Application of Welsh-Powell algorithm to graph coloring problem

Chen Tianxing, Liu Yifei

November 26, 2022

Abstract

In graph theory, graph coloring is a special case of graph labeling; it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices are of the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges are of the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

However, graph coloring is a kind of typical NP-Complete problem.

In this paper, we propose a feasible solution to the map coloring problem based on the Welsh-Powell algorithm and supplement it with a mathematical proof that it can find the approximate optimal feasible solution with O(n^2) time complexity.

In the final test, the model performed well, and it is proved to be correct and optimal solution in the test.

1 Introduction

Mathematical definition: Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the graph coloring problem is to divide V into K color groups, each of which forms an independent set, i.e., one in which there are no adjacent vertices. Its optimized version is desired to obtain the smallest value of K.

In real-life maps, coloring is often needed to distinguish regions. Also, graph coloring problems have important applications in combinatorial analysis, such as task scheduling, resource allocation, VLSI wiring and testing. Some classical combinatorial problems, such as maximum dominating set, quadratic assignment, and maximum coverage problems can be converted to graph coloring problems for study.

While the map least coloring problem is an NP-Complete problem, there is no effective way to solve it, and how to trade-off the number of coloring with the complexity of solving the problem becomes a major problem.

By simple abstraction, we can equate the map coloring problem with the dot coloring problem, i.e., if the contradiction of the dot coloring problem is solved, the contradiction of the map coloring problem can also be solved. The Welsh-Powell algorithm is a good solution to the balance between the superiority of the dot coloring problem solution and the feasibility of time.

With the Welsh-Powell algorithm, we are able to find the superior solution for the dot coloring problem with the desired time complexity of $O(n^2)$, which is equivalent to proposing a mature solution to the graph coloring problem.

2 Related research

2.1 An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems

An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems [1], a paper on the Welsh-Powll algorithm by Welsh, D.J.A. and Powell, M.B. published in The Computer Jounery in 1967. Its provides the theoretical support for this paper.

This paper points out the connection between the basic scheduling or timetabling problem with the well known problem of colouring the vertices of a graph in such a way that (i) no two adjacent vertices are the same colour and (ii) the number of colours used is a minimum. We give an algorithm for colouring a graph subject to (i) and give a new easily determined bound for the number of colours needed. This same bound is also a new upper bound for the chromatic number of a graph in terms of the degrees of its vertices.

2.2 Heuristic search ant colony algorithm for graph coloring problem

A Heuristic Search Ant Algorithm for the Graph Coloring Problem [2], published in 2005 by Feixiong Liao and Liang Ma of Shanghai University of Technology, China.

In the article, an ant colony algorithm-based solution strategy is proposed for the graph coloring problem using the then rapidly developing intelligent applications (represented by adaptive algorithms such as genetic algorithms and simulated annealing algorithms). Through extensive computations and tests, the performance of the algorithm is explored and compared with general solution algorithms, and the results converge quickly to better results in the same computing time, largely reflecting the advantages of this bionic algorithm.

While acknowledging the effectiveness of the algorithm, we are also aware of its drawbacks: the ant colony algorithm is essentially a stochastic optimization algorithm with a long computation time. With the development of parallel computing, the optimization of this algorithm can be prospected.

3 My system

3.1 From Graph-coloring problem to Dot-coloring problem

In the graph coloring problem, it is difficult for us to use a computer to directly characterize the adjacency of blocks with complex edges in a graph.We can abstract the graph coloring problem as a dot coloring problem, and the bordering relationship between the blocks in the graph can be abstracted as whether the points are connected to each other, which is obvious.

That is to say, graph-coloring problem and dot-coloring problem are equivalent.

3.2 My solution based on Welsh-Powell Algorithm

To solve dot coloring problem, we use Welsh-Powell Algorithm.

The Welsh-Powell algorithm is a greedy algorithm that finds a coloring scheme when the maximum number of colors is not limited. Time complexity O(n^2). Related proofs are detailed in the next subsection.

Algorithm Steps below:

- Step 1: All vertices are sorted according to the decreasing value of their degree in a list V.
- Step 2: Colors are ordered in a list C(color table).
- Step 3: The first non colored vertex v in V is colored with the first available color in C means a color that was not previously used by the algorithm.
- Step 4: The remaining part of the ordered list V is traversed and the same color is allocated to every vertex for which no adjacent vertex has the same color.
- Step 5: Steps 3 and 4 are applied iteratively until all the vertices have been colored.

With this algorithm, we can quickly obtain a feasible solution that approximates the optimal solution. Note that the algorithm does not always obtain the optimal solution, but most of the results do not deviate too far, which means it is a fast and effective algorithm.

3.3 Proof

3.3.1 Time Complexity Proof

First, it's easy to know that the max number of colors we need is maxDegree+1: every node connect with at most maxDegree nodes, which means maxDegree+1 colors must be the max number we need.

For a non-self-looping undirected graph G, we assume $V(G) := \{v_1, v_2, ..., v_n\}$ satisfy:

$$degree(v_i) \geq degree(v_{i+1}), \quad \forall 1 \leq i \leq n-1$$

According to the algorithm and Theorem above:

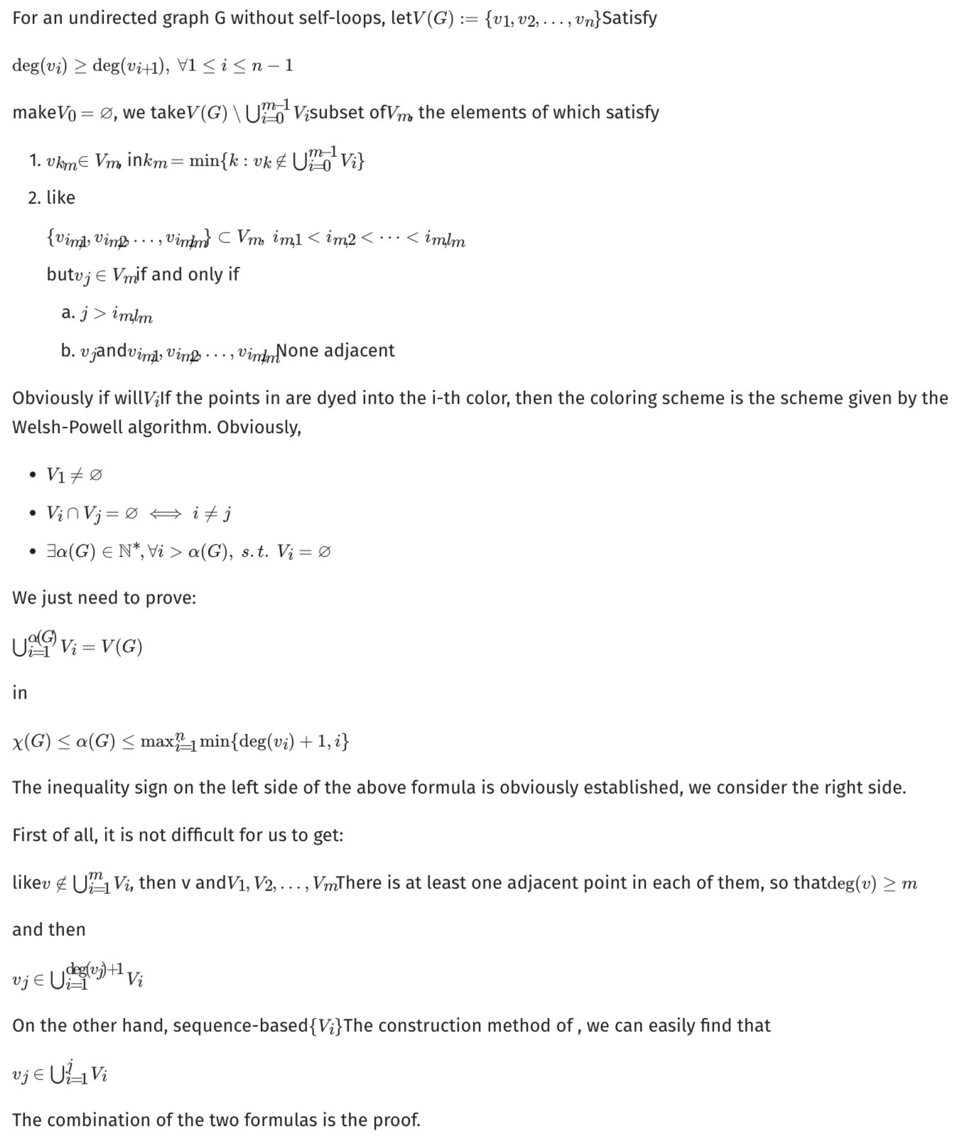
$$max_{i=1}^{n} min\{degree(v_i + 1, i)\}$$

So the time complexity of the algorithm:

$$O(n * max_{i=1}^{(} n) min\{degree(v_i + 1, i)\}) = O(n^2)$$

3.3.2 Correctness Proof

The following is a mathematical proof of the Welsh-Powell Algorithm from oiwiki.org.



For an undirected graph G without self-loops, let$V(G) := \{v_1, v_2, \dots, v_n\}$Satisfy

$\deg(v_i) \geq \deg(v_{i+1}),\ \forall 1 \leq i \leq n-1$

make$V_0 = \varnothing$, we take$V(G) \setminus \bigcup_{i=0}^{m-1} V_i$subset of$V_m$, the elements of which satisfy

1. $v_{k_m} \in V_m$, in$k_m = \min\{k : v_k \notin \bigcup_{i=0}^{m-1} V_i\}$
2. like

   $\{v_{i_{m,1}}, v_{i_{m,2}}, \dots, v_{i_{m,l_m}}\} \subset V_m,\ i_{m,1} < i_{m,2} < \dots < i_{m,l_m}$

   but$v_j \in V_m$if and only if

   a. $j > i_{m,l_m}$

   b. $v_j$and$v_{i_{m,1}}, v_{i_{m,2}}, \dots, v_{i_{m,l_m}}$None adjacent

Obviously if willV_iIf the points in are dyed into the i-th color, then the coloring scheme is the scheme given by the Welsh-Powell algorithm. Obviously,

- $V_1 \neq \varnothing$
- $V_i \cap V_j = \varnothing \iff i \neq j$
- $\exists \alpha(G) \in \mathbb{N}^*, \forall i > \alpha(G),\ s.t.\ V_i = \varnothing$

We just need to prove:

$\bigcup_{i=1}^{\alpha(G)} V_i = V(G)$

in

$\chi(G) \leq \alpha(G) \leq \max_{i=1}^{n} \min\{\deg(v_i) + 1, i\}$

The inequality sign on the left side of the above formula is obviously established, we consider the right side.

First of all, it is not difficult for us to get:

like$v \notin \bigcup_{i=1}^{m} V_i$, then v and$V_1, V_2, \dots, V_m$There is at least one adjacent point in each of them, so that$\deg(v) \geq m$

and then

$v_j \in \bigcup_{i=1}^{\deg(v_j)+1} V_i$

On the other hand, sequence-based$\{V_i\}$The construction method of , we can easily find that

$v_j \in \bigcup_{i=1}^{j} V_i$

The combination of the two formulas is the proof.

3.4 Code

Pseudo code blow: Python code below(core):

```
color_kind = 0
SortedNode = copy.deepcopy(node)
# Sort in descending order by degree
SortedNode = sorted(SortedNode, key=cmp_to_key(compare))

while SortedNode:
    color_kind += 1
    i = 0
    while i < len(SortedNode):
        check = True
        for j in range(len(SortedNode[i].connect)):
            if node[SortedNode[i].connect[j]].color == color_kind:
                check = False
                break
        if check:
            node[SortedNode[i].node_number].color = color_kind
            SortedNode.remove(SortedNode[i])
            i -= 1
        i += 1
```

4 Experiment

We take solving the coloring scheme of the administrative division map of some provinces and cities of China map as a sample test case.

We try to color the following Chinese map by the above model and require an acceptable number of color kinds.

The original uncolored map below:

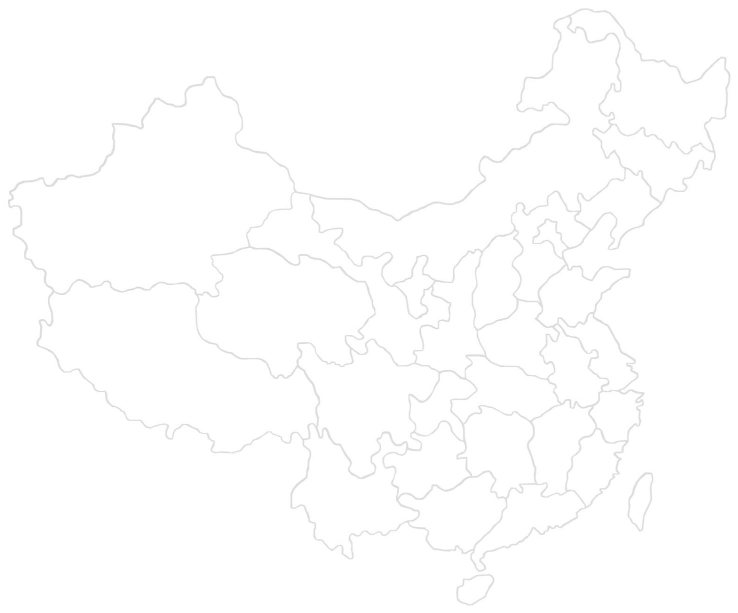


Figure 1: Map of China without color.

According to step 1, we order all the blocks, and get graph below: (Note that the number of Chinese provinces and cities shown in the chart is only 30, and because of the small size of regions such as Macau, they are not counted)

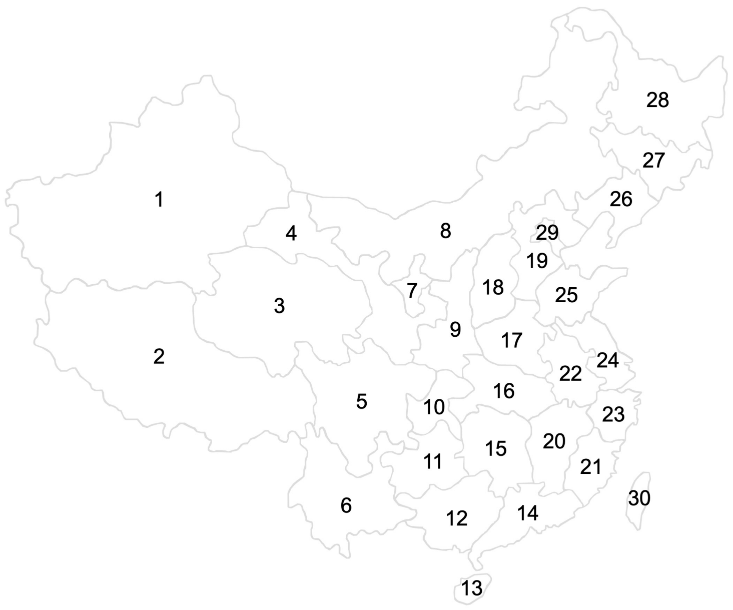


Figure 2: Map of China with number.

Then, as we said before, we want to convert blocks into point sets and describe the bordering of blocks in the graph with connected edges.

Abstract the graph and describe the adjacency matrix:

Number	Adjacency Matrix	Number	Adjacency Matrix
1	011100000000000000000000000000	16	000000001100001010010100000000
2	101011000000000000000000000000	17	000000001000000101100101100000
3	110110000000000000000000000000	18	000000011000000010100000000000
4	101011111000000000000000000000	19	000000010000000011000010000010
5	011101001110000000000000000000	20	000000000000011100001110000000
6	010010000011000000000000000000	21	000000000000010000010010000001
7	000100011000000000000000000000	22	000000000000000110010011100000
8	000100101000000001100000011100	23	000000000000000000011101000000
9	000100110100000111000000000000	24	000000000000000000000110100000
10	000010001010001100000000000000	25	000000000000000010100101000000
11	000011000101001000000000000000	26	000000010000000000100000001000
12	000001000010011000000000000000	27	000000010000000000000000010100
13	000000000000010000000000000000	28	000000010000000000000000001000
14	000000000001101000011000000000	29	000000000000000000100000000000
15	000000000111010100010000000000	30	000000000000000000001000000000

Table 1: Adjacency Matrix Table.

In the adjacency matrix, $Node_i$is connected with $Node_j$if and only if $matrix[i][j] = 1$.

Finally, we get the answer:

Number	Color
1	1
2	2
3	3
4	2
5	1
6	3
7	4
8	1
9	3
10	4
11	2
12	4
13	1
14	2
15	1
16	2
17	1
18	4
19	2
20	3
21	1
22	4
23	2
24	1
25	3
26	3
27	2
28	3
29	1
30	2

Table 2: Answer color table.

We show the result on the graph:

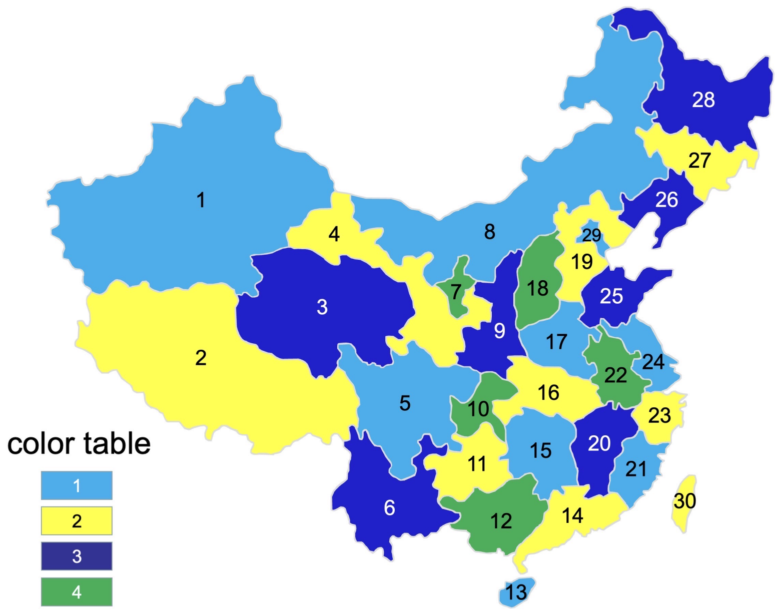


Figure 3: Map of China colored(Visualization results).

Only 4 kinds of color are used. It's easy to know 4 is the best answer since blocks 4, 7, 8, 9 are interconnected. Which means in this case, the solution works.

5 Conclusion

This study set out to propose a solution to the graph coloring problem (and its abstractable related problems).

In this paper, we propose an $O(n^2)$ time complexity approximate optimal solution to the graph coloring problem, convert the NP-Complete problem into a stable algorithm that is acceptable in time and space, and prove mathematically that this approach yields an approximate optimal solution as a trade-off between solvability and solution complexity, and the final experiment result indicates that the solution works well.

The limitation of this solution is that when the number of points and their adjacent edges reaches a certain level (10^8), this solution is not so good for general computers. For solving different types of problems, a more suitable algorithm should be used, and a solution algorithm for graph coloring problems that achieves excellent results in terms of both solution superiority and time complexity remains to be found.

6 Acknowledgement

We would like to thank our teacher Peng Xiaogang for his great help.

References

[1] Welsh, D. J. A.; Powell, M. B. (1967), "An upper bound for the chromatic number of a graph and its application to timetabling problems", The Computer Journal 10 (1): 85–86, doi:10.1093/comjnl/10.1.85

[2] Feixiong Liao, Liang Ma. Heuristic search ant colony algorithm for graph coloring problem[J]. Computer Engineering, 2007, 33(16):3.