Project1 Sudoku Challenge: A Evaluation Survey

Date: 5/16/2019

Group member: Ziteng Pang, Yuan Fu, Junlin Wang, Yitan Ze

Methods:

L1 weighted linear programming[1]

Warm restart strategy[1]

Naked single

Our implementation is largely based on *A Warm Restart Strategy for Solving Sudoku by Sparse Optimization Methods* by Yuchao Tang with some minor improvements.

First of all, we run a forward checking based on the naked single algorithm to get the possible outcomes for the unknown spots by checking duplicates. We first replace unknown spots with the possible value list of 1-9, and eliminate the possible spot value by checking the duplicates of rows, columns and boxes. If the list of possible values is eliminated to one number, we add this number to this spot as a new clue. By increasing the number of clues at the start, we're able to have more constraints when implementing linear programming method later.

In our setting, the most sparse solution is guaranteed to be the correct solution. Ideally, we would want to do L0 minimization to ensure our sparsity. However, since it is NP-hard, we do the L1 minimization instead. To encourage the sparsity, we add different weights W to X suggested by[1]. We calculate the corresponding weight coefficients

based on the relative reciprocal of the previous X in a sense that we push small X's to 0.

For hard problems, weight L1 minimization is no enough to achieve the correct solution, so we implemented the Warm restart strategy suggested by[1]. After the first attempt of solving the problem, we check if our solution meets the sudoku requirements. If not, we delete all the duplicated entries and add back potentially deleted clues to solve the LP problem one more time. After the second attempt, we solve the problem in the same setting another time if it was not correct. After the third attempt, we approach the problem slightly different since the above approach will not likely to produce further improvements. We first delete all the duplicates entries in the solution, and then choose one entry from the solution along with all the clues to form a new problem. We believe entries near the corners are more likely to be correct, so we run the same attempting 4 times for 4 corners if the previous attempting is not correct already.

Result:

The best result for Set A (small 1) is 24/24 = 100%. The best result for Set B (small 2) is 815 / 1011 = 80.6 %.

Since the number of clues given in each quiz in Set B is much smaller than that given in Set A, the forward checking algorithm mostly does not make progress by giving new clues for each quiz in Set B.

Set	Success Rate
Set A (small 1)	24/24

Set A (small 2)	805/1000
Set B (large 1)	963/1000
Set B (large 2)	1000/1000

Future work:

During our investigation of solving the sudoku challenge, there are several methods that we think we should have tried that may provide some insights if given more time:

1. **Quadratic Programming**

2. **CNN with a modified objective function**

One obvious approach is to use quadratic programming in place of linear programming. This requires us to change the objective function to x(1-x)and the matrix notation becomes 12xTPx + qTx where P is a diagonal matrix with -2 on the diagonal and q is a diagonal matrix with 1 on the diagonal. However, during our limited testing time, we keep encountering issues involving incorrect ranks.

Another approach is to use CNN. We explored this option a bit but quickly realized training a somewhat decent model requires more computational resource than we could offer. But our idea is to change the objective function from percentage of correct cells filled to a binary function which the loss is 1 if the output is incorrect and 0 if the output is correct.

Reference:

- [1]https://arxiv.org/pdf/1507.05995.pdf

- [2]https://colab.research.google.com/drive/1RNR3bkramHc2iVacBwHDR13IW78k7fWs#scrollTo=ZyqfPChDdRNu