Depth First Search (Graph: Map Coloring)*

Nurkhairina Balqis Binti Mohammad Joe
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220206@student.uthm.edu.my

Nurulaina Nisa Binti Sahanuan
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220179@student.uthm.edu.my

Nurul Jannah Binti Kamarul Zaman
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220147@student.uthm.edu.my

Serena Ng Yen Xin
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220061@student.uthm.edu.my

Tuan Nur Rifaqiah Binti Tuan Hanizi
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220040@student.uthm.edu.my

Tuan Khalidah Syazwana Binti Tuan Mohd Kasmawi
Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
ai220118@student.uthm.edu.my

***Abstract*—This case study explores the application of depth-first search (DFS) in the context of map coloring. The process of coloring a map entails giving each region a unique hue such that no neighboring parts have the same color. DFS is an efficient algorithm that solves this problem by methodically exploring the graph that represents the map and making sure that neighboring parts are colored suitably while using the fewest possible colors. This study also explores the theoretical underpinnings of DFS in map coloring and emphasizes how effective it is in producing the most ideal and aesthetically acceptable color allocations using Python language. Furthermore, real-world examples and implementations show how flexible the approach is with different map structures.**

Keywords—map coloring, graph, depth-first search, color allocations, algorithm, Python language

I. Introduction

Depth First Search (DFS) is a recursive algorithm to search all the vertices of a tree data structure or a graph. The depth-first search (DFS) algorithm starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children [12]. This approach is often used in many applications such as map coloring. Traversing the graph representing the map and assigning color to the region by applying Depth First Search will solve any issues containing regions that share the same color. Depth First Search is useful for exploring different possibilities and finding a valid coloring solution. Map coloring problems help illustrate Depth First Search in solving real-world problems including scheduling and network routing. The algorithm's ability to systematically explore possibilities makes it an useful tool in various ways for graph theory.

II. Background

A. Depth First Search

Depth First Search (DFS) is a popular graph traversal approach for addressing the map coloring issue. A map is represented as a graph in this context, with regions as nodes and connections between regions as edges. The objective is to color the map so that no two nearby sections have the same color. DFS begins in a region, travels as far as feasible down each branch, and returns when color assignment problems develop. This recursive procedure is repeated until a valid coloring is identified or all alternatives have been exhausted. Because of its simplicity and versatility, DFS is an efficient technique for methodically solving map coloring challenges and guaranteeing that neighboring regions have unique colors. Application of the Depth First Search algorithm such as finding path, test if the graph is bipartite, finding the strongly connected components of a graph and detecting cycles in a graph [12]. In our case study, Depth First Search is applied to have no two subjects at the same time for exam scheduling.

B. Map Coloring

In graph theory, the most important one is map coloring. Map coloring is the procedure of assignment of colors to each vertex of a graph such that no adjacent vertices get the same color [9]. In the case of map coloring, the goal is to assign colors to regions on a map such that no two adjacent regions share the same color [11]. Algorithms like DFS are usually used to represent a graph with regions as vertices and shared borders as edges. For example, graph coloring in the context of exam timetabling is a scheduling technique where each exam represented as a vertex in a graph is assigned a specific time slot while ensuring that there are no conflicts between two subjects that share the same time slots.

III. Review Of Literature

General graph coloring techniques have already been thoroughly investigated by scholars. Numerous methods have already been employed to address the scheduling issues.

A. University Exam scheduling system using graph coloring algorithm and RFID technology

Aklubut and Yilmaz have taken into consideration a mechanism designed to concurrently arrange several exams in the same halls. The primary goal was to make better use of the hall's capacity and reduce attempts at cheating significantly. In their efforts, they have identified the university final exam weeks as their primary issue that has to be resolved. Students who are taking separate tests can sit next to each other, increasing the hall's capacity. But now is when the students' seating arrangements will start to get disorganized. They came up with the idea of assigning an

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

RFID tag to each pupil as a means of identification. The card reader reads the specific student's scheduled tests and shows them on a screen. Two algorithms were employed: a Hybrid Approach and their Graph Coloring Algorithm [1].

B. Examination Scheduler Using A Linear-Time Graph Coloring Algorithm

Debabrata and Rush came to the conclusion that a lightning-fast solution that runs in linear time may be obtained by utilizing a hybrid technique for the beginning point that combines backtracking and a brute force algorithm, all the while taking into account the symmetry of the CBCS curriculum followed in India. They first employ the backtracking technique to arrive at a preliminary answer, and then they apply the suggested methodology by utilizing the bipartite property of graphs. Subsequently, they found that the suggested technique was more effective than the backtracking algorithm that was initially employed. The technique can also be used to schedule exams in institutions, which can greatly reduce the strain of its staff. By utilizing the same, universities can quickly and efficiently arrange their semester and mid-semester tests. It will be possible for the students to take their exams for the topics they have chosen, with ease and without scheduling conflicts [2].

C. Greedy Graph Coloring Algorithm Based on Depth First Search

Gupta and Singh discovered an efficient technique of graph coloring based on the greedy approach, with the Depth First Search determining the order of execution. In order to reduce the computation time, a greedy approach is used. The coloring of the vertices in their Depth-First search ordering maintains connectivity between the coloured vertices in sequence. The suggested approach computes virtually all instances in a relatively short amount of time. The suggested approach is evaluated on challenging graph examples, and the results reveal that for the majority of the graph, known chromatic numbers are matched. The comparison with the present algorithm demonstrates that the goal is justified. When compared to previous algorithms, the suggested algorithm matched the already known chromatic number for 54 graph examples and reduced the time for the remaining graph instances [3].

IV. Methodology for Problem-Solving

The method that we used to solve the exam scheduling problem is graph coloring by implementing the depth first search algorithm in Python language. In order to create a fair exam schedule, we first have to tabulate and collect data of the subjects, sections and courses from the Faculty of Computer Science and Information Technology's (FSKTM) trimester 2 students. There are also several constraints that need to be followed as we schedule the exam[4]:-

Hard Constraints:

- Exam constraint - there is only one exam for each subject.
- Student conflict - a student cannot take two exams at the same time or slot.
- Seating restriction - the number of students seated for an exam cannot exceed the room capacity

Soft constraints:-

- A student should not have more than one exam per day
- Only one exam is happened at the same time
- Exams should not be split across rooms

A. Decomposition of Subjects

Depending on the course they are taking, students are enrolled in several subjects. Some might have the same subjects and some might not. In their course portions, they have been grouped into sections. Therefore, we sort the subjects and sections based on the Table 4.1 below:-

COURSE & SECTIONS	BIS			BIP			BIW			BIM			BIT		
SUBJECT	S1	S2	S10	S3	S4	S11	S5	S6	S13	S7	S8	S14	S1T	S2T	S3T
Discrete Structure	X	X	X	X	X	X	X	X	X	X	X	X			
Operating System	X	X		X	X		X	X		X	X		X	X	
Object-Oriented Programming	X	X		X	X		X	X		X	X		X	X	
Human Computer Interaction	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Requirement Engineering				X	X	X									
Software Security	X	X	X												
Web Application							X	X	X						
Computer Graphic										X	X	X			
Mobile Application Development														X	
Entrepreneurship													X		

Table 4.1: Subjects and Course Taken by Each Sections

B. Conflict Matrix

One of the most crucial components of the test scheduling problems is the conflict matrix, which shows a hard constraint or a pair of exams that conflict. The process of creating a conflict matrix aids in identifying the requirements that prohibit a student from taking more than one exam at once. If a student takes two subjects, then those subjects are in conflict with one another. In order to determine whether two exams clash with one another or with the examiner and invigilator, researchers must create a conflict matrix. Table 4.2 and 4.3 show the subjects that have conflicts with one another.

Code	Subject	Clashes with (code)
T1.	Discrete Structure	T2, T3, T4, T5, T6, T7, T8
T2.	Operating System	T1, T3, T4, T5, T6, T7, T8, T9, T10
T3.	Object-Oriented Programming	T1, T2, S4, T5, T6, T7, T8, T9 ,T10
T4.	Human Computer Interaction	T1, T2, T3, T5, T6, T7, T8, T9, T10
T5.	Requirement Engineering	T1, T2, T3, T4, T9, T10
T6.	Software Security	T1, T2, T3, T4
T7.	Web Application	T1, T2, T3, T4
T8.	Computer Graphic	T1, T2, T3, T4
T9.	Mobile Application Development	T2, T3, T4
T10.	Entrepreneurship	T2, T3, T4

Table 4.2: Clash Subjects with Code

Subject Code	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Clash instances
T1		X	X	X	X	X	X	X			7
T2	X		X	X	X	X	X	X	X	X	9
T3	X	X		X	X	X	X	X	X	X	9
T4	X	X	X		X	X	X	X	X	X	9
T5	X	X	X	X					X	X	6
T6	X	X	X	X							4
T7	X	X	X	X							4
T8	X	X	X	X							4
T9		X	X	X	X						4
T10		X	X	X	X						4
Clash instances	7	9	9	9	6	4	4	4	4	4	

Table 4.3: Conflict Matrix

C. Depth First Search

In this research, a depth first search algorithm by using Python language is used to assign the color of the time slot for each subject. Here provide pseudocode of the depth first search algorithm:-

```
class Exam:
    // Constructor to initialize Exam object
    function Exam(name, color, sections):
        self.name = name
        self.color = color
        self.sections = sections

// Function to build the exam graph based on shared sections
function build_exam_graph(exams):
    graph = {exam: set() for exam in exams}
    for exam1 in exams:
        for exam2 in exams:
            // Check if exams share at least one section
                    if exam1 != exam2 and any(section in exam1.sections for section in exam2.sections):
                // Add an edge between exams in the graph
                graph[exam1].add(exam2)
                graph[exam2].add(exam1)
    return graph

// Function to recursively schedule exams using graph coloring
function schedule_exams(graph, exams, current_exam_index, time_slots, assigned_colors):
    // Base case: All exams are scheduled
    if current_exam_index == len(exams):
        return true

    // Get the current exam
    current_exam = exams[current_exam_index]

    // Try assigning colors to the current exam
    for color in time_slots:
        // Check if the color is not used by neighboring exams
        if all(color != assigned_colors[neighbor] for neighbor in graph[current_exam]):
            // Assign the color to the current exam
            assigned_colors[current_exam] = color

            // Recursively schedule the next exam
                        if schedule_exams(graph, exams, current_exam_index + 1, time_slots, assigned_colors):
                return true

            // Backtrack: Undo the color assignment
            assigned_colors[current_exam] = None

    // Unable to schedule the current exam
    return false

// Function to print the final exam schedule
function print_exam_schedule(exams, assigned_colors):
    print("Exam Schedule:\n")
        print("| {:<35} | {:<18} | {:<70} |".format("Exam Name", "Color", "Sections"))
    print("|" + "-"*37 + "|" + "-"*20 + "|" + "-"*72 + "|")

    // Print details of each exam and its assigned color
    for exam, color in zip(exams, assigned_colors):
        sections = ', '.join(exam.sections)
        print("| {:<35} | {:<18} | {:<70} |".format(exam.name, color, sections))

// Create exams and their sections
exams = [
        Exam("Discrete Structure", "Red", ["S1", "S2", "S3", "S4", "S5", "S6", "S7", "S8", "S9", "S10", "S11", "S13", "S14"]),
      Exam("Operating System", "Yellow", ["S1", "S2", "S3", "S4", "S5", "S6", "S7", "S8", "S1T", "S2T"]),
    // ... (other exams)
]

// Build the exam graph
exam_graph = build_exam_graph(exams)

// Define available time slots
time_slots = ["Red", "Yellow", "Orange", "Green", "Blue", "Purple", "Light Gray", "Brown", "Pink", "Dark Red"]

// Initialize assigned colors
assigned_colors = {exam: None for exam in exams}

// Call the graph coloring function to schedule exams
schedule_exams(exam_graph, exams, 0, time_slots, assigned_colors)

// Print the exam schedule
print_exam_schedule(exams, [assigned_colors[exam] for exam in exams])
```

The output of the algorithm is shown below in Figure 4.1:-

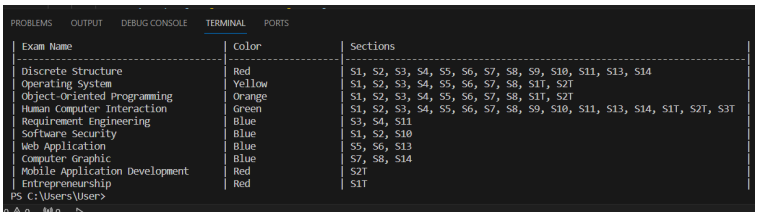

Figure 4.1: The Output of Depth First Search Algorithm

D. Graph Coloring

The graph coloring method is used to determine the order in which exams are chosen. Each exam is represented by a distinct vertex in the graph coloring technique, and the exam conflict is represented by the edges connecting the vertices [5], [6]. The procedure of coloring a graph involves giving each vertex a distinct color such that two neighboring vertices will have different colors. Each color corresponds to a particular exam period [7], [5].

Finding the least amount of color to apply to a graph's vertices in order to prevent two neighboring vertices from having the same color is the goal of graph coloring [8]. Based on the output from the Depth First Search algorithm, it shows that there are 4 colors used to show the subjects that have conflict with one another.

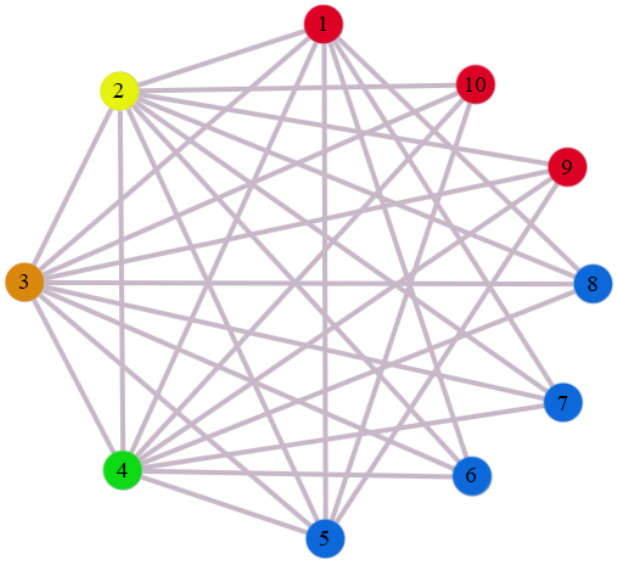

Figure 4.2: Graph Coloring Approach

Figure 4.2: is a coloured graph with 10 vertices representing each exam. The clash instances between exams is depicted by the edges between vertices. The use of different colors represents different time slots. In Figure 4.2, the time slots are conveyed as the colors red, yellow, orange, green, and blue.

V. Result And Discussion

In this study, data were collected on trimester two 2022/2023 courses in the Faculty of Computer Science and Technology of the University Tun Hussein Onn Malaysia. The data were collected from the subjects for each course, which are Information Security (BIS), Software Engineering (BIP), Web Technology (BIW), Multimedia Computing (BIM), and Information Technology (BIT). In the system, for this semester, subject timetables are generated without conflicts, and courses with high credit units do not run simultaneously to avoid a high workload on the students. The variables that were used during implementation include courses' names, numbers, sections, and subjects. The implementation was carried out in phases. At this phase, each subject was categorized by a different color.

In this study, all the 10 subjects will be grouped into their courses such as BIP and BIM which do not have the same characteristics. The entire subject will be grouped into three different clusters based on their courses and intake. The cluster represents colors such as red and yellow. For example subject discrete structure, intake trimester 2 can represent as color = red. Decomposition subjects will help research in reducing the problem size and it is very useful for determining the conflict matrix between subjects. Based on decomposition, researchers can define whether the subject can slot or assign in the same slot or not in the exam timetable.

After decomposition of subjects, conflict between subjects will be determined by using the conflict matrix table. For this problem, the maximum number of conflicts is 9 for subject T2, T3, and T4. Based on the conflict matrix, it shows that five colors are used in the graph coloring for this exam problem. These five colors represent five slots that should be used for this problem.

Exam slots (colors) are now arranged based on the number of students enrolled in each course after subjects were grouped according to a particular color for scheduling. The exam that has the most students enrolled in it should be scheduled first. The most important enrollment graph coloring heuristic is the constructive heuristic used by this research that ended up at an initial solution. The exam with the highest enrollment is assigned to the first available slot by this process. The exam is scheduled for the next available session if a slot is not available. If a slot fulfills the soft constraint stating that a student cannot take more than one exam on a single day, then it is acceptable. This heuristic determines the period with the least amount of consequences by evaluating the possible penalty for assigning an exam to each period.

A. Period Selection Using 5 Slots

The exam will be scheduled according to the number of students enrolled in the course. The algorithm will verify that a student is not permitted to take two tests in the same period or slot before allocating the exam. If it is not practical, it will advance to the following position until it reaches slot five, which is the last slot. After that, it will restart from a vacant slot, and this time, the exam penalty

assigned to each period will be computed and the period with the lowest penalty will be chosen.

As a result, using five slots, only 5 out of 10 subjects will violate the soft constraint. With five slots, all subjects that have conflict matrix will violate the soft constraint because the number of slots available is equal to the number of conflict matrices.

This period selection indicates that, in contrast to the original exam schedule, which used ten slots, the trimester 2 exam schedule for 2023–2024 used just five slots. By using this strategy, researchers are able to save five slots, which also saves resources like the invigilator and room.

B. Period selection using 10 slots (without penalty)

In this section, researchers will use the entire 10 slots provided by the management. In the current exam timetable, the university has allocated 10 slots for FSKTM students to sit for their exam in trimester 2, 2023/2024. All the process is the same as period selection using five slots and the only difference is the slot will be increased from five to 10 slots.

Experiments show that the second method would not have any penalty since all the soft constraints are not violated. Each subject would have their own slots and there would be no two exams happening at the same time.

C. Room Selection

A selection heuristic will be used to distribute students across the room after the examination timetables for each subject have been created. The challenge of placing students in rooms is similar to the knapsack filling issue, in which researchers must arrange a series of examinations into a range of rooms. In order to optimize the usage of each space, the goal is to accommodate as many students as feasible [13], [14], [15]. In order to solve this issue, a sorted list of subjects for certain periods of time will fit in the room according to:-

- Largest First – In order to maximize the use of available space and reduce the number of places and instructors, the largest spaces will be fitted first.
- Best Fit – Exams will be conducted in the least amount of room available in the room.

In order to allocate each examination to a room, the larger rooms are filled first, followed by those with the fewest amount of capacity available [15]. A significant issue with room selection is that too many students cannot be accommodated in one location or that many tests are scheduled in the same room at the same time [13], [14], [15]. The room with the lowest penalty is chosen in this room selection process, which calculates the penalty of fitting the identical topics in a different room.

CONCLUSION

This case study has provided that implementing a Depth First Search in Python language is an effective way to do graph coloring. The scheduling system was implemented in stages, with one particularly important phase utilizing the Depth First Search (DFS) method for graph coloring, namely in categorizing each subject with a distinct color. This deliberate use of DFS in the context of map coloring ensured that courses with possible conflicts were planned in such a way that overlaps were avoided and student discomfort was minimized. The study helps to organize academic calendars more efficiently by taking into consideration characteristics such as course titles, numbers, sections, and subjects for a more streamlined and student-friendly educational experience.

REFERENCES

[1] Akbulut, Akhan & Yilmaz, Guray. (2013). University Exam Scheduling System Using Graph Coloring Algorithm and RFID Technology. International Journal of Innovation, Management and Technology. 4. 66-72. 10.7763/IJIMT.2013.V4.359.
https://www.researchgate.net/publication/322131479_University_Exam_Scheduling_System_Using_Graph_Coloring_Algorithm_and_RFID_Technology

[2] Datta, Debabrata. (2022). Examination Scheduler Using A Linear-Time Graph Coloring Algorithm. 10.21917/ijsc.2022.0372.
https://ictactjournals.in/paper/IJSC_Vol_12_Iss_4_Paper_2_2678_2684.pdf

[3] Gupta, Sumit & Singh,. (2020). Greedy Graph Coloring Algorithm Based on Depth First Search. https://www.researchgate.net/publication/341371807_Greedy_Graph_Coloring_Algorithm_Based_on_Depth_First_Search

[4] E. K. Burke, D. Elliman, P. H. Ford and R. F. Weare, "Examination timetabling in British Universities: A survey," in The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference (Lecture Notes in Computer Science 1153), E. Burke and P. Ross, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 79–90.

[5] P. Cowling, S. Ahmadi, P. Cheng, R. Barone, "Combining Human and Machine Intelligence to Produce Effective Examination Timetables," The 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, 2002, pp. 662-666.

[6] M.Malkawi, M. A. Hassan and O. A. Hassan, " A New Exam Scheduling Algorithm Using Graph Coloring," The International Arab Journal of Information Technology, 2008, pp. 80-87.

[7] E. K. Burke, B. Mccollum, A. Meisels, S. Petrovic and R. Qu, "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems," European Journal of Operational Research 176, 2007, pp. 177-192.

[8] S. A. Rahman, A.Bargiela, E. K. Burke, E. Ozcan and B. McCollum, "Construction of Examination Timetables Based on Ordering Heuristics," in 24th International Symposium on Computer and Information Sciences , 2009 , pp. 680-685.

[9] The Graph Coloring. (n.d.). https://www.tutorialspoint.com/the-graph-coloring

[10] ZHANG, X., LIAO, P., & GUO, B. (2010, April 7). Depth-first search algorithm for mining frequent closed itemsets. Journal of Computer Applications, 30(3), 806–809. https://doi.org/10.3724/sp.j.1087.2010.00806

[11] Potdar, S. (2023, September 4). Map Coloring - Shantanu Potdar - Medium. Medium. https://medium.com/@co.2020.sppotdar/map-coloring-a677db08d632#:~:text=In%20the%20case%20of%20Map,regions%20share%20the%20same%20color.&text=1.,from%20a%20domain%20of%20colors

[12] Depth First Search (DFS) Algorithm. (n.d.). https://www.programiz.com/dsa/graph-dfs

[13] P. Cowling, S. Ahmadi, P. Cheng, R. Barone, "Combining Human and Machine Intelligence to Produce Effective Examination Timetables," The 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, 2002, pp. 662-666.

[14] M.Malkawi, M. A. Hassan and O. A. Hassan, " A New Exam Scheduling Algorithm Using Graph Coloring," The International Arab Journal of Information Technology, 2008, pp. 80-87.

[15] S. A. Rahman, A.Bargiela, E. K. Burke, E. Ozcan and B. McCollum, "Construction of Examination Timetables Based on Ordering Heuristics," in 24th International Symposium on Computer and Information Sciences , 2009 , pp. 680-685.