IE501, Optimization Models
Lights Out Project Report

Group Members

Sanskar Shaurya (22B0985), Arihant Vashista (22B0958),
Nivesh Aggarwal (22B0912), Atharva Bendale (22B0901)

Contents

1 Problem Statement

1.1 Motivation for the problem

The motivation behind addressing the problem described in this project proposal stems from the need to tackle complex scenarios that involve toggling the states of interconnected elements, such as a grid of lights. The problem can have wide-ranging practical applications like:

- Puzzle Solving and Games
- Efficient Resource Utilization
- Localized Actions with Global Impact
- Customer Differentiation and Market Segmentation:
- Effect of actions of social media influencers on social media.

In summary, the motivation behind addressing this project problem lies in its relevance to a diverse range of real-world applications. By finding efficient solutions to this problem, we can gain valuable insights into resource allocation, network connectivity, localized actions, and optimization strategies that can benefit both theoretical research and practical decision-making processes.

1.2 Detailed Description of the Problem

The original problem statement for this project is:

> Given a n x n grid where each block can be either on (1) or off (0), we can toggle the state of each block but doing so will change the state of all the blocks that share a border with that particular block. This counts as one operation. Find whether we can turn off all the blocks in the grid using some series of operations and if it is possible, return the minimum number of operations required for this.

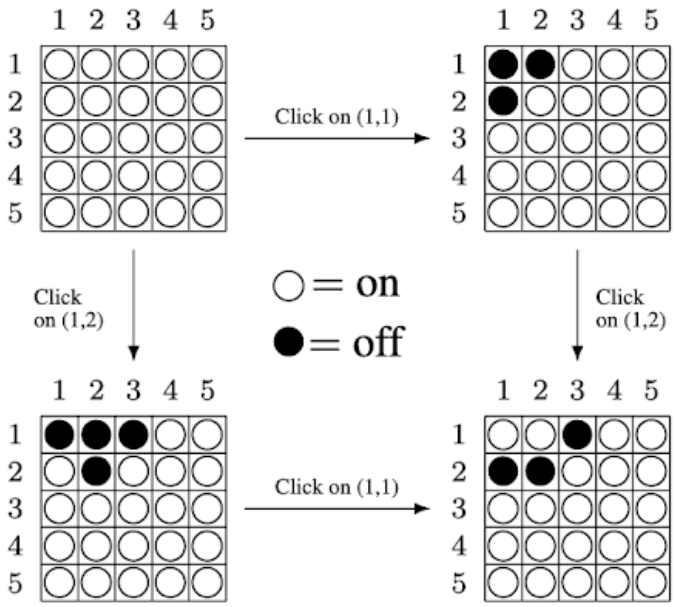

Figure 1: Diagram showing the toggle operation

1.3 Extensions

This problem for a n x n grid can be extended to a general graph where each block is a node and blocks sharing a border would have an edge between them. This problem can also be ex- tended to include 'k' states instead of just 2 (on and off). We have also extended this problem to the case where we are dealing with real variables and not integer variables. These sorts of problems have a lot of real life applications that we have formulated and discussed thoroughly in the later sections of this report.

2 Literature Survey

While searching for papers, reports, etc. covering the Lights Out game, we came across various facts and discoverings about the game that researchers have found out.

- Lights Out and its variations have not only been played by millions of people but have also been studied in detail by many. Several published contributors to the study of the mathematics behind the game include Jaap Scherphuis, Klaus Sutner, Don Pelletier, J. Goldwasser, W. Klostermeyer, G, Trapp and S. Kauffman. Their research extends the simplicity of the game to graph theory, linear algebra, polynomial sequences and many other topics
- Researchers have analyzed the computational complexity of solving "Lights Out" puzzles. They have classified the problem as NP-hard
- The game of Lights out, by Rebecca S. Meyer, 2013 This paper analyzes in depth about the Lights out game and it's many variations by extensively using linear algebra and finding out various patterns between different states.

3 Formulation and Solving

3.1 Solving all-on configuration for a grid

We started by solving the all-on configuration of the Lights Out puzzle, where all the lights are "on" initially. If we observe the effects of toggling neighboring cells of any cell, we should toggle the neighboring cells and the cell itself an odd number of times. Because of the odd number of effects on the cell, its configuration will change to "off". Note that toggling any cell for more than n(>1) times is equivalent to toggling it for n mod(2) times, and as we are trying to minimize the number of toggles, we decide whether we will toggle a cell or not.

3.1.1 Formulation

Decision Variables:
Let x_{ij} be our binary decision variables, let it be equal to 1 if cell (i, j) is toggled once and 0 if it is not. Let N_{ij} denote the set of cell (i', j') that are neighbors of the cell (i, j). We use another variable $z_{ij} \in \{0, 1, 2\}$. Then we can formulate the Linear Programming Problem as shown below.

Objective Function:

$$\textbf{min} \sum_i \sum_j x_{ij}$$

Constraints:

$$x_{ij} + \sum_{(i',j') \in N_{ij}} x_{i'j'} = 2 \cdot z_{ij} + 1 \ \forall \ (i,j)$$
$$x_{ij} \in \{0,1\} \quad z_{ij} \in \{0,1,2\} \quad \forall \ (i,j)$$

3.2 Solving general configuration for a grid

The 'all-on' initial state can be converted into general initial state by making a minor change introducing a parameter 'alpha' for all the blocks, which tells us whether the corresponding block is on (1) or off (0). The formulation for the corresponding Linear Program is given below:

3.2.1 Formulation

Decision Variables:
All the decision variables will be the same, we have just introduced a new set of parameters α_{ij} for each block (i,j) which denotes the initial state of the block, 1 if on and 0 if off. The previous LP formulation can be adjusted slightly to give:

Objective Function:

$$\textbf{min} \sum_i \sum_j x_{ij}$$

Constraints:

$$x_{ij} + \sum_{(i',j') \in N_{ij}} x_{i'j'} = 2 \cdot z_{ij} + \alpha_{ij} \ \forall \ (i,j)$$
$$x_{ij} \in \{0,1\} \quad z_{ij} \in \{0,1,2\} \quad \forall \ (i,j)$$

3.3 Solving general configuration for any given graph

We can further extend the problem to any general graph, where the neighbor relation between two members is shown by an edge between the corresponding two nodes and toggling the state of a node will toggle the states of all the nodes which are adjacent to it. The formulation for the corresponding Linear Problem is:

3.3.1 Formulation

Decision Variables:
For any general graph $G(V,E)$, we define our binary decision variables as $x_i, i \in \{1,2,...,|V|\}$, which is 1 if the i^{th} vertex is toggled on else it is 0. Let $E_i = \{(i,j) : (i,j) \in E\}$ and α_i denote the initial state of the i^{th} vertex, which is 1 if the vertex is initially on else it is 0. The Linear Program will be:
Objective Function:

$$\textbf{min} \sum_i x_i$$

Constraints:

$$x_i + \sum_{(i,j) \in E_i} x_j = 2 \cdot z_i + \alpha_i \ \forall \ i$$
$$x_i \in \{0,1\} \quad z_{ij} \in \mathbb{Z} \quad \forall \ i$$

4 A Real Problem

4.1 Motivation for a real problem

The above work motivated us to consider a real problem which some advertising company might face, which can be solved by generalising the above formulations to include real numbers.

Suppose there is a city which is divided into several sectors. An advertising company can invest certain amount of resources in a sector which leads to some customer acquisition in that sector. This will also lead to some customer acquisition from the neighbouring sectors, but less than the main sector. Further the company can only spend amount from a certain range in any given sector. The goal of the company is to reach a customer acquisition threshold in each sector while minimising the amount of resources spent.

4.2 Modelling the problem

The group of sectors is considered as a graph G, where each sector represents a vertex and neighbouring sectors are connected by edges. To model the customer acquisition we have assumed a simplified model which assumes that spending an amount x_i in a sector leads to acquisition of $x_i \cdot u_i$ new customers. u_i is a constant determined from previous data. Further acquisition of $x_i \cdot u_i$ will further lead to acquisition of $x_i \cdot u_i \cdot w_{ij}$ customers from sector j which is a neighbour of sector i and w_{ij} represents the correlation between these sectors with respect to customer acquisition. α_i and β_i are the minimum and maximum amount the company can spend on some sector.

4.3 Linear Program Formulation

Decision Variables:
Let G(V,E) be the graph, let $x_i \in \mathbb{R}^+$ be the number of resources we spend in any vertex, let $w_{ij} \in \mathbb{R}$ be the correlation weight for the vertices i & j. Let y_i be their initial state and let the threshold we need to cross on each vertex is t, then the LP is:

Objective Function:

$$\textbf{min} \sum_{i=1}^{|V|} x_i$$

Constraints:

$$y_i + x_i \cdot u_i + \sum_{j=1, j\not=i}^{|V|} (w_{ij} \cdot x_j \cdot u_i) \geq t \quad \forall\ i \in \{1, 2, ..., |V|\}$$

$$\alpha_i \leq x_i \leq \beta_i \quad \forall\ i \in \{1, 2, ..., |V|\}$$

$$x_i \in \mathbb{R}^+$$

5 Coding Up The Solution

5.1 Python Script

We used the library **Pulp** in python for coding up the solution to our linear programs. The code is given below:

```python
import pulp as p

# Create a LP minimization problem
problem = p.LpProblem('Problem', p.LpMinimize)

n = int(input("n:"))
m = int(input("m:"))
alpha = [[None for i in range(m)] for i in range(n)]

print("Enter the initial state of the grid:")

for i in range(n):
    alpha[i] = [int(x) for x in input().split()]

#Create a 2D array of binary variables
x=[[None for i in range(m)] for i in range(n)]
z=[[None for i in range(m)] for i in range(n)]
# Create problem variables
for i in range(n):
    for j in range(m):
        x[i][j] = p.LpVariable(f"x_{i}{j}", cat=p.LpBinary)
        z[i][j] = p.LpVariable(f"z_{i}{j}", cat=p.LpInteger)

# Objective function
s = 0
for i in range(n):
    for j in range(m):
        s += x[i][j]

problem += s

# Constraints
for i in range(n):
    for j in range(m):
        neighboring_cells = [(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)]
        c = x[i][j]
        for neighbor in neighboring_cells:
            if 0 <= neighbor[0] < n and 0 <= neighbor[1] < m:
                c += x[neighbor[0]][neighbor[1]]
        problem += c == 2*z[i][j] + alpha[i][j]

solver = p.PULP_CBC_CMD()
problem.solve(solver)


# Print the status of the problem (should be 1 for Optimal)
print("Status:", p.LpStatus[problem.status])

# Print the values of the binary variables
for i in range(n):
    for j in range(m):
        print(x[i][j], "=", x[i][j].value())

# Print the optimized objective function value
print("Objective value:", p.value(problem.objective))
```

The code for all the extended variations that we discussed above can be found in https://github.com/SanskarShaurya/Lights_Out/

5.2 Usage

This python script asks the user for the grid size initially and then takes in the initial state of the grid.

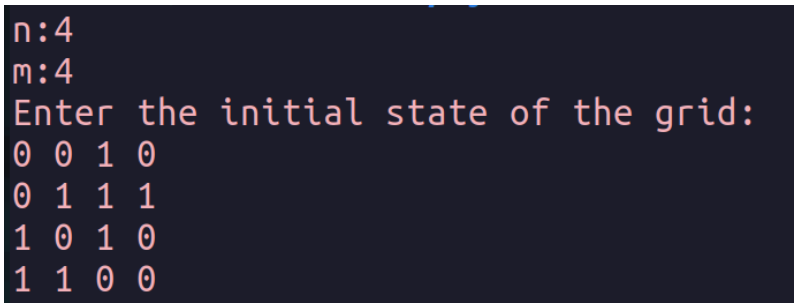

Figure 2: Taking input

After this it solves the Linear Integer Program and gives the values of the decision variables for which it is optimal (if there exists a solution, otherwise it says "infeasible"). The 1s represent the blocks that have to be toggled. The objective value represents the minimum number of moves to turn off the grid.

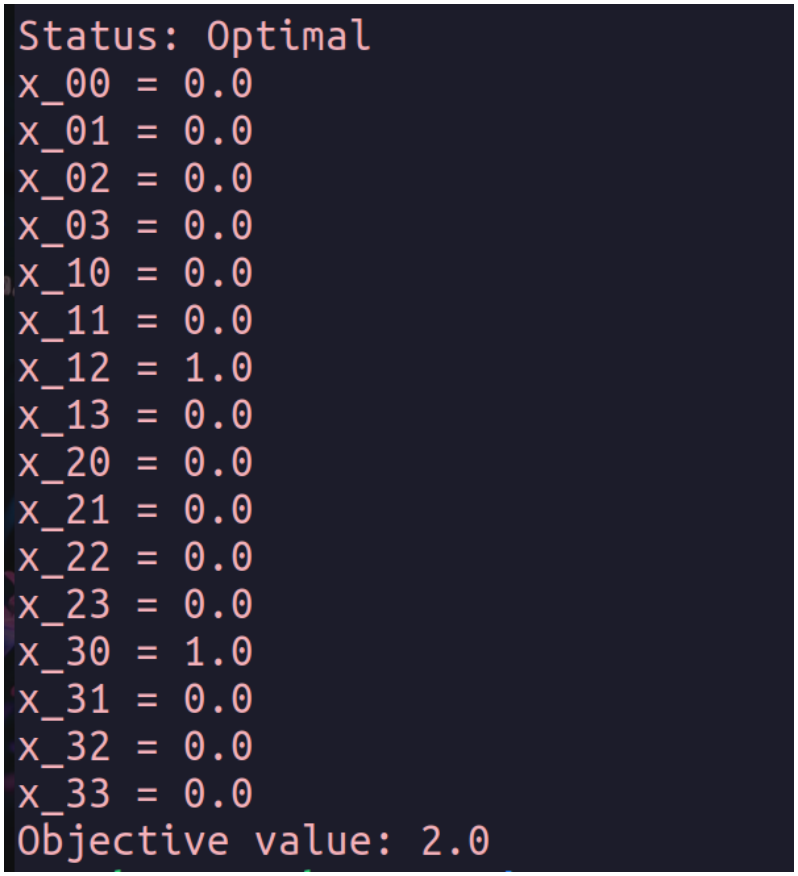

Figure 3: Output

After some modifications in the above code, we got the extended variants' solutions too using the same pulp library.

6 An interesting Observation

While trying different graphs , we observed that for any graph, if the intial condition is all on, then there exists a solution which makes the final condition all-off. This infact turned out to be true. Here's proof for the theorem inspired from [1]

Theorem *The all-on configuration is solvable for any undirected graph.*

Proof. If $n = 1$, then there is only one node in G. Selecting this node will flip its state from on to off.

Assume the claim is true for graphs with at most n nodes. Consider a graph G with $n + 1$ nodes. For any node v in G, let X_v be an activation set for the n-nodes graph $G - \{v\}$. If there exists a node v such that X_v is also a solution for G, then we are done. Otherwise, each X_v will turn all nodes off except node v.

If n is odd, then we apply all the activitation sets X_v, for all nodes v. Each node v will not change its state when we apply X_v, and it will change its state an odd number of times when we apply the union of all sets X_w, for all $w \neq v$. Since there is an odd number of such nodes w, v will change its state and turn off.

If n is even, then at least one node v of G has even degree (the sum of all node degrees in G equals twice the number of edges, i.e., it is even). We first activate v. As a result, all nodes in $N[v]$ will turn off. Then we apply the activation sets X_u, for all $u \in G - N[v]$. Since $n + 1$ and $|N[v]|$ are odd, there is an even number of such nodes u. Thus, the nodes in $N[v]$ are not affected and will remain off because they will change their state an even number of times, while the nodes in $G - N[v]$ will change their state an odd number of times (X_w does not change the state of w), i.e., they will also turn off.

References

[1] Andrej Brodnik Alejandro López-Ortiz Venkatesh Raman Alfredo Viola (Eds.) *Space-Efficient Data Structures, Streams, and Algorithms*. Springer.