Project 1 Write Up

8/30

Nuo Chen

Assuming the set of clues given is a well-defined condition, meaning that the solution is feasible and unique. Observing that the Sudoku problem fits the framework of a LP problem perfectly, but with integer constraints.

Objective Function: If all constraints are satisfied, the Sudoku quiz is solved. Thus, no objective function is needed for our case.

Decision Variables: Let X_{ijk} be the decision variable indicating the event of the (i,j) element of the Sudoku grid containing k, i.e. if k is the right choice, $X_{ijk} = 1$, otherwise $X_{ijk} = 0$. Since $1 \leq i,j,k \leq 9$, the model generates $9^3 = 729$ variables instead of 81 variables (one variable per grid representing the value in that grid). The advantage of doing so is that it's more efficient to pass 729 Boolean variables than comparing variables (worst case scenario) 729 times.

Constraints: There are $4 \times 81 = 324$ default constraints generated based on the Sudoku rules plus the number-of-clues constraints given by the quiz. The default constraints are generated as followed:

- Each cell is filled with only one value from 1 to 9.

$$\sum_{k=1}^{9} X_{ijk} = 1, \qquad 1 \leq i,j \leq 9$$

- Each row contains every number from 1 to 9 once.

$$\sum_{j=1}^{9} X_{ijk} = 1, \qquad 1 \leq i,k \leq 9$$

- Each column contains every number from 1 to 9 once.

$$\sum_{i=1}^{9} X_{ijk} = 1, \qquad 1 \leq j,k \leq 9$$

- Each 3×3 box contains every number from 1 to 9 once.

$$\sum_{j=3p-2}^{3p} \sum_{i=3q-2}^{3q} X_{ijk} = 1, \qquad 1 \leq k \leq 9, 1 \leq p,q \leq 3$$

The clue constraints are generated by filling the prefilled cells.

Solution: I used the PuLP package in Python to solve the above LP problem. With the built-in classes and functions of PuLP, I simply defined the LP problem (naming it), then added the $4 \times 81 + \#clues$ constraints to the pulp.LpConstraint class, and finally used the .solve() function to solve for the 729 variables.

Instead of constructing matrices A and B explicitly then solving it using the cvxpy or numpy package (as described in the Project Instruction), using pulp with its built-in features solves the LP problem more efficiently with high accuracy (the average time spent for each quiz in the Example is about 2s whereas the average time spent in my code using pulp is 0.12s).

The following table displays the accuracy of the code (#corrects/#attemps). The result matches the expectation as the constraints of the LP problem are set up correctly.

DATA SET	SUCCESS RATE
SET A	100%
SET B	100%