Math381 HW3

Siyue Zhu

January 29, 2022

Introduction

Four Color Theorem is a theorem that states any map can be colored using four or fewer colors, with regions or countries that share a border always colored differently. The theorem has been proved sufficiently, however many people think there is a better proof. We are going to color a map and see how many colors needed to color a whole map. Also, we are going to color the map in different ways, with more restriction putting into the requirement, we will see how the number of minimum colors would change.

Map Chosen

The map I chose is Yunnan, which is a province of China. Our graph is an order pair of sets (V,E) where V is the set of vertices and E is the set of edges. An edge is an unordered pair of distinct vertices. We generate our graph from the map of Yunnan. Each vertex is a integer correspond to a district, and we would connect two vertices as an edge if and only if two districts share border. Our V = {1,2,...,16}. And for E, if (a,b) is in E, then this tells us there is an edge between a and b, and the corresponding districts share a common border.

And here is our map of Yunnan and the graph:

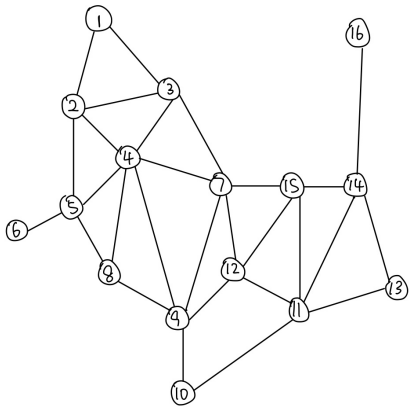


Figure 1: The map of Yunnan. Figure 2: 1. Doqing, 2. Nujiang, 3. Lijiang, 4. Dali, 5. Baoshan, 6. Dehong, 7. Chuxiong, 8. Lincang, 9. Pu'er(Simao), 10. Xishuangbanna, 11. Honghe, 12. Yuxi, 13. Wenshan, 14. Qujing, 15. Kunming, 16. Zhaotong

[1]

[1]Here is the link of the map https://teadb.org/yunnan/.

We can see from the graph, that Yuannan has 16 districts. And we are going to color the 16 districts in different ways in following parts.

1 Coloring map in real world mapping situation

Now, we are going to consider how to color the map with a minimum number of colors, with districts share border could not be in the same color. We are going to to use lp to solve this problem. According to the Four Color Theorem, all maps can be colored in four different colors, with regions share boundary be colored in different colors. Thus, we are going to use four colors and try to color the map.

I'm going to use two binary variables

$$x_{n,i}(1 \leq n \leq 16) \text{ and } (1 \leq i \leq 4)$$

to represent if a specific vertex n uses a specific color i, with 0 means not use and 1 means use. And

$$y_i(1 \leq i \leq 4)$$

to represent different colors is used, with 0 means color i is not be used, and 1 means color i is used.

Our goal in this question is to minimize the number of color we use to color the vertex, and see if four colors is enough to color the graph. Thus, the objective function is

$$min \sum_{i=1}^{4} y_i$$

And to required that each vertex can only take one color, we need to have the constraints that

$$\sum_{i=1}^{4} x_{ni} = 1(n = 1, ..., 16)$$

We also need to add a constraint that a vertex cannot be colored with an unused colored. If $x_{ni} = 1$ then $y_i = 1$, which mean if vertex n uses color i, then color i is used. So we need to add the constraints that

$$x_{ni} \leq y_i \text{ for n=1,...,16 and i=1,2,3,4}$$

One more important constraint is that vertices that connected each other cannot be colored in the same color, that means that districts that share border with each other cannot be colored in the same color. Thus, we need the constraint that

$$\text{for all } (a, b) \in E, x_{ai} + x_{bi} \leq 1 \text{ and i=1,2,3,4}$$

To speed up our calculation, we can add one more constraint. We will not use color 2 if color 1 is not be used, and we will not use color 3 is color 2 is not be used, etc. For example, if color 2 has not be used yet, we can save time on looking for the way of using color 3 and color 4. This constraint can save a lot of time finding other combination of colors, which would be able to save a lot of time of running the lpsolve.

$$y_k \leq y_{k-1} \text{ k=2,3,4}$$

Last be not least, all variables should be binary.

The lp should look like this: Minimize:

$$\sum_{i=1}^{4} y_i$$

Subject to:

$$\sum_{n=1}^{16} x_{ni} = 1(i = 1, 2, 3, 4)$$

$$x_{ni} \leq y_k \text{ for n=1,...,16 and i=1,2,3,4}$$

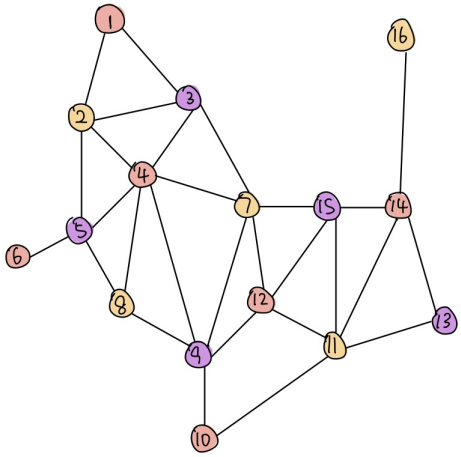
$$\text{for all } (v_a, v_b) \in E, x_{ai} + x_{bi} \leq 1 \text{ and k=1,2,3,4}$$

$$y_k \leq y_{k-1} \text{k=2,3,4}$$

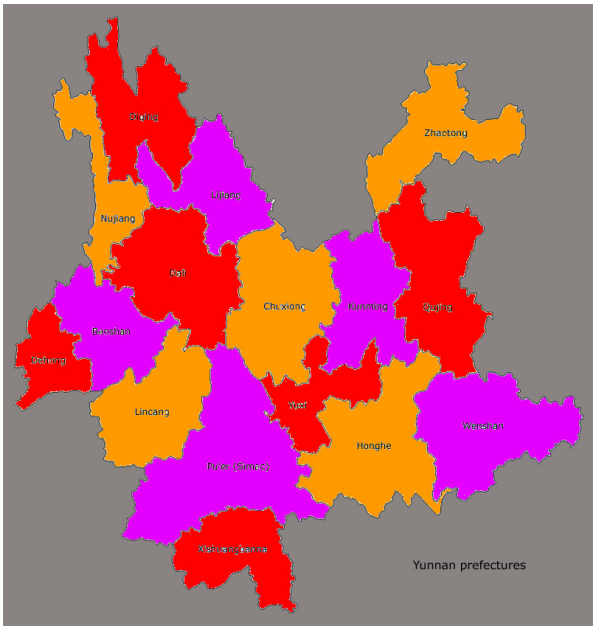
$$x_{ni}, y_i \in 0, 1 \text{ for n=1,...,16 and i=1,2,3,4}$$

The python code, lp input file and the result can be find in the appendix.

According to the result, we only need three colors to color the map, which match with the Four Color Theorem. We can see vertices 1, 2, and 3 are three vertices connected to each other, which requires at least three colors to color the map. Since our graph only requires three colors, which means there is no other sub-graphs that is more complex than sub-graph of vertices 1, 2, and 3. Thus, we can conclude that our result of using three colors to color the map is the optimised result. Here is the how the graph and the map look like:



(a)



(b)

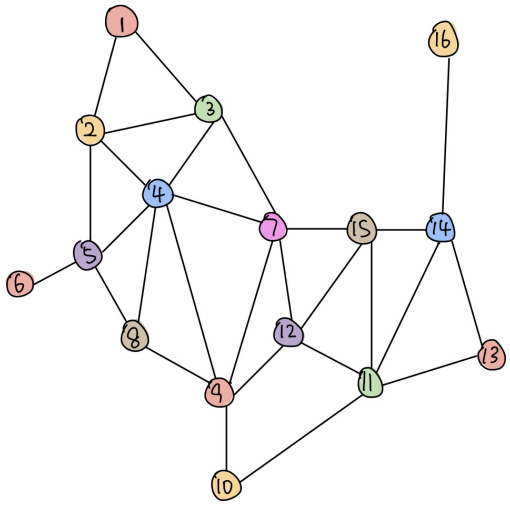
2 Add Additional Constraint

Now, we are going to add one more additional constraint to the lp, which requires that two regions that border the same region cannot be colored the same. In order to satisfy this constraint, I change the edge set E to E' which contains more edges than before. E' contains all the edges from E, and (i,j) such that region i and j both share border with region k. The only thing need to be changed in the python code is add E'.

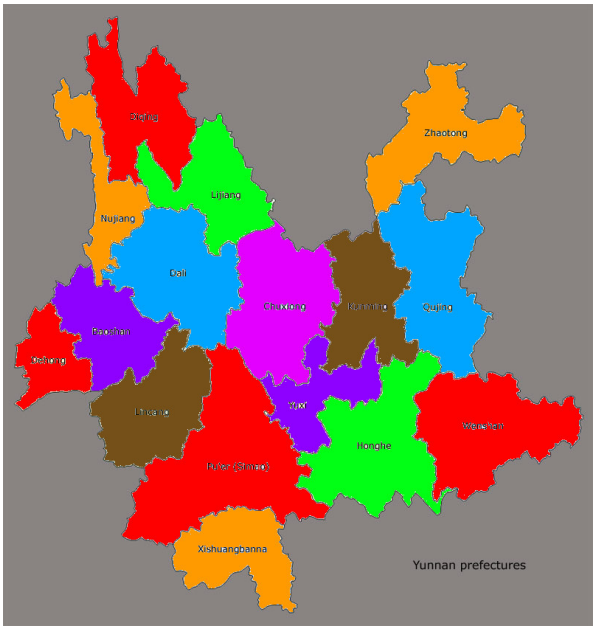
The python code, lp input file and the result can be find in the appendix.

This time, we use seven colors in total to color the map. We can see from the map that Dali, which is vertex 4 connects six districts. Our new constraints says that two regions that border the same region cannot be colored the same, those six districts plus Dali itself cannot be colored in the same color. Thus, we need minimum of 7 colors. Also, we can see there is no other sub-graph can be more complicated of vertex 4 and all vertices connected to vertex 4, so there is no any other sub-graph that requires more than 7 colors. Thus, we can conclude that our result of using 7 colors to color the map is the optimised result.

And the here is the how the graph and the map look like:



(c)



(d)

3 Additional Constraint: Two colors Need to be Used to One District

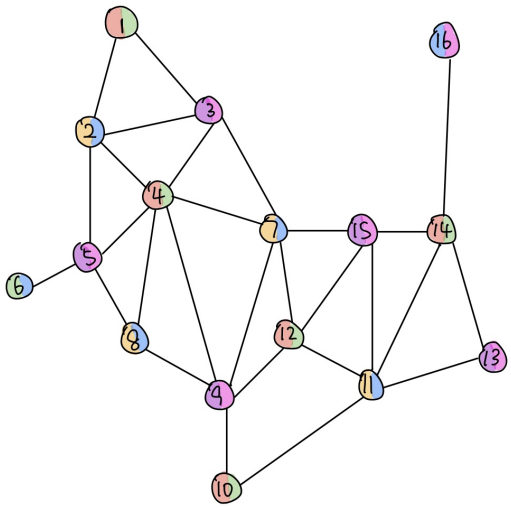
Now, we are going to consider what is we use two different colors to color one district, and adjacent vertices have different colors. This time, we need E instead of E'. And also, the constraint that requires all vertices to be colored with exactly one color should be changed to following:

$$\text{for all } (v_a, v_b) \in E, x_{ai} + x_{bi} \leq 2 \text{ and k=1,2,3,4}$$

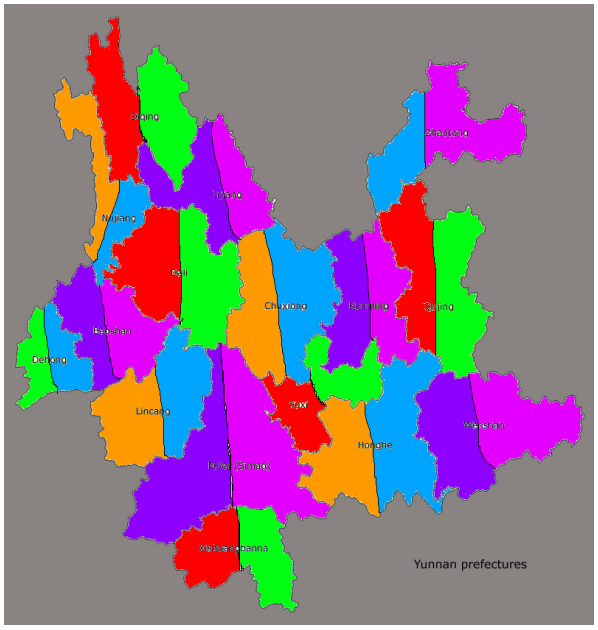
The python code, lp input file and the result can be find in the appendix.

In this situation, the minimum number of colors we need is six. For example, vertex 1, 2, and 3 are three vertices that connect to each other, and in this case, we need six different colors to color them. Thus, the minimum number we need is six. And there is no other sub-graph that need more colors to be colored. Thus, we can conclude that our result of using six colors to color the map is the optimised result.

And the here is the how the graph and the map look like:



(e)



(f)

A python Code

```
import math
# number of vertex
n=16
# number of color
i=4

# define vertex
E=[[1,2],[1,3],[2,3],[2,4],[2,5],[3,4],[3,7],[4,5],[4,7],[4,8],[4,9],[5,6],
    [5,8],[7,9],[7,12],[7,15],[8,9],[9,10],[9,12],[10,11],[11,12],[11,13],
    [11,14],[11,15],[12,15],[13,14],[14,15],[14,16]]
```

```
# objective function
output = ""
for a in range(1,i+1):
    output += "+y_"+str(a)
print("min: "+output+";")

# requires all vertices to be colored with exactly one color
output = ""
for a in range(1,n+1):
    output = ""
    for b in range(1,i+1):
        output += "+x_"+str(a)+"_"+str(b)
    print(output+"=1;")

# requires that a vertex cannot be colored with an unused colored
for a in range(1,n+1):
    for b in range(1,i+1):
        print("x_"+str(a)+"_"+str(b)+"<=y_"+str(b)+";")

# requires that adjacent vertices have different colors
for edge in E:
    for b in range(1,i+1):
        print("x_"+str(edge[0])+"_"+str(b)+"+x_"+str(edge[1])+"_"+str(b)+"<=1;")

# we will not use color 2 if we do not use color 1, etc.
for a in range(2,i+1):
    print("y_"+str(a)+"<="+"y_"+str(int(a)-1)+";")

# all variables are binary
output = "bin "
for a in range(1,n+1):
    for b in range(1,i+1):
        if (a>1 or b>1):
            output += ","
        output += "x_"+str(a)+"_"+str(b)
print(output+";")

output = "bin "
for a in range(1,i+1):
    if (a>1):
        output += ","
    output += "y_"+str(a)
print(output+";")
```

B Lp input for part 1

```
(1 line of the following type:
this is the object function)
min: +y_1+y_2+y_3+y_4;

(16 lines of the following type:
ensure all vertices to be colored with exactly one color)
+x_1_1+x_1_2+x_1_3+x_1_4=1;

(64 lines of the following type:
ensure that a vertex cannot be colored with an unused colored)
x_1_1<=y_1;

(112 lines of the following type:
ensure that adjacent vertices have different colors)
x_1_1+x_2_1<=1;

(3 lines of the following type:
ensure we will not use color 2 if we do not use color 1, etc.)
y_2<=y_1;

(2 lines of the following type:
ensure all variables are binary)
bin x_1_1,x_1_2,.....,x_16_3,x_16_4;
bin y_1,y_2,y_3,y_4;
```

C Output for part 1

```
Value of objective function: 3.00000000

Actual values of the variables:
y_1                             1
y_2                             1
y_3                             1
x_1_1                           1
x_2_2                           1
x_3_3                           1
x_4_1                           1
x_5_3                           1
x_6_1                           1
x_7_2                           1
x_8_2                           1
x_9_3                           1
x_10_1                          1
x_11_2                          1
x_12_1                          1
x_13_3                          1
x_14_1                          1
x_15_3                          1
x_16_2                          1
```

D New code used for part 2

```
E' = [[1,2],[1,3],[2,3],[2,4],[2,5],[3,4],[3,7],[4,5],[4,7],[4,8],[4,9],
[5,6],[5,8],[7,9],[7,12],[7,15],[8,9],[9,10],[9,12],[10,11],[11,12],
[11,13],[11,14],[11,15],[12,15],[13,14],[14,15],[14,16],[1,4],[1,5],[1,7],
[2,7],[2,8],[2,9],[2,6],[3,5],[3,8],[3,9],[3,4],[3,12],[3,15],[4,6],
[4,12],[4,10],[4,15],[5,7],[5,9],[6,8],[7,8],[7,10],[7,11],[8,10],[8,12],
[9,15],[9,11],[10,12],[10,13],[10,14],[10,15],[11,16],[12,13],[12,14],
[13,15],[13,16],[15,16]]
```

E Lp input for part 2

```
(1 line of the following type:
this is the object function)
min: +y_1+y_2+y_3+y_4;

(16 lines of the following type:
ensure all vertices to be colored with exactly one color)
+x_1_1+x_1_2+x_1_3+x_1_4=1;

(64 lines of the following type:
ensure that a vertex cannot be colored with an unused colored)
x_1_1<=y_1;

(260 lines of the following type:
ensure that adjacent vertices have different colors)
x_1_1+x_2_1<=1;

(3 lines of the following type:
ensure we will not use color 2 if we do not use color 1, etc.)
y_2<=y_1;

(2 lines of the following type:
ensure all variables are binary)
bin x_1_1,x_1_2,.....,x_16_3,x_16_4;
bin y_1,y_2,y_3,y_4;
```

F Output for part 2

```
Value of objective function: 7.00000000

Actual values of the variables:
y_1                            1
y_2                            1
y_3                            1
y_4                            1
y_5                            1
y_6                            1
y_7                            1
x_1_1                          1
x_2_2                          1
x_3_3                          1
x_4_4                          1
x_5_5                          1
x_6_1                          1
x_7_6                          1
x_8_7                          1
x_9_1                          1
x_10_2                         1
x_11_3                         1
x_12_5                         1
x_13_1                         1
x_14_4                         1
x_15_7                         1
x_16_2                         1
```

G New python code for part 3

```
# requires all vertices to be colored with exactly two color
output = ""
for a in range(1,n+1):
    output = ""
    for b in range(1,i+1):
        output += "+x_"+str(a)+"_"+str(b)
    print(output+"=2;")
```

H Lp input for part 3

```
(1 line of the following type:
this is the object function)
min: +y_1+y_2+y_3+y_4;

(16 lines of the following type:
ensure all vertices to be colored with exactly one color)
+x_1_1+x_1_2+x_1_3+x_1_4=2;

(64 lines of the following type:
ensure that a vertex cannot be colored with an unused colored)
x_1_1<=y_1;

(112 lines of the following type:
ensure that adjacent vertices have different colors)
x_1_1+x_2_1<=1;

(3 lines of the following type:
ensure we will not use color 2 if we do not use color 1, etc.)
y_2<=y_1;

(2 lines of the following type:
ensure all variables are binary)
bin x_1_1,x_1_2,.....,x_16_3,x_16_4;
bin y_1,y_2,y_3,y_4;
```

I Output for part 3

```
Value of objective function: 6.00000000

Actual values of the variables:
y_1                            1
y_2                            1
y_3                            1
y_4                            1
y_5                            1
y_6                            1
x_1_1                          1
x_1_3                          1
x_2_2                          1
x_2_4                          1
x_3_5                          1
x_3_6                          1
x_4_1                          1
x_4_3                          1
x_5_5                          1
x_5_6                          1
x_6_3                          1
x_6_4                          1
x_7_2                          1
x_7_4                          1
x_8_2                          1
x_8_4                          1
x_9_5                          1
x_9_6                          1
x_10_1                         1
x_10_3                         1
x_11_2                         1
x_11_4                         1
x_12_1                         1
x_12_3                         1
x_13_5                         1
x_13_6                         1
x_14_1                         1
x_14_3                         1
x_15_5                         1
x_15_6                         1
x_16_4                         1
x_16_6                         1
```