CS404-A2

Görkem Yar, Kaan Bilgili

April 2024

1 Solving Hashi Puzzle as a CSP

In the Hashi puzzle, some of the cells contain a number in the range 1 to 8 both inclusive which represents the amount of bridges that this island is connected to. The aim is to connect all the islands into a single connected component using bridges.
Input: A nxn grid that shows the location of the islands with the number of bridges the island is required to make.
Output: A grid that shows the islands with the bridges. Bridges are represented as:

- One horizontal bridge "-"
- Two horizontal bridges "="
- One vertical bridge "l"
- Two vertical bridges "x"

To solve this problem as a CSP problem, we created a variable matrix called bridgesBetween to represent the number of bridges between two islands. In this context, bridgesBetween will be addressed as BB. BB is a MxM matrix where BB_{ij} represents the number of bridges between island i and island j.
Domain: Dom[BB_{ij}] = 0, 1, 2. Since the number of bridges between two islands is restricted to the maximum amount of 2.

1.1 Constraints

1.1.1 A bridge must begin and end at distinct islands as a straight line in between.

To ensure this constraint we need to enforce $BB_{ii} = 0$; i ϵ {1,2..,M}. For the other variables BB_{ij} where i != j, the domain is not restricted in this step.

1.1.2 A bridge may only run vertically or horizontally.

For the islands that do not share the same column or row index, the BB_{ij} constraint is set to 0. For the ones that share a column or a row, the BB_{ij} is given a constraint that implies the BB_{ij} should be smaller or equal to the minimum(Bridges(i), Bridges(j)), where Bridges represent the initial number of bridges that are given to each island.

- (i.row != j.row) and (i.column != j.column) $\Rightarrow BB_{ij} = 0$
- (i.row == j.row) or (i.column == j.column) $\Rightarrow BB_{ij} \leq$ minimum(Bridges(i), Bridges(j))

1.1.3 The number of bridges connected to each island must match the number on that island.

Since the BB_{ij} shows the number of bridges between the island i and j, we can calculate the number of bridges that are connected to the island i by summing BB_{ij} for all j.

- $\forall_i \; \Sigma_{j=1}^{M} BB_{ij}$ = Bridges(i)

This constraint is added to imply the necessary amount of bridges.

1.1.4 Bridges are symmetric

The bridges between two islands are symmetric meaning that $BB_{ij} = BB_{ji}$ for all i and j.

1.1.5 A bridge must not cross any other bridges or islands

Assuming that there is a bridge between island i and j, $BB_{ij} > 0$. Then for the all islands k and l, there should not be any bridges that intersect the bridge between i and j. There are four sub-constraints to imply the non-intersection of the bridges and islands, namely row-col, col-row, row-row, and col-col intersections.

Constraint for Row-Col Intersections: Assuming that there exists a horizontal bridge between island i and j, i.row = j.row. For all islands k and l that differ from i and j, we cannot create a vertical bridge that has the following conditions.

- i.row = j.row, k.col = l.col. i, j, k, l. All different.

- c1 = min(i.col, j.col), c2 = max(i.col, j.col), r1 = min(k.row, l.row), r2 = max(k.row, l.row).
- r1 $\leq$ i.row $\leq$ r2 and c1 $\leq$ k.col $\leq$ c2

If all these conditions are satisfied, $BB_{kl} = 0$, to prevent the intersection of bridges. The equality in the last remark implies the island bridge intersections as well.

Constraint for Col-Row Intersections: Assuming that there exists a vertical bridge between island i and j, i.col = j.col. For all islands k and l that differ from i and j, we cannot create a horizontal bridge that has the following conditions.

- i.col = j.col, k.row = l.row. i, j, k, l. All different.
- r1 = min(i.row, j.row), r2 = max(i.row, j.row), c1 = min(k.col, l.col), c2 = max(k.col, l.col).
- r1 $\leq$ k.row $\leq$ r2 and c1 $\leq$ i.col $\leq$ c2

If all these conditions are satisfied, $BB_{kl} = 0$, to prevent the intersection of bridges. The equality in the last remark implies the island bridge intersections as well.

Constraint for Row-Row Intersections: Assuming that there exists a horizontal bridge between island i and j, i.row = j.row. For all islands k and l that differ from i and j, we cannot create a horizontal bridge that has the following conditions.

- i.row = j.row = k.row = l.row. At least three of the i, j, k, l should be different.
- c1 = min(i.col, j.col), c2 = max(i.col, j.col), c3 = min(k.col, l.col), c4 = max(k.col, l.col).
- c1 $\leq$ c3 $\leq$ c2 or c3 $\leq$ c1 $\leq$ c4
- Also, c1 != c4 and c2 != c3 since it will lead to straight bridges between three islands.

If all these conditions are satisfied, $BB_{kl} = 0$, to prevent the intersection of bridges.

Constraint for Col-Col Intersections: Assuming that there exists a vertical bridge between island i and j, i.col = j.col. For all islands k and l that differ from i and j, we cannot create a vertical bridge that has the following conditions.

- i.col = j.col = k.col = l.col. At least three of the i, j, k, l should be different.
- r1 = min(i.row, j.row), r2 = max(i.row, j.row), r3 = min(k.row, l.row), r4 = max(k.row, l.row).
- r1 $\leq$ r3 $\leq$ r2 or r3 $\leq$ r1 $\leq$ r4
- Also, r1 != r4 and r2 != r3 since it will lead to straight bridges between three islands.

If all these conditions are satisfied, $BB_{kl} = 0$, to prevent the intersection of bridges.

1.1.6 All islands should be in the same connected component.

To check this condition, we created a recursive function that returns a boolean variable that checks the connectivity between two islands. This function recursively checks the connected islands and returns true if the destination is reachable. Let's name this function as checkConnected(start, destination). It checks the reachability of the destination from the start.

Using this function we implied connectivity in the following way:

- Since we want all the islands to be connected, all islands need to be connected to island 1.
- Using the checkConnected we must imply all the islands are connected to the island 1, checkConnected(1, otherIslands).
- If the above condition is correct for all islands, then all islands should be connected via the following logic.
- The connectivity is symmetric so if 1 is connected u then u is connected to 1.
- The connectivity is transitive so if 1 is connected to u and v is connected to 1, then v is connected to u.
- For this reason if the implication holds, all islands are connected.