Bob Lin
Math381

Homework2

The problem is to find out the chromatic number when n = 1,2,3…, Suppose we have a graph G = (V, E) with n vertices where n is the number of vertex in a graph and V = $\{1, \dots, n\}$, and edge E defined as:

$$\mathrm{E} = \{(\mathrm{i}, \mathrm{j}) : \sin(\mathrm{i}) + \sin(\mathrm{j}) < 0, 1 \leq \mathrm{i}, \mathrm{j} \leq \mathrm{n}\}$$

The chromatic number is the minimal number of color needed to color all of vertices which any of two vertices connected by an edge cannot be same color.

Define $x_{ik} \in \{0,1\},\ i, k = 1, \dots, n\ with\ x_{ik} = 1$ iff we use color k to color vertex i.

Define $y_k \in \{0,1\},\ k = 1, \dots, n\ with\ y_k = 1$ iff we use color k.

To solve this LP problem of finding chromatic number in a graph G, first, I will build my objective function to minimize the color we need, which is:

$$\sum_{k=1}^{n} y_k$$

Then, there are totally five constraints that we need to consider:

First, because all variables are binary about if we use the color, we will make sure:

$$x_{ik}, y_k \in \{0,1\}$$

Second, for any vertex i, it need to be colored with exactly one color:

$$\sum_{k=1}^{n} x_{ik} = 1$$

Third, we want to keep track of which color is used, and we have following constraint for all i and k, so this make y_k = 1 when x_{ik} = 1 for any i

$$x_{ik} \leq y_k$$

Forth, we will use color in increasing order because it will be easier for us to read how many color is adapt when number of color increases. For example, we will use color1 before color2, and so on. Then the first $y_k = 0$ will implies us that k – 1 is the number of color needed. Also, because we always choose the first color available which is greedy algorithm, so this save us a lot of time from running it, so we set the constraint which is:

$$y_k \leq y_{k-1}\ with\ k = 2, \dots, n$$

Last and most important that we need to color nearby vertex with different color:

$$x_{ik} + x_{jk} \leq 1 \; for \; all \; (v_i, v_j) \in E \; and \; k = 1, \ldots, n$$

To carefully and fully adapt these condition, I wrote following code with example of five vertices:

```
import math

n = 5
V = []
E = []
color = []

for i in range(1, n + 1):
    V.append(i)
    color.append(i)

'''E = {(i,j): sin(i)+sin(j)<0, 1 ≤ i,j ≤ n}'''
for i in V:
    for j in range(i + 1, n + 1):
        if math.sin(i) + math.sin(j) < 0:
            E.append((i, j))

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

'''constraint1'''
for i in range(1, n + 1):
    output1 = ""
    for k in range(1, n + 1):
        output1 = output1 + "+x_" + str(i) + "_" + str(k)
    print(output1 + "=1;")

'''constraint2'''
for k in range(1, n + 1):
    for i in range(1, n + 1):
        print("+x_" + str(i) + "_" + str(k) + "<=y_" + str(k) + ";")

'''constraint3'''
for (i, j) in E:
    for k in range(1, n + 1):
        print("+x_" + str(i) + "_" + str(k) + "+x_" + str(j) + "_" + str(k) + "<=1;")

'''constraint4'''
for k in range(1, n):
    print("y_" + str(k+1) + "<=y_" + str(k) + ";")

'''constraint5'''
output = "bin "
for k in range(1, n + 1):
```

```
    output = output + "y_" + str(k) + ","
    for i in range(1, n):
        output = output + "x_" + str(i) + "_" + str(k) + ","
output = output + "x_" + str(n) + "_" + str(n)
print(output + ";")
```

The LP input file for n = 5:

min: +y_1+y_2+y_3+y_4+y_5;

(5 lines of the following type:

ensure that each vertex has been colored just one time)

+x_1_1+x_1_2+x_1_3+x_1_4+x_1_5=1;

.

.

(25 lines of the following type:

ensure that each vertex has been colored by a color i if and only if we use that color)

+x_1_1<=y_1;

.

.

(25 lines of the following type:

ensure that every nearby vertex is been colored by different color)

+x_1_1+x_5_1<=1;

.

.

(4 lines of the following type:

ensure that we use color in increasing order)

y_2<=y_1;

.

.

(ensure all variable are binary)

bin x_1_1, … , x_5_5, y_1, …, y_5;

This gives us the result:

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_1	1
x_3_1	1
x_4_3	1
x_5_2	1

(all other variables are zero.)

To test how n affect the chromatic number, I find out the chromatic number started from n = 1:

n	Chromatic number
1	1
2	1
3	1
4	2
5	3
6	4
7	4
8	4
9	4
10	5

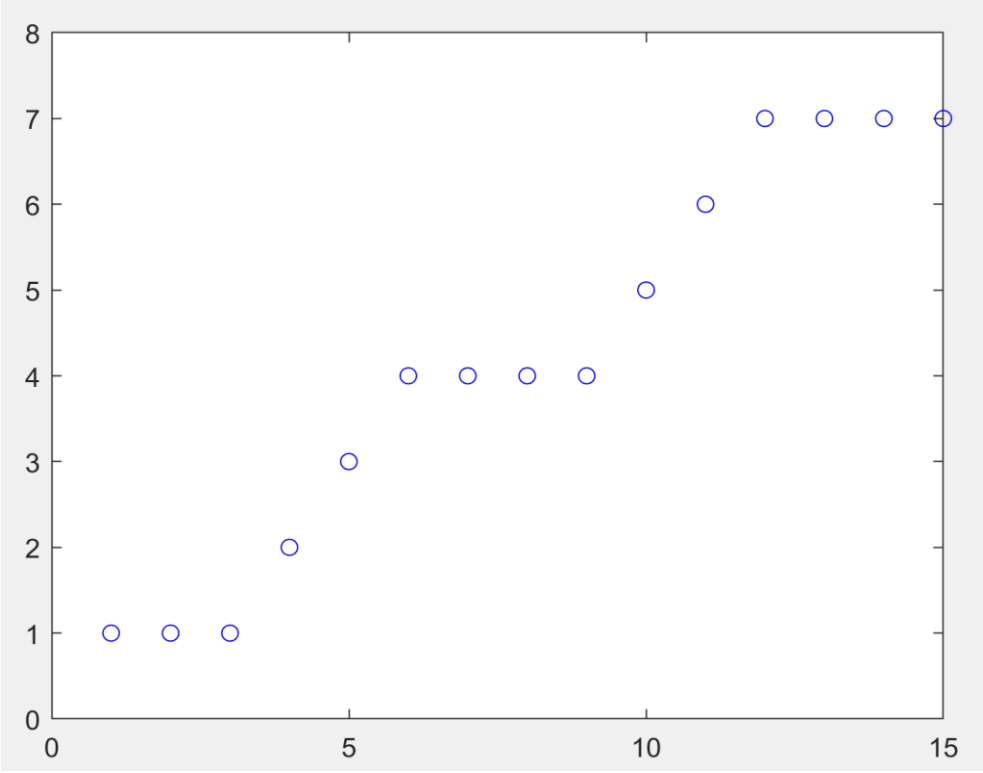
11	6 (here computer become very slow, but run is still fine)
12	7
13	7(it runs about 30-60 seconds)
14	7
15	7

For n = 15, the edge set is below:

{(1, 5), (1, 11), (2, 5), (2, 11), (3, 4), (3, 5), (3, 6), (3, 10), (3, 11), (3, 12), (4, 5), (4, 6), (4, 7), (4, 9), (4, 10), (4, 11), (4, 12), (4, 13), (4, 15), (5, 6), (5, 7), (5, 9), (5, 10), (5, 11), (5, 12), (5, 13), (5, 15), (6, 10), (6, 11), (6, 12), (7, 11), (8, 11), (9, 10), (9, 11), (9, 12), (10, 11), (10, 12), (10, 13), (11, 12), (11, 13), (11, 14), (11, 15), (12, 13)}

We find that vertex 11 has most degree which is 14 and vertex 14 has least degree which is 1. This is because sin 11 is the smallest number which is -0.999… and sin14 is the largest number which is 0.9906. Therefore, sin 11 add anything else will less than 0 and thus satisfy the condition of building an edge. Also, sin14 can only create an edge when adding sin11, because only sin 14 + sin 11 < 0 and satisfy our condition to build an edge

Here is the plot(with x-axis = n y-axis = chromatic number):



For the part of drawing a figure, I will draw the n = 5,6,7 and compare the result.

By python I see when n = 5 its edge is:

[(1, 5), (2, 5), (3, 4), (3, 5), (4, 5)]

When n = 6, its edge is:

[(1, 5), (2, 5), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (5, 6)]

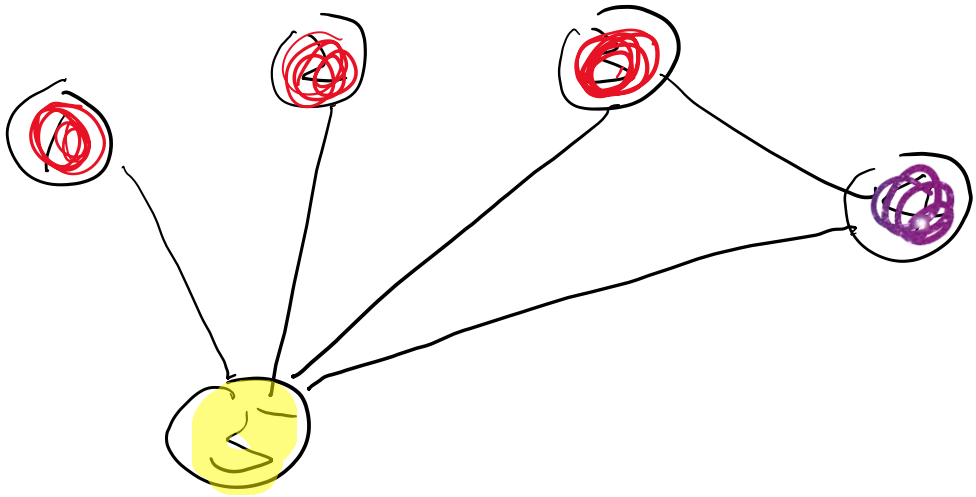
When n = 7, its edge is:

[(1, 5), (2, 5), (3, 4), (3, 5), (3, 6), (4, 5), (4, 6), (4, 7), (5, 6), (5, 7)]

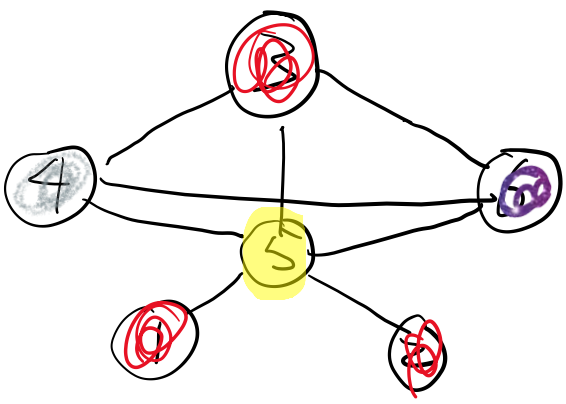
by observation I found that when n become larger, the previous edge will remain and only new edge satisfy the condition will be added, so this will only make number of edge remain the same or increase, then I draw the graph as following:

The graph is here, and it is correct chromatic number because we know that for a complete graph, the n is just equal the number of vertices, so we can find out the small cycle with most vertices form a complete graph, and the number of vertices is chromatic number, for example, for n = 5, 3,4,5 form a complete graph, so number of color is 3, and for n = 6, 3,4,5,6 form a complete graph and thus 4. To compare with the data I calculate with code, it gives me the same result of chromatic number, thus both the code and graph work!

n = 5:



n = 6:



n = 7: