Sudoku-Pair Solver and Generator

(Using SAT solver)

Team Members:

- Gavish Garg (200385)
- Bhavya Garg (200270)

1. Introduction

This is the brief introduction of the SUDOKU pair solver and generator by encoding the problems to the propositional logic, using SAT solver(https://pypi.python.org/pypi/pycosat).

Here, we have used a SAT solver to solve the given pair of sudoku. The size of the sudoku is k^2 x k^2. In the first problem, we are the given the input of two partially filled sudokus and we need to solve that pair of sudokus such that :

1. Each of the sudoku is correctly filled according to the rules.
2. Each corresponding cell of the solved pair of sudokus doesn't have the same value.

							2	
5			6			3		7
			3			9		
9				5				
		3		6		2	4	5
2		1			3	6		
		9						
						8		2
							1	6

	8	7					1	
6								
						6	4	5
				7				
		5			9	8		3
1				4	6			
	1						3	6
4								
				3	5		8	

2. Code

2.1. Encoding in SAT

Variables

x_{ijd} : For d ranging from 1 to k^2, x_{ijd} represents the value that will be assigned to the cell [i,j] = d.

Constraints

1. Each cell will have at least one value : $\forall_{i,j}$ (x_{ij1} V x_{ij2} V x_{ij3} …… $x_{ijk\text{^}2}$)
2. Each cell will have utmost one color : $\forall_{i,j}$ $U_{1\le d1<d2\le k*k}(\sim x_{ijd1}$ V $\sim x_{ijd2})$
3. A row cannot have a repeating number : $\forall_{i,d}$ $U_{1\le j1<j2\le k*k}(\sim x_{ij1d}$ V $\sim x_{ij2d})$
4. A column cannot have a repeating number : $\forall_{j,d}$ $U_{1\le i1<i2\le k*k}(\sim x_{i1jd}$ V $\sim x_{i2jd})$
5. A block (k x k) cannot have a repeating number :
6. Corresponding cells of two sudoku can't have a same value : $\forall_{i,j,d}$ ($\sim x1_{ijd}$ V $\sim x2_{ijd}$)

2.2. Working

We have used pycoSAT to solve this hard problem.

Problem 1.

Initially we create an empty list S [ ] and then we start to add clauses in that list using above constraints. After all the necessary constraints have been added for each variable we take the input sudoku pair.

In the input sudoku pair, some values are non zero which means that we have to assign only that value to the corresponding cell.
Therefore we convert this also into a literal and add it to the main list of clauses. It takes care that those corresponding cells are assigned only that values which are given in the input.

Now using pycosat.solve(S) we get the desired model if possible otherwise it returns UNSAT.
UNSAT signifies that there is no possible interpretation for which our formula is True.
The model obtained is used to update the given sudokus.

4	3	6	8	9	7	5	2	1
5	9	2	6	1	4	3	8	7
7	1	8	3	2	5	9	6	4
9	6	4	7	5	2	1	3	8
8	7	3	9	6	1	2	4	5
2	5	1	4	8	3	6	7	9
6	2	9	1	7	8	4	5	3
1	4	7	5	3	6	8	9	2
3	8	5	2	4	9	7	1	6

.

9	8	7	6	5	4	3	1	2
6	5	4	3	2	1	9	7	8
3	2	1	9	8	7	6	4	5
8	9	6	5	7	3	4	2	1
7	4	5	2	1	9	8	6	3
1	3	2	8	4	6	5	9	7
5	1	8	4	9	2	7	3	6
4	7	3	1	6	8	2	5	9
2	6	9	7	3	5	1	8	4

Problem 2.

For the problem 2 we have to create a puzzle consisting of a pair of sudoku such that :

1. Given puzzle has a unique solution in the way described in the Problem1.
2. Given puzzle should have a maximum number of holes.

For this we create a random pair of sudoku using a random generator code, which generates a fully solved pair of sudoku as described in Problem1.

Now we create a list of the cells of the given pair of sudokus (2 x k^4 cells in total). Using a shuffler in python we shuffle that list randomly.

After that we start popping out the first element from that list and check whether the puzzle generated after removing the cell (that has been popped out), gives a unique solution or not. If it maintains the uniqueness of the solution, then we empty that cell from the initial puzzle we generated, else we keep that cell as it is.
Now again we shuffle the list (after popping) and again pop the first element from the list and follow the above same procedure.

- Finally after checking for every cell in the given list, we have got our final pair of sudokus with the maximum number of holes.
- This pair of sudoku also has a unique solution as described in problem1.

Assumptions

Problem 1.

1. The value of k entered must be a positive integer greater than 1.
2. The given input csv files satisfies the 6^{th} constrained i.e corresponding cells don't have the same value.

Problem 2.

1. The value of k entered must be a positive integer greater than 1.

Limitations

1. Time Complexity :
   It takes a large time for the SAT solver to solve the given formula for values of k larger than 4. And If k is a very large number then the time taken is extremely large.

2. Space complexity :
   As we are assuming different variables corresponding to each cell (k^6 variables), the size of the list can be very large as increasing the size of k will increase both the variables and total number of clauses.

3. Sudoku pair solver (Problem1.) provides only one solution even if there are other solutions possible.

4. In Problem2. , we check each cell in a random manner which implies we are not considering every possible permutation.
   Thus there is a possibility of a puzzle that could have been generated with more holes.