Assignment 1: Sudoku pair: solving and generation

Jaya Gupta (200471)
Harshit Bansal (200428)

February 2, 2022

1 Sudoku Pair Solver

Given two half filled sudokus $S1, S2$ (both of dimension k^2) , the program solves the two sudokus under the following constraints

- Individual sudoku properties should hold
  - Only the number $[1, k^2]$ can be used to fill each cell.
  - Each $k * k$ box , row and column can only contain each number from 1 to k^2 once.
- For each empty cell $S1[i, j] \neq S2[i, j]$, where i is row and j is column.

Implementation

- Used propositional logic to solve the given problem.
- Generated clauses in the CNF format and feed them to the SAT solver.

Encoding in SAT

- Variables:
  - p_{ij}^d : (i, j) cell in $S1$ which has a value d where $d \in [1, k^2]$
  - q_{ij}^d : (i, j) cell in $S2$ which has a value d where $d \in [1, k^2]$
- Constraints :
  - Each cell in both sudokus $(S1, S2)$ have atleast one number($[1, k^2]$).

$$\forall_{(i,j)\in[1,k^2]}(p_{ij}^1 \vee p_{ij}^2 \vee p_{ij}^3 .....p_{ij}^{k^2})$$

$$\forall_{(i,j)\in[1,k^2]}(q_{ij}^1 \vee q_{ij}^2 \vee q_{ij}^3 .....q_{ij}^{k^2})$$

  - Each cell in both sudokus should have atmost one number($[1, k^2]$).

$$\forall_{(i,j)\in[1,k^2]}\forall_{1\leq d<d'\leq k^2}(\neg p_{ij}^d \vee \neg p_{ij}^{d'})$$

$$\forall_{(i,j)\in[1,k^2]}\forall_{1\leq d<d'\leq k^2}(\neg q_{ij}^d \vee \neg q_{ij}^{d'})$$

  - Both sudokus should individually satisfy the properties of a sudoku.

$$valid(x_1, x_2, ..., x_{k^2}) = \forall_{v,w}\forall_{d\in[1,k^2]}(\neg x_v^d \vee \neg x_w^d)$$

    $where(x_1, x_2, ..., x_{k^2})$ are cells of a given row,column or block.
  - For each empty cell $S1[i, j] \neq S2[i, j]$, where i is row and j is column.

$$\forall_{(i,j)\in[1,k^2]}\forall_{d\in[1,k^2]}(\neg p_{ij}^d \vee \neg q_{i,j}^d)$$

Implementation in Python

- Representation of CNF clauses in the DIMACS format

– The above mentioned variables were represented as numbers using the given function

$$v(i, j, d, m) = 2 * (pow(k^2, 2) * (i - 1) + k^2 * (j - 1) + d) + m$$

where i = row number
j = column number
d = number($[1, k^2]$) filled in cell (i, j)
m = sudoku number (either 1 or 2 for $S1$ or $S2$)

This will ensure that each cell in both sudokus are represented by a unique number.Negation of any literal was expressed using a negative sign before the respective number.

– The clauses mentioned above (in Constraints Section) were given as list of lists to the SAT solver.

– Single literal clauses representing the already filled cells of both the sudokus were appended to the above list . This made sure that the already filled cells are assigned the same number filled in them.

- Obtaining the model (in case it exists) and returning the filled sudokus (and None if do not exist)
  – If a model exists (means the formula is solvable for some interpretation) then the cells were filled with their corresponding values.

$$\forall_{i,j,d} \text{ if } v(i, j, d, 1) \text{ exists in model then } S1[i - 1][j - 1] = d$$

$$\forall_{i,j,d} \text{ if } v(i, j, d, 2) \text{ exists in model then } S2[i - 1][j - 1] = d$$

2 Sudoku Pair Generator

- We have to write a k^2-sudoku puzzle pair generator. The puzzle pair must be maximal (have the largest number of holes possible) and must have a unique solution.
- All the constraint mentioned above in Sudoku Pair Solver are applicable here while generating a valid sudoku pair.

Implementation

1. First of all two completely filled random sudokus were generated which satisfies the above mentioned constraint by using the following trick.
   - The first cell (1,1) and last cell (k^2, k^2) of $S1$ were randomly filled with number in range $[1, k^2]$.
   - Then the first sudoku ($S1$) was fed to the single sudoku solver which completely filled the entire sudoku satisfying all the properties of a sudoku.
   - The second sudoku ($S2$) was filled entirely with 0.
   - Then $S1$ and $S2$ were fed to the Sudoku Pair solver described above which gave us a completely filled sudoku $S2$. Completely filled sudokus $(S1_{complete}, S2_{complete})$ were generated randomly satisfying the given constraints.
2. Then list of indexes of both the sudokus was made.
3. Any index was randomly picked and poped from the list.Then the value of the cell picked was made 0 and the sudoku pair $(S1, S2)$ was fed to the Sudoku Pair Solver.

$$S_{ij}^m = 0$$

m = 1 or 2
i,j = index picked randomly from the list

4. The clauses that were added in Solver were:

- All the clauses mentioned above in Constraints Section of Sudoku Solver.
- Negation of the entries of the completely filled sudokus ($S1_{complete}, S2_{complete}$) generated in Step 1 .This step ensures that when our SAT solver returns a model then there is not a unique solution for our generated sudokus.
- Single literal clauses representing the already filled cells(which are not 0) of both the sudokus were appended to the above clause list . This made sure that the already filled cells are assigned the same number filled in them.

5. If our SAT solver does not return any model (means a unique solution exists), then again we go to Step 3 and continue. Else if the SAT solver returns any model then it means that there is not a unique solution to our generated Sudoku Pair and we revert that cell back to its old value and move to next cell and repeat from Step 3 until entire index is exhausted.

$$S_{ij}^{m} = \begin{cases} 0 & \text{if SAT solver does not return any model} \\ S_{ijcomplete}^{m} & \text{Solution exists} \end{cases}$$

This gave us two randomly generated sudoku pait which satisfies the given constraints.

Limitations of Sudoku Pair Generator

- **Hardware Constraint**
  - Generator for k = 6 and larger k values requires approximate 12 GB RAM which is not feasible to run on local machines .
  - Execution time for k = 6 and onwards is quite large .