Università degli Studi di Padova
Dipartimento Matematica

Yashi Game

Knowledge Representation and Learning Project

Dejan Dichoski

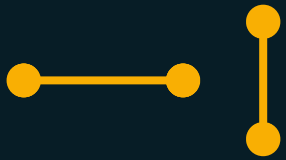
September 2023

Introduction

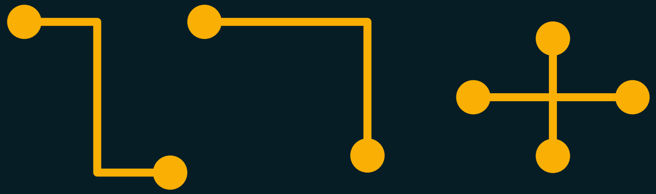
- YASHI is the Contest Center's line-drawing logic puzzle.
- Goal: Connect all of the dots using horizontal and vertical lines and without crossing lines. There must be exactly one path connecting any dot to any other.

Rules

- Only straight lines are allowed: horizontal or vertical.
    - Not allowed: diagonal lines or curved lines, L-shapes, no Z-shapes, paths that have angles or corners.
    - Each dot may be connected to 1, 2, 3 or 4 other dots, either horizontally, vertically or both.
- Lines cannot cross each other.
    - They cannot pass through other dots, or go on top of other lines.



Allowed



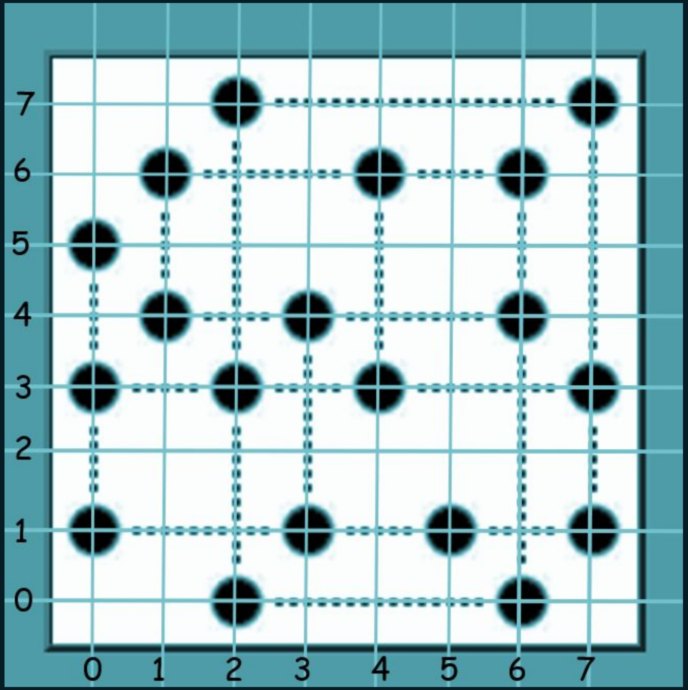
Not allowed

Rules

- There should be exactly one path from any dot to any other dot.
  - May pass through any number of intermediate dots.
- Lines cannot form any closed loops. For example, there could not be a path: Dot A - Dot B - Dot C - Dot D - Dot A.
- The number of lines must always be 1 less than the number of dots.
  - More lines => There are some closed loops.
  - Fewer lines => The network has several unconnected pieces.

Notation

- Grid: NxN
- Points: $P = \{(i, j) \mid 0 \leq i < N, 0 \leq j < N\}$
- Lines: $L = \{(p1, p2) \mid p1, p2 \in P, p1 \neq p2\}$



8x8 grid

Start

Decide if there is a solution.

Version 1

Outcome

Any solution

Version 2

Outcome

A minimum-length solution.

Possible approaches

How to automate the Yashi game?

1. Programming approach - write a program that will exhaustively search for a solution.
2. Constraint-solving approach - write constraints that model the conditions for a Yashi solution, and then let a constraint solver do all the computation - thus avoiding the hard and error-prone work of programming.

Constraints (rules) that must be satisfied

1. Diagonal lines aren't allowed:
   - Restrict movement to UP, DOWN, LEFT and RIGHT.
2. All points must be connected.
   - Use a graph based approach.

3. Lines cannot cross each other.
4. Exactly n−1 lines must be used (n is the number of points).
5. No closed loops.

SAT solver

Using a SAT solver

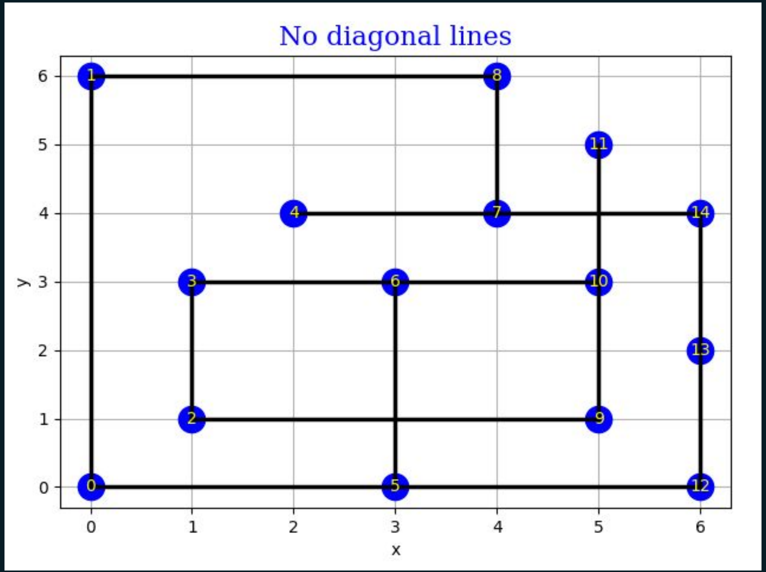
- Use the segment names as Boolean variables and then write propositional formulas that enforce the selection of a subset that satisfies the constraints.

- If Boolean variable is assigned truth-value True (resp. False), then the corresponding line segment is included (resp. excluded) from the solution.

- Moreover, since we want to use a SAT solver to solve the constraints, it is better if these propositional formulas are in CNF.

Constraint “no diagonal lines”

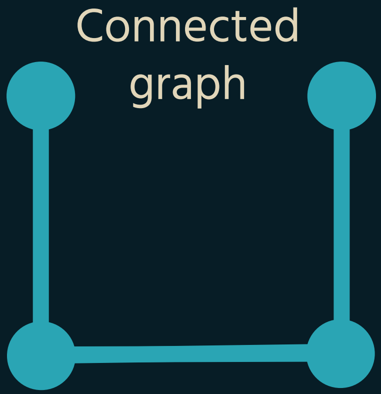
- Construct a dictionary of lines.
    - For every point, move in all four allowed directions: ↑, ↓, ← and →.
    - If you encounter another point before reaching the grid boundary, draw a line.
- Note: The output configuration violates other constraints, which will be addressed in the following slides.

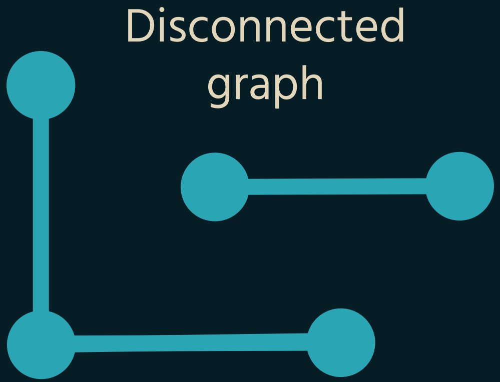


A fully-connected grid

All points must be connected - DFS

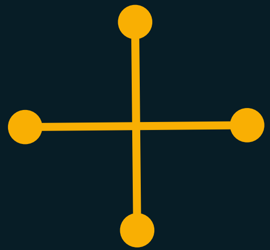
- There is a path between every pair of vertices.
  - Number of vertices in the graph = number of visited vertices by DFS.
- The lines must form exactly one graph.
  - Number of isolated subgraphs = 1.
- Complexity: O(n), where n is the number of vertices.





Constraint “no crossing lines”

- Find “programmatically” all pairs of lines that cross:
  - Library: `from shapely.geometry import LineString`
  - Check: `line_string_1.intersects(line_string_2) and not line_string_1.touches(line_string_2)`



- Add a constraint for each pair of crossing lines:

$$\varphi_{no\_crossing} = \mathrm{CNF}\left(\bigwedge_{\substack{L_i,\ L_j \\ L_i \neq L_j \\ is\_crossing(L_i, L_j)}} \overline{L_i \wedge L_j} \right) = \bigwedge_{\substack{L_i,\ L_j \\ L_i \neq L_j \\ is\_crossing(L_i, L_j)}} \overline{L_i} \vee \overline{L_j}$$

Constraint “exactly n-1 lines”

- Use “Exactly k” constraint, with n = #lines, and k = #points - 1.

$$\varphi_{exactly\_n-1\_lines} = \underbrace{\left(\bigwedge_{\substack{I \subseteq [n] \\ |I| = n-k+1}} \bigvee_{i \in I} L_i \right)}_{\text{At least k}} \wedge \underbrace{\cancel{\left(\bigwedge_{\substack{I \subseteq [n] \\ |I| = k+1}} \bigvee_{i \in I} \overline{L_i} \right)}}_{\text{At most k}}$$

- Improvement: "At least k” is sufficient because the “no closed loops” constraint ensures “at most k” lines.

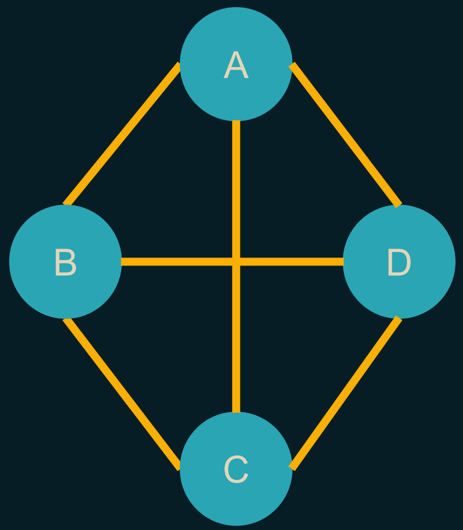
Constraint “no cycles”

- Find all cycles C using a graph-theory-based approach.
- Add a constraint for all the lines L_i belonging to a cycle c_i:

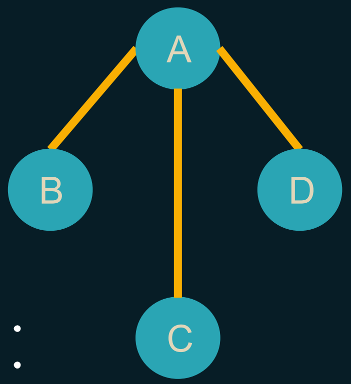
$$\varphi_{no\_cycles} = CNF\left(\bigwedge_{c_i \in C} \overline{\bigwedge_{L_i \in c_i} L_i}\right) = \bigwedge_{c_i \in C} \bigvee_{L_i \in c_i} \overline{L_i}$$

Finding cycles (loops)

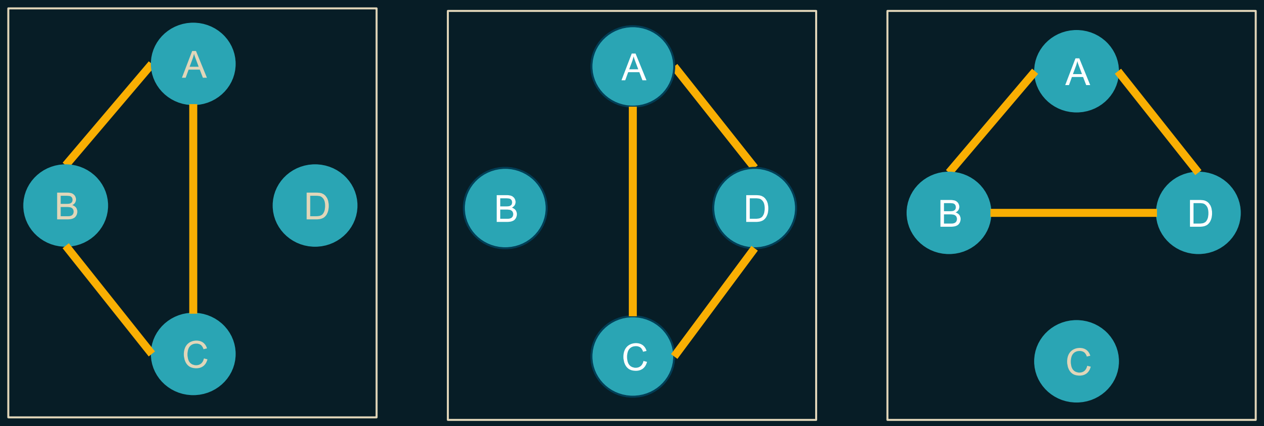
- Represent the grid of points as a graph:
  - points = vertices, lines = edges.
- Build a spanning tree.
- Find all fundamental cycles.
  - Look for all edges which are present in the graph but not in the tree.
  - Adding one of the missing edges to the tree will form a fundamental cycle.
- $N_{FC} = E - V + 1$, E = #edges, V = #vertices.
- Use XOR operator to merge the cycles.
- Downside: The code scales exponentially with the number of fundamental cycles in the graph.



Finding fundamental cycles in a graph

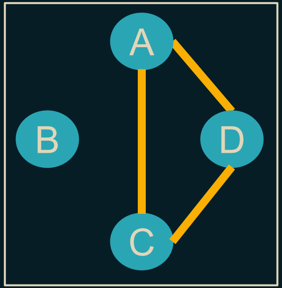


Spanning tree

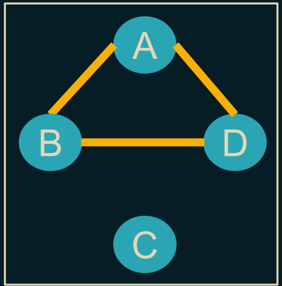


Fundamental cycles

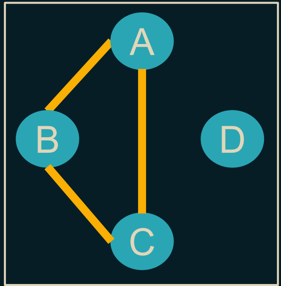
Finding all cycles in a graph



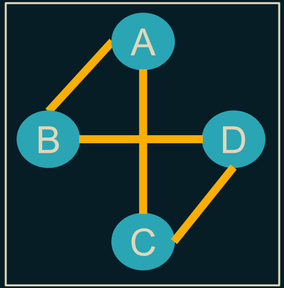
(1)



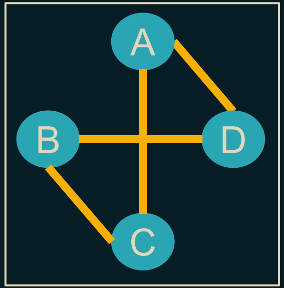
(2)



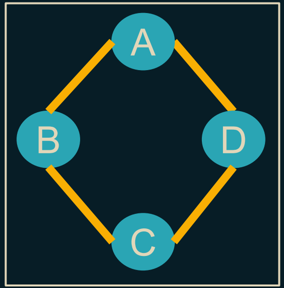
(3)



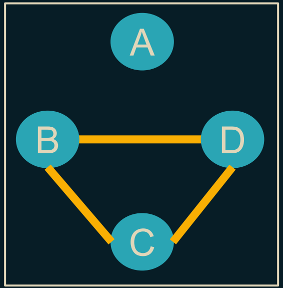
(1) ⊕ (2)



(2) ⊕ (3)



(1) ⊕ (3)



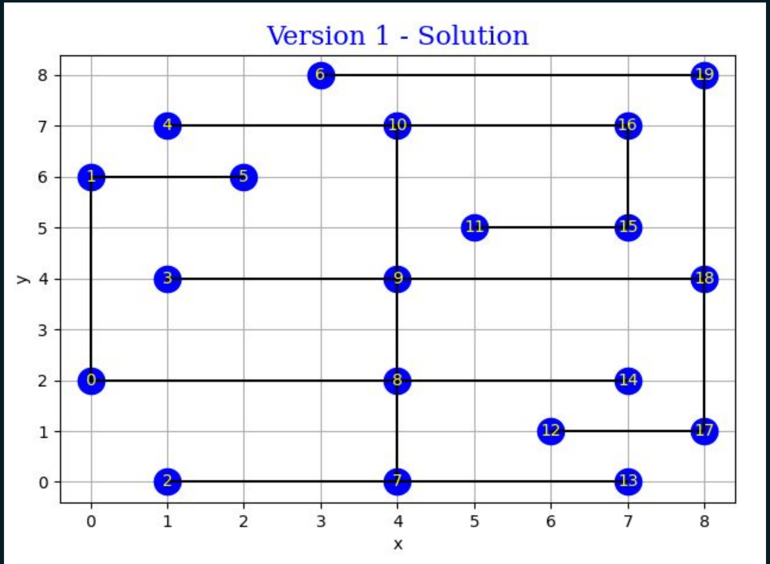
(1) ⊕ (2) ⊕ (3)

Combine the fundamental cycles.

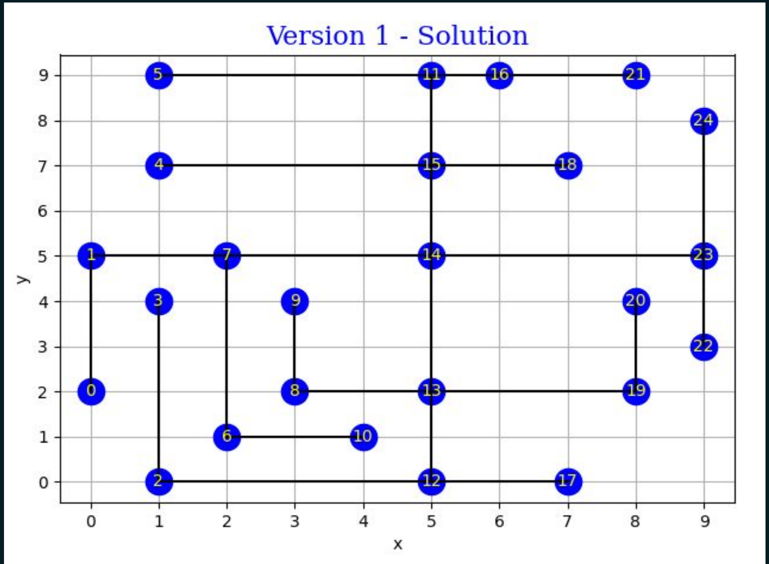
Yashi Game Version 1

1. Read input data
2. Construct lines
3. Check graph connectivity using DFS
4. Generate hard constraints
5. Pass to MiniSat
6. Check whether a solution is found
7. Get a model from the solver
8. Output solution

Yashi Game Version 1



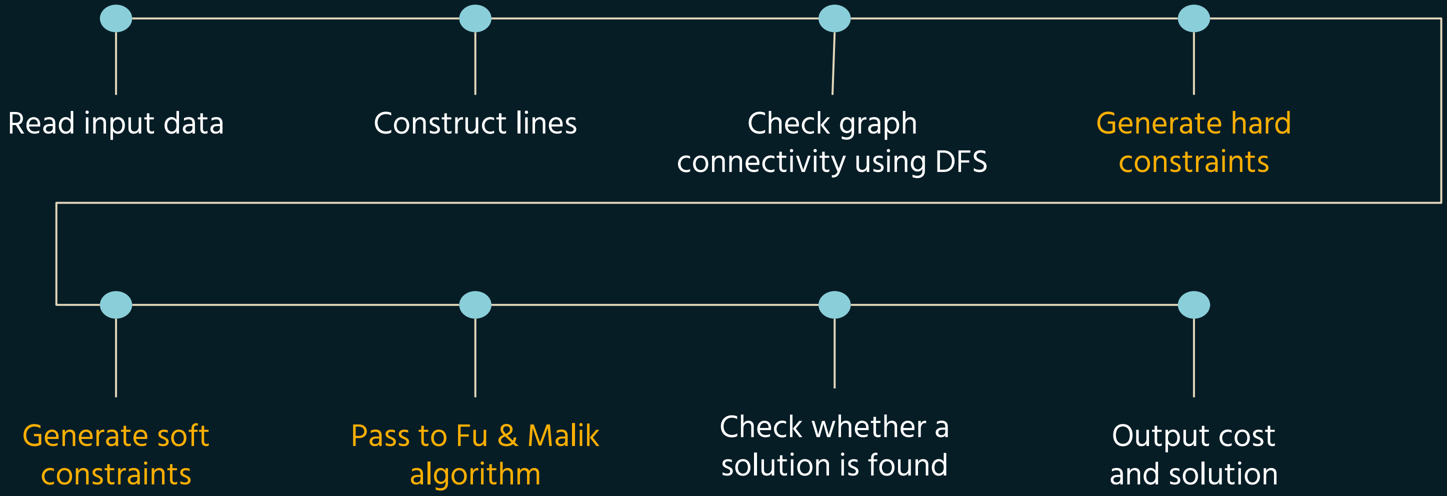
Solution for 9x9 game



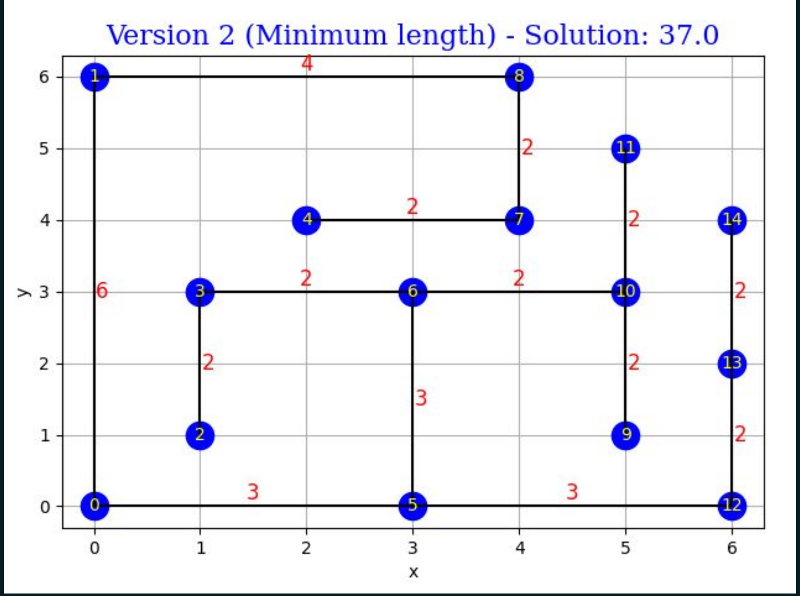
Solution for 10x10 game

Yashi Game Version 2

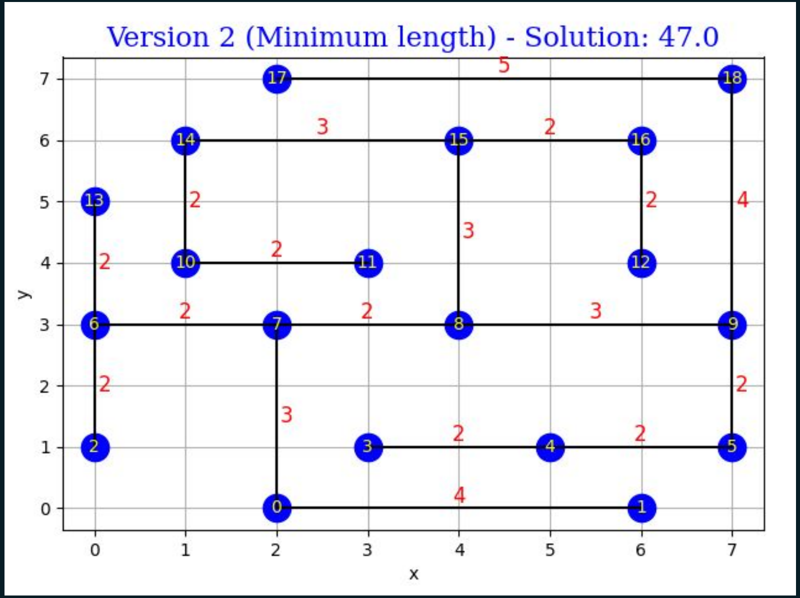
- Version 2: If there is a solution, return a minimum-length solution.
  - Minimize the sum of the lengths of the lines.
  - Cost of the solution = negative Euclidean distance.



Yashi Game Version 2



Solution for 7x7 game



Solution for 8x8 game

Thanks for your attention.

Do you have any questions?

Let’s run some examples.

Demo time