CS404 - Homework 2 Kemal Sarper Yücel - syucel - 21031

Solving Akari Using CSP

To model the given akari problem, the problem had to be modelled as a matrix:

```
inp = [['B', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', '1'],
       ['W', '1', 'W', 'W', 'W', '3', 'W', 'B', 'B', 'W'],
       ['W', 'B', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W'],
       ['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W'],
       ['W', '3', 'W', 'W', 'B', 'B', 'W', 'W', 'W', 'W'],
       ['W', 'W', 'W', 'W', 'B', 'B', 'W', 'W', '4', 'W'],
       ['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
       ['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'B', 'W']
       ['W', 'B', '2', 'W', '3', 'W', 'W', 'W', 'B', 'W']
       ['B', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'B']]
```

‘W’ denotes white tiles, ‘B’ denotes black tiles without a number, and numbered tiles denote black tiles with a number.
Then each white tile(denoted W) became the variables. Each black tile was recorded alongside including each numbered tile to provide necessary constraints which are as follows:

1) Each sub row and sub column can have at most 1 lamp.
2) Each white tile must be illuminated.
3) Each numbered tile must have ‘number’ lamps next to it.

(sub row/column can be defined as white tiles in a row/column which are not separated by black tiles.)
The domain i used for the variables was {1, 0}, 1 meaning the tile has a lamp and 0 meaning the tile is empty. I, then, introduced the constraints above respectively as:

1) Sum of tiles in a sub row/column <= 1
2) Sum of the tiles in the sub row and column of each tile >= 1
3) Sum of tiles above, below, right and left of a numbered tile == Number in the tile

I reached 1 solution after defining this as a CSP and using the python-constraint package provided in the pdf to solve it. The output of the program is:
‘L’ denotes light bulbs since ‘O’ could easily be mixed with ‘0’.
Solution 1:

```
['B', 'W', 'L', 'W', 'W', 'W', 'B']
['W', 'L', '4', 'L', 'W', 'W', 'W']
['0', 'W', 'L', 'W', '1', 'B', 'L']
['W', 'W', 'W', '1', 'L', 'W', 'W']
['L', 'B', '1', 'W', 'W', 'L', 'B']
['W', 'W', 'L', 'W', 'B', 'W', 'L']
['1', 'L', 'W', 'W', 'W', 'W', 'W']
```

The input and solution above is also Sample 1.

Sample 2:

```
inp = [['W', 'W', 'W', 'W', 'W', 'W', 'W'],
       ['W', '4', 'W', 'W', 'W', '2', 'W'],
       ['W', 'W', 'W', 'B', 'W', 'W', 'W'],
       ['W', 'W', 'B', 'W', '1', 'W', 'W'],
       ['W', 'W', 'W', '3', 'W', 'W', 'W'],
       ['W', 'B', 'W', 'W', 'W', 'B', 'W'],
       ['W', 'W', 'W', 'W', 'W', 'W', 'W']]
```

Output:

Solution 1:

```
['W', 'L', 'W', 'W', 'W', 'W', 'W']
['L', '4', 'L', 'W', 'W', '2', 'L']
['W', 'L', 'W', 'B', 'W', 'L', 'W']
['W', 'W', 'B', 'L', '1', 'W', 'W']
['W', 'W', 'L', '3', 'W', 'W', 'W']
['W', 'B', 'W', 'L', 'W', 'B', 'W']
['W', 'W', 'W', 'W', 'L', 'W', 'W']
```

(puzzle taken from https://www.puzzle-light-up.com/)

Sample 3:

```
inp = [['W', '2', 'W', 'W', 'W', 'W', 'W'],
       ['W', 'W', 'B', 'W', 'W', 'W', '3'],
       ['W', 'W', 'W', 'W', 'W', 'B', 'W'],
       ['W', 'W', 'W', '4', 'W', 'W', 'W'],
       ['W', 'B', 'W', 'W', 'W', 'W', 'W'],
       ['B', 'W', 'W', 'W', '2', 'W', 'W'],
       ['W', 'W', 'W', 'W', 'W', 'B', 'W']]
```

Output:

Solution 1:

```
['L', '2', 'W', 'W', 'W', 'W', 'L']
['W', 'L', 'B', 'W', 'W', 'L', '3']
['W', 'W', 'W', 'L', 'W', 'B', 'L']
['W', 'W', 'L', '4', 'L', 'W', 'W']
['W', 'B', 'W', 'L', 'W', 'W', 'W']
['B', 'L', 'W', 'W', '2', 'L', 'W']
['W', 'W', 'W', 'W', 'L', 'B', 'W']
```

(puzzle taken from https://www.puzzle-light-up.com/)

This problem has strict constraints and requires to be modelled as a Constraint Satisfaction Problem in order to be solved. If we tried to use a heuristic to model this problem, the only reasonable heuristic would have been if this problem satisfies the constraints, which is called Forward Checking and it is a feature of CSPs. Also, we don't have optimality issues in this problem because all possible solutions are going to be returned by the algorithm. Another fundamental feature of CSPs is Backtracking Search, which is necessary to solve this problem in a relatively short time. Whenever the algorithm violates a constraint it backtracks and tries other possible moves from there on, since it uses trial and error to solve this problem.

Also, Akari's can be viewed as an enhanced N-Queens Problem. The first rule mentioned above is almost identical to the first rule of N-Queens. The numbered black tiles can be viewed imaginary tiles on the chess board which require some number of queens next to them and so on. In the lectures, you have demonstrated that solving N-Queens problem with a CSP is much more effective in terms of time and space complexity.