International Journal of Mathematics Trends and Technology
ISSN: 2231 – 5373/doi:10.14445/22315373/IJMTT-V68I1P507
Volume 68 Issue 1, 61-69, January, 2022

Original Article

Course Scheduling of Undergraduate Study Program of Mathematics Universitas Padjadjaran Case Using Graph Coloring with Modified Algorithm

Ugi Abdul Muchyi Sunanto[1], Herlina Napitupulu[2], Ema Carnia[3], Asep Kuswandi Supriatna[4]

[1]*Undergraduate Student,* [2,3,4]*Academic Faculty*
[1,2,3,4]*Departement of Mathematics, Universitas Padjadjaran, Bandung, Indonesia*

Abstract *- Course scheduling problems is the most routine problems faced by academic institutions in every new semester. Scheduling is done by taking into resources such as students, lecturers, courses, and rooms whose purpose is to avoid conflicts by satisfying various preferential constraints. The presence of the various resources leads to difficulty in generating a schedule in a limited period. Room is assumed as zoom meeting accounts so that they are not limited by capacity. Graph coloring is one decent method to deal with a timetable scheduling problem. In this work, graph vertex coloring is applied for generating the schedule from the given data of Undergraduate Study Program of Mathematics, Universitas Padjadjaran. The scheduling problem is divided into a common case and a special case which has additional constraints on the special case. The scheduling problem is solved by the combination and modification of Bania-Duarah and Greedy algorithm. The workflow is initiated by constructing a graph and its matrix adjacency then the workflow completed by Greedy algorithm on the coloring phase. The main results from this work are a modified Python program for scheduling problems and the schedule of both cases. This work provides an alternative solution to the scheduling problem by using the concept of graph theory applying graph coloring for similar cases.*

Keywords — *Scheduling, Timetable, Graph Coloring, Greedy, Bania-Duarah.*

I. INTRODUCTION

Timetable scheduling concerns every educational institution. The particular instance we are dealing with here is the university course timetabling problem. In every semester, universities need to make a schedule for teaching and learning activities. The presence of various constraints such as a large number of students, number of courses held, and limited period of time, resulting the possibility of colliding course and difficult to schedule without any conflict. If the schedule is generated manually, it will take a lot of time and energy which leads to inefficiency. Scheduling theory problems usually give a large interval of NP-hard combinatorial optimization problems being widely used ([1]). Therefore, an appropriate method is needed to work on scheduling more efficient.

Many research conducted in solving scheduling problems. Several variety techniques are developed and used such as graph theory algorithms ([2]). Genetic Algorithm and its development ([3]), Particle Swarm Optimization ([4]), Ant Colony Optimization ([5,6]), Firefly Algorithm ([7]), Bat Algorithm ([8]), Cuckoo Search ([9]), Tabu Search Algorithm ([10]), a hybrid whale optimization algorithm ([11]) and more. In spesific, the optimal partitioning of mutually exclusive events, usually can be solved by of graph coloring, such as in course or examination timetable scheduling ([12,13]).

Graph Coloring is a method of assigning color to certain graph elements by meeting certain constraints. Thus, the optimal solution to a problem such as scheduling can be found by determining the minimal color (chromatic number) of the corresponding graph ([14]). This work examines the determination of the minimum color in the related graph. Thus, in this research, the scheduling problem is solved using two methods by coloring the graph vertices. The methods namely the Bania-Duarah Algorithm ([15]) and the Greedy Algorithm, where the two algorithms have different processing steps. The Bania-Duarah Algorithm starts by forming a graph and an adjacency matrix. Secondly, assigning the color. Greedy Algorithm starts by considering the vertices of the graph as a sequence and filling each vertex with the first available color. The two algorithms were chosen to be modified using the Python programming language and the final results obtained are chromatic numbers and class schedules. Before going into further discussion, some of the materials used in this study will be discussed.

II. GRAPH, COLORING, AND TIMETABLE PROBLEM

Solving timetabling problems through the application of modern technology has a long and varied history. In 1967, Welsh and Powell ([16]) illustrated the relationship between timetabling and graph coloring, and developed a new general graph coloring algorithm to solve the minimum coloring problem more efficiently. In 1994, university timetabling system based on

graph coloring and constraint manipulation is introduced by [17]. In 2008, Malkawi et al [18] proposed an algorithm to solve exam time table schedule problem which consists two major steps. The first step, generate an undirected graph and a weight matrix, and the last step assign the color to different vertices of that undirected graph. In 2018, Bania and Duarah improved algorithm from [18] by adding sorting color value using standard merge sort algorithm to reduce computation time in designing the exam timetable.

In the following section we will studied some theoretical background of Graph, Graph Coloring and timetable problem and solutions. The further studied can be refer to [19-25].

A. Graph and Graph Coloring ([26])

A graph G consists of a set of objects $V = \{v_1, v_2, v_3, \ldots\}$ called vertices (also called points or nodes) and other set $E = \{e_1, e_2, e_3, \ldots\ldots\}$ whose elements are called edges (also called lines or arcs). The set $V(G)$ is called the vertex set of G and $E(G)$ is the edge set. Usually, the graph is denoted as $G = (V, E)$.

Minimum number of colors that you require to assign colors to graph G with n vertices such that no two adjacent vertices have the same color is called coloring problem.

Chromatic number of a graph is the smallest number of colors with which the graph can be properly colored. The chromatic number of a graph G is usually denoted by $\chi(G)$ ([26]).

B. Timetable Problem and Solution's Algorithms

A timetable is a placement of a set of meetings in time. A meeting is a combination of resources (e.g. rooms, people, and items of equipment), some of which may be specified by the problem, and some of which must be allocated as part of the solution ([18]).

Greedy Algorithm arranges the existing vertices with certain rules v_1, ..., v_n and fills v_i with available colors that are not used by all neighbors v_i, between v_1, ..., v_{i-1}. If needed, a new color can be added to the vertices being read.

Bania and Duarah Algorithm is an algorithm to find optimal solution of timetabling problem at university level. The algorithm consists of two phases, the first phase is graph forming and the second phase is graph coloring.

III. METHODOLOGY

A simple graph model is made for the problem of courses scheduling in Undergraduate Study Program of Mathematics, Universitas Padjadjaran. Let G be the graph for this scheduling problem. The vertices in graph G represent the courses and the edges represent the relationship between two vertices, for example a student takes two different courses and a lecturer teach two different courses. Two vertices can have a color if they are not neighbors. Two vertices that are not adjacent can be scheduled in the same time slot. Room is assumed to be a zoom account which is not limited by capacity.

There are several data that need to be input:

1. Courses for each semester.
2. Courses Type, compulsory courses and elective courses.
3. Lecturer's code.
4. Available slot time for each day and hour

The purpose of this work is to find the minimum color of the course graph (chromatic number) by using modified Bania-Duarah algorithm ([14]) and Greedy algorithm. Then look for the most optimal schedule where all schedules can be held without any conflict.

A. Modified Algorithm

Based on Bania-Duarah algorithm there are several phases to be done to obtain a proper schedule. Firstly, generate the course graph and the course matrix. Secondly, do the color assignment using Greedy Algorithm. Lastly, generate the schedule by sorting the subjects based on the assigned color value.

The data is input, then it is represented to the form of a graph using Python software by performing the first phase, the steps are as follows:

Step 1: Create an empty graph G

Step 2: Take the vertex V_i and find the adjacency to the vertex V_j to form an edge between them, which V represents the course

Step 3: To determine the adjacency of V_i, perform a searching base on the specified constraints

Step 3.1: If the constraint is met then add edges on V_i and V_j

Step 3.2: If the constraint is not met then edges are not added to V_i and V_j

Step 4: Repeat steps 2 and 3 so that all pairs of courses have been searched for neighbors.

After the graph is formed, form the adjacency matrix table with the following steps:

Step 1: create an empty two-dimensional array $n \times n$ $course\_matrix[n][n]$.

Step 2: To determine the value of the adjacency matrix of V_i, perform a searching in all the vertices present.

Step 2.1: If vertices V_i are adjacent to V_j, assign $course\_matrix[i][j] = 1$

Step 2.2: If vertices V_i are not adjacent to V_j assign $course\_matrix[i][j] = 0$

Step 3: Repeat the step 2 until all the pairs of courses have a value of '0' or '1'

Based on the course matrix the algorithm will detect colliding courses and then color the vertices which represent the courses. Two courses are said to collide with each other if they are adjacent. Adjacent courses cannot be colored with the same color. Vertex coloring with Greedy algorithm is performed based on the following steps:

Step 1: Form an empty array $g[n]$ where n is the number of classes to be held. Perform a searching on $course\_matrix[i][j]$ to list the neighbors of each vertex.

Step 2: Form an empty array $result[n]$ and assign the value of array $result[n]$ to '-1'. Array is used to store colors for each vertex.

Step 3: Assign a value of '0' to the $result[0]$ array which means that the first vertex color is '0'.

Step 4: Form an array $available[n]$ and assign the array value $available[n]$ with 'False', array is used to give the color availability status.

Step 5: For coloring the vertex, perform a search based on an array of $g[n]$ with due attention to the color of the neighboring vertex and the availability of colors

Step 6: Repeat step 5 until all courses have a color.

After each course has its own color, courses with the same color can be scheduled at the same time slot. To schedule courses that have the same color into the same time slot can be performed with the following steps:

Step 1: Form an empty two-dimensional array $p \times q$ $data[p][q]$, where p is the number of time slots and q is the number of rooms

Step 2: Form an empty array $used$, the array is $used$ for select the same color courses.

Step 3: To place a vertex in a time slot, run a search on V_i by color C_k

Step 3.1: If V_i is C_k then V_i is placed in time slot k and used array is filled with V_i

Step 3.2: If V_i is not colored C_k, the search continues until it finds V_i which is color C_k

Step 4: Repeat step 3 until all vertices fill the data array

Modified Algorithm is completed and the results of scheduling is the number of courses in each time slot and rooms that must be provided.

B. Case Study

In Undergraduate Study Program of Mathematics, Universitas Padjadjaran the courses held every semester are 29 courses. There are 46 classes with every compulsory course divided into two classes, class A and class B. The following is given lecture data in Table 1 and data time slot in Table 2.

Table 1. Courses Schedule for Even Semester Academic Year 2020/2021 Undergraduate Study Program of Mathematics, Universitas Padjadjaran

Semester	Class	Courses	Lecturer 1/ Lecturer 2	Semester	Class	Courses	Lecturer 1/ Lecturer 2
2	A	Mathematical Logic	12	4	B	Mathematical Statistics	18
2	B	Mathematical Logic	9	4	A	Special Functions	17/22
2	A	Geometry	25	4	B	Special Functions	10
2	B	Geometry	26	4	A	Complex Functions	23
2	A	Calculus 2	7/18	4	B	Complex Functions	7/25
2	B	Calculus 2	15	6	A	Real Analysis 2	5
2	A	Statistics Methods	28	6	B	Real Analysis 2	12
2	B	Statistics Methods	29	6	A	Combinatorial Analysis	14
2	A	Algorithms and Programming	30	6	B	Combinatorial Analysis	27

Semester	Class	Courses	Lecturer 1/ Lecturer 2	Semester	Class	Courses	Lecturer 1/ Lecturer 2
2	B	Algorithms and Programming	30	6	A	Entrepreneurship	19
2	A	Operation Research Introduction	2	6	B	Entrepreneurship	26
2	B	Operation Research Introduction	8/ 34	6	A	Research Methods	20
2	A	Discrete Mathematics	5/ 27	6	B	Research Methods	2
2	B	Discrete Mathematics	15	6	A	Mathematical Population	1/20
4	A	Forecasting Methods	8	6	A	Simulations and Models	2
4	B	Forecasting Methods	24	6	A	Nonlinear Programming	21
4	A	Database	31	6	A	Difference Equations	1
4	B	Database	32	6	A	Linear Algebra	33
4	A	Ordinary Differential Equations	13	6	A	Fractal Geometry	25/27
4	B	Ordinary Differential Equations	13	6	A	Survival Models	23
4	A	Algebra Structure	6	6	A	Multivariate Statistics	17
4	B	Algebra Structure	14	6	A	Investment Models and Asset Management	4
4	A	Mathematical Statistics	24	6	A	Time Series Analysis	3

Table 2. Available Time Slot

Time Slot	Day and Hour	Time Slot	Day and Hour
Slot 1	Monday, 7.30 – 10.00	Slot 8	Wednesday, 10.15 – 12.45
Slot 2	Monday, 10.15 – 12.45	Slot 9	Wednesday, 13.15 – 15.45
Slot 3	Monday, 13.15 – 15.45	Slot 10	Thursday, 7.30 – 10.00
Slot 4	Tuesday, 7.30 – 10.00	Slot 11	Thursday, 10.15 – 12.45
Slot 5	Tuesday, 10.15 – 12.45	Slot 12	Thursday, 13.15 – 15.45
Slot 6	Tuesday, 13.15 – 15.45	Slot 13	Friday, 7.30 – 10.00
Slot 7	Wednesday, 7.30 – 10.00	Slot 14	Friday, 13.15 – 15.45

In generating the schedule there are several constraints which are divided into 2 cases. The simple case subjected to the following constraints.

1. Compulsory courses in each semester in the same class, namely class A or class B.
2. Compulsory and elective courses in semester 6th.
3. Some lecturers who teaches 2 to 3 courses.

The special case subjected to the following additional constraints.

1. The 6th semester students take 4th semester course, namely Algebra Structure
2. The 4th semester students take 6th semester course, namely Entrepreneurship.

C. Modified Algorithm Implementation on Case Study

Based on the modified algorithm, the first stage is forming a graph from the data and constraints, to get a graph representation with the syntax in Figure 1 and Figure 2.

```
tabel = pd.read_csv(r'E:\Dokumen\Kuliah\Tingkat 4\Skripsi\Python\Tabel_MK.csv', delimiter = ',')
```

```
G = nx.Graph() #Graf Kosong
G
G.nodes(), G.edges()

for i in range(46): # constraint: Compulsory courses in class A and class B
    for j in range(i+1,46):
        if tabel.iloc[i][0] == tabel.iloc[j][0] and tabel.iloc[i][2] == tabel.iloc[j][2]:
            G.add_edge(tabel.iloc[i][5],tabel.iloc[j][5])
            print(tabel.iloc[i][5],tabel.iloc[j][5])

for i in range(46): # constraint: Compulsory and specialization courses in semester 6
    for j in range(i+1,46):
        if(tabel.iloc[i][0] == tabel.iloc[j][0]) and (tabel.iloc[i][4] != tabel.iloc[j][4]):
            G.add_edge(tabel.iloc[i][5],tabel.iloc[j][5])
            print(tabel.iloc[i][5],tabel.iloc[j][5])

for i in range(46): # constraint: Lecturer who teaches 2 to 3 courses
    for j in range(i+1,46):
        if (tabel.iloc[i][6] == tabel.iloc[j][7] or tabel.iloc[i][6] == tabel.iloc[j][6] or
            tabel.iloc[i][7] == tabel.iloc[j][6] or tabel.iloc[i][7] == tabel.iloc[j][7]):
            G.add_edge(tabel.iloc[i][5],tabel.iloc[j][5])
            print(tabel.iloc[i][5],tabel.iloc[j][5])
```

Fig 1. Syntax for Representation of Simple Case Graph

```
for i in range(46): # constraint: The 6th semester students take 4th semester course, Algebra Structure A
    if (tabel.iloc[i][0] == tabel.iloc[28][0] and tabel.iloc[i][2] == tabel.iloc[20][2]):
        G.add_edge(tabel.iloc[20][5],tabel.iloc[i][5])
        print(tabel.iloc[20][5],tabel.iloc[i][5])
    if (tabel.iloc[i][0] == tabel.iloc[28][0] and tabel.iloc[i][4] != tabel.iloc[20][4]):
        G.add_edge(tabel.iloc[20][5],tabel.iloc[i][5])
        print(tabel.iloc[20][5],tabel.iloc[i][5])
for i in range(46): # constraint: The 6th semester students take 4th semester course, Algebra Structure B
    if (tabel.iloc[i][0] == tabel.iloc[28][0] and tabel.iloc[i][2] == tabel.iloc[21][2]):
        G.add_edge(tabel.iloc[21][5],tabel.iloc[i][5])
        print(tabel.iloc[21][5],tabel.iloc[i][5])
    if (tabel.iloc[i][0] == tabel.iloc[28][0] and tabel.iloc[i][4] != tabel.iloc[21][4]):
        G.add_edge(tabel.iloc[21][5],tabel.iloc[i][5])
        print(tabel.iloc[21][5],tabel.iloc[i][5])


for i in range(46): # constraint: The 4th semester students take 6th semester course, Entrepreneurship B
    if (tabel.iloc[i][0] == tabel.iloc[14][0] and tabel.iloc[i][2] == tabel.iloc[20][2]):
        G.add_edge(tabel.iloc[32][5],tabel.iloc[i][5])
        print(tabel.iloc[32][5],tabel.iloc[i][5])
for i in range(46): # constraint: The 4th semester students take 6th semester course, Entrepreneurship B
    if (tabel.iloc[i][0] == tabel.iloc[14][0] and tabel.iloc[i][2] == tabel.iloc[21][2]):
        G.add_edge(tabel.iloc[33][5],tabel.iloc[i][5])
        print(tabel.iloc[33][5],tabel.iloc[i][5])
```

Fig 2. Syntax for Representation of Special Case Graph

After data is represented in the form of a graph, then an adjacency matrix is formed by converting the existing graph with the syntax in Figure 3.

```
course_matrix =  [[0 for i in range(46)] for j in range(46)]
```

```
M=nx.to_numpy_matrix(G)
for i in range(46):
     for j in range(46):
        course_matrix[i][j] = M[i,j]


mat=pd.DataFrame(course_matrix, columns = G.nodes, index = G.nodes)
pd.set_option('display.max_columns', 50)
```

Fig 3. Adjacency Matrix Syntax

After the adjacency matrix is formed, all vertices are colored with a greedy algorithm with the syntax in Figure 4.

```
def addEdge(adj, v, w):

    adj[v].append(w)
    adj[w].append(v)
    return adj
```

```
if __name__ == '__main__':

    g = [[] for i in range(46)]

for i in range(45): # i=0..45
      for j in range(i+1, 46):
        if(course_matrix[i][j]==1):
            g = addEdge(g,tabel.iloc[i][1],tabel.iloc[j][1])
```

```
result = [-1] * len(V)

result[0] = 0;

available = [False] * len(V)

for u in range(1, len(V)):
    for i in g[u]:
        if (result[i] != -1):
            available[result[i]] = True

    cr = 0
    while cr < len(V):
        if (available[cr] == False):
            break

        cr += 1

    result[u] = cr

    for i in g[u]:
        if (result[i] != -1):
            available[result[i]] = False
```

Fig 4. Greedy Algorithm Syntax

After all vertices are colored, the results of the courses schedule can be formed by sorting them based on the color values of each vertex with the syntax shown in Figure 5.

```
data = [["" for i in range(8)] for j in range(7)]
used=[]

for i in range(0,8):
    a = 0
    for k in range (0,46):
        if result[k] == i and V[k] not in used:
            data[a][i]=(V[k])
            used.append(V[k])
            a+=1

final = pd.DataFrame(data)
```

Fig 5. Scheduling Syntax

IV. RESULT AND DISCUSSION

The modified algorithm implementation on case study written above resulting the colored graph in Figure 6 and Figure 7, thus the resulting schedule for each case presented in Table 3 and 4 for simple case and special case respectively.

The schedule of simple case requires seven rooms and seven time slots and the schedule of special case requires eight rooms and nine time slots. In this work there are forty-six vertices that represent the number of courses or classes held. For each course that is in the same time slot means that it has same color and the number of rooms required is equal to the number color. The program can solve both cases properly, resulting eight and nine colors for each case. By solving the special case with constraint written above, the program seems possible to solve more realistic case with more complex constraints.

Table 3. Scheduling Result for Simple Case

	Time Slot 1	Time Slot 2	Time Slot 3	Time Slot 4	Time Slot 5	Time Slot 6	Time Slot 7	Time Slot 8
Room 1	Mathematical Logic A	Geometry A	Calculus 2 A	Statistics Methods A	Algorithms and Programming A	Operation Research Introduction A	Discrete Mathematics A	Time Series Analysis
Room 2	Mathematical Logic B	Geometry B	Calculus 2 B	Statistics Methods B	Operation Research Introduction B	Algorithms and Programming B	Discrete Mathematics B	Fractal Geometry
Room 3	Forecasting Methods A	Database A	Ordinary Differential Equations A	Algebra Structure A	Mathematical Statistics A	Special Functions A	Complex Functions A	
Room 4	Forecasting Methods B	Database B	Algebra Structure B	Ordinary Differential Equations B	Mathematical Statistics B	Special Functions B	Complex Functions B	
Room 5	Real Analysis 2 A	Combinatoria l Analysis A	Entrepreneurs hip A	Research Methods A	Mathematical Population	Nonlinear Programming	Simulations and Models	
Room 6	Combinatorial Analysis B	Real Analysis 2 B	Entrepreneurs hip B	Research Methods B	Survival Models	Investment Models and Asset Management	Multivariate Statistics	
Room 7					Linear Algebra	Difference Equations		

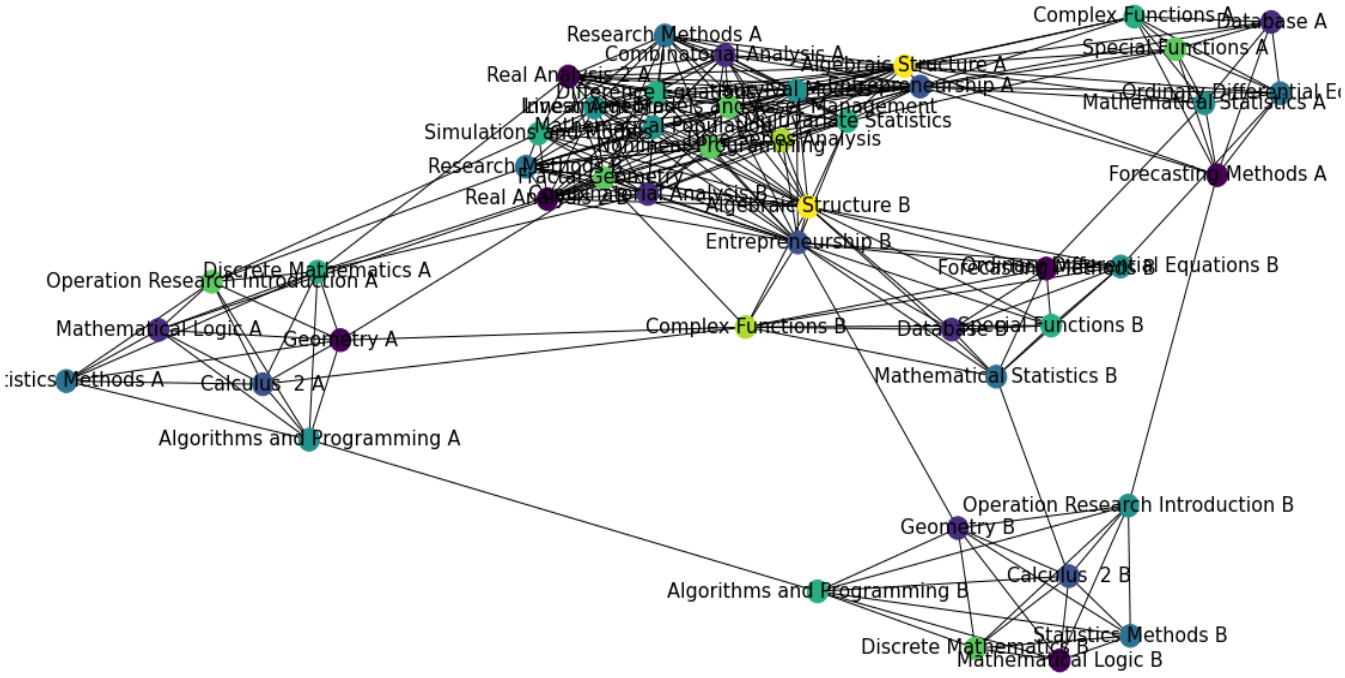

Fig 7. Graph with Special Case

Table 4. Scheduling Result for Special Case

	Time Slot 1	Time Slot 2	Time Slot 3	Time Slot 4	Time Slot 5	Time Slot 6	Time Slot 7	Time Slot 8	Time Slot 9
Room 1	Mathematical Logic A	Geometry A	Calculus 2 A	Statistics Methods A	Algorithms and Programming A	Operation Research Introduction A	Discrete Mathematics A	Entrepreneurship A	Time Series Analysis
Room 2	Mathematical Logic B	Geometry B	Calculus 2 B	Statistics Methods B	Operation Research Introduction B	Algorithms and Programming B	Discrete Mathematics B	Entrepreneurship B	Fractal Geometry
Room 3	Forecasting Methods A	Database A	Ordinary Differential Equations A	Algebra Structure A	Mathematical Statistics A	Special Functions A	Complex Functions A		
Room 4	Forecasting Methods B	Database B	Algebra Structure B	Ordinary Differential Equations B	Mathematical Statistics B	Special Functions B	Complex Functions B		
Room 5	Real Analysis 2 A	Combinatorial Analysis A	Research Methods A	Research Methods B	Mathematical Population	Nonlinear Programming	Simulations and Models		
Room 6	Combinatorial Analysis B	Real Analysis 2 B			Survival Models	Investment Models and Asset Management	Multivariate Statistics		
Room 7					Linear Algebra	Difference Equations			

V. CONCLUSION

In this work, the combination of Bania-Duarah and Greedy algorithm and modification were made to solve the case of scheduling courses in the Undergraduate Study Program of Mathematics, Universitas Padjadjaran. With this algorithm, the chromatic number obtained represents the number of time slots required, while the same number of colors in each color group represents the number of zoom accounts required in each time slot. There are two cases scheduling created in this study, namely the simple case and the special case (the problem with students repeating and taking an ahead semester course). The chromatic number for the simple case is 8 and a maximum of 7 zoom accounts are required. Whereas in special cases, the chromatic number obtained is 9 and a maximum of 7 zoom accounts are required. The modified algorithm succeeded in giving the results of scheduling courses that were subject to the given constraints without any conflicts.

ACKNOWLEDGEMENTS

The Author thanks Universitas Padjadjaran Indonesia that supported this work through Academic Leadership Grant research with Contract No: 1959/UN6.3.1/PT.00/2021.

REFERENCES

[1] Diveev, A. I., and O. V. Bobr, Variational genetic algorithm for np-hard scheduling problem solution, Procedia Computer Science 103 (2017) 52-58.
[2] F. Zhu, Z. Liu, Z. Yan and Y. Liang, A study into employee scheduling problem based on graph theory algorithms, 2018 IEEE Integrated STEM Education Conference (ISEC), (2018) 213-215, doi: 10.1109/ISECon.2018.8340484.
[3] F. A. Omara, M. M. Arafa, Genetic Algorithms for Task Scheduling Problem. In: Abraham A., Hassanien AE., Siarry P., Engelbrecht A. (eds) Foundations of Computational Intelligence Volume 3. Studies in Computational Intelligence, vol 203. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01085-9_16
[4] J. Kennedy & R. Eberhart, Particle swarm optimization, In Proceedings of ICNN'95-international conference on neural networks (4) (1995) 1942–1948, https://doi.org/10.1109/ICNN.1995.488968
[5] M. Dorigo, M. Birattari, & T. Stutzle, Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4) (2006) 28–39. https://doi.org/10.1109/CI-M.2006.248054
[6] W. Deng, J. Xu and H. Zhao, An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem, in IEEE Access, (7) (2019) 20281-20292 doi: 10.1109/ACCESS.2019.2897580.
[7] X. S. Yang & S. Deb, Cuckoo search via Lévy flights. In 2009 World Congress on nature & biologically inspired computing (NaBIC) (2009) 210–214, IEEE.
[8] X. S. Yang, A new metaheuristic bat-inspired algorithm. In Proceedings of nature inspired cooperative strategies for optimization (NICSO 2010) (2010) 65–74, Springer.
[9] X. S. Yang & S. Deb, Cuckoo search via Lévy flights. In 2009 World Congress on nature & biologically inspired computing (NaBIC) (2009) 210–214). IEEE.
[10] M. Dell'Amico, M. Trubian, Applying tabu search to the job-shop scheduling problem. Ann Oper Res (41) (1993) 231–252 https://doi.org/10.1007/BF02023076
[11] Abdel-Basset, Mohamed, G. Manogaran, D. El-Shahat, and S. Mirjalili., A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem." Future Gener. Comput. Syst. 85 (2018) 129-145.
[12] T.A. Redl, University Timetabling via Graph Coloring: An Alternative Approach, Congressus Numerantium, (187) (2007) 174-186.
[13] D. C. Wood, A System for Computing University ExaminationTimetables, The Computer Journal (11) (1968) 41-47

[14] J. R. Allwright, R. Bordawekar, P. D. Coddington, K. Dincer & C. L. Martin, A Comparison of Parallel Graph Coloring Algorithms. SCCS-666 (1995) 1-19.
[15] R. K . Bania & P. Duarah. Exam Time Table Scheduling using Graph Coloring Approach. International Journal of Computer Sciences and Engineering, 6(5) (2018) 84-93.
[16] D.J.A. Welsh, & M. B. Powell, An Upper Bound for the Chromatic Number of a Graph and it's Application to Timetabling Problems. The Computer Journal. ,Vol.10, No.1, pp. (1967)85-86.
[17] E. K. Burke, D. G. Elliman, & R. Weare, A University Timetabling System Based on Graph Colouring and Constraint Manipulation. Journal of research on computing in education, 27(1) (1994) 1-18.
[18] M. Malkawi, M.A. Hassan, & O. A. Hassan, A New Exam Scheduling Algorithm using Graph Coloring. International Arab Journal of Information Technology (IAJIT), 5(1)(2008) 80- 86.
[19] E. K. Burke, K. Jackson, J.H. Kingston, & R. F. Weare, Automated University Timetabling: The State of the Art. Compute. J., 40(9) (1997) 565–571.
[20] T. Harju, Oxide Networks, Graph Theory, and The Passivity of Ni-Cr-Mo Ternary Alloys. Corrosion Science, 50(12) (2008) 3622–3628.
[21] R. M. R. Lewis, A Guide to Graph Coloring. In A Guide to Graph Coloring, 7(2016)
[22] K. Ruohonen, Graph Theory (Chapter 2). Network Science, 72(8)(2015) 1–35.
[23] R. J. Trudeau, Introduction to Graph Theory. Dover books on advanced mathematics. (1993)
[24] R. J. Wilson, Introduction to Graph Theory. Longman. (1996)8-14
[25] D. B. West, Graph Coloring. In Graph Theory with Applications (2nd ed.). (2000)1-19
[26] C. Vasudev, Graph Theory with Applications. New Age International (P). (2006) 153-155
[27] J. Zhang, An Introduction to Chromatic Polynomials. Journal of Combinatorial Theory, 4(1)(1968) 52–71.