Assignment 2: Solving Hashi Puzzle as a CSP

Meltem Ceylantekin 28089

Ayşenur Güller 27796

Variables:

- b[i, j]: This array represents the board itself. Each element b[i, j] holds an integer value between -1 and 8.
  - -1: Empty square (not an island and potential bridge)
  - 1..8: Island square with a number indicating the required number of bridges.
  - B (defined as -1): Potential bridge square.
- horizontal[i, j]: Boolean variable indicating a horizontal bridge at position (i, j). True if there's a horizontal bridge, False otherwise.
- vertical[i, j]: Boolean variable indicating a vertical bridge at position (i, j). True if there's a vertical bridge, False otherwise.

Domains:

- b[i, j]: Domain is {-1, 1, 2, ..., 8, B}
- horizontal[i, j]: Domain is {True, False}
- vertical[i, j]: Domain is {True, False}

Constraints:

- Islands have no bridges:
  - *forall(i in H, j in W where b[i,j] != B)(horizontal[i,j] == false /\ vertical[i,j] == false)*
  - Ensures islands don't have bridges.
- Mutual exclusion for bridges:
  - *forall(i in H, j in W) ((horizontal[i,j] -> not vertical[i,j]) /\ (vertical[i,j] -> not horizontal[i,j]))*
  - Guarantees only one bridge type (horizontal or vertical) can exist at a position.
- Bridge connections to islands:
  - *forall(i in H, j in W where b[i,j] in I)(...)*
  - This constraint ensures the number of bridges connected to an island matches its number.
- Horizontal bridge continuity: These constraints ensure horizontal bridges connect islands horizontally:
  - *forall(i in H, j in W where b[i,j] in I) (...)*
  - Checks if a horizontal bridge at an island connects to another island or bridge on the left and right.
  - *forall(i in H, j in W where j > 1 /\ j < w) (...)*
  - Enforces continuity between bridges within a row (not at the edges).
  - *forall(i in H, j in W) (horizontal[i, j] -> (sum(j1 in W)(b[i, j1] in I) >= 2))*
  - Ensures at least two islands on the same line for a horizontal bridge.
- Horizontal bridge placement:
  - *forall(i in H, j in W where j >= 1 /\ j < w) (...)*
  - Ensures islands on both sides of a horizontal bridge (not at the edges).
  - Constraints forall(i in H) (not horizontal[i, 1]) and forall(i in H) (not horizontal[i, w]) - Prevent bridges at the board's edges.
- Vertical bridge continuity:

- Similar to horizontal bridges, these constraints ensure vertical bridges connect islands vertically:
  - *forall(i in H, j in W where b[i,j] in I) (...) -*
  - Checks for island or bridge connections above and below a vertical bridge at an island.
  - *forall(i in H, j in W where i > 1 /\ i < h) (...) -*
  - Enforces continuity between bridges within a column (not at the top and bottom).
- Vertical bridge placement:
- *forall(j in W) (not vertical[1, j]) and forall(j in W) (not vertical[h, j])*
- Prevent bridges at the top and bottom edges of the board.

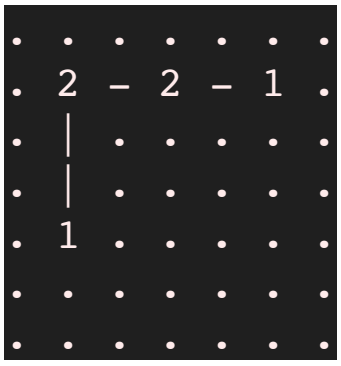
This representation ensures that the puzzle rules (such as bridge connections, island constraints, and bridge continuity) are enforced, leading to a solution where all islands are connected by the correct number of bridges. Adjustments to this model can be made based on specific puzzle requirements and desired behaviors.

Input:
-1 Represents empty cells, numbers represents islands.

EASY BOARD: *There is smaller number of possible bridge.*
Input

```
h = 7;
w = 7;
b = [| -1,-1,-1,-1, -1,-1,-1
    | -1,2,-1,2,-1,1,-1
    |  -1,-1,-1,-1 ,-1,-1,-1
    | -1,-1, -1,-1, -1,-1,-1
    | -1,1,-1, -1,-1,-1,-1
    | -1,-1,-1,-1,-1,-1,-1
    |  -1,-1, -1,-1,-1,-1,-1 |];
```

Output



MEDIUM BOARD: *There is more possible bridge.*

Input

```
h = 7;
w = 7;
b = [| 1,-1,-1,-1, -1,-1,1
     |-1,-1,-1,-1,-1,-1,-1
     | -1,2,-1,1 ,-1,-1,-1
     |-1,-1, -1,-1, -1,-1,-1
     |-1,2,-1, 3,-1,-1,2
     |-1,-1,-1,-1,-1,-1,-1
     | -1,-1, -1,2,-1,-1,2 |];
```

Output