Haley Feng
May 15, 2019

Project 1: Sudoku Challenges

Method

I used linear programming (LP) method with implementation of recursion and weight to solve all four sets of data. After running the necessary function *fixed_constraints* and *clue_constraint,* I made some changes to the starter code of LP by adjusting the solver as a function *solver(input_).* Inside this function, there's an implementation of weight using the code provided from the paper "A Warm Restart Strategy for Solving Sudoku by Sparse Optimization Methods". The weight would be combine with the original constant *c* and solved using LP from scipy.optimize. After computing the solution *x_new*, it gets check to see if there's are repeated variable inside the solution. To do so, I use the function *repeats(matrix, original)* that will find the repeated values and replace those values with zero. The function will give me a new matrix which I can then plug it back in to the function of *solver* and get a new solution until the norm of the difference between new solution and old solution, *x_new* and *x_ori* respectively, is less than epsilon=10e^-10.

Data Sets	**Total**	**Success**	**Baseline**
small_1	24	100%	24/24
small_2	1011	63%	324/1000
large_1	1000 (random sample)	93%	818/1000
large_2	1000 (random sample)	100%	1000/1000

Expectation/Improvement

While the implementation is not the best, it is still pretty good since there's an increase in success rate for data sets small_2 by 31% and large_1 by 12%. I expected a higher percentage by implementing the procedure of checking for repeated values since the data from the paper has a 99% performance for their result. I don't know if there are any improvement I would make to the code, but if more time are given I will check to see if the result without the weight versus with weight pair along to checking for repeated values will be any different to the result I get now.