Volume 14, Issue 3, Page 46-56, 2025 https://doi.org/10.46810/tdfd.1629184 Research Article

BİNGÖL UNIVERSITY

Türk Doğa ve Fen Dergisi
Turkish Journal of Nature and Science
www.dergipark.gov.tr/tdfd

Graph-Based Course Scheduling Using the Malatya Vertex Coloring Algorithm for Constraint Optimization

Cezayir KARACA[1*], Selman YAKUT[2]

[1] İnönü University, Faculty of Engineering, Department of Computer Engineering, Malatya, Türkiye
[2] İnönü University, Faculty of Engineering, Department of Software Engineering, Malatya, Türkiye
Cezayir KARACA ORCID No: 0009-0007-9250-2612
Selman YAKUT ORCID No: 0000-0002-0649-1993

**Corresponding author: cezayirkaraca0242@gmail.com*

(Received: 29.01.2025, Accepted: 10.07.2025, Online Publication: 26.09.2025)

Keywords
Course scheduling, Centrality, Malatya coloring, Timetabling

Abstract: The course timetabling problem is a significant combinatorial optimization problem that has attracted the attention of researchers since the second half of the 20th century. Traditionally managed through manual methods, the scheduling process is time-consuming, challenging, and prone to errors. Therefore, with technological advancements, various algorithms have been developed to offer more efficient and faster solutions. In this study, the "MVC Algorithm" is applied to the course timetabling problem. The algorithm operates in two main steps: first, the Malatya Centrality(MC) values of the nodes in the timetable graph are calculated; then, the node with the highest centrality is selected and labeled with an appropriate color. Throughout the process, the main objective is to minimize course conflicts and to generate a valid timetable that complies with defined constraints. The MVC Algorithm stands out with its predictability of procedural steps and its potential to operate in polynomial time, thus offering an effective alternative to classical and heuristic methods proposed in the literature.

Kısıt Optimizasyonu için Malatya Vertex Coloring Algoritması Kullanılarak Graf Tabanlı Ders Çizelgeleme

Anahtar Kelimeler
Ders çizelgeleme, Merkezilik, Malatya coloring, Zaman çizelgesi

Öz: Ders çizelgeleme problemi, 20. yüzyılın ikinci yarısından itibaren araştırmacıların dikkatini çeken önemli bir kombinatoryal optimizasyon problemidir. Geleneksel olarak manuel yöntemlerle yürütülen çizelgeleme süreci, zaman alıcı ve zorlayıcı olmakla birlikte hata yapmaya açık bir yapıdadır. Bu nedenle, teknolojik ilerlemelerle birlikte çeşitli algoritmalar geliştirilerek daha etkili ve hızlı çözümler sunulmaya çalışılmıştır. Bu çalışmada, "Malatya Vertex Coloring(MVC) Algoritması" ders çizelgeleme problemine uygulanmaktadır. Algoritma, iki temel adımda çalışmaktadır: ilk olarak, çizelge grafındaki düğümlerin Malatya Merkezilik değerleri hesaplanmakta; ardından en yüksek merkeziliğe sahip düğüm seçilerek uygun bir renkle etiketlenmektedir. Süreç boyunca temel hedef, ders çakışmalarını en aza indirmek ve tanımlı kısıtlamalara uyumlu, geçerli bir çizelge üretmektir. MVC Algoritması, işlem adımlarının öngörülebilirliği ve polinom zamanda çalışabilme potansiyeliyle dikkat çekmekte, bu yönüyle literatürde önerilen klasik ve sezgisel yöntemlere etkili bir alternatif sunmaktadır.

1. INTRODUCTION

The Course Scheduling Problem (CSP) is a complex optimization problem that has been extensively studied in the fields of operations research and artificial intelligence since the 1960s. This problem aims to assign courses to specific time slots, classrooms, and instructors without conflicts. Particularly in educational institutions, due to the necessity of managing limited resources (classrooms, instructors, time slots) along with numerous constraints, the scheduling process becomes highly time-consuming and error-prone when handled manually[1].

Technological advancements have enabled the development of various algorithms to automate and improve this process. Among the solution methods proposed in the literature, heuristic algorithms, metaheuristic approaches (e.g., genetic algorithms, tabu search)[2], and graph-based algorithms (e.g., bipartite edge coloring, vertex coloring)[3] are particularly

prominent. However, the high computational power required by existing algorithms when applied to large datasets poses a significant challenge, especially in university settings where course repetitions and student enrollments are dense.

Course scheduling is not merely a matter of assigning courses to time slots. It also involves ensuring compliance with various strict constraints, such as preventing conflicts and adhering to class capacities and lecture hours. Some fundamental challenges frequently encountered in universities include:

- Lecture durations not being uniform,
- Courses sharing common students or instructors not being scheduled simultaneously,
- The necessity for certain courses to be repeated more than once a week.

In this context, there is a growing need for more flexible and computationally efficient algorithms that can reduce course conflicts and generate optimal timetables under defined constraints.

In response to these needs, this study proposes a novel approach called the MVC Algorithm. The algorithm consists of two fundamental phases. In the first phase, the Malatya Centrality (MC) values of the nodes representing the courses are calculated. In the second phase, the node with the highest MC value is selected, assigned a color, and removed from the graph. This process continues iteratively, ensuring that adjacent nodes are assigned different colors. The main advantage of the proposed algorithm lies in the predictability of all procedural steps and its ability to produce solutions in polynomial time.

The remainder of this article is organized as follows: Section 2 reviews the related literature and graph-coloring-based approaches. Section 3 presents the method and proposed approach. Experimental results are provided in Section 4. Finally, Section 5 offers interpretations of the findings and discusses recommendations for future work.

2. LITERATURE REVIEW

CSP is a classical combinatorial optimization problem that aims to ensure the efficient and conflict-free use of resources in educational institutions. As the number of courses and the constraints to be applied increase, the problem becomes increasingly complex. Therefore, numerous studies based on various algorithms have been conducted. These studies can be categorized according to the methods employed.

Graph-coloring-based algorithms are widely used to prevent conflicts through structures in which nodes and edges correspond to time slots. Rina I. proposed a solution using the Welch-Powell algorithm, treating each course as a node and assigning the same time slot to nodes with the same color. This method yielded effective results for basic scheduling problems due to its simplicity[4]. A resource scheduling algorithm proposed by Egwuneche successfully scheduled tasks without resource conflicts based on graph coloring[5]. Bania and Duarah developed an examination scheduling model using graph coloring with institutional data and achieved satisfactory results under both hard and soft constraints[6]. Mursyidah (2019) discussed the application of graph coloring to schedule courses effectively and efficiently based on student needs and available spaces at the Department of Mathematics Education, University of Muhammadiyah Surabaya[7]. In their study, the authors used graph coloring to solve the CSP by dividing the data into three sets: time slots, classrooms, and the courses assigned to lecturers[8]. Mishkhal I. and colleagues attempted to solve CSP using a graph coloring approach implemented in the JEdit programming language[9].

These methods aim to scan the solution space more rapidly and manage complex constraints more effectively. Han and Wang developed a hybrid method, POGA-DP, which combines genetic algorithms and dynamic programming for university course timetabling. This method significantly improves scheduling quality, especially in cases where multiple classes attend a single course, and ensures more efficient classroom usage. Tests on real datasets demonstrated substantial performance improvements compared to traditional methods[10]. Budiarto applied genetic algorithms (GA) to school-level CSP. In a study conducted at SMPK Santo Yoseph School in Bali, the difficulties of manual scheduling were highlighted, and a fitness value of 0.0880 was achieved after 100 generations. This method reduced the number of daily classes from seven to four or five, resulting in a more balanced and efficient schedule[11]. Rahardjo and Zulkifli successfully created conflict-free course schedules by integrating dynamic matching with Vertex Graph Coloring, one of the most effective methods for CSP[12]. Burke and his colleagues applied heuristic variants of graph coloring methods in curriculum-based course scheduling problems, modeling constraint sets in the classroom-time-teacher triangle[13].

Mathematical programming-based approaches are often preferred in scheduling problems where precision and optimality are required. E. Rappos and colleagues developed a Mixed Integer Programming (MIP) model to solve the university timetabling problem. The model simultaneously optimizes student, classroom, and time assignments, addressing both student conflicts and classroom usage constraints. Due to the large size of the model, a two-phase solution was proposed: first, a feasible initial solution is obtained, followed by refinement using a local search algorithm. The model yielded strong results when tested on ITC 2019 competition datasets[14]. Steiner and his team used a three-stage modeling process and integer programming to reduce conflicts caused by complex structures such as double major selections at the University of Graz[15]. Subulan argued that timetables should not only balance scheduling but also aim for skill development. The model developed for this purpose was solved using multi-objective mathematical programming and produced promising results[16]. In another study, the authors successfully applied linear programming—an effective method for addressing CSP—to prepare class

schedules at the Sirindhorn International Institute of Technology[17].

Due to the increasing volume and complexity of data, hybrid solutions combining multiple methods have become more prevalent. Gunawan A. developed a hybrid algorithm to solve faculty assignment and course scheduling problems simultaneously. By combining integer programming, greedy heuristics, and a modified simulated annealing method, this approach delivered effective results even on large datasets. The method was tested on data from a university in Indonesia[18]. The authors proposed a two-phase hybrid algorithm combining exact and heuristic methods for university course scheduling. In the first phase, courses were clustered using graph coloring; in the second phase, a tabu search algorithm assigned these clusters to time slots. The model also incorporated additional constraints based on teacher and student preferences to generate realistic schedules[1].

In another study, the authors utilized a hybrid method combining graph coloring methodology and heuristic techniques to solve curriculum-based CSP[19]. Fizzano and Swanson developed an algorithm that assigns only one student group per class at a specific time and day, optimizing the allocation of student groups to classrooms based on this constraint[20].

In this study, a novel algorithm based on the graph coloring approach—Malatya Vertex Coloring (MVC) Algorithm—is proposed to solve the CSP. Unlike many widely used methods in the literature, this algorithm prioritizes the most critical nodes by calculating their Malatya Centrality (MC) values. As a result, course conflicts are minimized, and more efficient timetables are produced using fewer time slots. Furthermore, the algorithm enables the easy integration of additional constraints such as instructor availability, increasing its applicability in highly constrained environments. Test results on real-world datasets demonstrate that the algorithm provides an effective solution in terms of both accuracy and scheduling quality.

3. PROPOSED METHOD

3.1. Problem Definition and Approach

The Course Scheduling Problem (CSP) is a classical and computationally complex scheduling problem commonly encountered in universities and educational institutions. The CSP requires the allocation of courses, instructors, classrooms, and students to appropriate time slots while satisfying a variety of constraints. Due to its broad applicability across different academic domains, the CSP has led to the development of numerous algorithms and solution strategies in the literature. With the advancement of computer science and optimization techniques, increasingly effective and efficient algorithms have been proposed to solve CSPs. Notably, the CSP shares considerable similarities with the Examination Timetabling Problem (ETP). Although some constraints differ between the two, many of the solution approaches are similar, which has led several academic studies to treat them within a unified framework.

In this study, the Malatya Vertex Coloring (MVC) algorithm is applied to address the CSP. The MVC algorithm aims to achieve an optimal or near-optimal coloring of graph vertices, representing courses, in such a way that the number of required time slots is minimized. By efficiently assigning colors (i.e., time slots) to conflicting courses, the algorithm ensures a feasible and conflict-free timetable, ultimately enhancing the utilization of limited temporal resources.

3.2. Application of the MVC Algorithm to the Course Scheduling Problem

In this study, the Malatya Vertex Coloring (MVC) algorithm was employed to develop a solution for the university-level Course Scheduling Problem (CSP). MVC is a graph-theoretic approach that focuses on identifying the most influential vertex during the coloring process. For this purpose, a Malatya Centrality (MC) value is calculated for each vertex, and the vertex with the highest MC value is selected as the next candidate to be colored[21].

The MVC algorithm relies on the centrality values of vertices to determine their relative influence within the graph. The MC value of a vertex is defined as the sum of the ratios between the vertex's degree (i.e., the number of neighbors) and the degrees of its neighboring vertices[22]. This formulation prioritizes vertices that are highly connected but surrounded by relatively less connected neighbors. Consequently, such vertices are selected earlier in the coloring sequence.

The functioning of the MVC algorithm is illustrated through an example graph consisting of six vertices, each representing a course: "Programming," "Mathematics," "Data Structures," "Physics," "Database Systems," and "Discrete Mathematics." Edges between vertices indicate that the corresponding courses are taken by at least one common student, implying a scheduling conflict if assigned to the same time slot. In other words, the existence of an edge between two courses signifies a hard constraint that they must not be scheduled simultaneously.

The resulting graph structure, as shown in Figure 1, models the CSP as a graph coloring problem, where the goal is to assign time slots (colors) to each course (vertex) such that no adjacent courses share the same slot.

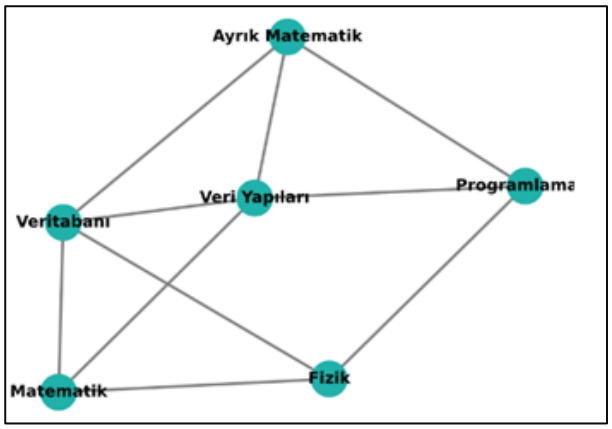


Figure 1. Graph model of a sample course schedule

In order to illustrate how the MC value, which lies at the core of the MVC algorithm, is calculated, the Data Structures course from the sample course graph has been selected. The degree of each node represents the number of edges (i.e., neighbors) it possesses. The degree values of the relevant nodes in the graph are as follows:

Deg(Discrete Mathematics) = 3
Deg(Database) = 4
Deg(Mathematics) = 3
Deg(Physics) = 3
Deg(Programming) = 4
Deg(Data Structures) = 4

$$MC(v_i) = \sum_{v_j \in N(v_i)} \frac{\text{degree}(v_i)}{\text{degree}(v_j)}$$

- v_i: The node for which the MC value is calculated
- $N(v_i)$: The set of neighbors of node v_i
- deg(v): The degree of node v (i.e., the number of edges)

When examining the neighbors of the Data Structures course, it is observed that it is directly connected to the courses Programming, Discrete Mathematics, Mathematics, and Database. Therefore, the MC value of the Data Structures node is calculated by taking the sum of the ratios of its degree (i.e., number of edges) to the degree values of its neighboring nodes.

$$MC(\text{Data Structures}) = \sum (\frac{4}{3} + \frac{4}{3} + \frac{4}{3} + \frac{4}{4})$$
$$MC(\text{Data Structures}) = 1,\overline{3} + 1,\overline{3} + 1,\overline{3} + 1$$
$$MC(\text{Data Structures}) = 5$$

This calculation indicates that the Data Structures node is highly connected, while some of its neighbors exhibit relatively lower connectivity. As a result, within the MVC algorithm, the Data Structures node is prioritized among those to be colored first.

To implement the MVC algorithm, a duplicate of the original graph is initially created. This duplicate graph serves to retain the information of the removed nodes and facilitates comparison with the colors of adjacent nodes during the coloring process. In contrast, the original graph is used for computing MC values and removing nodes as the algorithm progresses.

The procedure begins with the computation of MC values for all nodes in the graph. After these values are calculated, the node with the highest MC value, representing the most influential node, is identified. The colors already assigned to its adjacent nodes are then examined. Starting from the beginning of the color list, the first color that has not been used by any neighboring node is selected and assigned to the current node[23]. If the selected color is already present in a neighboring node, the next color in the list is considered and the same verification is performed. Once a color distinct from all neighboring nodes is found, it is assigned to the selected node. Following the color assignment, the node is removed from the original graph.

Subsequently, the MC values of the remaining nodes are recalculated. The node with the newly highest MC value is then identified and the same color assignment process is repeated. This node is also removed from the graph and the algorithm continues iteratively. This cycle is repeated until all nodes have been colored. By the end of the process, all nodes in the duplicate graph will have been successfully colored with valid assignments.

Algorithm 1. Pseudocode of the MVC Algorithm

```
MVC Algorithm
Input: Graph g = (V, E)
Output: Colored graph f
f ← Copy(g)
Function CentralityCalculate(g):
 // Calculates MC value for each node and finds the one with the
//highest centrality.
    MaxCentrality ← 0
    For each vertex i in V(g):
        Centrality ← 0
        For each neighbor j in Neighbors(i):
            Centrality ← Centrality + Degree(i) / Degree(j)
        If Centrality > MaxCentrality:
            MaxCentrality ← Centrality
            MaxVertex ← i
    Return MaxVertex
// Return the vertex with the highest MC value.

While VertexCount(g) > 0:
 // Repeat until all vertices are removed (colored).
    vertex ← CentralityCalculate(g)
    For each color c in ColorList:   // Iterate over the list of colors.
        If c not in Colors of Neighbors(vertex) in f:
            Assign color c to vertex in f
            Break
    Delete vertex from // Remove the colored vertex from the graph.

Show(f)
```

The detailed steps and control structure of the MVC algorithm are presented with the help of the pseudocode shown in Algorithm 1. The algorithm consists of two fundamental components: the CentralityCalculate function and the main body. The CentralityCalculate function is responsible for calculating the MC values of all nodes in the graph and for selecting the most influential node. In the main body, a copy of the original graph is first created. In each iteration, the most influential node is determined and assigned a color. During the coloring process, it is checked whether there is any conflict with the colors of adjacent nodes. These control operations are

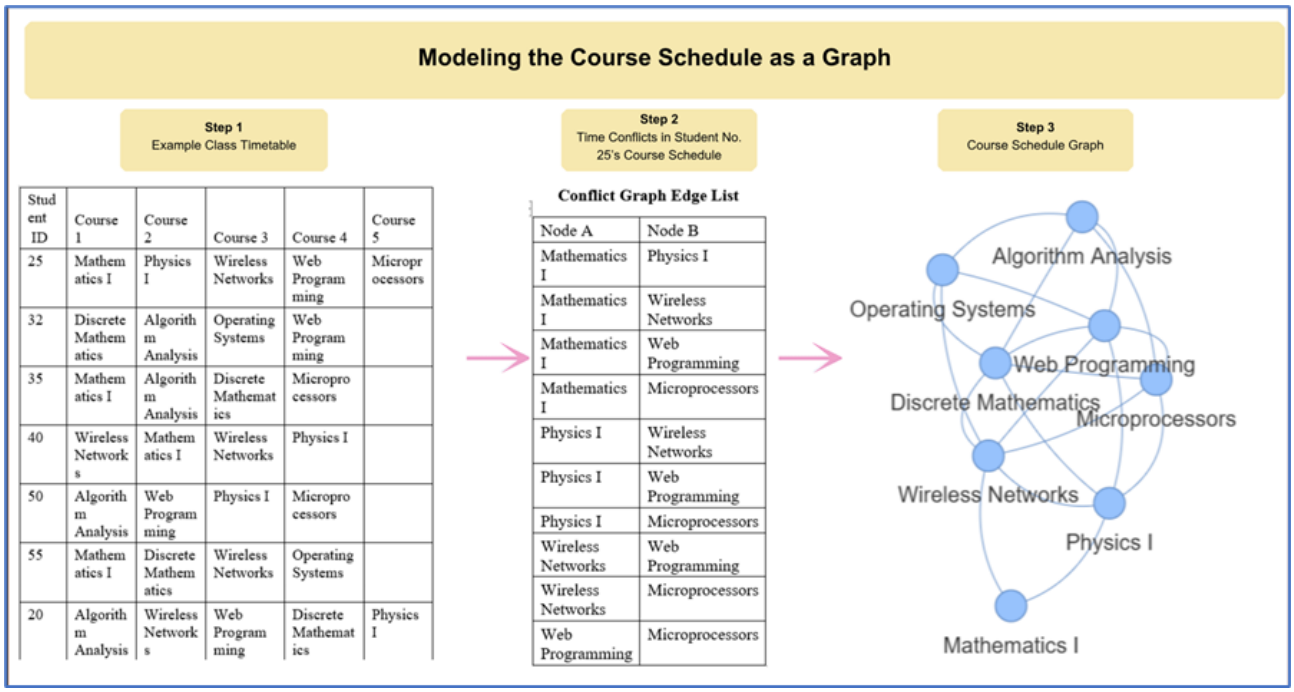
carried out using the copy graph. Once the color is assigned, the node is removed from the original graph. The original graph plays an active role in the removal of nodes and the recalculation of MC values, whereas the copy graph is used solely for tracking color assignments. The coloring process continues until there is only one start to finish, including MC computation, color comparison, color assignment, and node removal from the graph. Additionally, a small sample graph with calculated MC values and step-by-step illustrations of the coloring process will support the reader in better understanding how the algorithm functions.

The primary constraints encountered in solving course timetabling problems are that a student cannot attend multiple classes at the same time and that each course must be offered only once per week. These constraints

node left in the original graph. Once all nodes have been colored, the process ends, and the nodes in the copy graph are left with valid color assignments. To visualize the effectiveness and procedural operation of the MVC algorithm, it is recommended to include a flowchart. This diagram should clearly represent the entire process from

significantly increase the complexity of the timetabling problem.

In this study, the MVC algorithm was employed to address these challenges. During the implementation of the algorithm, each course was modeled as a node, and edges were established between courses taken by the same student, thereby forming a graph structure as illustrated in Figure 2. Through the coloring process applied to this graph structure, time conflicts between courses were effectively avoided[24].



Student ID	Course 1	Course 2	Course 3	Course 4	Course 5
25	Mathematics I	Physics I	Wireless Networks	Web Programming	Microprocessors
32	Discrete Mathematics	Algorithm Analysis	Operating Systems	Web Programming	
35	Mathematics I	Algorithm Analysis	Discrete Mathematics	Microprocessors	
40	Wireless Networks	Mathematics I	Wireless Networks	Physics I	
50	Algorithm Analysis	Web Programming	Physics I	Microprocessors	
55	Mathematics I	Discrete Mathematics	Wireless Networks	Operating Systems	
20	Algorithm Analysis	Wireless Networks	Web Programming	Discrete Mathematics	Physics I

Node A	Node B
Mathematics I	Physics I
Mathematics I	Wireless Networks
Mathematics I	Web Programming
Mathematics I	Microprocessors
Physics I	Wireless Networks
Physics I	Web Programming
Physics I	Microprocessors
Wireless Networks	Web Programming
Wireless Networks	Microprocessors
Web Programming	Microprocessors

Figure 2. Graph-Based Modeling Process of the Course Timetabling Problem

The course timetable datasets used in this study were obtained from the UNITIME.org platform, specifically the Computer Science Department Spring: [pu-spr07-cs] and Fall: [pu-fal07-cs] datasets[25]. In these datasets,

each course is represented as a node, and edges are created between courses taken by the same student, resulting in a relational graph-based structure.

Table 1. Courses Taken by Students

Student No	Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5
25	Mathematics 1	Physics 1	Wireless Networks	Web Programming	Microprocessors
32	Discrete Mathematics	Algorithm Analysis	Operating Systems	Web Programming	
35	Mathematics 1	Algorithm Analysis	Discrete Mathematics	Microprocessors	
40	Wireless Networks	Mathematics 1	Wireless Networks	Physics 1	
50	Algorithm Analysis	Web Programming	Physics 1	Microprocessors	
55	Mathematics 1	Discrete Mathematics	Wireless Networks	Operating Systems	
20	Algorithm Analysis	Wireless Networks	Web Programming	Discrete Mathematics	Physics 1

The MVC algorithm was applied to a graph structure representing student-course relationships, assigning different colors to potentially conflicting courses. This approach prevented students from being scheduled for multiple classes at the same time. Moreover, by employing a node selection strategy based on MC values, the algorithm effectively grouped interconnected courses into a minimal number of sets. As a result, the timetabling process was optimized by utilizing fewer time slots[21].

Table 2. List Made for the Graph Structure of the Courses Taken by Student No. 25

Edges	
Node	**Node**
Mathematics 1	Physics 1
Mathematics 1	Wireless Networks
Mathematics 1	Web Programming
Mathematics 1	Microprocessors
Physics 1	Wireless Networks
Physics 1	Web
Physics 1	Microprocessors
Wireless Networks	Web Programming
Wireless Networks	Microprocessors
Web Programming	Microprocessors

The primary constraints to be considered in the Course Scheduling Problem (CSP) are as follows:

- Courses taken by the same student must not be scheduled in the same time slot.
- Each course must be offered only once per week.

In accordance with these constraints, the MVC algorithm was implemented by modeling each course as a node (vertex), and establishing edges between courses taken by the same student so that these courses are treated as adjacent nodes. Table 2 illustrates the transformation of the course list of Student 25 into a graph structure. All possible pairwise combinations among this student's courses were generated to form the edge list. This process was repeated for all students to create a comprehensive list of course relationships, which was then used to construct a graph model. The MVC algorithm was applied to this graph. In the graph structure, nodes (courses) sharing the same color do not have a direct connection (edge), meaning that these courses can be scheduled in the same time slot. Conversely, differently colored courses are directly connected, indicating that they are taken by the same student and therefore must be scheduled in separate time slots to avoid conflicts. As such, minimizing the number of required time slots depends on reducing the number of colors used. The MVC algorithm provides efficient results in this context. The student-course data presented in Table 1 was converted into pairwise relationships as shown in Table 2 and then imported into the system. The data was first transformed into a graph structure, and subsequently, the MVC algorithm was applied to this graph. In each step of the algorithm, the MC values of all nodes were calculated as shown in Table 3. Based on these calculations, the iteration process proceeded as follows:

In the first iteration, the course "Wireless Networks", having the highest MC value, was assigned the first color in the list (e.g., green) and then removed from the graph. In the second iteration, the course with the highest updated MC value was "Physics 1" (MC = 6.6). Since its neighbor "Wireless Networks" had been assigned green, the next available color, purple, was assigned to "Physics 1." In the third iteration, three nodes shared the highest MC value, so the one that appeared earlier in the system, "Web Programming", was selected. Its neighbors had green (Wireless Networks) and purple (Physics 1) colors, so the next available color, blue, was assigned. In the fourth iteration, since green had not been used among the neighbors of "Algorithm Analysis", this course was assigned green. In the fifth iteration, "Discrete Mathematics" had neighbors with green, purple, and blue, thus the next suitable color, turquoise, was assigned. In the sixth iteration, the neighbors of "Microprocessors" had green, purple, blue, and turquoise, so the next available color, light green, was chosen. In the seventh iteration, "Operating Systems" had neighbors colored green, blue, and turquoise. Since purple had not been used among its neighbors, it was assigned purple. Finally, in the eighth iteration, "Mathematics 1" had neighbors with green and purple; thus, blue was assigned as the next available color. With this step, the coloring process was completed, and all nodes were removed from the graph. As a result of executing the MVC algorithm, a total of five distinct colors were used, indicating that at least five time slots are needed for scheduling. Once these time slots are created, courses with the same color are scheduled in the same slot, while courses with different colors are assigned to different time periods.

Table 3. MC Values in Each Iteration

.	Mathematics 1	Discrete Mathematics	Algorithm Analysis	Web Programming	Physics 1	Wireless Networks	Micro processors	Operating Systems
1st Iteration	0.73	7.4	3.13	7.4	6.0	8.9	4.75	2.9
2nd Iteration	0.25	6.4	3.9	6.4	6.6		3.59	1.95
3rd Iteration	0	4.6	4.6	4.6			2.25	2.25
4th Iteration	0	4.0	4.0				1.3	1.3
5th Iteration	0	4.0					0.5	0.5
6th Iteration	0						0	0
7th Iteration	0							0
8th Iteration	0							

In solving the CSP, the MVC algorithm prioritizes the coloring of the region with the highest MC value at each step. The coloring process of the map following each execution step is visualized through the stages presented in Figure 3.

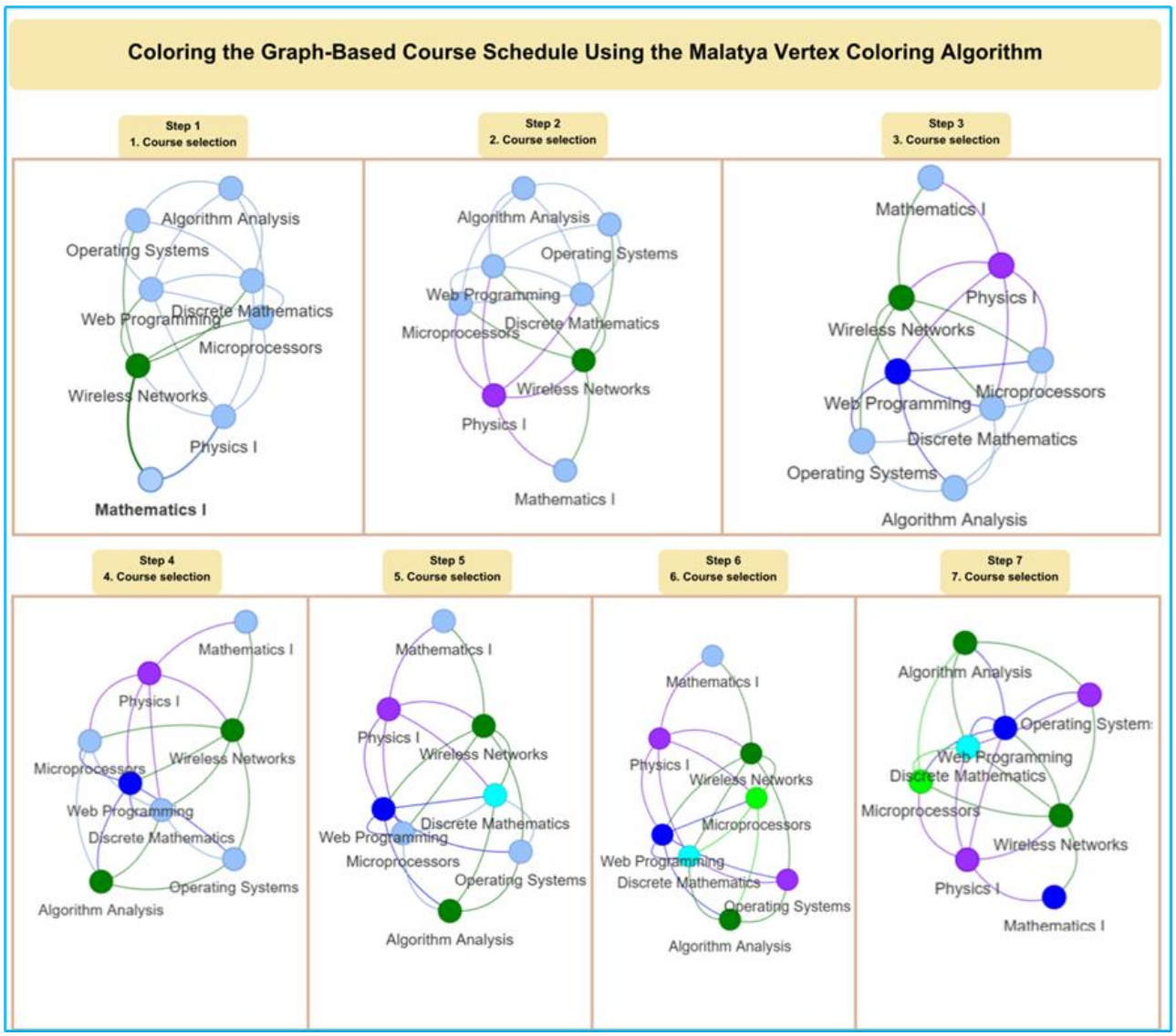


Figure 3. Color Assignment Process According to the Steps of the MVC Algorithm

This figure illustrates, step by step, which node is selected and which color is assigned in each iteration of the MVC algorithm. Starting from the top-left and proceeding to the bottom-right, the sequential stages clearly demonstrate the node selection logic based on centrality (MC) values and the corresponding color assignment process. This procedure effectively avoids color conflicts and ensures minimal color usage.

4. EXPERİMENTAL RESULTS

In this study, the "[pu-fal07-cs]" dataset, which belongs to the Computer Science Department of the University of Purdue and is publicly available on the Unitime.org platform, was utilized. Based on this dataset, a graph structure representing student conflicts between courses was constructed. The resulting graph consists of 466 nodes (courses) and 3,819 edges (student overlaps between courses). The MVC algorithm was applied to this large-scale graph structure, resulting in a coloring with 12 distinct colors. Since nodes with the same color are not adjacent to each other, these courses can be scheduled in the same time slot but in different classrooms. Conversely, courses with different colors are directly connected, indicating that they must be scheduled in separate time slots. As shown in Figure 4, the MVC algorithm enabled the scheduling of all 466 courses using only 12 time slots for this dataset. Additionally, four other commonly used graph coloring algorithms from the literature—DSATUR, Welsh–Powell, RLF, and Greedy—were also applied to the same dataset. As presented in Table 4, the MVC, DSATUR, Welsh–Powell, and Greedy algorithms all produced solutions using 12 colors, whereas only the RLF algorithm required 13 colors.

Table 4. Comparison of the Number of Colors Used by Different Algorithms in the Fall Semester Course Schedule of the Computer Science Department

Algorithm	Number of Colors Used
MVC (Malatya Vertex Coloring)	12
DSATUR (Degree of Saturation Order)	12
Welsh–Powell	12
RLF (Recursive Largest First)	13
Greedy (Rastgele sıra ile)	12

These results demonstrate that the MVC algorithm performs comparably to, or even better than, traditional algorithms, offering an effective approach to the timetabling process.

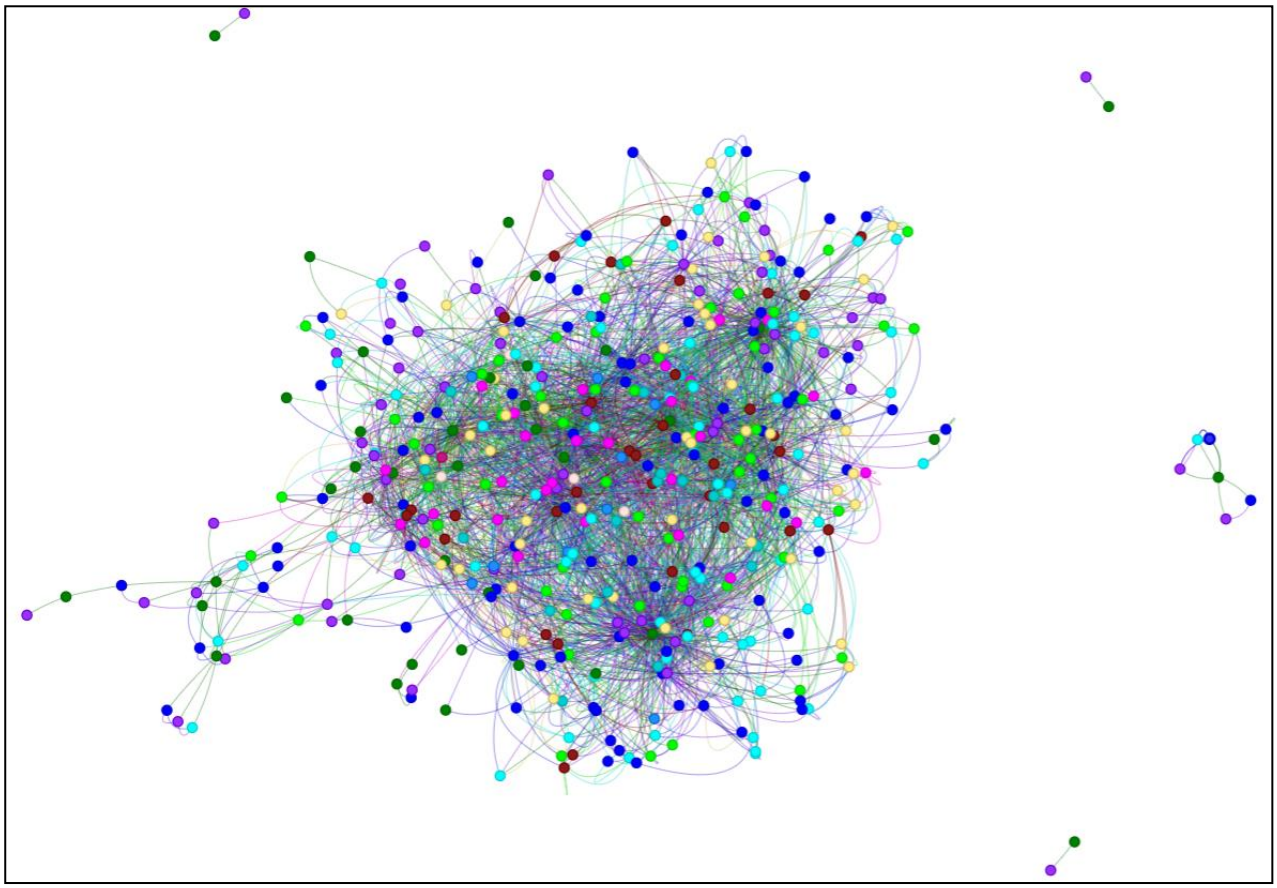


Figure 4. Computer science department fall term coloring

In the second phase of the study, the spring semester course scheduling dataset, labeled "[pu-spr07-cs]" and obtained from the Unitime.org platform, was utilized. In this dataset, each course is modeled as a vertex, and an edge is established between courses that are taken by the same student. As a result, a graph structure consisting of 884 vertices and 9,190 edges was constructed, based on 884 courses and 725 students. The MVC algorithm was applied to this large-scale graph, and a successful coloring was achieved using 14 distinct colors. Each color represents a separate time slot. Since there are no direct connections between vertices with the same color, these courses can be scheduled within the same time period. As illustrated in Figure 5, the MVC algorithm enabled a conflict-free scheduling of all courses in this dataset using only 14 time slots. The aforementioned dataset was also tested with other algorithms, and the corresponding results are presented in Table 5.

Table 5. Comparison of the Number of Colors Used by Different Algorithms in the Spring Semester Course Schedule of the Computer Science Department

Algorithm	Number of Colors Used
MVC (Malatya Vertex Coloring)	14
DSATUR (Degree of Saturation Order)	14
Welsh–Powell	13
RLF (Recursive Largest First)	16
Greedy (Rastgele sıra ile)	13

These comparisons demonstrate that the MVC algorithm yields results equivalent to those of traditional algorithms and can be effectively applied, particularly to large-scale datasets.

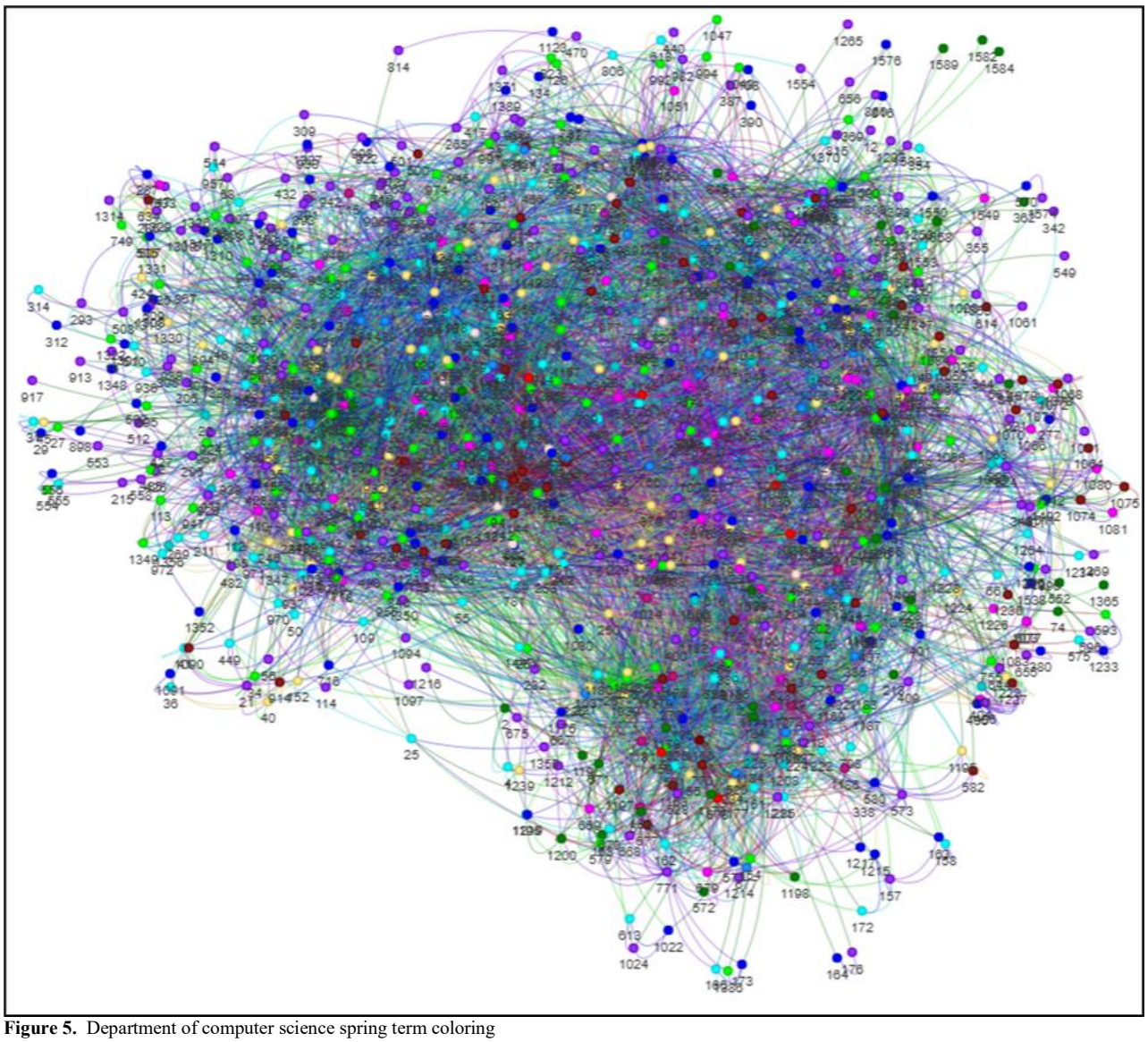


Figure 5. Department of computer science spring term coloring

4. CONCLUSION

This study addresses the complex and frequently encountered Course Scheduling Problem (CSP) in universities by employing a novel graph-based approach, the Malatya Vertex Coloring (MVC) algorithm. The MVC algorithm is designed to minimize conflicts between courses based on their centrality (MC) values. Its core principle is to prioritize the coloring of the most influential (central) nodes, thereby aiming to solve the scheduling problem using the minimum number of colors (i.e., time slots). Experimental applications on real-world datasets demonstrate the effectiveness of the MVC algorithm. Using two distinct semester datasets (Fall and Spring) from the Computer Science Department of Purdue University, the algorithm successfully generated conflict-free schedules with only 12 and 14 colors, respectively. These results indicate that even in large-scale systems involving hundreds of courses and thousands of students, conflict-free scheduling can be achieved with as few as 12–14 time slots. Furthermore, comparative analyses reveal that the MVC algorithm performs comparably to, or better than, traditional algorithms such as DSATUR, Welsh–Powell, Greedy, and RLF. The success of the MVC algorithm lies in its node selection strategy based on MC values. These values measure a node's relative impact within the graph, allowing for the prioritization of the most critical courses during the scheduling process. This mechanism not only optimizes time slot usage but also facilitates the integration of other scheduling actors, such as instructors, into the same framework. In fact, instructors' courses can be modeled in a similar way to students within the graph structure, enabling conflict-free scheduling within a unified algorithmic framework.

In addition, the operational steps of the algorithm were illustrated in detail through examples and visualizations. The clear definition of stages such as MC value calculation, usage of a duplicate graph, and color control mechanisms contributes to both the theoretical strength and practical applicability of the algorithm.

The advantages offered by the MVC algorithm can be summarized as follows:

- Ability to function under complex constraints,

- Conflict-free scheduling with a minimal number of time slots,
- Flexibility to model both student and instructor data within a single structure,
- Applicability to large datasets due to its polynomial-time execution.

In conclusion, the MVC algorithm offers a robust alternative to conventional and heuristic methods for solving CSPs. Its simplicity, predictability, and efficiency in practice suggest that this approach has strong potential for widespread application in educational institutions' scheduling systems.

Future research may focus on extending the algorithm to incorporate additional constraints (e.g., classroom capacities, course priorities, student preferences), performing multi-objective optimization with data from various departments, and testing the algorithm in real-time scheduling environments. Moreover, combining the algorithm with parallel computing techniques could significantly improve its performance on larger-scale problems.

REFERENCES

[1] Xiang, K., Hu, X., Yu, M., & Wang, X. (2024). Exact and heuristic methods for a university course scheduling problem. Expert Systems with Applications, 248. https://doi.org/10.1016/j.eswa.2024.123383.

[2] Zhaohui, F., 2000, A. L.-T. with A. Intelligence. I., & 2000, undefined. (n.d.). Heuristics for the exam scheduling problem. Ieeexplore.Ieee.OrgF Zhaohui, A LimProceedings 12th IEEE Internationals Conference on Tools with, 2000•ieeexplore.Ieee.Org. Retrieved October 21, 2024, from https://ieeexplore.ieee.org/abstract/document/889864/

[3] Pillay, N., & Banzhaf, W. (2007). A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4874 LNAI, 223–234. https://doi.org/10.1007/978-3-540-77002-2_19.

[4] Rina, I., Sulistiowati, D., & Raudhatuloktavi, D. (2022). Graph Coloring Applications in Scheduling Courses using Welch-Powell Algorithm - A Case Study. Proceeding - 2022 International Symposium on Information Technology and Digital Innovation: Technology Innovation During Pandemic, ISITDI 2022, 131–135. https://doi.org/10.1109/ISITDI55734.2022.9944511

[5] Egwuche, O. S. (2020, March 1). Examination Timetabling with Graph Coloring for Emerging Institutions. 2020 International Conference in Mathematics, Computer Engineering and Computer Science, ICMCECS 2020. https://doi.org/10.1109/ICMCECS47690.2020.246988

[6] Bania, R. K., & Duarah, P. (2018). Exam Time Table Scheduling using Graph Coloring Approach. International Journal of Computer Sciences and Engineering, 6(5), 84–93. https://doi.org/10.26438/ijcse/v6i5.8493

[7] Mursyidah, H. (2019). Graph edges coloring to determine lecture classroom of mathematics education department at muhammadiyah university of surabaya. Journal of Physics: Conference Series, 1188(1). https://doi.org/10.1088/1742-6596/1188/1/012096

[8] Lampung, U. B., Kesuma, R., #2, W., & Yusman, M. (n.d.). The Use of Edge Coloring Concept for Solving The Time Schedule Problem at Senior High School (Case Study at SMAN 9 Bandarlampung).

[9] 2019 1st AL-Noor International Conference for Science and Technology (NICST). (2019). IEEE.

[10] Han, X., & Wang, D. (2025). Gradual Optimization of University Course Scheduling Problem Using Genetic Algorithm and Dynamic Programming. Algorithms, 18(3). https://doi.org/10.3390/a18030158

[11] Budiarto, A. S., Satvika Iswari, N. M., & Dharma, E. M. (2024). Application of Genetic Algorithm for School Timetable Scheduling. 2024 International Conference on Informatics Electrical and Electronics, ICIEE 2024 - Proceedings. https://doi.org/10.1109/ICIEE63403.2024.10920446

[12] Rahardjo, E. Tjipto., & Zulkifli, F. Yuli. (2013). 2013 International Conference on QiR (Quality in Research). Faculty of Engineering, Universitas Indonesia.

[13] Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research, 176(1), 177–192. https://doi.org/10.1016/j.ejor.2005.08.012

[14] Rappos, E., Thiémard, E., Robert, S., & Hêche, J. F. (2022). A mixed-integer programming approach for solving university course timetabling problems. Journal of Scheduling, 25(4), 391–404. https://doi.org/10.1007/s10951-021-00715-5

[15] Steiner, E., Pferschy, U., & Schaerf, A. (2024). Curriculum-based university course timetabling considering individual course of studies. Central European Journal of Operations Research. https://doi.org/10.1007/s10100-024-00923-2

[16] Subulan, K. (2024). A multi-objective mathematical programming model for a novel capability-based university course timetabling problem. Journal of the Faculty of Engineering and Architecture of Gazi University, 40(1), 365–379. https://doi.org/10.17341/gazimmfd.1391236

[17] Nakasuwan, J., Asia, P. S.-S. & T., & 1999, undefined. (n.d.). Class scheduling optimization. Thaiscience.Info. Retrieved January 28, 2025, from https://www.thaiscience.info/Journals/Article/TSTJ/10480612.pdf

[18] Gunawan, A., Leng, K., Gunawan, A. ;, Ng, M. ;, & Poh, K. L. (2007). Solving the teacher assignment-

course scheduling problem by a Solving the teacher assignment-course scheduling problem by a hybrid Algorithm hybrid Algorithm Kien Ming NG Citation Citation Solving the teacher assignment-course scheduling problem by a hybrid Algorithm. In International Journal of Computer, Information, and Systems Science, and Engineering (Vol. 1, Issue 2). https://ink.library.smu.edu.sg/sis_research

[19] Computers & Informatics (ISCI), 2013 IEEE Symposium on. (2013). Institute of Electrical and Electronics Engineers.

[20] Fizzano, P., & Swanson, S. (2000). Scheduling Classes on a College Campus. In Computational Optimization and Applications (Vol. 16).

[21] Özavci, A., & Yakut1, S. (n.d.). Based on Malatya Centrality Algorithm Development of Suggestion System in Social Platforms and Commercial Applications.

[22] Karci, A., Yakut, S., & Öztemiz, F. (2022). A New Approach Based on Centrality Value in Solving the Minimum Vertex Cover Problem: Malatya Centrality Algorithm. Computer Science. https://doi.org/10.53070/bbd.1195501

[23] Yakut, S., Öztemiz, F., & Karci, A. (2023). A new robust approach to solve minimum vertex cover problem: Malatya vertex-cover algorithm. Journal of Supercomputing, 79(17), 19746–19769. https://doi.org/10.1007/s11227-023-05397-8

[24] YAKUT, S., & BAKAN, C. T. (2023). Development of Text Summarization Method based on Graph Theory and Malatya Centrality Algorithm. Computer Science. https://doi.org/10.53070/bbd.1350971

[25] UniTime | University Course Timetabling Benchmark Benchmark Datasets. (n.d.). Retrieved January 28, 2025, from https://www.unitime.org/uct_datasets.php