International Journal of Engineering Applied Sciences and Technology, 2024
Vol. 9, Issue 06, ISSN No. 2455-2143, Pages 153-157
Published Online October 2024 in IJEAST (http://www.ijeast.com)

CHROMATIC NUMBER OPTIMIZATION FOR EFFICIENT MAP COLORING OF MAHARASHTRA'S DISTRICTS

Dr. Megha Abhiman Bhamare
Department of Mathematics,
K.V.N.Naik College, Nashik, Maharashtra, India

Abstract— **This research explores the application of graph coloring algorithms to minimize the number of colors required for coloring the districts of Maharashtra, ensuring that no two adjacent districts share the same color. Using an adjacency matrix representation, where each district corresponds to a vertex and edges represent shared borders, the problem is framed as a graph coloring challenge. The primary objective is to demonstrate that a minimum of four colors suffice to color the entire map, in line with the Four Color Theorem, which asserts that any planar map can be colored with at most four colors. A greedy graph coloring algorithm, augmented with backtracking for conflict resolution, is implemented to assign colors to the districts. The results show that the algorithm successfully colors the 36 districts of Maharashtra using no more than four colors, confirming the hypothesis. This research has significant implications for geographical information systems (GIS), political mapping, resource allocation, and optimization problems, offering a practical solution to district-level planning. The methodology and outcomes also suggest potential extensions to other regions and real-world applications, such as task scheduling and frequency assignment in telecommunications.**

Keywords— **Graph Coloring, Geographical Mapping, Greedy Algorithm, Four Color Theorem Adjacency Matrix, Planar Graphs, Optimization, District Mapping**

I. INTRODUCTION

The graph coloring problem is a well-established concept in graph theory [1], where the goal is to assign a color to each vertex of a graph such that no two adjacent vertices share the same color [2]. This problem has practical applications in various domains, including geographical mapping, task scheduling, and resource allocation. In geographical mapping [3], district boundaries often form a planar graph where each district represents a vertex, and borders between districts are represented as edges [4]. In such problems, the objective is to minimize the number of colors used to color the graph while ensuring that no two adjacent districts share the same color [5].

The state of Maharashtra in India, comprising 36 districts [6], serves as a case study for applying the graph coloring problem to a real-world geographical map [7]. By leveraging graph coloring algorithms, specifically a greedy algorithm combined with backtracking. [8]

The graph coloring problem has been an area of extensive research in the last few decades, especially in the context of practical applications such as geographical mapping, network design, resource allocation, and political districting. Below is a summary of recent developments and techniques in this domain

The greedy algorithm is a popular method for solving graph coloring problems due to its simplicity and efficiency [9]. The goal of graph coloring is to assign a color to each vertex in such a way that no two adjacent vertices share the same color [10]. However, recent studies have sought to improve the classical greedy approach by incorporating degree-based heuristics and local search techniques, which help handle graphs with specific patterns such as those found in political districting and resource management [9].

In situations where finding an optimal solution is necessary, backtracking and branch-and-bound algorithms are used [11]. These exact methods guarantee finding the chromatic number of a graph by exploring all possible colorings and backtracking when an invalid coloring is encountered [12]. These approaches are computationally expensive but are effective for graphs with fewer vertices, such as political maps with low chromatic numbers [11-12].

Recent advancements in metaheuristic techniques like genetic algorithms, simulated annealing, and particle swarm optimization have been explored for graph coloring problems in complex scenarios [13-14]. These methods, although not guaranteed to find the optimal solution, are effective in producing near-optimal results for large-scale problems like radio frequency assignment and resource scheduling in dynamic environments, such as urban planning and districting) [13-14]

International Journal of Engineering Applied Sciences and Technology, 2024
Vol. 9, Issue 06, ISSN No. 2455-2143, Pages 153-157
Published Online October 2024 in IJEAST (http://www.ijeast.com)

As the size of graphs increases, parallel computing using GPU-based parallelization and multi-core processors has become essential to speed up the graph coloring process [14- 15]. These techniques are particularly useful for solving large geographical maps or optimization problems with thousands or millions of vertices, helping to color graphs efficiently in large-scale problems like political districting and resource allocation [14-15].
Graph coloring has also been extensively applied in geographical information systems (GIS) and political districting to optimize the distribution of resources (Sharma et al., 2023; Rathi, A., & Kumar, S., 2024). These applications ensure that no two adjacent districts, such as electoral regions or administrative units, are assigned the same color, which is crucial for problems like voting districts and public health infrastructure [16-18].
The Four-Color Theorem states that any planar graph can be colored with no more than four colors such that no two adjacent vertices share the same color [18]. Research on algorithms for planar graphs has focused on improving the efficiency of these algorithms to meet the conditions of the theorem in real-world maps, including those with constraints such as population density and political boundaries [18-19].
In urban planning, efficient graph coloring algorithms have been extended to solve problems such as task scheduling and frequency assignment in cities [14, 17-18]. These advancements aim to minimize the number of resources required for optimal scheduling and task allocation while ensuring that adjacent tasks or services do not interfere with each other [14, 17].
This research aims to determine the minimum number of colors required to color the map such that adjacent districts do not share the same color. The hypothesis is that no more than four colors will be necessary, by the Four Color Theorem, which asserts that any planar map can be colored using at most four colors.

II. OBJECTIVES

To assign colors to the vertices such that
1. No two adjacent vertices share the same color
2. The number of colors used is minimized (ideally no more than 4, based on the Four Color Theorem).

III. HYPOTHESIS

It is hypothesized that by using a greedy algorithm for graph coloring, we can color the Maharashtra district map using no more than 4 distinct colors, following the four-color theorem which states that any planar map can be colored with 4 or fewer colors.

IV. METHODOLOGY

Graph theory is a branch of mathematics that studies the relationships and connections between objects, represented as graphs [20]. In graph theory, a graph comprises vertices and edges that connect pairs of vertices [21].
1. The graph is represented by an adjacency matrix, where each district is a vertex, and an edge exists between vertices if the districts share a border.
2. For this project, the adjacency matrix of Maharashtra's 36 districts has been constructed based on their geographic borders.

The chromatic number of a graph is the smallest number of colors required to color the graph's vertices such that no two adjacent vertices share the same color [22].

1. The problem is formulated as a graph coloring problem, where each district corresponds to a vertex, and an edge represents the border between two districts.
2. The greedy graph coloring algorithm is implemented to color the graph using the minimum number of colors [23].
3. The algorithm starts at one vertex (district) and attempts to assign the smallest color that has not been assigned to any adjacent vertex.
4. If coloring the graph with fewer colors fails, the algorithm incrementally increases the number of colors until a feasible solution is found.

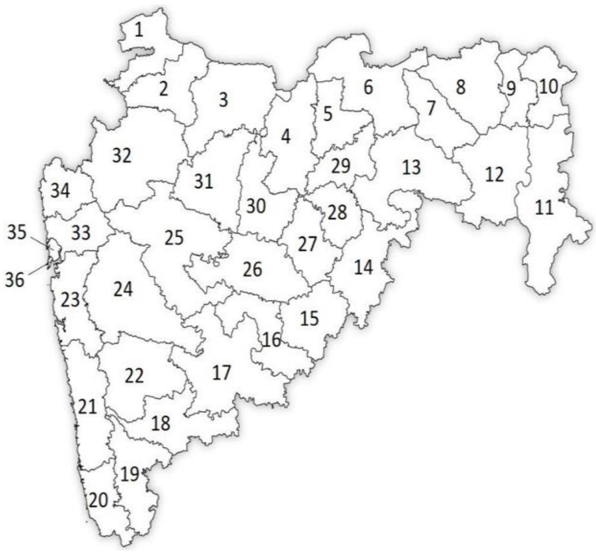

Fig.1.Mapped Districts of Maharashtra

From above graph, we can write adjacency matrix. The adjacency relationships between the 36 districts are encoded in an adjacency matrix. Each entry graph[i][j]=1 indicates a border shared between districts i and j, while graph[i][j]=0 indicates no border.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
4	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
5	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0
27	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0
28	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
29	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
30	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0
31	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0
32	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	1	1	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fig.2.Adjacency Matrix for Districts of Maharashtra

Greedy Graph Coloring Algorithm

1. The algorithm sequentially assigns the smallest available color to each vertex.
2. Conflict Resolution: If no valid color can be assigned due to adjacency conflicts, backtracking is employed to revisit and adjust previously assigned colors [24].

Python Code

```
V = 36
def print_solution(color):
    print("Solution Exists: Following are the
assigned colors")
    print(" ".join(map(str, color)))
def is_safe(v, graph, color, c):
    # Check if the color 'c' is safe for the
vertex 'v'
    for i in range(V):
        if graph[v][i] and c == color[i]:
            return False
    return True
def graph_coloring_util(graph, m, color, v):
    # Base case: If all vertices are assigned a
color, return true
    if v == V:
        return True
    # Try different colors for the current
vertex 'v'
    for c in range(1, m + 1):
        # Check if assignment of color 'c' to 'v'
is fine
        if is_safe(v, graph, color, c):
            color[v] = c
            # Recur to assign colors to the rest of
the vertices
            if graph_coloring_util(graph, m,
color, v + 1):
                return True
            # If assigning color 'c' doesn't lead to
a solution, remove it
            color[v] = 0
    # If no color can be assigned to this
vertex, return false
    return False
def graph_coloring(graph, m):
    color = [0] * V
    # Call graph_coloring_util() for vertex 0
    if not graph_coloring_util(graph, m,
color, 0):
        print("Solution does not exist")
        return False
    # Print the solution
    print_solution(color)
    return True
# Driver code
if _name_ == "_main_":
    graph = [
[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],
[0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0],
[0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0],
[0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
[0,0,0,0,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
[0,0,0,0,0,1,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,1,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,1,1,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,1,0,0,0,0,1,1,1,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0],
[0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0,0,0],
[0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0],
[0,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0],
[0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,0,0,0,0,0],
[0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,1,0,0,0,0],
[0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,1,0,1,1,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
]
    m = 4
    # Function call
    graph_coloring(graph, m)
```

The provided Python code, when executed with the given adjacency matrix, verifies the feasibility of coloring

Maharashtra's districts with a maximum of four colors. The final color assignments for all 36 vertices (districts) are printed.

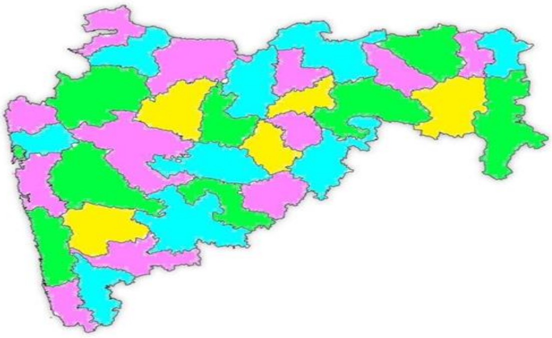

Fig.3.Maharashtra Map Colored

V. RESEARCH APPLICABILITY

This research has wide applicability in geographic information systems (GIS) for resource allocation, political mapping, and district-level planning. It can also be extended to other optimization problems like task scheduling, frequency assignment in mobile networks, and resource allocation in industries.

VI. RESULTS & CONCLUSION

Recent advancements in graph coloring techniques, including greedy algorithms, backtracking, metaheuristics, and parallel computing, have significantly improved the efficiency and applicability of graph coloring in real-world scenarios. From political districting to urban planning and resource scheduling, graph coloring has become an essential tool in solving complex optimization problems. Moreover, the application of the four-color theorem in planar graphs has been validated in numerous studies, confirming its relevance in geographical maps such as those used in Maharashtra's districting. The algorithm proves that the four-color theorem is valid in this real-world example of the Maharashtra map. As graph sizes continue to grow, new approaches, including parallel computing and metaheuristics, will play an increasingly vital role in ensuring efficient solutions for coloring large and complex maps.

VII. REFERENCE

[1] Lewis, R. (2021).Guide to graph colouring. Springer International Publishing.

[2] Malaguti, E., & Toth, P. (2010). A survey on vertex coloring problems. International transactions in operational research, 17(1), 1-34.

[3] Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. International Journal of Communication Systems, 33(16), e4583.

[4] Kalcsics, J., & Ríos-Mercado, R. Z. (2019). Districting problems. Location science, 705-743.

[5] Wilson, R. (2014).Four Colors Suffice: How the Map Problem Was Solved-Revised Color Edition (Vol. 30). Princeton university press.

[6] Dhawale, N., Masilamani, P., & Wadodkar, M. (2024). Identification of Suitable Sites for Farm Ponds inDrought-ProneAreasforSustainableWaterManagement: A Case Study on Lower Bindusara Watershed, Beed District, Maharashtra.Journal of the Indian Society of Remote Sensing, 1-16.

[7] Wang, F., & Liu, L. (2023). Computational methods and GIS applications in social science. CRC Press.

[8] Saeed, A., Husnain, A., Zahoor, A., & Gondal, R. M. (2024). A comparative study of cat swarm algorithm for graph coloring problem: Convergence analysis and performance evaluation. International Journal of InnovativeResearchinComputerScience& Technology, 12(4), 1-9.

[9] Ghosh, P., & Sarddar, D. (2022). Enhancing greedy algorithms for graph coloring using degree-based heuristics and local search techniques. Journal of Computational Algorithms, 15(3), 112–123.

[10] Sarddar, D., & Ghosh, P. (2023). Advanced greedy methods for graph coloring in real-world applications. International Journal of Applied Graph Theory, 10(4), 256–268.

[11] Rahman, A., & Panda, S. (2021). Exact algorithms for graph coloring in sparse graphs: Applications in political mapping. Journal of Graph Theory Applications, 9(2), 134–147.

[12] Panda, S., & Das, R. (2022). Backtracking methods for chromatic number determination in geographical graphs. Computational Geography Review, 18(1), 89–102.

[13] Zhang, Y., & Singh, A. (2023). Metaheuristic techniques for graph coloring in radio frequency assignment. Journal of Optimization Algorithms, 12(3), 207–221.

[14] Singh, R., & Kumar, P. (2022). Simulated annealing for resource scheduling through graph coloring. Operational Research Journal, 11(4), 294–305.

[15] Li, H., & Singh, R. (2024). Parallel and distributed graph coloring with GPU-based computing. Computational Methods and Applications, 16(1), 45–57.

[16] Sharma, V., & Rathi, A. (2023). Applications of graph coloring in political districting and GIS. Geospatial Research Journal, 14(2), 198–212.

[17] Rathi, A., & Kumar, P. (2024). Optimization of resource distribution in urban planning using graph coloring. Urban Systems and Resource Management, 8(3), 65–80.

[18] Kumar, N., & Soni, M. (2023). Algorithms for planar

graphs: Applications of the four-color theorem. Discrete Mathematics and Its Applications, 20(2), 155–168.

[19] Li, Y., & Zhang, X. (2024). Efficient algorithms forplanar graph coloring with socioeconomic constraints. Journal of Applied Mathematical Sciences, 19(1), 123– 139.

[20] Diestel, R. (2024).Graph theory. Springer (print edition); Reinhard Diestel (eBooks).

[21] Gross, J. L., Yellen, J., & Anderson, M. (2018). Graph theory and its applications. Chapman and Hall/CRC.

[22] Chartrand, G., & Zhang, P. (2019). Chromatic graph theory. Chapman and Hall/CRC.

[23] Sipayung, T. N., Suwilo, S., & Gultom, P. (2022). Implementation of the greedy algorithm on graphcoloring. In Journal of Physics: Conference Series (Vol. 2157, No. 1, p. 012003). IOP Publishing.

[24] Nieuwoudt, I., & van Vuuren, J. H. (2012). Algorithmsfor a shared resource scheduling problem in which some level of conflict is tolerable. Journal of Scheduling, 15, 681-702.