https://journal.uinsgd.ac.id/index.php/kubik/index
Vol. 10, No. 1, 2025, pp. 56-65
e-ISSN: 2686-0341 p-ISSN: 2338-0896

Courses Scheduling Using Graph Coloring

Rismawati Ramdani[1, *)], Salwa Nursyahida[2)]

[1,2] *Department of Mathematics, UIN Sunan Gunung Djati Bandung*

*) *email: rismawatiramdani@uinsgd.ac.id*

Abstract

At the beginning of each academic semester, universities are routinely required to develop course schedules that minimize or eliminate conflicts. Scheduling conflicts typically arise when multiple courses are taught by the same lecturer, taken by the same group of students, or require the use of the same classroom. As a result, an efficient and systematic method is needed to generate conflict-free schedules while optimizing the use of available time slots. One alternative approach is to apply graph theory, particularly graph coloring techniques, to the scheduling process. In this approach, each course is represented as a vertex in a graph, and an edge is established between two vertices if the corresponding courses cannot be held simultaneously. Graph coloring is then used to assign different time slots (represented as colors) to adjacent vertices, ensuring that no conflicting courses are scheduled at the same time. This paper proposes a course scheduling algorithm based on graph coloring, aiming to produce feasible schedules that reduce conflicts and enhance resource utilization. The approach provides a mathematical framework that can support automated and scalable scheduling systems in academic institutions.

Keywords: graph, graph coloring, scheduling
MSC2020: 05C78, 05C85, 05C90

Abstrak

Pada awal setiap semester akademik, perguruan tinggi secara rutin dihadapkan pada kebutuhan untuk menyusun jadwal perkuliahan yang bebas dari konflik. Konflik penjadwalan umumnya terjadi ketika beberapa mata kuliah diajarkan oleh dosen yang sama, diikuti oleh kelompok mahasiswa yang sama, atau membutuhkan penggunaan ruang kelas yang sama pada waktu yang bersamaan. Oleh karena itu, diperlukan suatu metode yang efisien dan sistematis untuk menghasilkan jadwal yang optimal serta minim konflik. Salah satu pendekatan alternatif yang dapat diterapkan adalah melalui teori graf, khususnya teknik pewarnaan graf (graph coloring). Dalam pendekatan ini, setiap mata kuliah direpresentasikan sebagai titik (vertex) dalam suatu graf, dan sisi (edge) dibentuk apabila dua mata kuliah tidak dapat dijadwalkan secara bersamaan. Pewarnaan graf kemudian digunakan untuk menetapkan slot waktu yang berbeda (dilambangkan sebagai warna) bagi titik-titik yang saling terhubung, sehingga tidak terjadi tumpang tindih. Makalah ini mengusulkan algoritma penjadwalan mata kuliah berbasis pewarnaan graf yang bertujuan untuk menghasilkan jadwal yang layak, mengurangi potensi konflik, dan mengoptimalkan pemanfaatan sumber daya yang tersedia.

Kata kunci: graf, pewarnaan graf, penjadwalan
MSC2020: 05C78, 05C85, 05C90

Citation: R. Ramdani and S. Nursyahida, "Courses Scheduling Using Graph Coloring", *KUBIK J. Publ. Ilm. Mat.*, Vol. 10, No. 1, pp. 56-65, 2025.

*) Corresponding Author
Received: 25-07-2024, Accepted: 09-08-2025, Published: 12-08-2025

Introduction

One of the most popular topics of graph theory research is graph coloring. Graph coloring has many applications in various aspects including scheduling, registration allocation, and so on [1], [5]. Graph coloring can be applied to varied problems, including optimization problems [3]. Recent research on specific graph structures has also expanded the theoretical foundations of graph coloring and labeling. For instance, Medika et al. [17] investigated the rainbow connection number in book graphs, while Sari et al. [18] explored k-th prime Fibonacci labeling on H graphs and caterpillar graphs, both of which enrich the theoretical background relevant to scheduling applications.

Recent studies also show its effectiveness in educational timetabling, where various algorithms such as Welsh–Powell, DSATUR, and hybrid approaches combining graph coloring with integer linear programming have been explored [12], [13], [14]. Moreover, modern approaches using metaheuristics and graph neural networks have further enhanced the efficiency and scalability of graph coloring solutions [15], [16]. One example of an optimization problem is courses or exams time-table scheduling [3], [10], [11].

One of the researchers who popularized the scheduling problem with graph coloring is Carter [4]. His research aims to provide conflict-free scheduling [6]. Kiaer and Yellen use a graph coloring approach to find approximate solutions to the course scheduling problem [7]. Burke et al. put a manipulative constraint on the plan of a course scheduling system [3]. Research on course scheduling using graph coloring is still developing both in terms of scheduling methods, problem constraints, and problems that depend on real scheduling cases.

Course scheduling activities at universities are routine activities carried out at each university. There are many aspects that must be considered when compiling a lecture schedule at a college or university, one of which is several courses taught by the same lecturer. In addition, several courses are held in the same classroom, these courses must be scheduled at different time intervals. It is not uncommon for some students to be required by the curriculum to take two or more different but related courses, and must be taken simultaneously in the same semester. Such cases also need to be carefully scheduled in order to avoid conflicts. Graph coloring is one way to solve the problem of determining the number of realistic minimum time intervals in scheduling [2], [9], [11]. In this article, we will construct a course scheduling algorithm that utilizes graph coloring to prevent or minimize conflicts caused by scheduling problems.

Methods

To determine the solution to the problem of scheduling courses using graph coloring, the problem is first formulated in the form of a graph where the courses act as vertices. The edge of the graph will be defined according to the existing boundaries where these boundaries create conflicts. After defining the edges, a graph will be formed that represents the course scheduling problem. Next, an algorithm for determining the solution of this scheduling problem is formed which applies the concept of graph coloring. Some scheduling problems involve few resources, and other require many resources at once. Subject can be conflicted because the lecturer are the same, the classrooms are the same, and students are the same. In such a case, graph that is formed consider the conflict of courses, lecturers, and lecture halls simultaneously

Results and Discussion

Creating Conflict Graph

The first thing that needs to be done is to record which courses have a possible intersection of conflicts, namely by making a conflict graph. Suppose G is a simple graph with vertices in G representing courses while edges in G represent conflicts between courses. Suppose $v_1, v_2, \dots, v_n$ are courses that are in an institution.

Definition 1. Let $A(G)$ be the adjacency matrix of G, is defined by

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

with

$$a_{ij} = \begin{cases} 0 & \text{if } v_i \text{ has no conflict with } v_j \\ 1 & \text{if } v_i \text{ has a conflict with } v_j \end{cases}$$

The table 1 shows a simple example of data collection of conflicts that occur between five courses, for example S_1, S_2, S_3, S_4, and S_5.

Table 1. Table of five courses

	S_1	S_2	S_3	S_4	S_5
S_1		√	√	√	√
S_2	√				
S_3	√			√	√
S_4	√		√		√
S_5	√		√	√	

Then, the table 1 is represented to adjacency matrix of the graph. Let *A* is adjacency matrix of *G*. Then

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

So, the illustration of the graph G can be seen in the **Figure 1**.

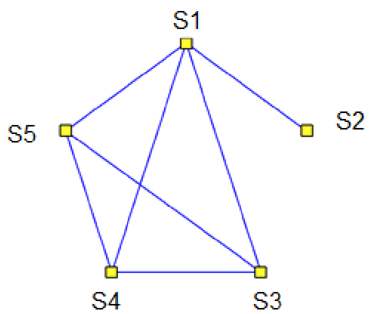

Figure 1. Conflict graph of the five courses

Coloring of The Conflict Graph

Coloring of the graph can be done when the conflict graph has been formed with the aim of minimizing the number of time slots needed to schedule all available courses. In this article the coloring process will be carried out with the following algorithm [2]:

Algorithm 1.

(1) Begin
(2) Construct conflict graph G.
(3) Let $\deg(v)$ be degree of vertex v in G.
(4) $i := 0$
(5) $S := V(G)$
(6) While $S \neq \emptyset$ do
(7) Begin
(8) $i = i + 1$
(9) Let v be vertex with maximum degree in S in G
(10) Give color i to the vertex v
(11) Let $N(v)$ be set of all vertices adjacent with v in S in G
(12) Give color i to all vertices in $N(v)$
(13) $S := G - N(v)$
(14) End
(15) So that, i is the minimum number of colours needed to coloring graph G.
(16) End

If the algorithm 1 is applied to the example in Table 1, it takes a minimum of four colors to color the graph G with the following illustration:

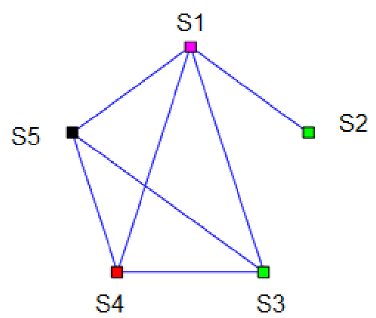

Figure 2. Coloring of conflict graph with the five courses

It means, courses with the same color can be scheduled in the same or different time slots, while courses with different colors must be scheduled in different time slots.

Table 2. Time slot allocation for Scheduling 5 Courses

1	2	3	4
S_1	S_2	S_4	S_5
	S_3		

Graph Coloring for Lecture Scheduling using Maple 18

By using the Graph Theory Package in the Maple 18 application, the conflict graph coloring algorithm is simplified to:

Algorithm 2.

(1) Construct of the conflict matrix A
(2) Construct of Conflict Graph G from matrix A with the Graph(A) command
(3) Determine the chromatic number for G with the command ChromaticNumber(G)

For example, in a department at university A, there are 30 courses available in the current semester. Course scheduling will be carried out using graph coloring.

Problem constraints defined in this schedule:

(1) Courses taught by the same lecturer cannot be scheduled in the same time slot.
(2) Courses contracted by at least one student cannot be scheduled in the same time slot.
(3) Each course must be scheduled in exactly one-time slot.
(4) Potentially conflicting courses must be scheduled in different time slots.

Suppose the available courses are $S_1, S_2, \dots, S_{30}$. Suppose A is a conflict matrix between 30 courses as follows:

1. *With (Graph Theory)*
2. With conflict data collection represented by matrix A, it is possible to construct a conflict graph G with A as the adjacency matrix of G, with the following algorithm:

$$A := \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix};$$

3. *G=Graph(A):*
4. *DrawGraph(G);*

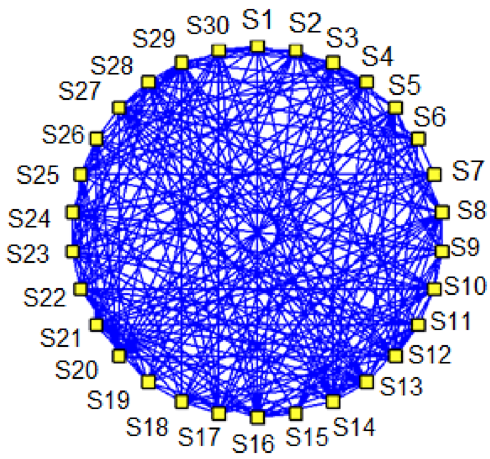

Figure 3. Conflict graph for 30 courses

Then the conflict graph G is colored to get the course scheduling. More effective than the previous algorithm, the graph coloring process in Maple 18 can be done with one command, which is to find the chromatic number for graph G.

5. *ChromaticNumber(G', colour');*

7

Obtained a minimum of 7 colors to color graph G with one of the following coloring combinations

6. *Colour*;

[[S1, S2, S18, S23, S27], [S4, S5, S21, S24], [S3, S7, S12, S13, S19], [S11, S15, S20, S28, S30], [S6, S10, S14, S25], [S8, S22, S29], [S9, S16, S17, S26]]

If it is represented in a colored graph, it can be presented in Figure 4.

7. *HighlightVertex(G, col[1], magenta); HighlightVertex(G, col[2], green); HighlightVertex(G, col[3], red); HighlightVertex(G, col[4], black); HighlightVertex(G, col[5], white); HighlightVertex(G, col[6], brown); HighlightVertex(G, col[7], yellow);*
8. *DrawGraph(G):*

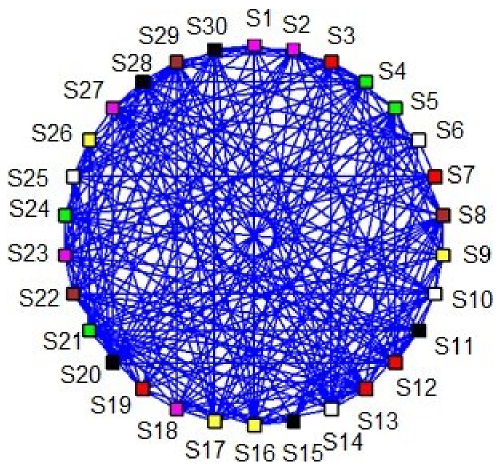

Figure 4. Coloring of Conflict graph for 30 courses

The coloring process has been completed, so that the minimum time slot allocation table for 30 courses is obtained as follows;

Table 3. Time slot allocation for scheduling 30 courses

1	2	3	4	5	6	7
S_1	S_4	S_3	S_{11}	S_6	S_8	S_9
S_2	S_5	S_7	S_{15}	S_{10}	S_{22}	S_{16}
S_{18}	S_{21}	S_{12}	S_{20}	S_{14}	S_{29}	S_{17}
S_{23}	S_{24}	S_{13}	S_{28}	S_{25}		S_{26}
S_{27}		S_{19}	S_{30}			

As an illustration of scheduling, suppose there are 10 time slots with the provisions of 5 days a week and 2 time slots each per day, so the scheduling alternatives are as follows.

Table 4. Scheduling of 30 courses

	Monday	Tuesday	Wednesday	Thursday	Friday
09.00 - 12.00	S_1	S_4	S_3	S_{11}	S_6
	S_2	S_5	S_7	S_{15}	S_{10}
	S_{18}	S_{21}	S_{12}	$S20$	S_{14}
	S_{23}	S_{24}	S_{13}	S_{28}	S_{25}
	S_{27}		S_{19}	S_{30}	
13.00 - 16.00	S_8	S_9			
	S_{22}	S_{16}			
	S_{29}	S_{17}			
		S_{26}			

It should be noted that this scheduling results in a minimum time slot, but scheduling can be done according to the needs and conditions of the institution, provided that each subject that is given a different color cannot be scheduled in the same time slot, while for courses with the same color it can be scheduled with different colors. same or different slots. So, for example, if you add a stipulation that there can only be 4 parallel courses related to space limitations, then courses with the same color can be divided into empty time slots, for example, table 4 can be modified as follows.

Table 5. Modified Schedulling of 30 courses

	Monday	Tuesday	Wednesday	Thursday	Friday
09.00-12.00	S_1	S_4	S_3	S_{11}	S_6
	S_2	S_5	S_7	S_{15}	S_{10}
	S_{18}	S_{21}	S_{12}	S_{20}	S_{14}
		S_{24}			S_{25}
13.00-16.00	S_8	S_9	S_{23}	S_{13}	S_{28}
	S_{22}	S_{16}	S_{27}	S_{19}	S_{30}
	S_{29}	S_{17}			
		S_{26}			

The results obtained in this study are consistent with recent research that applied graph coloring to university course timetabling. For example, Biswas et al. [12] and Perera & Lanel [13] achieved similar improvements in minimizing time slots by applying optimized coloring heuristics. Likewise, reviews such as Babaei et al. [14] emphasize that integrating graph coloring with other optimization techniques can address more complex scheduling constraints, while recent advances using genetic algorithms [15] and graph neural networks [16] open promising directions for future work.

Conclusion

This study has developed and implemented a course scheduling algorithm based on graph coloring to minimize scheduling conflicts among courses, lecturers, and classrooms. By representing courses as vertices and conflicts as edges, the proposed method determines the chromatic number of the resulting graph, thereby obtaining the minimum number of time slots required. The application of Maple 18 simplified the coloring process and provided an efficient means to handle larger scheduling instances, as demonstrated in both small-scale and 30-course case studies.

The results confirm that graph coloring is an effective approach for constructing conflict-free timetables and optimizing the use of available time slots. These findings align with previous studies and recent developments in educational timetabling that employ enhanced heuristics, hybrid methods, and machine learning approaches. Future research could extend this work by incorporating additional real-world constraints—such as room capacities, lecturer preferences, and course priorities—and by exploring the integration of metaheuristic optimization or graph neural network techniques to further improve scalability and adaptability.

Acknowledgments

The authors wish to acknowledge Department of Mathematics UIN Sunan Gunung Djati Bandung, which supports and funds this research.

References

[1] G. D. Birkhoff, “A determinant formula for number of ways of coloring a map”, *Annals of Mathematics*, vol. 14, pp. 42–46, 1912. https://doi.org/10.2307/1967597.

[2] S. Bondy, “Final Examination Scheduling”, *Communications of the ACM*, vol. 22, no. 7, pp. 494–498, 2002.

[3] E. K. Burke, D. G. Elliman, and R. Weare, “A university timetabling system based on graph coloring and constraint manipulation”, *Journal of Research on Computing in Education*, vol. 27, no. 1, pp. 1–18, 1994. https://doi.org/10.1080/08886504.1994.10782112.

[4] M. W. Carter, G. Laporte, and S. Lee, “Examination timetabling: Algorithmic strategies and applications”, *Scale Timetabling Problems Society*, vol. 47, no. 3, pp. 373–383, 1996.

[5] N. Deo, *Graph theory with applications to engineering and computer science*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[6] R. M. Karp, “Reducibility among combinatorial problems”, in *Symposium on the Complexity of Computer Computations*, New York, NY, 1972.

[7] L. Kiaer and J. Yellen, “Weighted graphs and university timetabling”, *Computers and Operations Research*, vol. 19, no. 1, pp. 59–67, 1992.

[8] S. K. Miner, S. Elmohamed, and H. W. Yau, "Optimizing timetabling solution using graph coloring", *NPAC*, vol. 23, pp. 99–106, 1995.

[9] J. E. L. Peck and M. R. Williams, "Examination scheduling", *Communications of the ACM*, vol. 9, no. 6, pp. 433–434, 1966.

[10] T. A. Redl, "University timetabling via graph coloring: An alternative approach", *Congressus Numerantium*, vol. 187, pp. 174–186, 2007.

[11] D. C. Wood, "A system for computing university examination timetables", *The Computer Journal*, vol. 11, pp. 41–47, 1968.

[12] S. Biswas, A. Bhattacharya, P. Mitra, and M. K. Naskar, "Graph coloring in university timetable scheduling", *International Journal of Intelligent Systems and Applications*, vol. 15, no. 3, pp. 16–32, 2023. https://doi.org/10.5815/ijisa.2023.03.02.

[13] M. T. M. Perera and G. H. J. Lanel, "A model to optimize university course timetable using graph coloring and integer linear programming", *IOSR Journal of Mathematics*, vol. 12, no. 5, pp. 13–18, 2016. https://doi.org/10.9790/5728-1205031318.

[14] H. Babaei, J. Karimpour, and A. Hadidi, "A survey of approaches for university course timetabling problem", *Computers & Industrial Engineering*, vol. 86, pp. 43–59, 2015. https://doi.org/10.1016/j.cie.2014.11.010.

[15] E. Demirović, M. J. Sann, and P. Stuckey, "An overview of graph colouring: Problems, applications and algorithms", *European Journal of Operational Research*, vol. 296, no. 1, pp. 1–19, 2022. https://doi.org/10.1016/j.ejor.2021.03.045.

[16] A. Assadi, S. Khanna, and Y. Li, "Graph neural networks for graph coloring", *Physical Review Research*, vol. 4, no. 4, 043131, 2022. https://doi.org/10.1103/PhysRevResearch.4.043131.

[17] G. H. Medika, Z. B. Tomi, and E. R. Wulan, "Rainbow Connection Number in Book Graphs", KUBIK: Jurnal Publikasi Ilmiah Matematika, vol. 9, no. 1, pp. 86–101, 2024. https://doi.org/10.15575/kubik.v9i1.30406.

[18] N. Y. Sari, E. Noviani, and F. Fran, "k-th Prime Fibonacci Labeling on H Graph and Caterpillar Graph H_n", KUBIK: Jurnal Publikasi Ilmiah Matematika, vol. 8, no. 2, pp. 89–98, 2023. https://doi.org/10.15575/kubik.v8i2.29290.