Fırat Üniversitesi Müh. Bil. Dergisi
37(2), 611-621, 2025

Araştırma Makalesi
https://doi.org/10.35234/fumbd.1646674

An Innovative Solution to the Map Coloring Problem Using the Malatya Vertex Coloring Algorithm

Cezayir KARACA[1*], Selman YAKUT[2]
[1] Computer Engineering, Faculty of Engineering, İnönü University, Malatya, Türkiye
[2] Software Engineering, Faculty of Engineering, İnönü University, Malatya, Türkiye
[*1] cezayirkaraca0242@gmail.com, [2] selman.yakut@inonu.edu.tr

(Geliş/Received: 25/02/2025; Kabul/Accepted:13/06/2025)

Abstract: The map coloring problem is a classical NP-complete problem that requires adjacent regions to be colored differently and is encountered in many real-world applications. Numerous algorithms have been developed to solve this problem. In this study, the Malatya Vertex Coloring (MVC) Algorithm, which presents a novel and original approach to solving the problem, is applied. This algorithm aims to identify influential vertices to reduce the number of colors used in graphs and to complete the coloring process more efficiently. Additionally, the applicability of the algorithm to real-world problems is also evaluated. The MVC Algorithm calculates the Malatya Centrality value for each vertex in the graph; it selects the vertex with the highest value, colors it with a color different from its neighbors, and then removes it from the graph. This process continues until all vertices are colored. The algorithm has been successfully applied to maps of Asia, Europe, districts of Istanbul, Turkey, U.S. states, and the world, and the results demonstrate the effectiveness of the algorithm. The advantages of the MVC Algorithm include its predictability, as well as its ability to operate in polynomial time and space. In this respect, the MVC Algorithm offers an alternative solution approach to the classical Four Color Theorem in the context of the map coloring problem.

Key words: Centrality, graph coloring, map coloring, planar graph.

Malatya Düğüm Renklendirme Algoritması Kullanılarak Harita Renklendirme Problemi için Yenilikçi Bir Çözüm

Öz: Harita renklendirme problemi, bitişik bölgelerin farklı renklerle boyanmasını gerektiren ve birçok gerçek hayat uygulamasında karşılaşılan klasik bir NP-tam problemdir. Bu problemin çözümü için birçok algoritma geliştirilmiştir. Bu çalışmada, ilgili problemin çözümüne yönelik yeni ve özgün bir yaklaşım ortaya koyan Malatya Vertex Coloring (MVC) Algoritması uygulanmıştır. Bu algoritma, graflarda kullanılan renk sayısını azaltmak için etkili düğümleri belirlemekte ve renklendirme sürecini daha kısa sürede tamamlamayı hedeflemektedir. Ayrıca algoritmanın, gerçek hayat problemlerine uygulanabilirliği de değerlendirilmiştir. MVC Algoritması, graf yapısındaki her düğüm için Malatya Merkezilik değerini hesaplar; en yüksek değere sahip düğümü seçerek komşularından farklı bir renkle boyar ve ardından bu düğümü graf üzerinden çıkarır. Bu işlem, tüm düğümler renklendirilene kadar devam etmektedir. Algoritma; Asya, Avrupa, İstanbul ilçeleri, Türkiye, ABD eyaletleri ve Dünya haritaları üzerinde başarıyla uygulanmış ve elde edilen sonuçlar algoritmanın etkinliğini ortaya koymuştur. MVC Algoritması'nın avantajları arasında öngörülebilir olması, polinom zamanda ve polinom uzayda çalışabilmesi yer almaktadır. Bu yönüyle, MVC Algoritması'nın harita renklendirme problemine ilişkin olarak klasik Dört Renk Teoremi'ne alternatif bir çözüm yaklaşımı sunduğu ortaya konulmuştur.

Anahtar kelimeler: Merkezilik, graf renklendirme, harita renklendirme, malatya coloring, düzlemsel graf.

1. Introduction

The Graph Coloring Problem (GCP) was first introduced in 1852 by Francis Guthrie while coloring a map of England. Guthrie noticed that adjacent cities could be assigned different colors and that this could be achieved using only four colors [1]. There are two fundamental principles in graph coloring. The first is the requirement that adjacent vertices must be assigned different colors. The second is the objective of using the minimum number of colors possible. Graph coloring has been applied in various domains, including map coloring [2], frequency assignment [3], Sudoku [4], scheduling [5], register allocation [6], and timetable generation [7].

Map coloring refers to the process of assigning different colors to neighboring regions within bounded areas. This method is studied as a mathematical problem and has applications in various fields. The main goal is to ensure that adjacent regions are assigned different colors while minimizing the total number of colors used. In 1977,

[*] Corresponding author: cezayirkaraca0242@gmail.com. Yazarların ORCID Numarası: [1] 0009-0007-9250-2612, [2] 0000-0002-0649-1993

Kenneth Appel and Wolfgang Haken proved that any planar graph can be colored using four colors. Assigning different colors to adjacent regions on a map while using the fewest colors has been proven to be an NP-complete problem [8].

Although various algorithms have been developed to solve the map coloring problem, certain limitations still persist. Most existing algorithms are optimized for specific types of maps or graph structures, resulting in limited generalizability. Furthermore, computational cost increases significantly in large and complex datasets, leading to longer solution times and sometimes suboptimal results. While heuristic and approximate algorithms may be effective in practice, they cannot guarantee the minimum number of colors in every case. Additionally, there are significant gaps in developing adaptive and real-time solutions for situations where region boundaries change dynamically.

To facilitate the coloring of maps by algorithms, maps were transformed into planar graphs. Planar graphs are defined as graphs in which edges intersect only at vertices—meaning no two edges cross each other when drawn [2]. In this representation, each region on the map corresponds to a vertex, and the adjacency between regions is represented as an edge [9]. In this context, graph coloring algorithms began to be used to solve the Map Coloring Problem (MCP). The literature indicates that no universally effective general solution exists for GCP, and various exact and approximate algorithms have been proposed for its resolution [10]. Methods such as Ant Colony Optimization [11], Tabu Search [12], Genetic Algorithms [13], and Quantum Search Algorithms [14] have been evaluated for this purpose. Each of these methods has its own strengths and weaknesses.

In this study, the MVC algorithm [15], which was previously proposed for the Graph Coloring Problem, is applied to provide a solution to the MCP. The MVC algorithm aims to color a given map using the minimum number of colors. During this process, the primary constraint is to prevent adjacent regions from being assigned the same color. The algorithm identifies the most critical region and assigns colors accordingly. Thus, optimal or near-optimal color usage is achieved between neighboring regions. The main advantages of the MVC algorithm include the predictability of all iteration and execution steps, as well as its ability to operate with polynomial time and space complexity. In this study, the MVC algorithm is adapted and used to solve the MCP.

The remainder of this paper is organized as follows: Section 2 presents the literature related to map and graph coloring. Section 3 introduces the proposed method and approach. Section 4 presents the results of the map coloring applications. Section 5 concludes the paper. According to the literature, MCP is identified as an NP-complete problem. With the advancement of computer science, numerous approaches and algorithms have been proposed to solve this problem.

2. Literature Review

Hong B. proposed the use of a genetic algorithm to solve the Graph Coloring Problem (GCP) [16]. Lui and Zang conducted research on a DNA-based algorithm for graph coloring. Ruxin Zhao and colleagues proposed the Selfish Herd Optimizer (DSHO) algorithm as an effective solution for GCP [17]. The Turbulent Particle Swarm Optimization (MTPSO) algorithm developed by Ling-Yuan et al. was shown to outperform the PSO algorithm developed by Cui et al. in 2008 [18]. Guangzhao Cui and colleagues developed a Modified PSO algorithm for solving the planar GCP. Test results demonstrated that the Modified PSO outperforms the classical PSO [10]. Colin Campbell and Edward Dahl proposed the Quantum Approximate Optimization Algorithm (QAQA), supported by application teams. In their article, they aimed to demonstrate that using higher-order terms can significantly enhance the performance of QAQA [14]. Kui and Hitoshi introduced the Discrete Firefly Algorithm (DFA) for GCP. Their study showed that the proposed DFA is based on similarity calculation, does not require defining extra operators, does not need the addition of new strategies, and is a non-hybrid algorithm [19].

Bui T.N., Nguyen T.H., and colleagues stated that the ant algorithm they proposed differs from previous ant-based algorithms, as each ant colors only a part of the graph using local information. This individual coloring forms a section of the graph. Despite certain limitations, the algorithm demonstrated good performance [20]. In the same study, the author proposed a local flower pollination algorithm to solve the planar GCP. The proposed algorithm achieved a higher success rate in coloring compared to Particle Swarm Optimization (PSO), Basic Flower Pollination Algorithm (FPA), and Differential Evolution (DE) [21]. Surbakti N. and Ramadhani F. addressed the map coloring problem using a greedy algorithm approach with a minimum number of colors. The map was colored using four colors, ensuring that neighboring provinces received different colors [22]. Lingli Zhao and colleagues applied greedy algorithms to MCP, arguing that such algorithms are suitable for many real-world problems. Although many algorithms have been used to solve this problem, they tend to be complex; in contrast, the greedy algorithm allowed for a simpler and more effective solution by prioritizing the provinces with the most connections [23].

Another study proposed an artificial neural network based on harmony theory to solve the map coloring problem. This approach determines whether MCP can be solved with a pre-specified number of colors and identifies the minimum number of colors required. The implementation directly encodes the problem into the neural network and represents the solution through node actions [24]. In that study, a perturbation factor was added to the algorithm. When the algorithm stops progressing toward a global optimum, the perturbation resets a particle's velocity with a certain probability, helping it escape local optima. This algorithm was applied to solve GCP. Later, Talavan proposed a neural network method for solving GCP [25].

In this study, the MVC algorithm applied for solving the MCP offers a unique strategy based on the MC value for node selection, aiming to minimize the number of colors used in map coloring problems. Unlike commonly used algorithms in the literature, such as genetic algorithms, ant colony optimization, particle swarm optimization, or classical greedy methods, MVC determines the order of color assignments based on a more systematic and predictable criterion. Most existing algorithms incorporate randomness in the selection of the initial node or make decisions independently of the graph's global structure. The MVC algorithm, however, selects the central node in each iteration and recalculates accordingly, allowing it to adapt to the dynamic structure of the graph and produce successful results with fewer colors by minimizing conflicts. Additionally, its polynomial time and space complexity offer advantages in both theoretical and practical applications.

3. Materials and Methods

The Graph Coloring Problem (GCP) is a significant optimization problem that has been widely studied and has numerous practical applications. In recent years, due to the expansion of its application domains and advancements in solution strategies, GCP has attracted considerable attention from researchers. The primary objective of GCP is to assign different colors to adjacent nodes while minimizing the total number of colors used.

The MVC (Malatya Vertex Coloring) algorithm, proposed as a solution to the GCP, is a coloring method that considers the centrality values of the nodes. The aim of the algorithm is to prioritize the coloring of the most influential nodes, thereby minimizing the total number of colors used and reducing conflicts. In order to apply the MVC algorithm to the Map Coloring Problem (MCP), the problem must first be represented in the form of a graph. To make real-world data such as maps suitable for algorithmic processing, some preprocessing steps are required.

As an example of this process, the district map of Adıyaman, shown in Figure 1, was used. In the first stage (Stage 1), each region or area is represented as a node, and the adjacency relationships between regions are defined as edges. In this way, the map is transformed into a graph structure consisting of nodes and edges (Stage 2). The MVC algorithm is then applied to the resulting graph to assign appropriate colors to each node, producing a colored graph as seen in Stage 3. In the final stage (Stage 4), the colors assigned to each node in the graph are mapped back to the corresponding regions on the original map, resulting in a successfully colored map.

The MVC algorithm is a graph coloring algorithm that operates in two main steps, based on the centrality (MC) values of the nodes. In the first step, the MC values of all nodes in the graph are calculated. In the second step, nodes are selected according to these values, and appropriate colors are assigned to complete the coloring process. The general operation of the algorithm is as follows: First, the MC values of all nodes in the input graph are calculated, and the node with the highest MC value is identified and assigned a color. This colored node is then removed from the graph. Next, the MC values of the remaining nodes are recalculated, and the node with the highest MC value is again selected. A color that does not conflict with the colors of its neighboring nodes is assigned to this node, which is then removed from the graph. This process repeats until all nodes are colored.

The pseudocode of the algorithm is presented in detail in Algorithm 1. The MVC algorithm consists of two main components: the Main Function and the CentralityCalculate function. The CentralityCalculate function computes the MC value for each node, identifies the node with the highest value, and returns it. This node is then assigned a color that does not conflict with those of its neighbors, and subsequently removed from the graph. This cycle continues until no uncolored nodes remain in the graph. Once the loop is completed, all nodes in the copied graph are assigned a color, and the graph is fully colored.

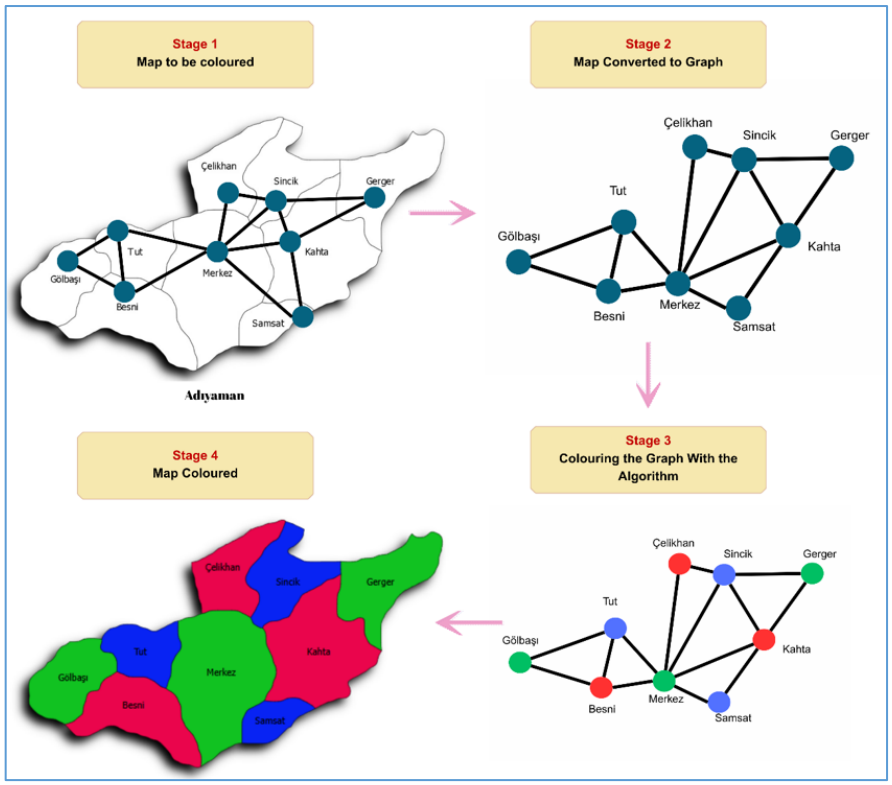

Figure 1. Coloring process of the district map of Adıyaman using the MVC Algorithm.

Algorithm 1. Pseudocode of the MVC Algorithm

MVC Algorithm

```
g ← (V, E)
f ← Copy(g)
CentralityCalculate ← function(g)
    MaxDegree ← -∞
    for i in V(g)
        Centrality ← 0
        for j in Neighbors(i)
            Centrality ← Centrality + Degree(i) / Degree(j)   // The MC value is calculated by
                                                              //summing the ratios of the node's
                                                              //degree to the degrees of its neighbors.
        if Centrality > MaxDegree
            MaxDegree ← Centrality
            Vertex ← i
    return Vertex
while VCount(g) > 0
    vertex ← CentralityCalculate(g)
    for i in Neighbors(f, vertex)
        Color(f[vertex]) ← Color(f[i])
    g ← DeleteVertex(g, vertex)
figure
Show(f)
```

The MC value is calculated as the sum of the ratios of the selected node's degree to the degrees of its neighboring nodes.

To demonstrate the application process of the MVC algorithm, a map of the Southeastern provinces of Turkey was used. The map was converted into a graph structure for coloring, and the adjacency table of the Southeastern provinces was input into the system. The MC (Modular Centrality) value of each node was calculated by summing the ratios of its degree to the degrees of its neighboring nodes. In this way, all nodes' MC values were computed. As shown in Table 1, the province with the highest value was Mardin. Therefore, a color was assigned to Mardin starting from the beginning of the color list, and then it was removed from the graph. After the graph was updated, all MC values were recalculated, and in the second iteration, Gaziantep was determined to have the highest MC value. While assigning a color to Gaziantep, care was taken to avoid using the same color as its neighboring provinces. Since none of Gaziantep's neighbors had yet been colored, the first color in the list was assigned to this node, and it was subsequently removed from the graph. Following the update and recalculation of MC values, Diyarbakır emerged as the province with the highest MC value in the third iteration. Since Mardin, a neighbor of Diyarbakır, had already been assigned a color, a different color from the list was selected for Diyarbakır, which was then removed from the graph. In the fourth iteration, Siirt had the highest MC value among the remaining nodes. To determine the color for Siirt, the colors of its neighboring provinces were reviewed. Since only Mardin had been assigned a color among Siirt's neighbors, a color different from that of Mardin was chosen. In the fifth iteration, Adıyaman and Şanlıurfa had equal MC values. The province that comes first alphabetically, Adıyaman, was selected, and its coloring process was carried out. Upon reviewing Adıyaman's neighbors, it was found that Gaziantep and Diyarbakır had already been colored; hence, a color different from these was assigned to Adıyaman. The remaining provinces had an MC value of zero since their neighbors had been removed from the graph. Each of these was subsequently colored using a color not used by its neighbors.

As a result, the Southeastern region of Turkey was successfully colored using the MVC algorithm. The changes in MC values at each step of the algorithm are presented in Table 1.

Table 1. MC values of southeastern provinces.

.	Adıyaman	Şanlıurfa	Gaziantep	Diyarbakır	Kilis	Mardin	Batman	Siirt	Şırnak
Iteration-1	2.5	4.46	4.75	4.46	0.3	8.3	2.35	3.1	1.06
Iteration-2	3.0	3.0	5.0	3.5	0.3		1.6	3.0	0.5
Iteration-3	1.6	1.6		4.5	0		1.6	3.0	0.5
Iteration-4	1.0	1.0			0		0.5	4.0	0.5
Iteration-5	1.0	1.0			0		0		0
Iteration-6		0			0		0		0
Iteration-7					0		0		0
Iteration-8							0		0
Iteration-9									0

In each step of the MVC algorithm, the coloring of the province with the highest MC value was prioritized. Following each execution step, the map coloring process was visualized through the stages illustrated in Figure2.

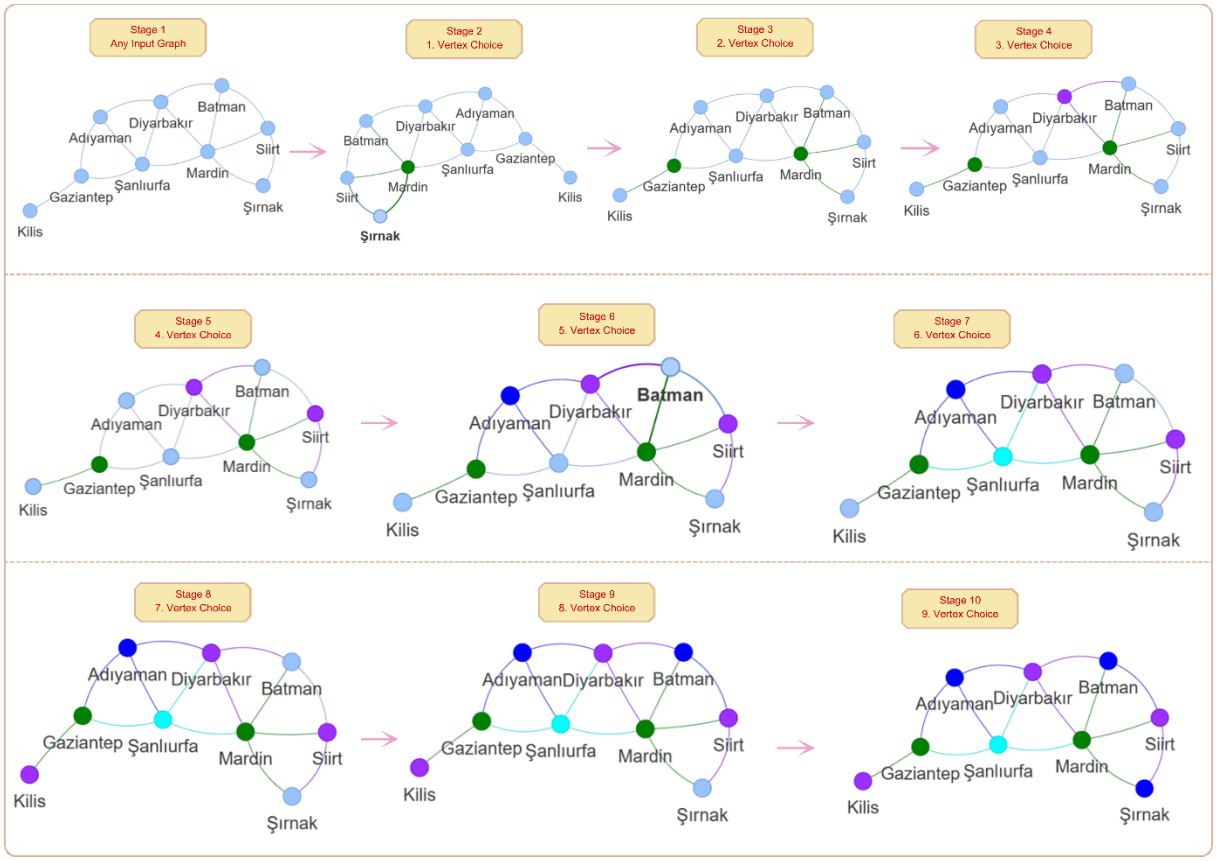

Figure 2. Step-by-step coloring of the Southeastern provinces using the proposed method.

The algorithm was applied to the Asia Map, Europe Map, Istanbul Districts Map, Turkey Map, World Map, and finally the US States Map for the purpose of map coloring, yielding successful results.

4. Map Application Coloring Results

In the graph modeling performed on the map, each region separated by borders was defined as a node, and the relationships between adjacent regions were represented by edges. In the application specifically conducted on the Asia map, 44 countries were modeled as nodes, and the 81 adjacency relationships between them were converted into a graph structure. The MVC algorithm was applied to this graph to ensure that neighboring countries were represented with different colors. As a result of the application, the entire map was successfully colored using only four distinct colors. This outcome is visualized in Figure 3, where each country is represented by a different color, and the principle that neighboring countries do not share the same color is maintained.

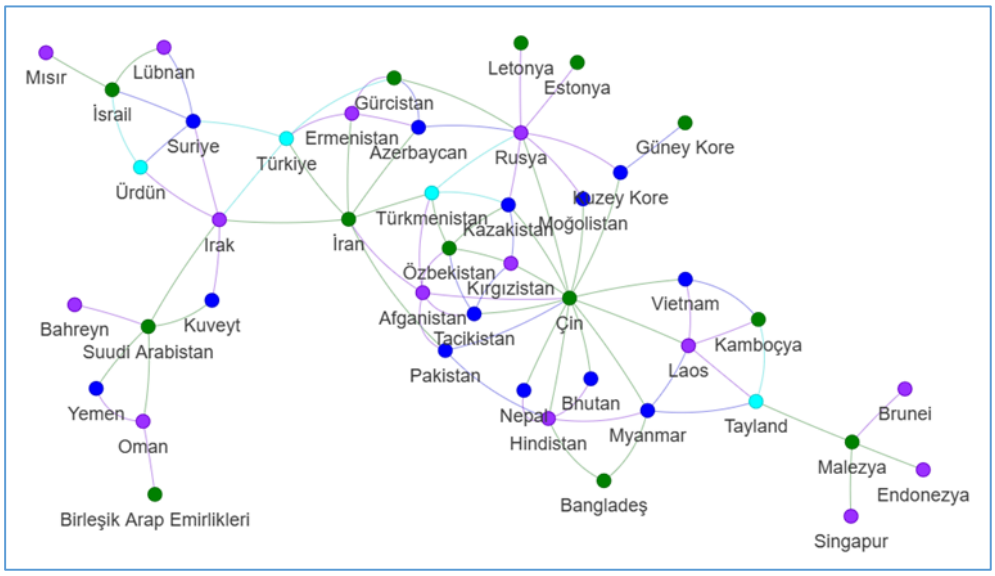

Figure 3. Coloring of the Asia Map.

The Europe map was transformed into a graph structure consisting of nodes representing 40 countries and edges depicting 72 adjacency relationships between them. The MVC algorithm was applied to this graph to ensure that neighboring countries were represented by different colors. As a result of the coloring process, an optimal color distribution was achieved on the map using only four colors. The obtained results are visualized in Figure 4, demonstrating that adjacent countries do not share the same color and that the minimum number of colors was successfully utilized.

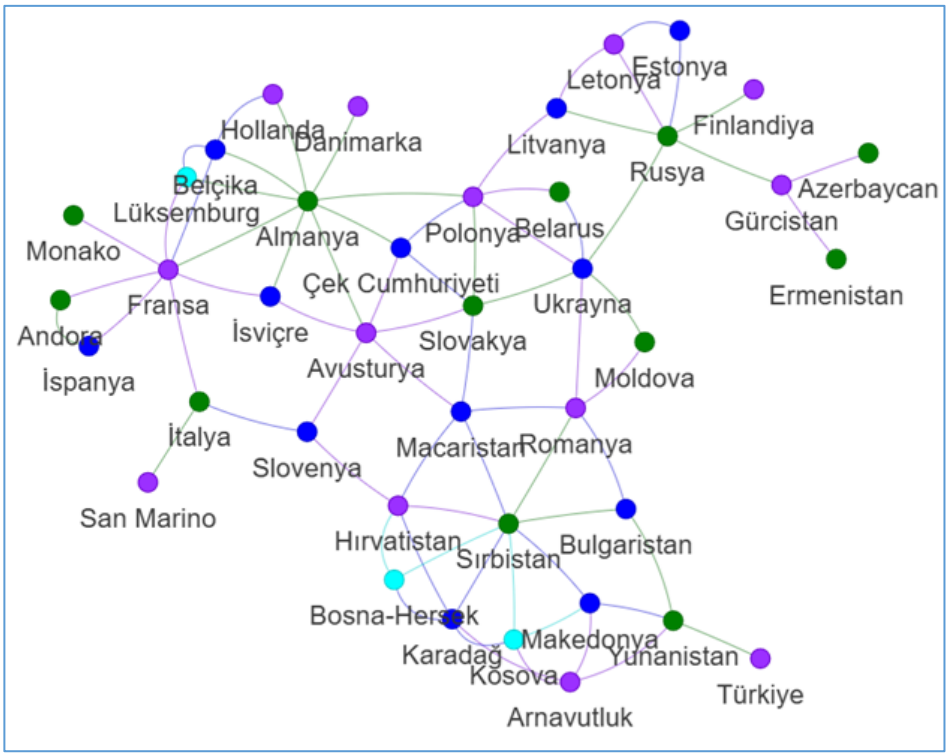

Figure 4. Coloring of the Europe Map.

The map covering Istanbul's 39 districts was transformed into a graph structure, where the districts were represented as nodes and the 76 adjacency relationships between them as edges. By applying the MVC algorithm to this graph, neighboring districts were assigned different colors, achieving a successful coloring using a total of four colors. According to the results visualized in Figure 5, adjacent districts do not share the same color, and an effective solution was obtained with minimal color usage.

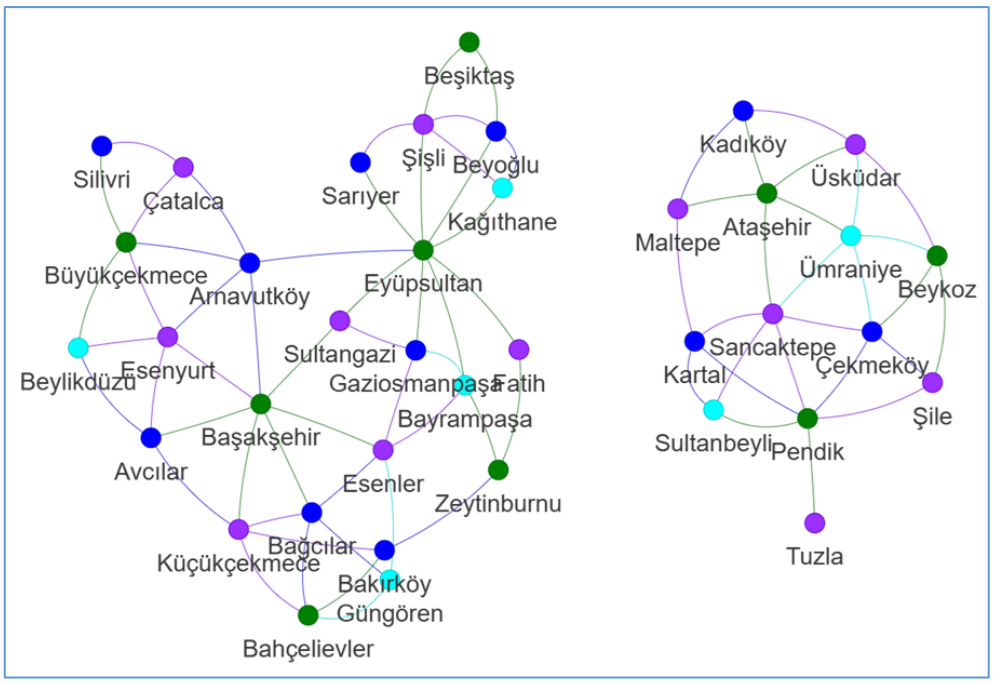

Figure 5. Coloring of the Istanbul Districts Map

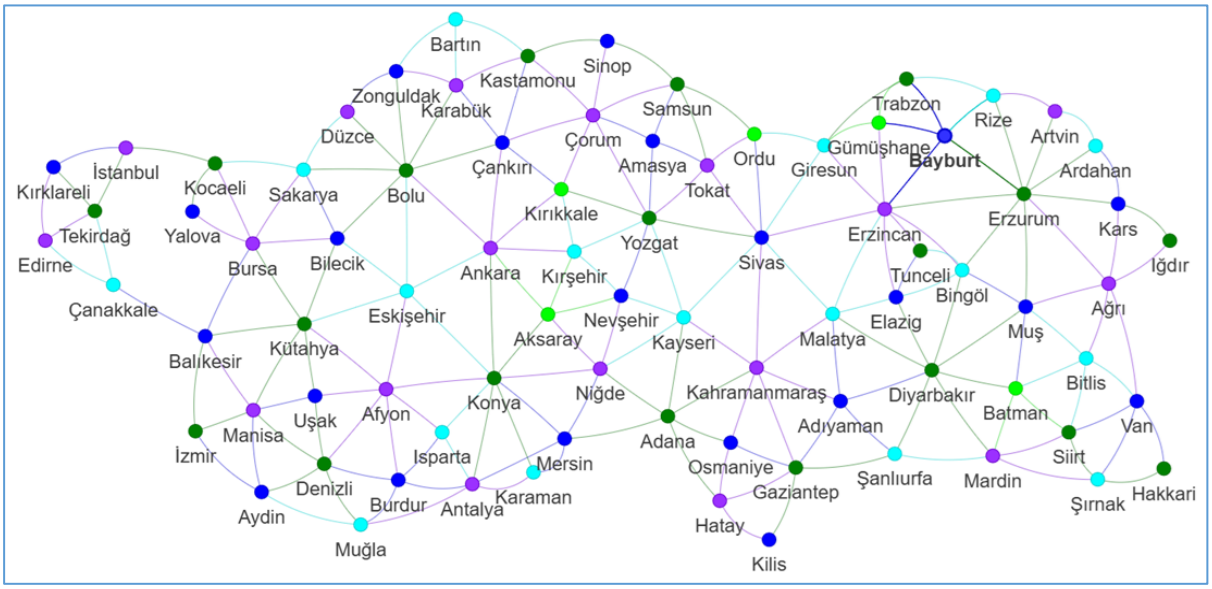

Figure 6. Coloring of Turkey Map

The map including the US states was modeled as a graph structure consisting of 51 nodes and 108 edges. By applying the MVC algorithm to this graph, neighboring states were colored with different colors, and the coloring process was completed using a total of five distinct colors. The obtained results are visualized in Figure7.

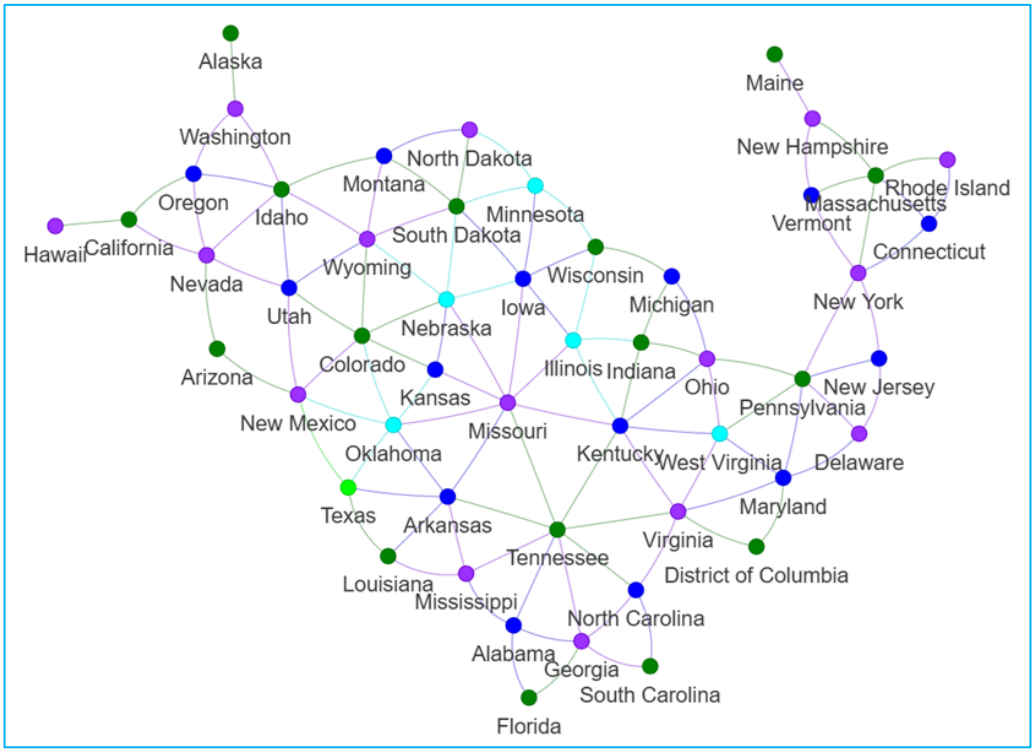

Figure 7. Coloring of the US States Map.

Finally, the algorithm was applied to the world map. In this application, 249 countries were modeled as nodes, and 728 adjacency relationships between the countries were represented by edges. The coloring process performed using the MVC algorithm ensured that all adjacent countries were displayed with different colors using only five distinct colors. The colored world map is presented in Figure 8.

Table 2 presents a comparative summary of the coloring results obtained by applying different algorithms to various geographical maps. The MVCA algorithm produced successful outcomes by using a number of colors equal to or comparable with those of existing classical algorithms. Notably, in the US States map, while the RLF algorithm utilized five colors, the MVCA algorithm achieved better performance using only four colors. For the other maps, MVCA yielded results on par with competing algorithms in terms of the number of colors used. These findings demonstrate that the MVC algorithm offers a competitive and effective alternative for map coloring problems.

Table 2. MC Values of Southeastern Provinces

Applied Maps	Number of Nodes	Number of Edges	Welsh-Powell	Greedy Color	Dsatur	RLF	MVCA
Asia Map	44	81	4	4	4	4	4
Europe Map	40	72	4	4	4	4	4
Istanbul Districts Map	39	76	4	4	4	4	4
Turkey Map	81	200	5	5	5	5	5
US States Map	51	108	4	4	4	5	4
World Map	249	320	5	5	5	5	5

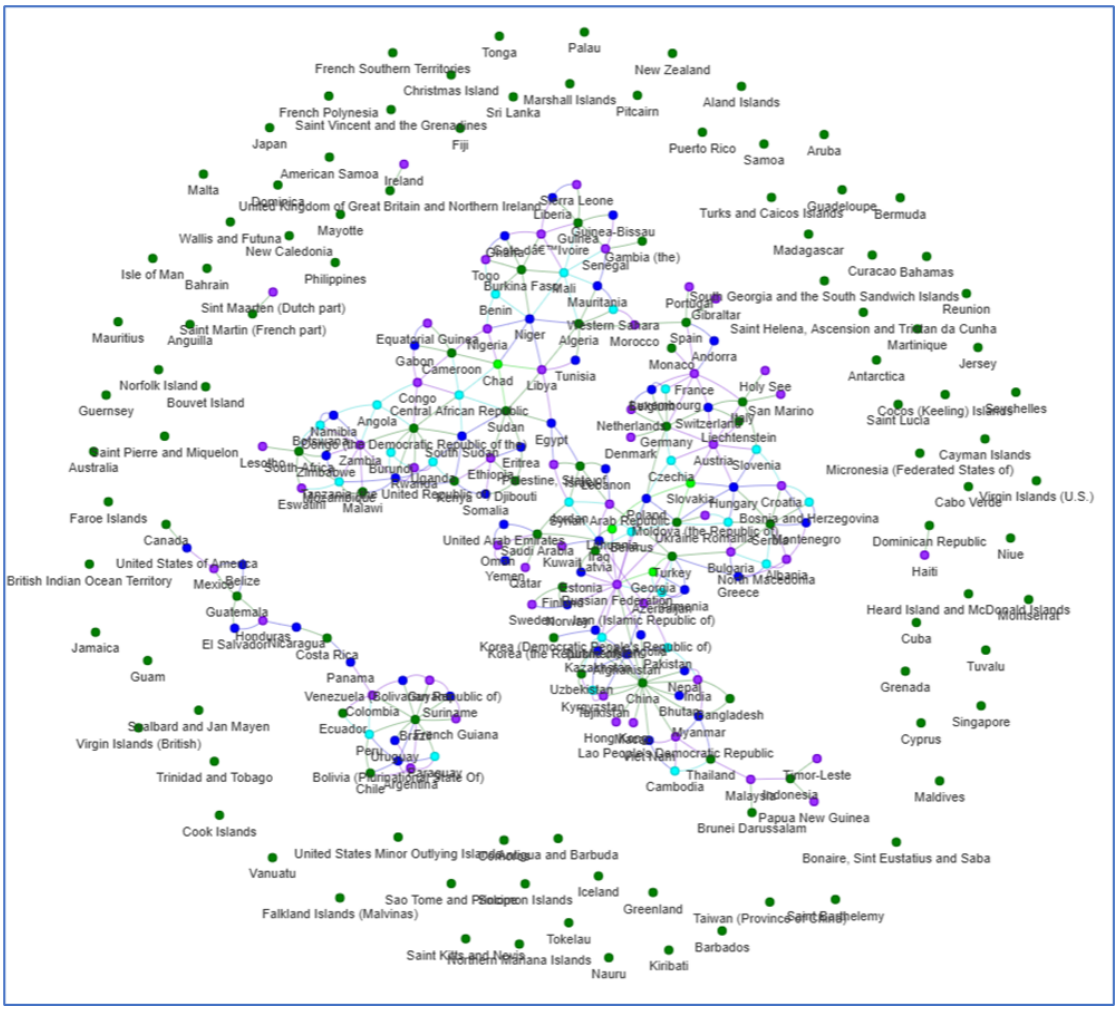

Figure 8. Coloring of the World Map.

5. Conclusion

In this study, a novel and original approach called the Malatya Vertex Coloring (MVC) Algorithm was developed and applied to solve the map coloring problem (MCP). Unlike classical graph coloring algorithms, the proposed algorithm assigns colors based on the MC values of the nodes, starting from the most influential node. This approach ensures that adjacent nodes receive different colors while minimizing the total number of colors used.

The MVC algorithm consists of two main steps: First, the centrality values of all nodes are calculated; then, a color is assigned to the node with the highest value in a way that does not conflict with the colors of its neighbors, and the node is removed from the graph. This process continues until all nodes are colored.

The distinctive feature of this study compared to existing literature is the use of a systematic strategy based on centrality values for selecting the initial node instead of randomness, making the coloring process predictable and reproducible. Additionally, the MVC algorithm operates with polynomial time and space complexity, demonstrating its theoretical and practical applicability.

The algorithm was tested on various maps of different scales and complexities, including Asia, Europe, Istanbul districts, Turkey, US states, and the world. The results obtained are as follows:

* The Asia map was modeled as a graph with 44 nodes and 81 edges and colored using 4 colors.
* The Europe map, represented by 40 nodes and 72 edges, was also colored with 4 colors.
* The Istanbul districts map, composed of 39 nodes and 76 edges, was successfully colored with 4 colors.
* The Turkey map, modeled with 81 nodes and 393 edges, required 5 colors.
* The US states map, represented by 51 nodes and 108 edges, used 5 colors.

* Finally, the world map, converted into a graph with 249 nodes and 728 edges, was colored using 5 colors.

In the specific case study of Southeastern Anatolian provinces, the algorithm selected the most influential node in each iteration, resulting in an effective color distribution.

As demonstrated in the comparative tables, the MVC algorithm provides efficient solutions using the same or fewer colors compared to classical algorithms. Notably, in certain maps (e.g., the US states map), it achieved near-optimal results using fewer colors than other methods.

In conclusion, the MVC algorithm stands out as a method with strong theoretical foundations and successful practical applications, offering an effective and alternative solution to the map coloring problem. Future work may focus on applying this algorithm to different problem types or enhancing its structure through hybrid systems.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Selman Yakut, for his valuable contributions and guidance throughout every stage of this study, and to my family for their unwavering support.

References

[1] Fritsch R, Fritsch G. Four-Color Theorem. New York, NY, USA: Springer-Verlag; 1998.
[2] Zhou Y, Zheng H, Luo Q, Wu J. An improved Cuckoo Search Algorithm for Solving Planar Graph Coloring Problem. Appl Math Inf Sci 2013; 7(2): 785-792.
[3] Hale WK. Frequency assignment: Theory and applications. Proc IEEE 1980; 68(12): 1497-1514.
[4] Ono S, Miyamoto R, Nakayama S, Mizuno K. Difficulty estimation of number place puzzle and its problem generation support. In: 2009 ICCAS-SICE; August 2009; Fukuoka, Japan. New York, NY, USA: IEEE. pp. 4542-4547.
[5] Chmeit N. Using simulated annealing and ant-colony optimization algorithms to solve the scheduling problem. PhD Thesis, Notre Dame University-Louaize, Lebanon, 2012.
[6] Chaitin GJ, Auslander MA, Chandra AK, Cocke J, Hopkins ME, Markstein PW. Register allocation via coloring. Comput Lang 1981; 6(1): 47-57.
[7] Altunay H, Eren T. A literature review for course scheduling problem. Pamukkale Univ J Eng Sci 2017; 23(1): 55-70.
[8] Appel K, Haken W. Every planar map is four colorable. In: Mathematical Solitaires and Games. New York, NY, USA: Routledge; 2019. pp. 145-152.
[9] Saxena S, Thapar A, Bansal R. Total fuzzy graph coloring. J Hyperstructures 2022; 11(1): 84-108.
[10] Cui G, Qin L, Liu S, Wang Y, Zhang X, Cao X. Modified PSO algorithm for solving planar graph coloring problem. Prog Nat Sci 2008; 18(3): 353-357.
[11] Ahn S, Lee S, Chung TC. Modified ant colony system for coloring graphs. In: Fourth Int Conf Information, Communications and Signal Processing; 2003; Singapore. New York, NY, USA: IEEE.
[12] Marappan R, Sethumadhavan G. Solution to graph coloring using genetic and tabu search procedures. Arab J Sci Eng 2018; 43(2): 525-542.
[13] Ardelean SM, Udrescu M. Graph coloring using the reduced quantum genetic algorithm. PeerJ Comput Sci 2022; 8: e836.
[14] Campbell C, Dahl E. QAOA of the highest order. In: 2022 IEEE 19th Int Conf Software Architecture Companion (ICSA-C); 2022; Montréal, Canada. New York, NY, USA: IEEE. pp. 141-146.
[15] Karcı A, Yakut S, Öztemiz F. A new approach based on centrality value in solving the minimum vertex cover problem: Malatya centrality algorithm. Computer Science 2022.
[16] Hong B. Generic algorithm of color planar graph. J Guizhou Univ (Nat Sci) 1999; 11(16): 232-297.
[17] Zhao R ve diğerleri. Discrete selfish herd optimizer for solving graph coloring problem. Appl Intell 2020; 50(5): 1633-1656.
[18] Hsu LY, Horng SJ, Fan P, Khan MK, Wang YR, Run RS, Chen RJ ve diğerleri. MTPSO algorithm for solving planar graph coloring problem. Expert Syst Appl 2011; 38(5): 5525-5531.
[19] Zhao R, Wang Y, Liu C, Hu P, Jelodar H, Rabbani M, Li H. Discrete selfish herd optimizer for solving graph coloring problem. Appl Intell 2020; 50(5): 1633-1656.
[20] Bui TN, Nguyen TH, Patel CM, Phan KAT. An ant-based algorithm for coloring graphs. Discrete Appl Math 2008; 156(2): 190-200.
[21] Wang R, Zhou Y, Zhou Y, Bao Z. Local greedy flower pollination algorithm for solving planar graph coloring problem. J Comput Theor Nanosci 2015; 12(11): 4087-4096.
[22] Surbakti NM, Ramadhani F. Implementation of the greedy algorithm for coloring graph based on four-color theorem. Sudo J Tek Inf 2022; 1(4): 178-182.
[23] Luo Q. The 2nd Conference on Environmental Science and Information Application Technology: ESIAT 2010; July 17-18, 2010; Wuhan, China. New York, NY, USA: IEEE; 2010.
[24] Tambouratzis T. A consensus-function artificial neural network for map-coloring. IEEE Trans Syst Man Cybern B Cybern 1998; 28(5): 721-728.
[25] Talaván PM, Yáñez J. The graph coloring problem: A neuronal network approach. Eur J Oper Res 2008; 191(1): 100-111.