**A description of my CSP model of the Slant puzzle**

We know that a CSP consists of

- a set of variables
- for each variable, a domain of possible values
- a set of constraints

In my solution, each cell is a variable i.e., if we have an NxN matrix for the corners, then we have (N-1)x(N-1) variables. Each variable can be either “/” or “\”. This means that for each variable the domain is {“/”, “\”}. However, in my code implementation, I used integer 1 for “/” and 2 for “\”. So, the domain became {1, 2}. As for the constraints, we have 3 constraints:

1. There must be a diagonal in each cell.
2. The number of lines meeting at a corner must be the same number written in that corner.
3. The diagonals should not form a loop.

In my code implementation, the first constraint is restricted while creating the variables. Since I create (N-1)x(N-1) cells, each one having the domain {1, 2}, none of them can be empty. The other two constraints are satisfied via different functions, namely num_intersects and has_loop.

**Easy Instances**

| Instance No | The number of variables | The number of constraints | The CPU time (in seconds) |
|---|---|---|---|
| 1 | 25 | 3 | 0.11 |
| 2 | 25 | 3 | 0.083 |
| 3 | 25 | 3 | 0.342 |
| 4 | 25 | 3 | 0.814 |
| 5 | 25 | 3 | 1.431 |

**Normal Instances**

| Instance no | The number of variables | The number of constraints | The CPU time (in seconds) |
|---|---|---|---|
| 1 | 25 | 3 | 2.036 |
| 2 | 25 | 3 | 1.574 |
| 3 | 25 | 3 | 2.014 |
| 4 | 36 | 3 | 2.521 |

| | | | |
|---|---|---|---|
| 5 | 36 | 3 | 4.088 |

**Difficult Instances**

| Instance no | The number of variables | The number of constraints | The CPU time (in seconds) |
|---|---|---|---|
| 1 | 36 | 3 | 4.546 |
| 2 | 36 | 3 | 7.143 |
| 3 | 49 | 3 | 7.373 |
| 4 | 49 | 3 | 15.198 |
| 5 | 49 | 3 | 69.632 |

**The observation about the scalability**

From the tables, it can be observed that the CPU time increases as the number of variables increases. Specifically, as we increase the number of variables linearly the CPU time increases exponentially which indicates that the CSP-based method's scalability is not very good. While the easy instances were solved relatively quickly, the normal and difficult instances took longer to solve. This observation suggests that as the size of the puzzle increases, the CSP-based method's computational time increases as well, making it less efficient for solving larger instances of the Slant puzzle.

**A discussion on whether A* or CSP is more appropriate for solving this puzzle**

Both A* and CSP are useful algorithms for solving logic puzzles. However, we need to pick the one that matches best with the specifics of the problem. So far, we have used A* search to solve problems like shortest path, maze-solving, etc. We modeled the problem as a graph or a tree. However, the slant puzzle is a binary-determination problem. Meaning that, instead of following a path, we try to pick the optimal value for each variable which makes it more sensible to model this problem with a set of variables, a domain for each variable, and a set of constraints. On the other hand, the slant puzzle does not have a shorter path or a more optimal solution but a correct solution. This also implies that there is no point in using A* search to improve the solution.