statistics, 2(4):739–741, 2006.
[21] Linda Ouerfelli and Hend Bouziri. Greedy algorithms for dynamic graph coloring. In 2011 International Conference on Commu- nications, Computing and Control Applications (CCCA), pages 1–5. IEEE, 2011.
[22] NS Narayanaswamy and R Subhash Babu. A note on first-fit coloring of interval graphs. Order, 25(1):49–53, 2008.
[23] Stavros D Nikolopoulos and Charis Papadopoulos. On the performance of the first-fit coloring algorithm on permutation graphs. Information Processing Letters, 75(6):265–273, 2000.
[24] Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 10(1):85–86, 1967.
[25] Dahlan Abdullah, Marsali Yaton, Heri Sujatmiko, Sepyan Purnama Kristanto, Hendra Nazmi, Irma Lisa Sridanti, AndangSuhendi, AbdurrozzaqHasibuan, Renny Kurniawati, Darmadi Erwin Harahap, et al. Lecture scheduling system us- ing welch powell graph coloring algorithm in informatics engineering departement of universitas malikussaleh. In Journal of Physics: Con- ference Series, volume 1363, page 012074. IOP Publishing, 2019.