

Slitherlink

Anh Minh TRAN

Solving Slitherlink Puzzle with Linear Programming

Monday 3rd March, 2025

Table of contents

- 1 Introduction to Slitherlink Game Rules
- 2 Building a Linear Programming Model to Solve the Slitherlink Game
 - Linear Programming
 - Setting up Variables for the Model
 - Setting up Constraints for the Model
 - Setting up the objective function for the model
 - Handling disconnected cases
- 3 Model Implementation
- 4 Conclusion

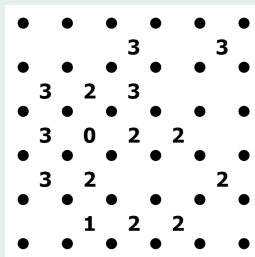
Table of contents

- 1 Introduction to Slitherlink Game Rules
- 2 Building a Linear Programming Model to Solve the Slitherlink Game
 - Linear Programming
 - Setting up Variables for the Model
 - Setting up Constraints for the Model
 - Setting up the objective function for the model
 - Handling disconnected cases
- 3 Model Implementation
- 4 Conclusion

Introduction to Slitherlink Game

Slitherlink Game Rules

The basic premise of Slitherlink involves a rectangular grid of dots size $(n+1) \times (m+1)$, where certain cells in the grid contain numbers ranging from 0 to 4.



Introduction to the Slitherlink Game

Rules of Slitherlink

- Draw a single, continuous loop that does not intersect or branch off.
- The loop must pass through some of the dots on the grid, connecting adjacent dots horizontally or vertically.
- The loop must match the numbers in the cells, with each number indicating the exact count of loop segments around that cell.
- Any number of loop segments can surround cells without numbers.

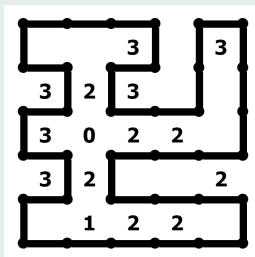


Table of contents

- 1 Introduction to Slitherlink Game Rules
- 2 Building a Linear Programming Model to Solve the Slitherlink Game
 - Linear Programming
 - Setting up Variables for the Model
 - Setting up Constraints for the Model
 - Setting up the objective function for the model
 - Handling disconnected cases
- 3 Model Implementation
- 4 Conclusion

Linear Programming

A linear programming (LP) model is a mathematical method for optimizing an objective, like maximizing profit or minimizing cost, subject to linear constraints. It consists of:







- **Decision Variables:** Variables that are adjusted to optimize the objective function.
- **Objective Function:** A linear equation to maximize or minimize a specific quantity.
- **Constraints:** Linear inequalities or equations that limit the values of the decision variables.

Example:

$$\begin{array}{ll}\max & x + y + z \\ \text{subject to} & 3x + 2y - z = 150 \\ & 4x - 3y + 2z \geq 180 \\ & 2x + 5y + 2z \leq 200 \\ & x \geq 0, \quad y \leq 0\end{array}$$

Setting up Variables for the Model

Setting up Variable System







Since the final result must be a single loop, each dot (i, j) can only have one of these states: , , , , , . We call this set of states Σ . A dot may also not have any edges connected to it.

Therefore, we will define a system of binary variables as follows:

$$x[i, j, \sigma] = \begin{cases} 1 & \text{if dot } (i, j) \text{ has state } \sigma \\ 0 & \text{if dot } (i, j) \text{ does not have state } \sigma \end{cases}$$

$$\forall 1 \leq i \leq n + 1; 1 \leq j \leq m + 1, \sigma \in \Sigma.$$

Setting up Variables for the Model

DotState						
(1, 1)	0	0	0	1	0	0
(1, 2)	0	0	1	0	0	0
(1, 3)	0	0	0	0	0	0
(2, 1)	0	0	0	0	1	0
(2, 2)	0	1	0	0	0	0
(2, 3)	0	0	1	0	0	0
(3, 1)	0	1	0	0	0	0
(3, 2)	0	0	0	0	0	1
(3, 3)	1	0	0	0	0	0

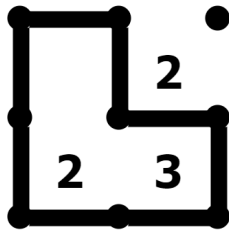


Figure: A solved puzzle and its representation

Setting up Constraints for the Model

Constraint 1

For each dot (i, j) , we can only fill in one state from the set $\Sigma = \{\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}, \begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \square \end{smallmatrix}, \begin{smallmatrix} \blacksquare & \blacksquare \\ \square & \blacksquare \end{smallmatrix}, \begin{smallmatrix} \blacksquare & \blacksquare \\ \square & \square \end{smallmatrix}, \begin{smallmatrix} \blacksquare & \square \\ \blacksquare & \square \end{smallmatrix}, \begin{smallmatrix} \blacksquare & \square \\ \square & \square \end{smallmatrix}\}$. So we have:

$$\sum_{\sigma \in \Sigma} x[i, j, \sigma] \leq 1$$

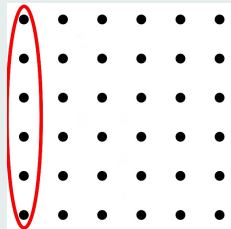
$$1 \leq i \leq n + 1; 1 \leq j \leq m + 1.$$

Setting up Constraints for the Model

Constraint 2




We observe that dots lying outside the left boundary cannot have states $\begin{smallmatrix} \blacksquare & \cdot \\ \cdot & \cdot \end{smallmatrix}$, $\begin{smallmatrix} \cdot & \blacksquare \\ \cdot & \cdot \end{smallmatrix}$, or $\begin{smallmatrix} \cdot & \cdot \\ \cdot & \blacksquare \end{smallmatrix}$, hence:

$$\begin{cases} x[i, 1, \begin{smallmatrix} \blacksquare & \cdot \\ \cdot & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1 \\ x[i, 1, \begin{smallmatrix} \cdot & \blacksquare \\ \cdot & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1 \\ x[i, 1, \begin{smallmatrix} \cdot & \cdot \\ \cdot & \blacksquare \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1. \end{cases}$$

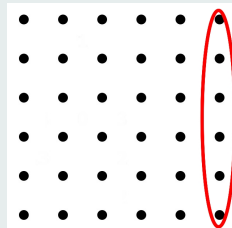


Setting up Constraints for the Model

Constraint 3

We observe that dots lying outside the right boundary cannot have states , , or , hence:

$$\begin{cases} x[i, m+1, \begin{smallmatrix} \bullet & \bullet \\ \bullet & \bullet \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1 \\ x[i, m+1, \begin{smallmatrix} \bullet & \bullet \\ \bullet & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1 \\ x[i, m+1, \begin{smallmatrix} \bullet & \cdot \\ \bullet & \bullet \end{smallmatrix}] = 0 & \forall 1 \leq i \leq n+1. \end{cases}$$

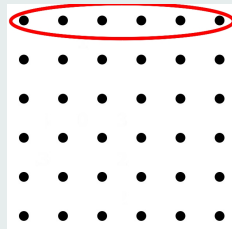


Setting up Constraints for the Model

Constraint 4

We observe that dots lying outside the top boundary cannot have states $\begin{smallmatrix} \square & \square \\ \cdot & \cdot \end{smallmatrix}$, $\begin{smallmatrix} \square & \cdot \\ \cdot & \cdot \end{smallmatrix}$, or $\begin{smallmatrix} \cdot & \cdot \\ \cdot & \cdot \end{smallmatrix}$, hence:

$$\begin{cases} x[1, j, \begin{smallmatrix} \square & \square \\ \cdot & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1 \\ x[1, j, \begin{smallmatrix} \square & \cdot \\ \cdot & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1 \\ x[1, j, \begin{smallmatrix} \cdot & \cdot \\ \cdot & \cdot \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1. \end{cases}$$

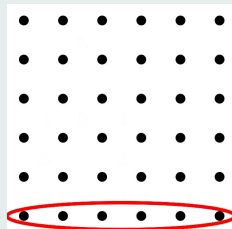


Setting up Constraints for the Model

Constraint 5

We observe that vertices lying outside the bottom boundary cannot have states $\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}$, $\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \text{r} \end{smallmatrix}$, or $\begin{smallmatrix} \blacksquare & \blacksquare \\ \text{l} & \blacksquare \end{smallmatrix}$, hence:

$$\begin{cases} x[n+1, j, \begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1 \\ x[n+1, j, \begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \text{r} \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1 \\ x[n+1, j, \begin{smallmatrix} \blacksquare & \blacksquare \\ \text{l} & \blacksquare \end{smallmatrix}] = 0 & \forall 1 \leq j \leq m+1. \end{cases}$$



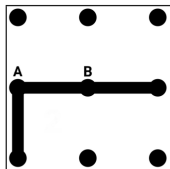
Setting up Constraints for the Model

Constraint 6

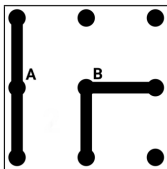
Adjacent dots horizontally must have compatible states, meaning the states of two adjacent dots must reflect whether their common edge is connected or disconnected:

$$x[i, j, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}] + x[i, j, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}] + x[i, j, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}] = x[i, j+1, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}] + x[i, j+1, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}] + x[i, j+1, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}]$$

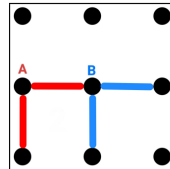
$$\forall 1 \leq i \leq n+1, 1 \leq j \leq m.$$



(a) Compatibly connected



(b) Compatibly disconnected



(c) Incompatible

Setting up Constraints for the Model

Constraint 7

Similarly, for two adjacent vertices vertically:

$$x[i, j, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}] + x[i, j, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}] + x[i, j, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}] = x[i+1, j, \begin{smallmatrix} \blacksquare \\ \blacksquare \end{smallmatrix}] + x[i+1, j, \begin{smallmatrix} \blacksquare \\ \square \end{smallmatrix}] + x[i+1, j, \begin{smallmatrix} \square \\ \blacksquare \end{smallmatrix}]$$

$$\forall 1 \leq i \leq n, 1 \leq j \leq m+1.$$

Setting up Constraints for the Model

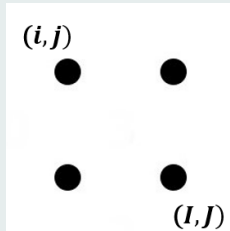
Constraint 8

For every cell with the top left dot at (i, j) and a value a_{ij} (where $0 \leq a_{ij} \leq 4$), we have:

$$\begin{aligned} & x[i, j, \begin{array}{|c|c|} \hline \bullet & \bullet \\ \hline \end{array}] + x[i, j, \begin{array}{|c|c|} \hline \bullet & \square \\ \hline \end{array}] + 2x[i, j, \begin{array}{|c|c|} \hline \square & \bullet \\ \hline \end{array}] + x[i, j, \begin{array}{|c|c|} \hline \square & \square \\ \hline \end{array}] + x[i, j, \begin{array}{|c|c|} \hline \bullet & \bullet \\ \hline \end{array}] \\ & + 2x[I, J, \begin{array}{|c|c|} \hline \bullet & \bullet \\ \hline \end{array}] + x[I, J, \begin{array}{|c|c|} \hline \bullet & \square \\ \hline \end{array}] + x[I, J, \begin{array}{|c|c|} \hline \square & \bullet \\ \hline \end{array}] + x[I, J, \begin{array}{|c|c|} \hline \square & \square \\ \hline \end{array}] + x[I, J, \begin{array}{|c|c|} \hline \bullet & \bullet \\ \hline \end{array}] = a_{ij}. \end{aligned}$$

Where

$$\begin{cases} I = i + 1 \\ J = j + 1 \\ 1 \leq i \leq n \\ 1 \leq j \leq m \end{cases}$$

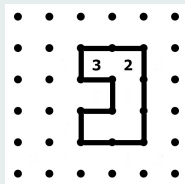


Setting up the objective function for the model

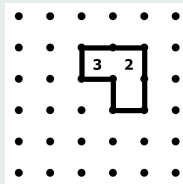
Setting up the objective function

Since each time $x[i, j, \sigma] = 1$ ($1 \leq i, j \leq n+1$, $\sigma \in \Sigma$), 2 additional edges are connected. Moreover, each edge is counted once at both its dots, so the sum of all variables in the model represents the number of connected edges. We need to minimize the number of connected edges.

$$\min \sum_{1 \leq i \leq n+1; 1 \leq j \leq m+1, \sigma \in \Sigma} x[i, j, \sigma].$$



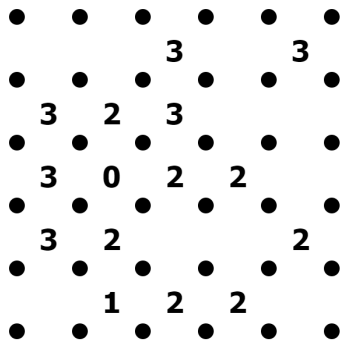
Suboptimal Solution



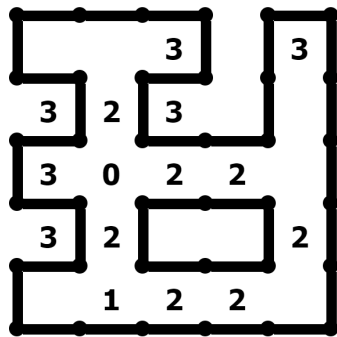
Optimal Solution

Handling disconnected cases

After optimizing the model with variables, constraints, and the objective function as above, disconnected components may still occur.



Problem



Disconnected Solution

Handling disconnected cases

Handling disconnected solutions

Let the set of states of the dots after the optimization model be V .

$$V = \{(i, j, \sigma) \mid x[i, j, \sigma] = 1; 1 \leq i \leq n + 1 \wedge 1 \leq j \leq m + 1 \wedge \sigma \in \Sigma\}.$$

We choose any connected component with dot set C . We call the set of states of this connected component as

$$V_C = \{(i, j, \sigma) \mid (i, j, \sigma) \in V \wedge (i, j) \in C\}.$$

Let the magnitude of V and V_C be $\overline{V}, \overline{V_C}$ ($\overline{V_C} \leq \overline{V}$). If $\overline{V_C} \neq \overline{V}$, we add the following constraint to the model and re-optimize:

$$\sum_{v \in V_C} x[v] \leq \overline{V_C} - 1.$$

Table of contents

- 1 Introduction to Slitherlink Game Rules
- 2 Building a Linear Programming Model to Solve the Slitherlink Game
 - Linear Programming
 - Setting up Variables for the Model
 - Setting up Constraints for the Model
 - Setting up the objective function for the model
 - Handling disconnected cases
- 3 Model Implementation
- 4 Conclusion

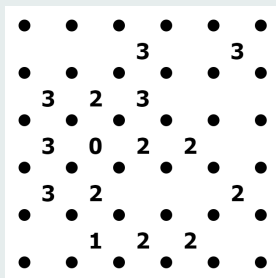
Model Implementation

Model Implementation

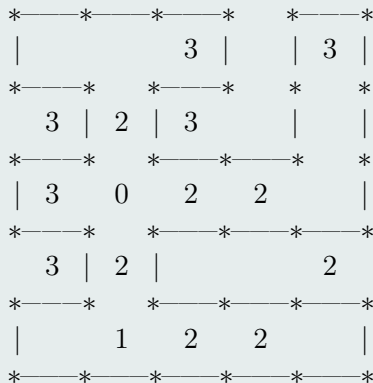
We implement the model using the gurobipy library.

Model Implementation

Program Output



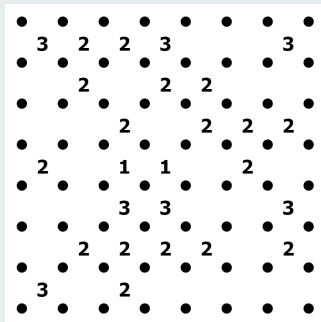
5 × 5 Problem



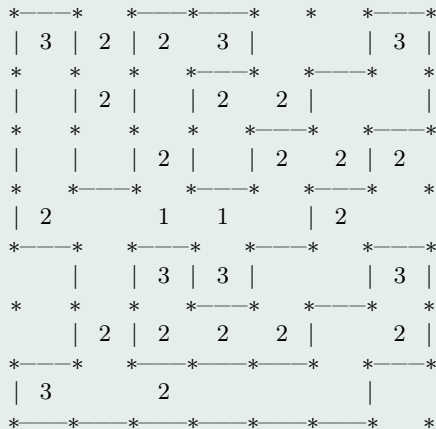
Program Output Solution

Model Implementation

Program Output



7 × 7 Problem



Program Output Solution

Table of contents

- 1 Introduction to Slitherlink Game Rules
- 2 Building a Linear Programming Model to Solve the Slitherlink Game
 - Linear Programming
 - Setting up Variables for the Model
 - Setting up Constraints for the Model
 - Setting up the objective function for the model
 - Handling disconnected cases
- 3 Model Implementation
- 4 Conclusion

Merci pour votre attention