Math 110B Project1: Sudoku Challenges Report

Meng Xu, Zijun Tan,

Yuxuan Wang, Minglu Chen

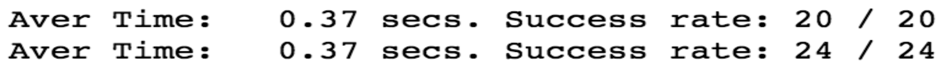
Introduction

In this project, we will apply optimization method to solve the 9x9 Sudoku problem with different level of difficulties. There are plenty of algorithms to approach the Sudoku problem, such as backtracking algorithm, integer programming, simulated annealing and constraint programming. In this project, we will be based on the starter code implemented with constraint linear programming using L1 sparse optimization method and modify it with a method introduced in one of the referenced paper. Our method is weighted L1 norm linear programming applied with twice deleting procedure of the repeated values and repeat adding one randomized new clue into the original Sudoku problem. The performance improved from 32.6% to 65.2% on the hardest Sudoku dataset and remain nearly 100% correctness on lower level difficulty dataset.

Results

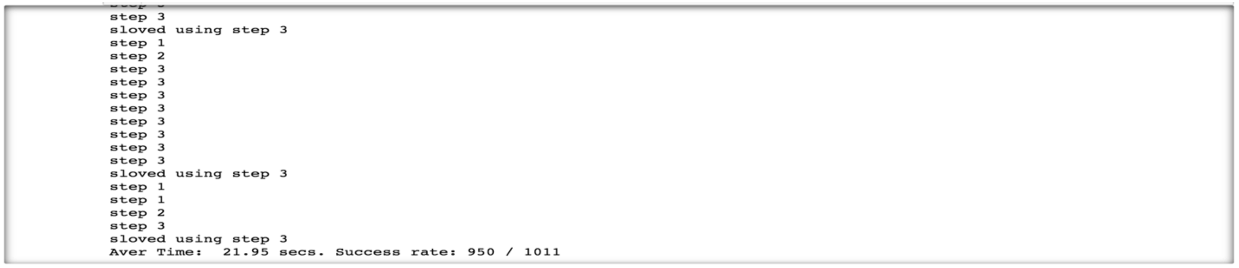
Methods & Accuracy		**Small 1 (preliminary) (Easy)**	**Small 2 (A) (Hard)**	**Large 1 (B) (Medium)**	**Large 2 (B) (Easy)**
Original sparse optimization model with L1 norm		100%	32.4%	81.8%	100%
Sparse optimization model with weighted L1 norm and warm restart modification (Weight = 0.01, 1 & 30**)**	ε				
	0.01	100%	43.3%	86.5%	100%
	1	100%	60.3%	92.7%	100%
	30	100%	65.2%	94.5%	100%
Sparse optimization model with weighted L1 norm and warm restart with repeated step3 (Weight = 1)		100%	93.96% (950/1011)	98.6%	100%

Small 1:



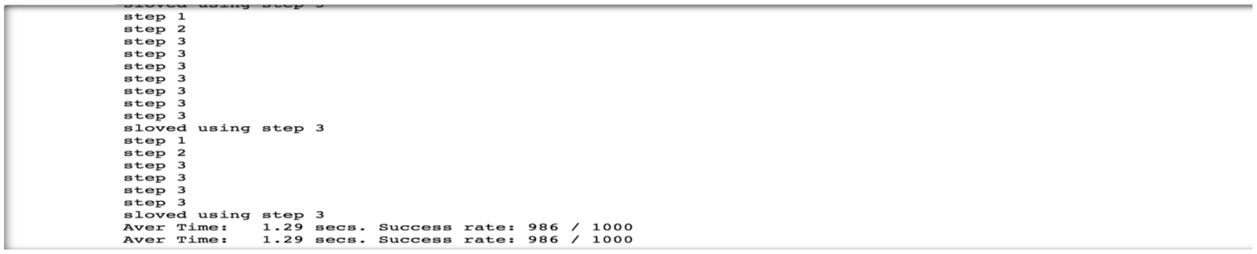
```
Aver Time:   0.37 secs. Success rate: 20 / 20
Aver Time:   0.37 secs. Success rate: 24 / 24
```

Small 2:



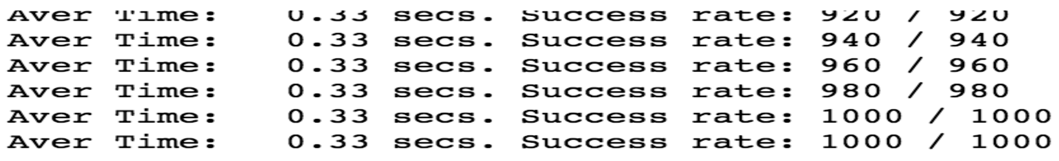
```
step 3
sloved using step 3
step 1
step 2
step 3
step 3
step 3
step 3
step 3
step 3
step 3
step 3
sloved using step 3
step 1
step 1
step 2
step 3
sloved using step 3
Aver Time:  21.95 secs. Success rate: 950 / 1011
```

Large 1:



```
step 1
step 2
step 3
step 3
step 3
step 3
step 3
step 3
step 3
sloved using step 3
step 1
step 2
step 3
step 3
step 3
step 3
sloved using step 3
Aver Time:   1.29 secs. Success rate: 986 / 1000
Aver Time:   1.29 secs. Success rate: 986 / 1000
```

Large 2:



```
Aver Time:   0.33 secs. Success rate: 920 / 920
Aver Time:   0.33 secs. Success rate: 940 / 940
Aver Time:   0.33 secs. Success rate: 960 / 960
Aver Time:   0.33 secs. Success rate: 980 / 980
Aver Time:   0.33 secs. Success rate: 1000 / 1000
Aver Time:   0.33 secs. Success rate: 1000 / 1000
```

Original linear programming model

- **Constraint Linear Programming with L1 norm**

The Sudoku problem can be formulated as a linear system solved by the corresponding sparse optimization model. Babu et al. [1] provided a method to transform the Sudoku problem into the linear system. We first change each number from 1 to 9 into a 9-dimensional binary vector such as 5 is represented as (0,0,0,0,1,0,0,0,0). Following that transformation, we will transform in total of 9x9 = 81 numbers into an 81x9 = 729 vector denoted as $\boldsymbol{X_{729x1}}$ which also denoted as the solution of the corresponding Sudoku problem. Therefore, based on the rules of the Sudoku puzzle game, we can formulate five constraints as following:

Let X_{ijk} as the event that the (i, j) cell of the Sudoku problem contains k, X_{ijk}= 0 if true and X_{ijk}= 0 if false. Thus,

- Column, $\sum_{i=1}^{9} Xijk = 1,$ $1 \leq j, k \leq 9$
- Row, $\sum_{i=1}^{9} Xijk = 1,$ $1 \leq i, k \leq 9$
- Grid, $\sum_{i=1}^{9} Xijk = 1,$ $1 \leq i, j \leq 9$
- Box, $\sum_{j=3p-2}^{3p} \sum_{i=3q-2}^{3q} Xijk = 1,$ $1 \leq k \leq 9$ and $1 \leq p, q \leq 3$
- Clues will be given in the puzzle with at least 17 clues, clues less than 17 will result no solution

Combining the constraints, then the linear equality constraints on x can be represented as:

$$A\mathbf{x} = \begin{pmatrix} A_{row} \\ A_{col} \\ A_{box} \\ A_{cell} \\ A_{clue} \end{pmatrix} \mathbf{X} = \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ \ldots \\ 1 \\ 1 \end{pmatrix}$$

A_{row}, A_{col}, A_{box}, A_{cell}, A_{clue} are matrices with the corresponding constraint of this Sudoku problem. Each matrix A is a 341 x 729 which is an under-determined that has infinity many solutions. By the Babu et.al [1] result which proved that "if the Sudoku puzzle has a unique solution, then the sparsest solution of the above equation is the solution for the Sudoku puzzle". Also, in order to find the sparest solution, we build up the L0 norm minimization model. Since it is nonconvex and NP-hard (non-deterministic polynomial-time hardness), we can replace this model with the L1 norm minimization model which is convex that can guarantee a unique solution. These two norm minimization models will get the approximately equivalence solution to the Sudoku problem which proved by the result of [3].

The L1 norm minimization model with $||x||$ as our objective function:

$$\min ||x||_1$$

$$\text{s.t } A\mathbf{x} = \mathbf{b}$$

$$\text{where } ||x||_1 = \sum_{i=1}^{n} |xi|$$

Therefore, we can have two linear programming models to solve the L1 norm minimization,

$$\min \mathbf{1}^T \binom{\hat{x}}{\check{x}}$$

$$\text{s.t. } [A \quad -A] \binom{\hat{x}}{\check{x}} = \mathbf{b}, \binom{\hat{x}}{\check{x}} \; in \; C$$

where $\hat{X}_{nx1}$, $\check{X}_{nx1}$ and C = {x| x ≥ 0}, let X = $\hat{X}_{nx1}$ - $\check{X}_{nx1}$, and

$$\min \; \mathbf{1}^T x$$

$$\text{s.t. } A\mathbf{x} = \mathbf{b}$$

$$x \text{ in } C, C = \{x|\ x \geq 0\}$$

Our Modification methods

- **Apply the "A warm restart strategy"**
  - **Step 1**
    After the first time using the original method calculated the Sudoku problem, we get a 9x9 matrix with 81 nonzero numbers. Then, we check if a number is repeated within its column, row and the 3x3 box. First, we mark those numbers' index and store into a matrix. Next, after we find all the repeated values and their corresponding locations, we set those numbers into zero which will give us a new Sudoku problem with new and more clues to solve and we take it to step 2. One thing we need to be careful is that this deleting strategy will probably delete the original clue if the value is repeated. Therefore, we have to add that miss deleting clue back into the problem.
  - **Step 2**
    This is a repeated step of the previous step which we are continuing deleting the repeated numbers by setting them into zero. Thus, we get another new Sudoku problem.
  - **Step 3**
    After step1 and 2, if we still didn't get the correct solution by deleting twice, we will apply the third strategy which is adding a new clue into the original Sudoku puzzle. In order to choose this new clue, we will randomly choose a value, which is not a clue, from the matrix calculated after the step 2. By observe the performance on the hardest dataset of adding only step 1, our method increased the accuracy rate from 32.4% to more than

60%. Therefore, we believe that those randomly chosen values have more than nearly 50% of the chance to be correct in aspect of its value and its position. Thus, adding this value into the original Sudoku problem, we formed a new puzzle with more clues and repeat step 1 once. If the solution is still not correct, we pass this specific problem.

- **Adding the Weighted L1 norm minimization**

After implemented the above "warm restart strategy", in order to improve the performance, we modified the original L1 normalization by adding weight. The algorithm of weighted L1 norm minimization method was proposed by Candes et al. [4] and we cited from [2], as following:

$$\min ||W\boldsymbol{x}||_1$$
$$\text{s.t. } A\mathbf{x} = \mathbf{b1}$$

where W = diag(w1, w2,…,wn), a diagonal matrix with $W_k = \frac{1}{|X_k|^{i-1} + \epsilon}$, $0 < \epsilon < 1$, k = 1,2,…,n and I is the iteration number.

<u>Algorithm: Solving the weighted L1- norm minimization problem</u>

```
Input: L = 10, $\hat{x} = 0, \check{x} = 0, x_{ori} = \hat{x} - \check{x}$, tol = $1 \times 10^{-10}$;
For i = 1 : L
W = diag ($\frac{1}{|X_{ori}| + \epsilon}$);
 Based on the first LP model, $x_{new} = \hat{x} - \check{x}$ is obtained by solving the following linear
programming problem:
min W$\binom{\hat{x}}{\check{x}}$, s.t. [A   -A] $\binom{\hat{x}}{\check{x}}$ = b, $\binom{\hat{x}}{\check{x}}$ $in\ C$ ;
if $||x_{new} - x_{ori}||$ < tol
    break;
    else
            $x_{ori} = x_{new}$;
End
End
```

By tuning the ϵ value for 0.01, 1 and 30, combining the steps, we get an increasing of the accuracy rate observed in the table.

- **Improvement of our method**

After modifying the naïve linear programming with weighted L1 norm and deleting procedures, our highest accuracy on the hardest dataset is still only 65.2%. Thus, in order to make some greater improvement, we tested each modification method on their contribution of calculating the correct solution, which means that we printed out how many correct solutions were calculated by step 1, step 2, step 3 and weighted L1-norm minimization. Before the testing, our expectation on step 1 is it should improve a lot and step 2 would enhance the performance and step 3 would contribute less since it is adding only one more possible correct clue. Lastly, the weighted L1-norm minimization would also enhance the performance a lot. After observing the results of the testing, we can tell that the step 3 contributed most as solving the correct answer. Step 2 literally has no contribution since the printout results didn't show any of the correct solutions were calculated by step 2. Also, we observe that by adding the weighted L1-norm, it actually doesn't effect the contribution of step 1 and step 3 which contradicted with our original expectation. Therefore, since step 3 contributed most, we will modify it by repeating step 3 more (we tested 5 times, 10 times, 15 times and 20 times). Surprisingly, with repeated running step 3 five times, our accuracy increased from 65.2% to 82%, running for ten times would give us 87.9% accuracy, with fifteen times we got to 91.1%, and with 20 times we got nearly 94%.

Conclusion

We mainly focused on linear programming method by using weighted L1 norm sparse optimization method for solving the Sudoku puzzles. Our method is weighted L1 norm linear programming applied with a deleting procedure of the repeated values and repeated adding one randomized new clue into the original Sudoku problem, which is eight times in our observations. The idea behinds it that would make it works is that we are creating a new Sudoku problem each time with a new clue value and its possibly correct position. By having one more clue, the solver would preform faster and has a higher correctness rate. In the future, we would find a pattern of at where the randomized new clue could be added in order to calculate the correct solution which could probably enhance our method accuracy rate.

Reference

1. P. Babu, K. Pelckmans, P. Stoica, J. Li, Linear systems, sparse solutions, and Sudoku, IEEE Signal Processing Letters, 17(1), 2010, 40-42.

2. Yuchao Tang1, Zhenggang Wu2, Chuanxi Zhu1, “A Warm Restart Strategy for Solving Sudoku by Sparse Optimization Methods”, June, 17, 2015

3. Linyuan Wang, Wenkun Zhang, Bin Yan, and Ailong Cai, “Equivalence of L0 and L1 Minimizations in Sudoku Problem”.

4. E. Candes, M. Wakin, S. Boyd, Enhancing sparsity by reweighted l1 minimization, Journal of Fourier Analysis Applications, 14(5), 2008, 877-905.