Exam Scheduling Using Greedy Graph Coloring Algorithm

Xingrou Mei
Dept. of Computer Information Science
Fordham Univeristy
New York, NY
xmei4@fordham.edu

***Abstract*—Every semester, there will be different students taking different courses. The conditions can also be different every year depending on the environment, faculties, and students. This is a concern that education institutions must solve. Exam schedules can be done manually, but they can be time-consuming and complicated. Imagine if a university has thousands of students, it will not be efficient. When it comes to making an exam timetable, the main objective is to avoid scheduling conflicts. This paper focuses on creating an exam schedule for school or university using a greedy graph coloring algorithm under some specific conditions.**

Keywords*—*Greedy Algorithm, Graph Coloring, Scheduling

I. INTRODUCTION

One of the first fields of graph theory is graph coloring. In 1852, Augustus de Morgan wrote a letter to his friend, William Hamilton, about if it was feasible to paint the districts of any map with four colors so that neighboring areas would have different colors. Arthur Cayley presented the four-color problem to the London Mathematical Society in 1878, bringing it to the attention of the scientific world. Although it was shown that five color areas are always sufficient, it took until 1977 for a widely acknowledged solution to the four-color problem to be published [5].

There are two types of graph coloring, vertex-coloring, and edge-coloring [5]. This paper focuses on vertex-coloring. Vertex-coloring assigns colors to a graph so that there are no two adjacent vertices have the same color. In this paper, we use an undirected graph. An undirected graph contains unordered pairs of (V, E). V stands for a set of vertices, and E stands for a set of non-directed edges between the vertices [6].

II. EXPERIMENTS

A. Dataset

The dataset for this paper is from Kaggle. It contains two hundred and fifty students, five majors and minors, and forty courses [1]. Student names are on the row index and course names are on the column index. The dataset has Boolean values of True or False. If True, students take that course. If False, otherwise. Each student has five True values, which indicates that students take five courses in total. Students with a minor take two courses from the minor-related courses and three courses from the major.

- Majors/Minors

  Biology, Computer Science, Environmental Science, Math, and Physics.

- Courses

'Biology of the Cell', 'Molecular Biology', 'Evolution', 'Biochemistry', 'Neurobiology', 'Animal Behavior', 'Genetics', 'Bioinformatics', 'Quantum Mechanics', 'Thermodynamics', 'Classical Mechanics', 'Programming for Physics', 'Linear Algebra for the Sciences', 'Complex Systems', 'Material Science', 'Nanotechnologies', 'Robotics', 'Calculus I', 'Calculus II', 'Probability I', 'Probability II', 'Statistics I', 'Statistics II', 'Linear Algebra', 'Geometry', 'Programming for Mathematics', 'Programming Introduction', 'Algorithms', 'Software Engineering', 'Programming in C++', 'Numerical Methods', 'Data Science', 'Machine Learning', 'Artificial Intelligence', 'Ecology', 'Chemical Geology', 'Physical Geology', 'Glaciology', 'Tectonics', 'Weather Systems'

The columns with majors and minors are removed from the dataset. Below is what the dataset looks like before continuing to the next step.

	Biology of the Cell	Molecular Biology	Evolution	Biochemistry	Neurobiology	Animal Behavior
VanessaHarris	False	False	False	False	False	False
JamesToliver	False	False	False	False	False	False
CarolTyer	False	False	False	False	False	False
BrookeMasters	False	False	False	False	False	False
MariaCope	False	False	False	False	False	False

Fig. 1. Example of the dataset

B. Methods

1) NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks [2]. This paper uses NetworkX to create graphs to show which courses can or cannot be taken together at the same time.

A graph is made of vertices and edges. In this case, all the courses are the vertices, and they are connected only when at least one student is taking both courses. For example, if one student is taking Biology of the Cell, Molecular Biology, Biochemistry, Genetics, Evolution, and Animal Behavior, then all these five courses will be connected. The number of edges is the combination of all the courses.

2) *Algorithms*

Graph coloring is an algorithm to color the vertices of an undirected graph so that its adjacent vertices have a different color. This algorithm does not necessarily need to be written in programming codes, and it can be done manually. However, if there are thousands or millions of vertices, then it will be useful to code it. The goal of this paper is to find the smallest possible color. A greedy algorithm will be the solution to this problem. Unfortunately, it does not guarantee the usage of minimum colors, but it does guarantee an upper bound on the number of colors. The algorithm never requires more than d+1 colors, where d is the highest degree of a vertex [4]. The degree of every vertex in a graph is equivalent to the number of edges which has that vertex [5].

Below is the pseudocode of the algorithm. The worst time complexity is at least O(V^2 + E) [4]. V as the vertex and E as the edge.

Algorithm 1: greedy algorithm, fill the colors
Input:
    [] = a list of colors
    [] = a list of courses
Output:
    all adjacent vertices have different color
function greedy_algorithm(graph, color)
  #there are three options
  1 = start with the first item
  2 = start with the last item
  3 = randomized the order
  for each node **do**
    [] create a list for the adjacent nodes
    [] create empty list to store used colors
    for each adjacent node **do**
      if adjacent has a color **then**
        add color to the list
      else
        continue
    for each color **do**
      if color is in the list **then**
        continue
      else
        set color to the node
        end

III. Result

As mentioned before, all forty courses are the vertices of the graph, and two vertices are connected only if one or more students take both courses. It is calculated using combinations of courses for each student. The total number of edges is two hundred and seventy-three. Below is what the graph looks like after applying all the vertices and edges from the dataset.

A. *Graphs after applying greedy algorithms*

For the greedy algorithm, a list of courses needs to create before running the algorithm. This list determines which vertex starts to color first. Therefore, this paper will run three different orders to verify whether the order matters.

First, the original list starts with the course called "Biology of the Cell." Most of the students take Biology of the Cell as a biology major. In this list, it seems like all the courses from the same major are next to each other. Based on **Fig. 2,** courses from the same major are mostly adjacent to each other. Second, this list is a reverse version of the first list, and it starts with a course called "Weather System." Lastly, this list is randomized. The start vertex is different every time and the order of the list is random. This means that it can start with one vertex and the next one can be somewhere far away and not adjacent to each other.

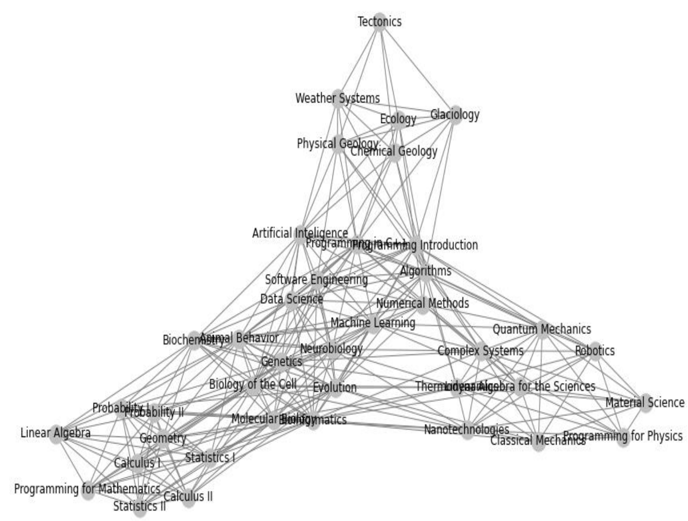

Fig. 2. Example of the graph (before applying colors)

1) *Original Order*

Using this order of the list, it uses eleven colors to fill the graph. In other words, it requires eleven time periods to complete all the exams.

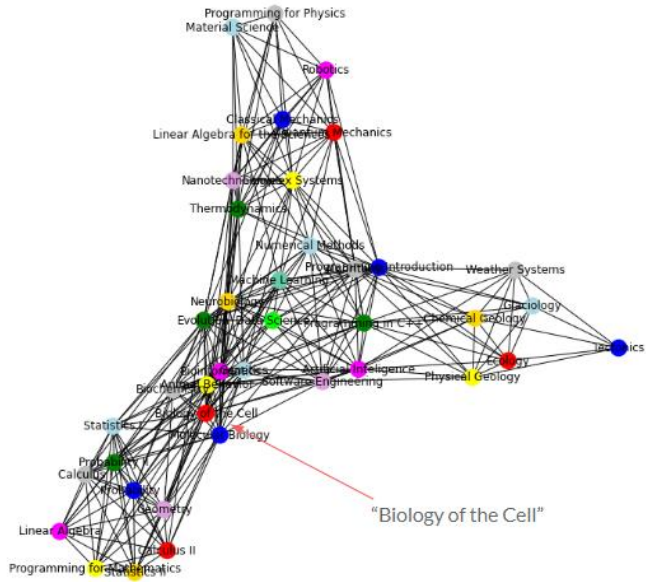

Fig. 3. Example of the colored graph *(Original order)*

2) Reverse Order

If the list is reversed, it uses thirteen colors to fill the graph. This also means that it needs thirteen time periods to complete all the exams.

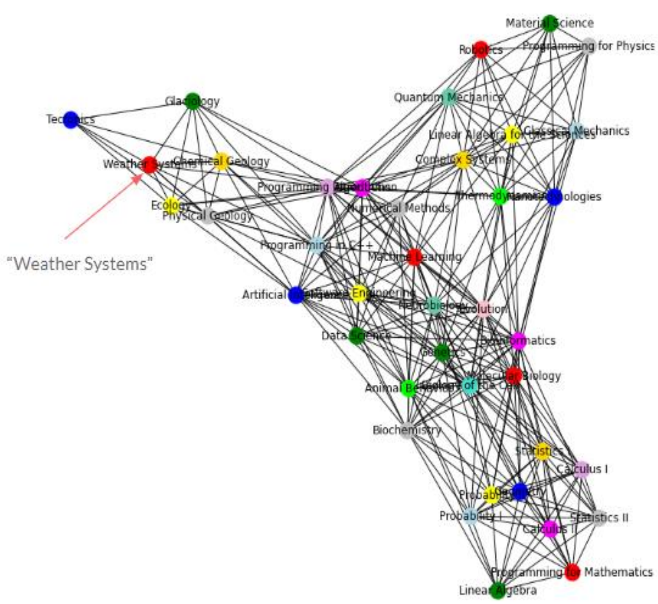

Fig. 4. Example of the colored graph *(Reverse order)*

3) Random Order

Since the order is randomized, it requires running multiple times to double-check whether it will return a similar result. Nine out of ten times, it returns eleven colors. One out of ten times, it returns the twelve colors.

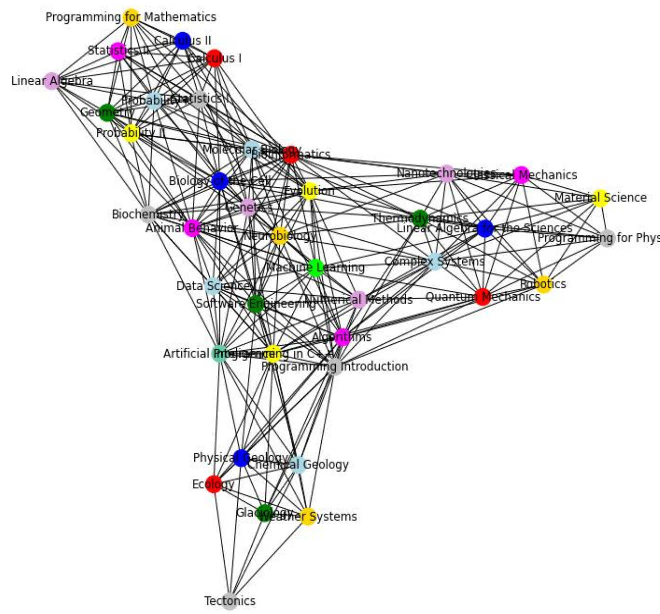

Fig. 5. Example of the colored graph *(Random order)*

B. Timetable

Before creating the exam schedule, it needs to build some conditions to mimic a real-life situation. The first exam time starts at 9:00 am, and the last exam starts at 6:00 pm. Each exam is about two hours long, and there is an hour break in between exams. In this case, the time frame is from 9:00 am to 8:00 pm, and there are four exams a day.

TABLE I.

Time	
9:00am	Exam 1
11:00am	Break
12:00pm	Exam 2
2:00pm	Break
3:00pm	Exam 3
5:00pm	Break
6:00pm	Exam 4
8:00pm	

1) Original Order

Based on **Fig. 6**, clearly shows that it takes about three days to finish scheduling all the exams.

	Room 1	Room 2	Room 3	Room 4	Room 5
2022-05-04 09:00:00	Biology of the Cell	Quantum Mechanics	Calculus II	Ecology	None
2022-05-04 12:00:00	Molecular Biology	Classical Mechanics	Probability I	Programming Introduction	Tectonics
2022-05-04 15:00:00	Evolution	Thermodynamics	Probability II	Programming in C++	None
2022-05-04 18:00:00	Biochemistry	Programming for Physics	Calculus I	Algorithms	Weather Systems
2022-05-05 09:00:00	Neurobiology	Linear Algebra for the Sciences	Statistics II	Chemical Geology	None
2022-05-05 12:00:00	Animal Behavior	Complex Systems	Programming for Mathematics	Physical Geology	None
2022-05-05 15:00:00	Genetics	Material Science	Statistics I	Numerical Methods	Glaciology
2022-05-05 18:00:00	Bioinformatics	Robotics	Linear Algebra	Artificial Inteligence	None
2022-05-06 09:00:00	Nanotechnologies	Geometry	Software Engineering	None	None
2022-05-06 12:00:00	Data Science	None	None	None	None
2022-05-06 15:00:00	Machine Learning	None	None	None	None
2022-05-06 18:00:00	None	None	None	None	None

Fig. 6. Schedule *(Original order)*

2) Reverse Order

In **Fig. 7**, the schedule for the reverse order takes about four days to complete all the exams.

	Room 1	Room 2	Room 3	Room 4	Room 5
2022-05-04 09:00:00	Molecular Biology	Robotics	Programming for Mathematics	Machine Learning	Weather Systems
2022-05-04 12:00:00	Nanotechnologies	Geometry	Artificial Inteligence	Tectonics	None
2022-05-04 15:00:00	Genetics	Material Science	Linear Algebra	Data Science	Glaciology
2022-05-04 18:00:00	Biochemistry	Programming for Physics	Statistics II	Numerical Methods	Physical Geology
2022-05-05 09:00:00	Complex Systems	Statistics I	Chemical Geology	None	None
2022-05-05 12:00:00	Linear Algebra for the Sciences	Probability II	Software Engineering	Ecology	None
2022-05-05 15:00:00	Classical Mechanics	Probability I	Programming in C++	None	None
2022-05-05 18:00:00	Bioinformatics	Calculus II	Algorithms	None	None
2022-05-06 09:00:00	Calculus I	Programming Introduction	None	None	None
2022-05-06 12:00:00	Animal Behavior	Thermodynamics	None	None	None
2022-05-06 15:00:00	Neurobiology	Quantum Mechanics	None	None	None
2022-05-06 18:00:00	Evolution	None	None	None	None
2022-05-07 09:00:00	Biology of the Cell	None	None	None	None
2022-05-07 12:00:00	None	None	None	None	None
2022-05-07 15:00:00	None	None	None	None	None
2022-05-07 18:00:00	None	None	None	None	None

Fig. 7. Schedule *(Reverse order)*

3) Random Order

Fig. 8 is generated using the eleven-color example of the random order. This schedule takes about three days to finish all the exams.

	Room 1	Room 2	Room 3	Room 4	Room 5
2022-05-04 09:00:00	Bioinformatics	Quantum Mechanics	Calculus I	Ecology	None
2022-05-04 12:00:00	Biology of the Cell	Linear Algebra for the Sciences	Calculus II	Physical Geology	None
2022-05-04 15:00:00	Thermodynamics	Geometry	Software Engineering	Glaciology	None
2022-05-04 18:00:00	Biochemistry	Programming for Physics	Statistics I	Programming Introduction	Tectonics
2022-05-05 09:00:00	Neurobiology	Robotics	Programming for Mathematics	Weather Systems	None
2022-05-05 12:00:00	Evolution	Material Science	Probability II	Programming in C++	None
2022-05-05 15:00:00	Molecular Biology	Complex Systems	Probability I	Data Science	Chemical Geology
2022-05-05 18:00:00	Animal Behavior	Classical Mechanics	Statistics II	Algorithms	None
2022-05-06 09:00:00	Genetics	Nanotechnologies	Linear Algebra	Numerical Methods	None
2022-05-06 12:00:00	Machine Learning	None	None	None	None
2022-05-06 15:00:00	Artificial Inteligence	None	None	None	None
2022-05-06 18:00:00	None	None	None	None	None

Fig. 8. Schedule *(Random order)*

According to the schedules, five exams can simultaneously take in five different rooms. The reason is that the max number of courses that share the same color is five. Courses that are connected cannot share the same color in the graph coloring algorithm. For this reason, courses with the same color can take it concurrently.

Below is a sample of the colors that the courses share from the original order.

```
Color : lime, Course: Data Science
Color : blue, Course: Molecular Biology
Color : blue, Course: Classical Mechanics
Color : blue, Course: Probability I
Color : blue, Course: Programming Introduction
Color : blue, Course: Tectonics
Color : lightblue, Course: Genetics
Color : lightblue, Course: Material Science
Color : lightblue, Course: Statistics I
Color : lightblue, Course: Numerical Methods
Color : lightblue, Course: Glaciology
```

C. Problems

Creating a schedule that can complete all the exams as soon as possible can be beneficial under a time limit. However, there is a problem with this schedule. Since the timetable is so packed that some students must take four exams a day back-to-back, this will not be ideal for students. Based on the original list, eighty students must take more than two exams from day one, seventy-six students from day two, and nineteen students from day three. A total of one hundred and seventy-five students must take more than two exams a day.

The second problem is from the dataset. Some students have to take Calculus I and Calculus II or Probability I and Probability II together in a semester.

IV. Related Works

There are many studies on exam scheduling and graph coloring algorithms. In 1967, Welsh and Powell mentioned a connection between the scheduling problem with the graph coloring approach such that no two adjacent nodes have the same color and find the minimum number of colors [8]. Another study is focusing on the average process speed and the efficiency of satisfaction for all constraints using different approaches. These approaches are blind search, brute force, map coloring, and manually solving the problem. The results show that all the algorithms satisfy the constraints 100% with a similar speed, but the manual schedule only satisfies 87.5% with the longest time [7].

V. Conclusion

The purpose of this paper is to schedule all the examinations in the shortest time possible. Depending on the conditions, the minimal number may be varied. Other constraints may include, for instance, the capacity limit of a room that can only fit thirty students, lack of rooms available for students at the time, and/or each student can only take two exams a day. In later work, solving the problems in this paper will achieve an ideal environment for students and find an optimal solution for any constraints at any time

References

[1] Y. Boutellier. “Synthetic School Enrollment Data.” Kaggle.com https://www.kaggle.com/datasets/yvesboutellier/synthetic-school-enrollment-data

[2] Networkx. “Overview – NetworkX v1.1 documentation.” Networkx.org https://networkx.org/documentation/networkx-1.1/index.html

[3] Cornell University (2020). A greedy graph-coloring algorithm [Online]. Available: https://www.cs.cornell.edu/courses/JavaAndDS/files/graphColoring.pdf

[4] GeeksforGeeks. “Graph Coloring | Set 2 (Greedy Algorithm).” GeeksforGeeks.org https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/

[5] S.N. Bharathi. “A Study on Graph Coloring.” in Int. J. of Scientific & Eng. Res., vol. 8, issue 5, May 2017, pp. 20-30.

[6] M. Malkawi, M. Al-Haj. Hassan, and O. Al-Haj. Hassan. “A New Exam Scheduling Algorithm Using Graph Coloring.” in The Int. Arab J. of Inf. Technol., vol. 5, no. 1, Jan. 2008, pp. 80-87.

[7] A.Akbulut and G. Yilmaz. “University Exam Scheduling System Using Graph Coloring Algorithm and RFID Technology.” in Int. J. of Innov., Manage. and Technol., vol. 4, no. 1, Feb. 2013, pp. 66-72.

[8] D. J. A. Welsh and M. B. Powell “An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems,” Comp. J., 1967.

[9] Y. Boutellier. “Graph Coloring with networkx.” Towardsdatascience.com https://towardsdatascience.com/graph-coloring-with-networkx-88c45f09b8f4