Math381 Assignment 2

Yuchen AN

June 28, 2023

For assignment 2, We try to solve a knight math problem. Suppose we have three colors of knight which are black, white and red. We want to find the maximum number of knights we can put on a chessboard with an equal number of each color. And no knights of differing colors attack each other.

Suppose we have a chessboard with size m×n, with square indexed from 0 to m-1 and from 0 to n-1. Each square on the board can be represented by an ordered pair (i, j) with 0 ≤ i ≤m-1 and 0 ≤ j ≤n-1. Let b_{ij}, w_{ij}, r_{ij} represent the black knight, white knight, and red knight respectively, where (i, j) is the location of that knight on the board. Also, b_{ij}, w_{ij}, r_{ij} are binary variables which indicate whether or not there is a knight with corresponding color at (i, j).

$$b_{ij} = 0 \text{ if black knight is not at } (i, j).$$
$$b_{ij} = 1 \text{ if black knight is at } (i, j).$$
$$w_{ij} = 0 \text{ if white knight is not at } (i, j).$$
$$w_{ij} = 1 \text{ if white knight is at } (i, j).$$
$$r_{ij} = 0 \text{ if red knight is not at } (i, j).$$
$$r_{ij} = 1 \text{ if red knight is at } (i, j).$$

Let k_{ij} be the set of locations from which a knight at (i, j) could attack.

$$k_{ij} = \{\text{i}\pm2, \text{j}\pm1\} \cup \{\text{i}\pm1, \text{j}\pm2\}$$

Let A_{ij} be actual location on the board from which a knight can attack (i, j).

$$A_{ij} = k_{ij} \cap \{(\text{i,j}):\ 0 \le \text{i} \le \text{m}-1,\ 0 \le \text{j} \le \text{n}-1\}$$

Here is the mathematical formulation of LP I am going to solve:
The objective function:

$$\text{maximize } \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} b_{ij} + w_{ij} + r_{ij}$$

Subject to constraints:
1. The number of knights of each color need to be same.

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} b_{ij} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} w_{ij}$$
$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} w_{ij} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} r_{ij}$$

2. There can be only one knight with any color or no knight on one square. For example, if black knight is at b_{ij}, white knight and red knight cannot be at (i, j). To do this, I add up the possible knights that can be on a square. Since b_{ij},w_{ij},r_{ij} are binary variables, they can only be 1 or 0. So their sum must be less than or equal to one, which means that there can be one knight or no knight on that square.

$$b_{ij}+w_{ij}+r_{ij} \le 1$$

3. No knights of differing colors can attack each other. To be specific, if there is a black knight at location (i, j), then there cannot be a white or red knight at any location that attacks (i, j).Let the maximal number of squares that a knight can possibly attack be n. n is the number of knight that attack (i, j). n = $|A_{ij}|$. Take one constraint below as an example, if $b_{ij} = 1$, the sum of the white knights that the black knight at (i, j) can attack must be 0. If $b_{ij} = 0$, which means there is no black knight at (i, j), the sum of the white knights that can attack (i, j) can be less than or equal to n.

$$\sum_{(a,b)\in A_{ij}} w_{ab} \leq \text{n - n}b_{ij}$$
$$\sum_{(a,b)\in A_{ij}} r_{ab} \leq \text{n - n}b_{ij}$$
$$\sum_{(a,b)\in A_{ij}} b_{ab} \leq \text{n - n}w_{ij}$$
$$\sum_{(a,b)\in A_{ij}} r_{ab} \leq \text{n - n}w_{ij}$$
$$\sum_{(a,b)\in A_{ij}} b_{ab} \leq \text{n - n}r_{ij}$$
$$\sum_{(a,b)\in A_{ij}} w_{ab} \leq \text{n - n}r_{ij}$$

The complete mathematical formulation of LP I am going to solve:

$$\text{maximize } \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} b_{ij} + w_{ij} + r_{ij}$$
$$\text{subject to: } \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} b_{ij} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} w_{ij}$$
$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} w_{ij} = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} r_{ij}$$
$$b_{ij}+w_{ij}+r_{ij} \leq 1$$
$$\sum_{(a,b)\in A_{ij}} w_{ab} \leq \text{n - n}b_{ij}$$
$$\sum_{(a,b)\in A_{ij}} r_{ab} \leq \text{n - n}b_{ij}$$
$$\sum_{(a,b)\in A_{ij}} b_{ab} \leq \text{n - n}w_{ij}$$
$$\sum_{(a,b)\in A_{ij}} r_{ab} \leq \text{n - n}w_{ij}$$
$$\sum_{(a,b)\in A_{ij}} b_{ab} \leq \text{n - n}r_{ij}$$
$$\sum_{(a,b)\in A_{ij}} w_{ab} \leq \text{n - n}r_{ij}$$
$$b_{ij}, w_{ij}, r_{ij} \in \{0,1\},\ 0 \leq \text{i} \leq \text{m-1},\ 0 \leq \text{j} \leq \text{n-1}$$

I use Python to create the lpsolve input file. The following code helps to form this LP problem.

```
# The following code generates a lpsolve input file.
# This file determines the location of the black kinghts, white knights, and red knights at chessboar
# Furthermore, this file specifies three limitations and identifies the quantities of black knights,


import math


# define a function to check if a position is actually on the board.
def on_board(i,j,m,n):
    if(i>=0 and j >= 0 and i<m and j<n):
        return(1)
    else:
        return(0)

## define the moves a knight can make
knightMoves = [[1,2],[1,-2],[-1,2],[-1,-2],[2,1],[2,-1],[-2,1],[-2,-1]]

# board size is mxn
m = 5
n = 5

color = ["b", "w", "r"]

#### generate lpsolve input file
```

```
# objective function
fullString = ""
for c in color:
    for a in range(m):
        for b in range(n):
            fullString += "+"+c +"_"+str(a)+"_"+str(b)
print("max: "+fullString+";")

## constraint: The number of knights of each color need to be same

# The sum of balck knight
blackString = ""
for a in range(m):
    for b in range(n):
        blackString += "+b"+"_"+str(a)+"_"+str(b)

# The sum of white knight
whiteString = ""
for a in range(m):
    for b in range(n):
        whiteString += "+w"+"_"+str(a)+"_"+str(b)

# The sum of red knight
redString = ""
for a in range(m):
    for b in range(n):
        redString += "+r"+"_"+str(a)+"_"+str(b)

print(blackString+"="+whiteString+";")
print(blackString+"="+redString+";")

# constraint: There can be only one knight or no knight on one square.

for a in range(m):
    one_knight = ""
    for b in range(n):
        one_knight= "+b"+"_"+str(a)+"_"+str(b)+"+w"+"_"+str(a)+"_"+str(b)+"+r"+"_"+str(a)+"_"+str(b)
        print(one_knight+"<="+"1"+";")

# constraint: no knights of differing colors attack each other.
for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+w_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"b_"+str(a)+"_"+str(b)+";")

for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+r_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"b_"+str(a)+"_"+str(b)+";")

for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+b_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"w_"+str(a)+"_"+str(b)+";")


for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+r_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"w_"+str(a)+"_"+str(b)+";")


for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+b_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"r_"+str(a)+"_"+str(b)+";")


for a in range(m):
    for b in range(n):
        constraintString=""
        count = 0
        for move in knightMoves:
            x = a+move[0]
            y = b+move[1]
            if(on_board(x,y,m,n)):
                count = count +1
                constraintString+="+w_"+str(x)+"_"+str(y)
        if constraintString != "":
            print(constraintString+"<=+"+str(count)+"-"+str(count)+"r_"+str(a)+"_"+str(b)+";")

# declare all varaibles as binary
binString = "bin "
for c in color:
    for a in range(m):
        for b in range(n):
            if (a>0 or b>0):
                binString += ","
            binString += c+"_"+str(a)+"_"+str(b)
    binString += ","
print(binString+";")
```

To get input file for the chessboard with size 5x5, I assign m=5, n=5.
The input file looks like this:

```
max:+b_0_0+b_0_1+...+b_4_3+b_4_4+w_0_0+w_0_1+...+w_4_3+w_4_4+r_0_0+r_0_1+r_0_2+...+r_4_3+r_4_4;
(Two lines of the following type:
ensure the number of each color on the chessboard is same)
+b_0_0+b_0_1+...+b_4_2+b_4_3+b_4_4=+w_0_0+w_0_1+w_0_2+...+w_4_3+w_4_4;
+b_0_0+b_0_1+...+b_4_1+b_4_2+b_4_3+b_4_4=+r_0_0+r_0_1+...+r_4_3+r_4_4;
(nine lines of the following type:
ensure there is only one knight or no knight on each square)
+b_0_0+w_0_0+r_0_0<=1;
.
.
+b_4_4+w_4_4+r_4_4<=1;
(fourty eight lines of the following type:
ensure a specific color knight can only attack its own color)
+w_1_2+w_2_1<=+2-2b_0_0;
+w_1_3+w_2_2+w_2_0<=+3-3b_0_1;
.
.
+w_3_1+w_2_4+w_2_2<=+3-3r_4_3;
+w_3_2+w_2_3<=+2-2r_4_4;
(The following line ensure all varaibles are binary)
bin b_0_0,b_0_1,...,b_4_2,b_4_3,b_4_4,w_0_0,w_0_1,w_0_2,...,w_4_3,w_4_4,r_0_0,r_0_1,...,r_4_3,r_4_4,;
```

Here is the output generated by the lpsolver. For the purpose of readability, we only include the the location of knights on the chessboard.

```
 Value of objective function: 15.00000000

Actual values of the variables:
b_0_0                         1
b_0_2                         1
b_0_4                         1
b_1_0                         1
b_1_2                         1
w_0_1                         1
w_0_3                         1
```

```
w_3_0                         1
w_3_4                         1
w_4_2                         1
r_3_2                         1
r_4_0                         1
r_4_1                         1
r_4_3                         1
r_4_4                         1

All other variables are zero.
```

By solving the problem with various board sizes, the table below summarize the results and helps understand what happens when the board size changes. I found that the computation time will get longer as the board size became larger. The computational time increases rapidly with even small changes in the size of the board. Also, the density of knights is mostly concentrated within the range of 60% to 75%.

Board size	Knights	Knight Density	Computation Time
3x3	6	66.7%	0m0.015s
4x4	12	75%	0m0.229s
5x5	15	60%	3m9.814s
3x4	6	50%	0m0.163s
3x5	9	60%	0m0.656s
3x6	12	66.7%	0m1.079s
4x5	15	75%	0m0.687s
4x6	15	75%	1m23.784s
5x6	21	70%	3m49.813s
6x3	12	66.7%	0m1.310s
6x4	15	62.5%	1m5.039s
2x8	12	75%	0m1.178s
8x2	12	75%	0m1.178s

To check my solutions are valid and get better understanding of the content in the table above, I draw all of my solution boards. Each chessboard meets all constraints and objective function.

3x3 board

		R
	W	B
B	R	W

3x4 board

	R		W
B	W	B	R

3x5 board

W				R
B	R		W	
B	W	B	R	

3x6 board

W					
B	R		W	B	R
B	W	B	R	R	W

4x4 board

B	W	R	B
R			W
W			R
B	R	W	B

4x5 board

B	R	W	B	R
W	B		R	W
R				B
B	W		W	R

4x6 board

W		R		W	B
R				R	W
B				B	W
B	R	B		W	R

5x5 board

B			W	R
W				R
B	B		R	W
W				R
B	B		W	R

5x6 board

W	R	W	B	R	R
W	B		R	W	B
R					
B	W				W
B	R	B	W	B	R

6x3 board

R	W	R
	R	
W		W
B	W	B
B	R	B

6x4 board

R	W	R	R
	R		W
W			
	B		
B	W	B	
B	R	B	W

2x8 board

R	W		B	R	W		R
B	B	R	W	B			W

8x2 board

	R
R	W
	W
W	R
B	B
R	W
B	B