Report

Jasjot (200468), Yash(201144)

February 2022

1 Implementation

Please make sure we have shown product symbol to represent conjunction nature and sum symbol for disjunction so as for easy visualisation.

a.The first problem of assignment is to solve sudoku pair with some extra constraints. So for this first of all we have identify our variables namely as,

$$x_{ijk}$$

where i represents i^{th} row of sudoku whose value ranges from $1 \leq i \leq 2*(size^2)$

j represents j^{th} column of sudoku whose value ranges from $1 \leq j \leq size^2$

k represnts number to be filled in sudoku at the cell (i,j) of sudoku whose value ranges from $1 \leq k \leq size^2$

Constraints on these set of variables-:

1. at least one number in each cell

$$\prod_{\forall_{i,j}}(\sum_{\forall_k} x_{ijk})$$

2. at most one number in cell

$$\prod_{\forall_{i,j}}(\prod_{\forall_k}(\neg x_{ijk_1} \vee \neg x_{ijk_2}))$$

3.unique numbers in each column, tested seperately for both sudoku.

$$\prod_{\forall_{j,k}}(\prod_{\forall_i}(\neg x_{i_1jk} \vee \neg x_{i_2jk}))$$

4.unique numbers in each row

$$\prod_{\forall_{i,k}}(\prod_{\forall_j}(\neg x_{ij_1k} \vee \neg x_{ij_2k}))$$

5.unique numbers in each subgrid i.e small box of SIZE=(size x size)

6.No two corresponding cell of both sudoku can be same.

$$\prod_{\forall_{i,j,k}} (\neg x_{ijk} \vee \neg x_{(i+size^2)jk})$$

And for envoking all these constraints we call out for card module of python-sat, which checks all cardinallity constarints effeciently and for displaying this effecient purpose we have also added two files for sudoku-solver.

NOTE-:to check effeciency one need to give large input like k=6.

b. In the second problem of assignment we have used standard algorithm of sudoku generator.

ALGORITHM-:First create a well defined sudoku pair starting from empty cells and then store all the id's of all the variables in list, in our Implementation lid. then shuffle the list lid with python shuffle module, so as to get randomize list. then corresponding to each identity store number of well defined solve sudoku(as they can be accessible via their ids) in seperate list name as lnum. Then iterate through list lid and remove that number from list lnum if unique slution continues to exist even on removing that number from sudoku. and if more solution begin to exist on removing that element then put it back, i.e don't remove it from lnum.

finally we have well defined list lnum with removed element representing as 0. so corresponding to their ids(as store in list lid) store them in seperate list of list, and then finally print them as .csv file if solution exist else print None.

Constraints on these set of variables-:

1. at least one number in each cell

$$\prod_{\forall_{i,j}} (\sum_{\forall_k} x_{ijk})$$

2. at most one number in cell

$$\prod_{\forall_{i,j}} (\prod_{\forall_k} (\neg x_{ijk_1} \vee \neg x_{ijk_2}))$$

3.unique numbers in each column, tested seperately for both sudoku.

$$\prod_{\forall_{j,k}} (\prod_{\forall_i} (\neg x_{i_1jk} \vee \neg x_{i_2jk}))$$

4.unique numbers in each row

$$\prod_{\forall_{i,k}}(\prod_{\forall_j}(\neg x_{ij_1k} \vee \neg x_{ij_2k}))$$

5.unique numbers in each subgrid i.e small box of SIZE=(size x size)

6.No two corresponding cell of both sudoku can be same.

$$\prod_{\forall_{i,j,k}} (\neg x_{ijk} \vee \neg x_{(i+size^2)jk})$$

7. Each time to find more number of sol. only add another clause as negation of original solution number at that particular cell, i.e,

$$(\neg x_{ijk})$$

for that particular number k in cell (i,j).

Here also we have envoked pysat.card module.

2 Assumption

1. file .csv is formed only if solution exist else just output None in console window.

2. As it is not mentioned that for what values we have to output our answer i.e no mentioning of value of k hence, we have shown our results for nominal values as we both our unable to install python-sat in our laptop and had worked on online platform and there is lack of time and space, and for showing our algorithm to be correct it is easy to ensure that it will give correct output for all values of k in nominal range, which we have make sure.

3 Limitation

1. As such no major limitation but, one limitation is that of time limit. As value of k increases more and more (exponentially or may be even faster) increment in time. And also as stated we have attach two files for sudoku solver so as to highlight major limitation in most of people approach as manually appending each clause is very inefficient approach as in that cases time limit exceeds even more faster.

4 Bonus Part

1. For the purpose of showing inefficiencies in various modules of python we have implemented both parts in two ways, but shown it for first part as a separate file, for more information refer Readme.md file.

2. It is to NOTE-: that we have call python random generator 3 times so as to bring a large diversity in number of sudokus formed, even sir had mentioned in class we want only different final answer but we have constructed even different sudoku starting from scratch.