CS202 Assignment 1

Group Members: Yash Raj Mittal (Roll no.: 201148)
Akash Biswas (Roll no.: 200074)

Constraints

Given the parameter k, we have k^4 cells in the sudoku. Each cell **can** (without taking into account the constraints of sudoku) take k^2 values, hence the number of **variables** required for a sudoku with parameter k is **at least k^6**. Also, the sudoku has k^2 rows and columns each. Also, there are k^2 blocks each having k^2 cells.

So, for representing the constraints in a more compact form, we introduce the Boolean variable $\boldsymbol{S_{i,j}^{c}} : 1 \leq c, i, j \leq k^2$. If the value of $\boldsymbol{S_{i,j}^{c}}$ is true for some value of i,j,c we can infer from it that the cell corresponding to the i^{th} row and j^{th} column has value c filled in it.

A completely filled sudoku must abide by the following rules:

1. Each cell must have **at least** one value filled in it. We can represent it concisely as $\bigcup_{\boldsymbol{c=1}}^{\boldsymbol{k^2}} \boldsymbol{S_{i,j}^{c}}$ **must be true** for all values of i and j.
2. Each cell must have **at most** one value, by coupling this condition with the first one, we can assure that each cell has **unique** value filled in it. We represent it as $\bigcap_{\boldsymbol{1 \leq c < c' \leq k^2}} (\neg \boldsymbol{S_{i,j}^{c}} \cup \neg \boldsymbol{S_{i,j}^{c'}})$ **must be true** for all values of i and j.
3. In a given **row**, each value occurs **only once.** Hence, $\bigcap_{\boldsymbol{1 \leq j < j' \leq k^2}} (\neg \boldsymbol{S_{i,j}^{c}} \cup \neg \boldsymbol{S_{i,j'}^{c}})$ **must be true** for all values of i and c.
4. Similarly, in each column, each value occurs **only once.** Hence, $\bigcap_{\boldsymbol{1 \leq i < i' \leq k^2}} (\neg \boldsymbol{S_{i,j}^{c}} \cup \neg \boldsymbol{S_{i',j}^{c}})$ **must be true** for all values of j and c.
5. Also, in the smaller block of size k^2, each value **occurs** only once. Hence, $\bigcap_{1+nk \leq i < i' \leq k+nk;\, 1+mk \leq j < j' \leq k+mk} (\neg \boldsymbol{S_{i,j}^{c}} \cup \neg \boldsymbol{S_{i',j'}^{c}})$ **must be true** for all values of c and all $1 \leq n, m \leq k-1$.

For a valid Sudoku all the above five conditions must hold true. As, we need to implement not just a Sudoku but, a Sudoku pair instead, so we denote it as pair $< S_{1_{i,j}}^{c}, S_{2_{i',j'}}^{c'} >$. Both sudoku $S_{1_{i,j}}^{c}$ and $S_{2_{i',j'}}^{c'}$ satisfy the original constraints mentioned above. Apart from that, it must satisfy an additional constraint given in the problem statement $S_{1_{i,j}}^{c} \neq S_{2_{i,j}}^{c}$ **for the filled sudoku pair** for all value of c,i,j. So, we represent it as $\cap_{1 \leq i < i' \leq k^2;\, 1 \leq j < j' \leq k^2}(\neg S_{1_{i,j}}^{c} \cup \neg S_{2_{i',j'}}^{c})$ **must be true** for all values of c.

Implementation

- ## Sudoku Pair Solver

After reviewing the constraints, we must now implement a **Python code** using **pysat** module. As our constraints are mostly in "**and of or"** form and also, pysat supports input in **CNF** form. So, we can represent clauses in form of a list. Also, we can represent variables using numbers in pysat so as discussed earlier, we need at least k^6 variables. So, we encode it as follows

If $S_{1_{i,j}}^{c}$ **is true**, then it corresponds to the number $k^4 \times (i-1) + k^2 \times (j-1) + c$.

Similarly, if $S_{2_{i,j}}^{c}$ **is true**, then it corresponds to the number $k^4 \times (i-1) + k^2 \times (j-1) + c + k^6$.

For proving the uniqueness of the above encoding, we assume that $S_{1_{i,j}}^{c} = S_{1_{i',j'}}^{c'}$

$$\text{So, } k^4 \times (i-1) + k^2 \times (j-1) + c = k^4 \times (i'-1) + k \times (j'-1) + c'$$

$$k^4 \times (i - i') + k^2 \times (j - j') + c - c' = 0$$

As, $0 \leq |i - i'|, |j - j'|, |c - c'| \leq k^2 - 1$

Now consider the minimum value $k^2 \times (j - j') + c - c'$ can take. So, we substitute $j - j' = c - c' = 1 - k^2$. We obtain it as $-k^4 + 1$, whereas $|k^4 \times (i - i')|$ takes values $0, k^4, 2k^4, \ldots, k^6 - k^4$. So, the **equation will never be satisfied**. Hence, we take the case when $i = i'$. We apply similar logic as $c - c'$ has minimum value $1 - k^2$ whereas, $|k^2 \times (j - j')|$ takes values $0, k^2, 2k^2, \ldots, k^4 - k^2$. So, again equation is not **satisfiable.**

Hence taking $i = i'$ and $j = j'$ automatically implies $c = c'$**.** Hence proving that the encoding is unique.

Now, for implementing the constraints, we simply substitute the encodings in place of $S1_{i,j}^{c}$ and $S2_{i,j}^{c}$**.**

We have passed the clauses in CNF form to the solver using list of lists data structure in python. But as we have not yet passed the additional information regarding the filled cells of the sudoku pair to the solver. We pass the info while solving the model in form of list to the **assumptions parameter** of solver.solve(). For filling the assumptions list, we read through the **CSV** file and for every **non-zero** entry, add the encoded number representation of the variable (say, $S1_{i,j}^{c}$ is True so, we add $k^4 \times (i-1) + k^2 \times (j-1) + c$) to the assumptions list. Having, passed the assumptions list to the solver, we get the corresponding solution and **not the solution generated for an empty sudoku pair.**

- ## Sudoku Pair Generator

We used the approach similar to above part. We first fill one row of the sudoku randomly using the **random** module and then pass it to the solver **assumptions** parameter to obtain a completely filled sudoku pair which will be a random one (out of $K^2!$ Possibilities). Now, to create a maximal sudoku, we iterate over each cell of the sudoku pair and check for uniqueness of sudoku pair model after removal of that cell, if not unique, we add the previous cell back. Then we move on to next cell and repeat the same process till we reached the end of sudoku. Thus, we generated a sudoku pair with maximal number of holes.

For checking the uniqueness, we first get the model with appropriate **assumptions** of the sudoku pair and store the variables other than that in assumptions in another list. We, then take the negative of all the variables in the list and add the obtained list as a clause to the solver and again solve with the same **assumptions** parameter. If the condition is **satisfiable**, we can infer that there exists more than one model for the given constraints else the model obtained earlier is the **unique** one.

Mathematical analysis

For a sudoku of size K, we have K^4 cells for a single sudoku. So, we have total $2K^4$ cells for a pair. The number of unique literals is K^2 for each cell we are using K^4 variables to store each value.

Calculation of Clauses

- At least one of the numbers is filled

  For each cell we have one clause. So, the number of clauses is K^4.
- At most one of the numbers is filled

  For each cell we are comparing for two d at a time. These can be done in $\binom{K^2}{2}$ ways. So, the total number is $K^4 \times \binom{K^2}{2}$.
- For no repetition in Rows, column and Block

  For each cell, we are comparing for two d at a time. These can be done in $\binom{K^2}{2}$ ways. So total number is 3 $K^4 \times \binom{K^2}{2}$
- For different values at same cell in two sudoku

  For each cell, we are comparing one d at a time. These can be done in K^2 ways. So total number is K^6.

  Hence, the sum over all clauses will be 2 K^8- K^6+ K^4.

Assumptions

1. Assumed K^2 literals for each cell to accommodate number for each cell.
2. We have assumed K > 1. As, for K=1, Sudoku Pair can't be satisfiable.

Limitations

1. For bigger K (>=6), the solver uses so many clauses and takes a little more time and due to lower system computation can't run efficiently.
2. Taking much memory for (K>=4) for sudoku generator and thus high system requirements.