Game of Fivers

March 23, 2020

1 Game of Fivers

Combinatorial Optimization course, FEE CTU in Prague. Created by Industrial Informatics Department.

1.1 Motivation

Riddle: On a square board of size $(n \times n)$ there lie n^2 stones. Every stone has two sides - white and black. In the beginning, all stones have the white side facing upwards.

You may turn the stone (white to black or black to white), but if you do that, all the stones in its 4-neighborhood will be turned as well. You want to reach the state in which all the stones have their black sides facing upwards.

What is the minimal number of moves you need to do?

1.2 Input

You are given a positive integer $n \geq 3$, representing the size of the board.

[1]:
```
n = 5
```

1.3 Output

You should find a minimal number of moves that need to be done to reach the final states (all stones black). Also, you should provide the moves (e.g., a list of positions of the stones to be turned over).

1.4 Model

[2]:
```
import gurobipy as g

m = g.Model()

x = m.addVars(n+2, n+2, vtype=g.GRB.BINARY, obj=1)
y = m.addVars(range(1,n+1), range(1,n+1), vtype=g.GRB.INTEGER)
```

```
for i in range(1,n+1):
    for j in range(1,n+1):
        m.addConstr(x[i,j] + x[i+1, j] + x[i-1,j] + x[i,j+1] + x[i,j-1] ==␣
 ↪2*y[i,j] + 1)

m.addConstr(x.sum(0,"*") + x.sum(n+1,"*") + x.sum("*",0) + x.sum("*",n+1) == 0)

m.optimize()

X = [[int(round(x[i,j].X)) for j in range(1,n+1)] for i in range(1,n+1)]
```

```
Academic license - for non-commercial use only
Optimize a model with 26 rows, 74 columns and 174 nonzeros
Variable types: 0 continuous, 74 integer (49 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 22 rows and 67 columns
Presolve time: 0.00s
Presolved: 4 rows, 7 columns, 16 nonzeros
Variable types: 0 continuous, 7 integer (7 binary)
Found heuristic solution: objective 15.0000000

Explored 0 nodes (0 simplex iterations) in 0.01 seconds
Thread count was 4 (of 4 available processors)

Solution count 1: 15

Optimal solution found (tolerance 1.00e-04)
Best objective 1.500000000000e+01, best bound 1.500000000000e+01, gap 0.0000%
```

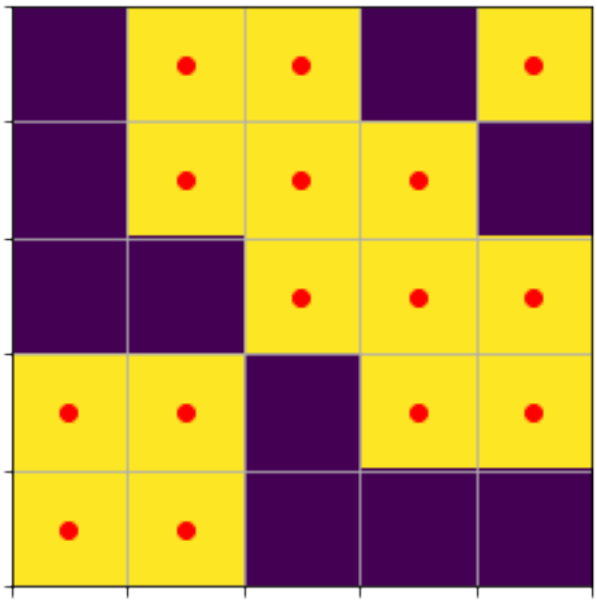
Visualization

[4]:
```
import matplotlib.pyplot as plt
import numpy as np

def visualize(board, n):
    board = np.array(board)
    clicks = np.argwhere(board == 1)
    plt.imshow(board, interpolation='none')
    plt.scatter(clicks[:, 1], clicks[:, 0], c='red')
    plt.gca().set_xticks(np.arange(-0.5, n, 1))
    plt.gca().set_xticklabels([])
    plt.gca().set_yticks(np.arange(-0.5, n, 1))
    plt.gca().set_yticklabels([])
```

```
    plt.grid()
    plt.show()

visualize(X, n)
```



1.5 Additional exercise

- Try to experiment with the model for different values of parameter n.
- See, how far is the model scalable (i.e., is it able to solve the problem for n ~ 10, n ~ 100, or even more in a reasonable time?).

[ ]: