Math 381 A
Sheng Yu

Homework #3 Writeup: Map Coloring

I. Introduction:

Map coloring is one of the many examples to demonstrate graphs. On the map, each location, city, or district is considered as a vertex, and when two vertices are sharing the same boarder, then we have an edge between the two vertices. In this paper, we want to find out the minimal number of colors required to color the map with different constraints applied. The map used here is my hometown, Henan province, China. There are totally 18 unique cities. We will first design a graph that illustrate the vertices and edges in the map, then with Linear Programming, we can compute the minimal number of colors required to color the graph and map.

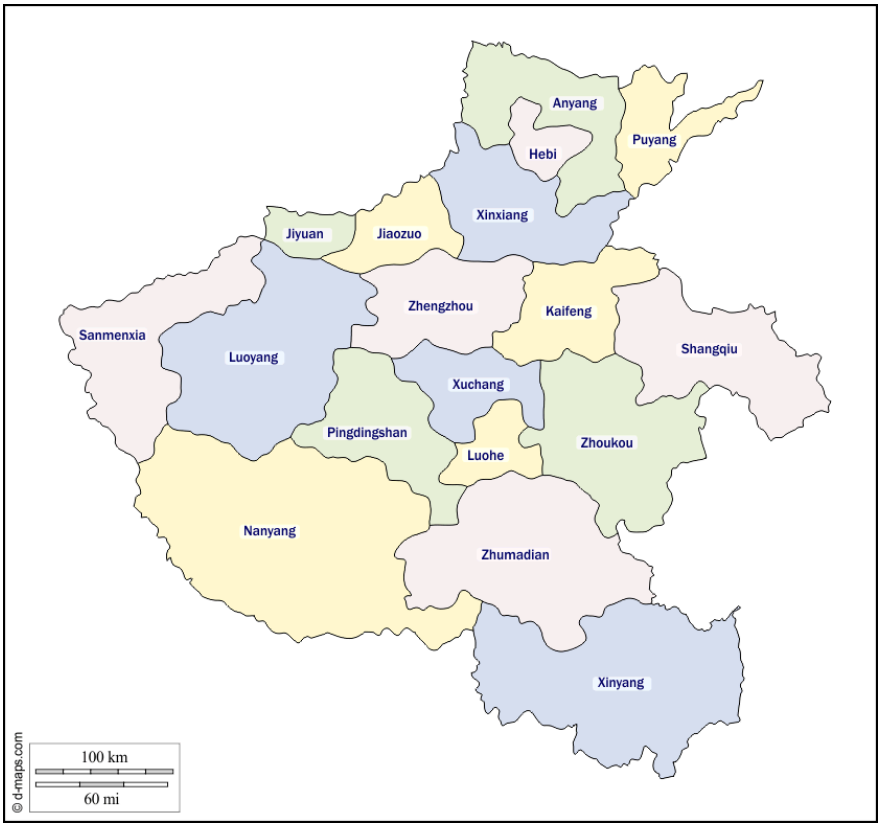


Image Source: d-maps.com

II. Graph:

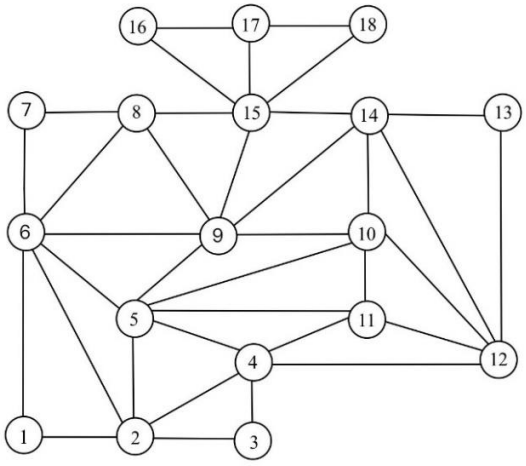
For simplicity, we will assign a number for each of the city in the map:

City	#	City	#
Sanmenxia	1	Xuchang	10
Nanyang	2	Luohe	11
Xinyang	3	Zhoukou	12
Zhumadian	4	Shangqiu	13
Pingdingshan	5	Kaifeng	14
Luoyang	6	Xinxiang	15
Jiyuan	7	Hebi	16
Jiaozuo	8	Anyang	17
Zhengzhou	9	Puyang	18

From here, we will use the numbers to reference the corresponding city names. We can get the following edge. Let E be the set of all edges below:

1	(1,2), (1,6)	10	(10,5), (10,9), (10,11), (10,12), (10,14)
2	(2,1), (2,3), (2,4), (2,5), (2,6)	11	(11,4), (11,5), (11,10), (11,12)
3	(3,2), (3,4)	12	(12,4), (12,10), (12,11), (12,13), (12,14)
4	(4,2), (4,3), (4,5), (4,11), (4,12)	13	(13,12), (13,14)
5	(5,2), (5,4), (5,6), (5,9), (5,10), (5,11)	14	(14,9), (14,10), (14,12), (14,13), (14,15)
6	(6,1), (6,2), (6,5), (6,7), (6,8), (6,9)	15	(15,8), (15,9), (15,14), (15,16), (15,17), (15,18)
7	(7,6), (7,8)	16	(16,15), (16,17)
8	(8,6), (8,7), (8, 9), (8,15)	17	(17,15), (17,16), (17,18)
9	(9,5), (9,6), (9,8), (9,10), (9,14), (9.15)	18	(18,15), (18,17)

Then, we can make a graph with the edges above like this:



III. Linear Programming:

We will define the following variables in the LP:

1. Let $y_i \in \{0,1\}, i = 1, \dots, 18$, with $y_i = 1$ iff we use color i.

2. Let $x_{ik} \in \{0,1\}, i = 1, \dots, 18, k = 1, \dots, 18$, with $x_{ik} = 1$ iff vertex i will have color k.

3. Let $(i,j) \in E, i = 1, \dots, 18, j = 1, \dots, 18$ be all the possible edges in the graph.

Since we trying to minimize the total number of colors used, then the objective function is:

$$\sum_{i=1}^{18} y_i$$

The first constraint defines that all vertices will be colored exactly with one color.

$$\sum_{k=1}^{18} x_{ik} = 1, i = 1, \dots, 18 \qquad (1)$$

The second constraint defines that if a color is never used, then a vertex cannot be colored with it. If $x_{ik} = 1$, then y_k must be one, which fulfills the requirement.

$$x_{ik} \leq y_k, i, k = 1, \dots, 18 \qquad (2)$$

The third constraint defines that if there is an edge between vertex i and j, then x_{ik} and x_{jk} cannot both be equal to one. For adjacent vertices, different colors will be assigned.

$$x_{ik} + x_{jk} \leq 1 \text{ for all } (i,j) \in E \text{ and } i, j, k = 1, \dots, 18 \qquad (3)$$

The fourth constraint defines that we will use adjacent colors only, this can speed up the computation. This is because it drastically decreases the possible sets of color. For instance, for 10 colors, there can be $2^{10} - 1 = 1023$ (we minus 1 because there must be at least one color) possible sets of colors since each color can be in the used set or not. With this constraint, the only possible sets are color 1, color 1 and 2, color 1, 2, and 3, etc. There are only 10 sets instead of 1023. For large graphs, we can reduce the space of possible solutions even more.

$$y_k \leq y_{k-1}, k = 2, \dots, 18 \qquad (4)$$

Now, we can summarize all the above information, and get the LP:

$Minimize$:

$$\sum_{i=1}^{18} y_i$$

$Subject\ to$:

$$\sum_{k=1}^{18} x_{ik} = 1, i = 1, \dots, 18 \qquad (1)$$

$$x_{ik} \leq y_k, i, k = 1, \dots, 18 \qquad (2)$$

$$x_{ik} + x_{jk} \leq 1 \text{ for all } (i, j) \in E \text{ and } k = 1, \dots, 18 \qquad (3)$$

$$y_k \leq y_{k-1}, k = 2, \dots, 18 \qquad (4)$$

$$x_{ik}, y_k \in \{0, 1\}, i = 1, \dots, 18, k = 1, \dots, 18$$

We can generate the LPSolve input file with the following MATLAB code. With so many constraints, the code is very long, so I will only clip the most important information here:

```
% Set up

n = 18;

% Coefficient for object function

co_obj = zeros(1, n^2);
co_obj(1, n^2+1:n^2+n) = ones(1, n);

% Coefficient for first constraint

co1 = zeros(n, n^2+n);
for i = 1:n
  co1(i, (i-1)*n+1:i*n) = ones(1, n);
end

% Coefficient for second constraint

co2 = zeros(n^2, n^2+n);

for i = 1:n
   co2((i-1)*n+1:i*n, n^2+i) = -1;
   for j = 1:n
      co2((i-1)*n+j, (j-1)*n+i) = 1;
   end
end

% Coefficient for third constraint
% Since (1, 2) and (2, 1) are the same, we only consider unique edges.

% For edge (1, 2)

x = 1; y = 2;
co3_1 = zeros(n, n^2+n);
for i = 1:n
   co3_1(i, (x-1)*n+i) = 1;
   co3_1(i, (y-1)*n+i) = 1;
end

% For edge (2, 3)

x = 2; y = 3;
co3_2 = zeros(n, n^2+n);
for i = 1:n
   co3_2(i, (x-1)*n+i) = 1;
   co3_2(i, (y-1)*n+i) = 1;
end
```

There are totally 36 unique edges in this graph, I only included the code to generate the third constraints for the edge $(1, 2)$ & $(2, 3)$. For the rest, the code is written in a similar way.

The same here when I add the third constraints, only first two included:

```
% Coefficient for the speed up constraint

co4 = zeros(n-1, n^2+n);

for i = 1:n-1
   co4(i, n^2+i) = 1;
   co4(i, n^2+i+1) = -1;
end

% Write the LP input file

lp = mxlpsolve('make_lp', 0, n^2+n);

mxlpsolve('set_binary', lp, (1:n^2+n));          % Bin x_i_k, and y_k
mxlpsolve('set_obj_fn', lp, co_obj);             % Set object fun

% Add constraints and name variables
% First constraint

for k = 1:n
   temp1 = co1(k, :);
   mxlpsolve('add_constraint', lp, temp1, 3, 1);
end

% Second constraint

for k = 1:n^2
   temp2 = co2(k, :);
   mxlpsolve('add_constraint', lp, temp2, 1, 0);
end

% Third constraint

for k = 1:n
   temp3 = co3_1(k, :);
   mxlpsolve('add_constraint', lp, temp3, 1, 1);
end

for k = 1:n
   temp3 = co3_2(k, :);
   mxlpsolve('add_constraint', lp, temp3, 1, 1);
end
```

```
% Fourth constraint

for k = 1:n-1
   temp4 = co4(k, :);
   mxlpsolve('add_constraint', lp, temp4, 2, 0);
end

% Name x_i_k, and y_k

start = 1;
for i = 1:n
   for j = 1:n
      names{start} = strcat('x_', num2str(i), '_', num2str(j));
      start = start + 1;
   end
end

for i = 1:n
   names{start} = strcat('y_', num2str(i));
   start = start + 1;
end

for i = 1:n^2+n
   mxlpsolve('set_col_name', lp, i, names{i});
end

% Write lpsolve input file

mxlpsolve('write_lp', lp, 'hw3');
```

The LPSolve input file looks like this:

```
min: +y_1 +y_2 +y_3 +y_4 +y_5 +y_6 +y_7 +y_8 +y_9 +y_10 +y_11 +y_12 +y_13 +y_14 +y_15 +y_16 +y_17 +y_18;
(18 lines of the following type:
ensure that all vertices will be colored exactly with one color.)
+x_1_1 +x_1_2 +x_1_3 +x_1_4 +x_1_5 +x_1_6 +x_1_7 +x_1_8 +x_1_9 +x_1_10 +x_1_11 +x_1_12 +x_1_13 +x_1_14 +x_1_15 +x_1_16 +x_1_17
+x_1_18 = 1;
.
.
+x_18_1 +x_18_2 +x_18_3 +x_18_4 +x_18_5 +x_18_6 +x_18_7 +x_18_8 +x_18_9 +x_18_10 +x_18_11 +x_18_12 +x_18_13 +x_18_14 +x_18_15
+x_18_16 +x_18_17 +x_18_18 = 1;
(324 lines of the following type:
ensure that if a color is never used, then a vertex cannot be colored with it.)
+x_1_1 -y_1 <= 0;
.
.
+x_18_18 -y_18 <= 0;
(688 lines of the following type:
ensure that for adjacent vertices, different colors will be assigned.)
+x_1_1 +x_2_1 <= 1;
.
.
+x_17_18 +x_18_18 <= 1;
(17 lines of the following type:
ensure that the use of adjacent colors only)
+y_1 -y_2 >= 0;
.
.
+y_17 -y_18 >= 0;
(ensure all variables are binary)
bin x_1_1, … , x_18_18, … , y_1, … , y_18;
```

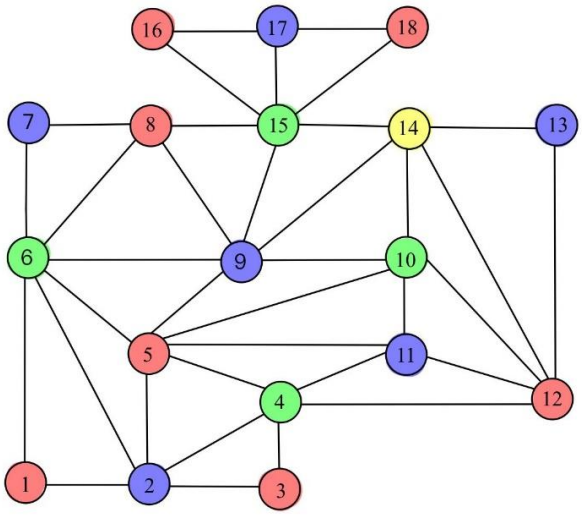
It took 0.077s to compute the following output file:

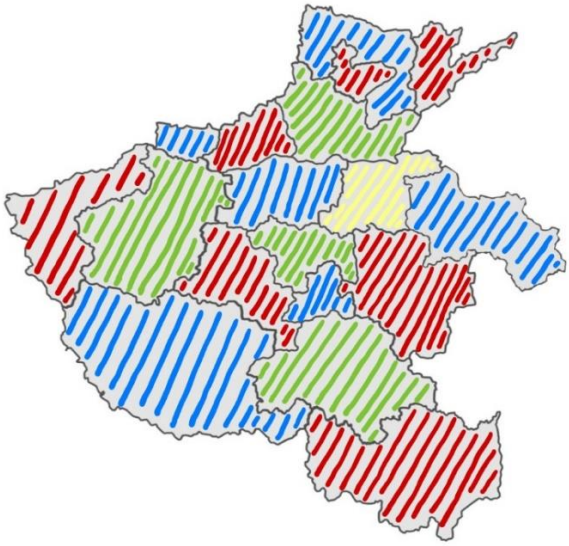
```
Value of objective function: 4.00000000

Actual values of the variables:
y_1                     1
y_2                     1
y_3                     1
y_4                     1
x_1_1                   1
x_2_2                   1
x_3_1                   1
x_4_3                   1
x_5_1                   1
x_6_3                   1
x_7_2                   1
x_8_1                   1
x_9_2                   1
x_10_3                  1
x_11_2                  1
x_12_1                  1
x_13_2                  1
x_14_4                  1
x_15_3                  1
x_16_1                  1
x_17_2                  1
x_18_1                  1

All other variables are zero.
```

We can demonstrate the output in the next page by coloring our graph and map. We use red (1), blue (2), green (3), and yellow (4) to represent the four colors used for color index $i = 1, 2, 3, 4$.





For the subgraph with vertices 5, 9, 10, 11, 12, and 14. If we start from 5, we see that it is in red. Since 9, 10, and 11 connect with 5, they cannot be colored with red. If we color 9 in blue, then 10 cannot be red or blue because 10 connects 9. So, we have 10 in green. Since 11 is not connected with vertex 9, we can have 11 in blue as well. Since 12 connects both 10 and 11, but not 5, we can color it with red. Now, we have 9 in blue, 10 in green, and 12 in red. Since 14 connects to 9 ,10, and 12, 14 cannot be colored with blue, green or red. It must be colored with the fourth color which in this case is yellow. Then for this subgraph, we show that exactly 4 colors are used, which proves that the 4 minimal colors set is the correct optimal solution.

IV. Exploration:

Now, we want to see that if we add more constraints to the LP we defined earlier, how will the minimal numbers of color change, and what will the graph look like. Our new constraint required that if two regions that border the same region, then they cannot be colored the same. For convenience, here is the table for E we defined in section II:

1	(1,2), (1,6)	10	(10,5), (10,9), (10,11), (10,12), (10,14)
2	(2,1), (2,3), (2,4), (2,5), (2,6)	11	(11,4), (11,5), (11,10), (11,12)
3	(3,2), (3,4)	12	(12,4), (12,10), (12,11), (12,13), (12,14)
4	(4,2), (4,3), (4,5), (4,11), (4,12)	13	(13,12), (13,14)
5	(5,2), (5,4), (5,6), (5,9), (5,10), (5,11)	14	(14,9), (14,10), (14,12), (14,13), (14,15)
6	(6,1), (6,2), (6,5), (6,7), (6,8), (6,9)	15	(15,8), (15,9), (15,14), (15,16), (15,17), (15,18)
7	(7,6), (7,8)	16	(16,15), (16,17)
8	(8,6), (8,7), (8, 9), (8,15)	17	(17,15), (17,16), (17,18)
9	(9,5), (9,6), (9,8), (9,10), (9,14), (9.15)	18	(18,15), (18,17)

For example, for vertex 1, there are two edges $(1, 2)$ & $(2, 6)$. With the edges we can see that both vertex 2 and vertex 6 border with vertex 1. Our new constraint requires vertices 2 and 6 to be in different colors even they may not border each other. However, we see that $(2, 6) \in E$. In our previous LP, we already limited that vertices 2 and 6 to be colored differently, so basically, we need to do nothing for this case.

Another example: for vertex 2, we require that vertices 1, 3, 4, 5, 6 need to all have different colors. Then we have the following vertices pairs that are not edges in our graph. To express the idea that those vertices pairs are not actual edges, we use [] instead of () like we used for edges:

$$[1, 3], [1, 4], [1, 5], [3, 5], [3, 6], [4, 6]$$

We can follow the example and found all the possible vertices pairs. There are 53 unique ones. Let F be the set of the 53 vertices pairs. Then our new constraint should look very similar with constraint (3) from the LP we have:

$$x_{ik} + x_{jk} \leq 1 \text{ for all } [i, j] \in F \text{ and } i, j, k = 1, \dots, 18$$

We add this new constraint to our LP. I created a MATLAB function to reduce the redundant work. The way to add constraint is the same so will not be shown in here again:

```
function coeff = co(x, y, n)

co = zeros(n, n^2+n);
for i = 1:n
    co(i, (x-1)*n+i) = 1;
    co(i, (y-1)*n+i) = 1;
end
coeff = co;

end
```

```
%% New constraint

co_new1 = co(1, 3, n);
co_new2 = co(1, 4, n);
co_new3 = co(1, 5, n);
co_new4 = co(1, 7, n);
co_new5 = co(1, 8, n);
co_new6 = co(1, 9, n);
co_new7 = co(2, 7, n);
co_new8 = co(2, 8, n);
co_new9 = co(2, 9, n);
co_new10 = co(2, 10, n);
co_new11 = co(2, 11, n);
co_new12 = co(2, 12, n);
co_new13 = co(3, 5, n);
co_new14 = co(3, 6, n);
co_new15 = co(3, 11, n);
co_new16 = co(3, 12, n);
co_new17 = co(4, 6, n);
co_new18 = co(4, 9, n);
co_new19 = co(4, 10, n);
co_new20 = co(4, 13, n);
co_new21 = co(4, 14, n);
co_new22 = co(5, 7, n);
co_new23 = co(5, 8, n);
co_new24 = co(5, 12, n);
co_new25 = co(5, 14, n);
co_new26 = co(5, 15, n);
co_new27 = co(6, 10, n);
co_new28 = co(6, 11, n);
co_new29 = co(6, 14, n);
co_new30 = co(6, 15, n);
co_new31 = co(7, 9, n);
co_new32 = co(7, 15, n);
co_new33 = co(8, 10, n);
co_new34 = co(8, 14, n);
co_new35 = co(8, 16, n);
co_new36 = co(8, 17, n);
co_new37 = co(8, 18, n);
co_new38 = co(9, 11, n);
co_new39 = co(9, 12, n);
co_new40 = co(9, 13, n);
co_new41 = co(9, 16, n);
co_new42 = co(9, 17, n);
co_new43 = co(9, 18, n);
co_new44 = co(10, 13, n);
co_new45 = co(10, 15, n);
co_new46 = co(11, 13, n);
co_new47 = co(11, 14, n);
co_new48 = co(12, 15, n);
co_new49 = co(13, 15, n);
co_new50 = co(14, 16, n);
co_new51 = co(14, 17, n);
co_new52 = co(14, 18, n);
co_new53 = co(16, 18, n);
```

It took the LPSolve client 8.53s to solve, and the output file looks like this:

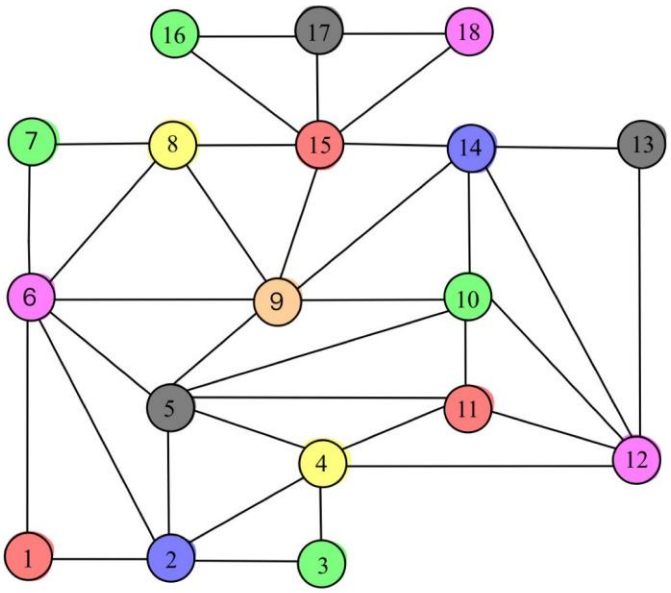
Value of objective function: 7.00000000

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_6	1
x_7_3	1
x_8_4	1
x_9_7	1
x_10_3	1
x_11_1	1
x_12_6	1
x_13_5	1
x_14_2	1
x_15_1	1
x_16_3	1
x_17_5	1
x_18_6	1

All other variables are zero.

We can see this this time we will need 7 different colors to color our graph to adapt the new constraint. We will use red (1), blue (2), green (3), yellow (4), black (5), pink (6), and beige (7), to represent the seven colors used for index $i = 1, 2, 3, 4, 5, 6, 7$. The graph and map is on the next page.



Notice that for vertices 5, 6, 8, 9, 10, 14, 15, there are exactly seven colors used in this sub-graph. Since vertices 5, 6, 8, 10, 14, 15 are all bounded with vertex 9, they all need to have different colors. This supports that we need minimal 7 different colors in this graph and map with this constraint. Combine with the graph and map, we can see the 7 different colors is the optimal solution.

V. Comment:

Comparing with the first map and the second map after we implemented the new constraint, the second map looks more appealing to me since there are not many overlaps in the colors. We can easily find out which are the cites border to the same city by looking at the colors.

However, the case may vary depend on the map, and the functionality of such map. For example, if we want to color a subway map, we will want more unique colors since we want people to distinguish from different metro lines by simply looking at their colors. For country maps, if we just want to slightly distinguish its states/provinces from each other, we may want to minimize the number of colors used. This is because a country usually has a lot of sub-district (states, province, etc.), the more colors we use, the more chaotic the map may look like. So, we want to use the minimal number of colors to distinguish between the different sub-district. But in real life, there are many conditions we need to consider, with different constraints applied, we can color the map to best serve for a particular purpose.