Sudoku Problem

MCEN 5125 Project #3

Hanwen Zhao

Abstract

In this report we describe how to use linear programming (LP) to solve sudoku problem. We will start by solving a simple 4x4 size sudoku and expand techniques we used for solving 9x9 sudoku.

1 Introduction

Numerical optimization is an extremely powerful tool for solving big and complex problem that would be difficult or costly to solve by conventional methods. In this report, we will solving a Sudoku problem by formulate is as a Linear Programming(LP) problem so that we can use the state-of-the-art solvers that are available in Matlab to solve it efficiently. Unlike previous truss topology or hand written recognition problems which were hard for human to solve or unapproachable for human being. The Sudoku problem is relative simple but here we are expanding our ability to use LP to solve more practical problems.

The Sudoku is a logical-based, combinatorial number-placement puzzle. The objective is to fill a 9x9 grid with digits from 1 to 9 so that each column, each row, and each 3x3 subgrids contains all the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. A typical Sudoku puzzle and its solution as shown in Figure 1.

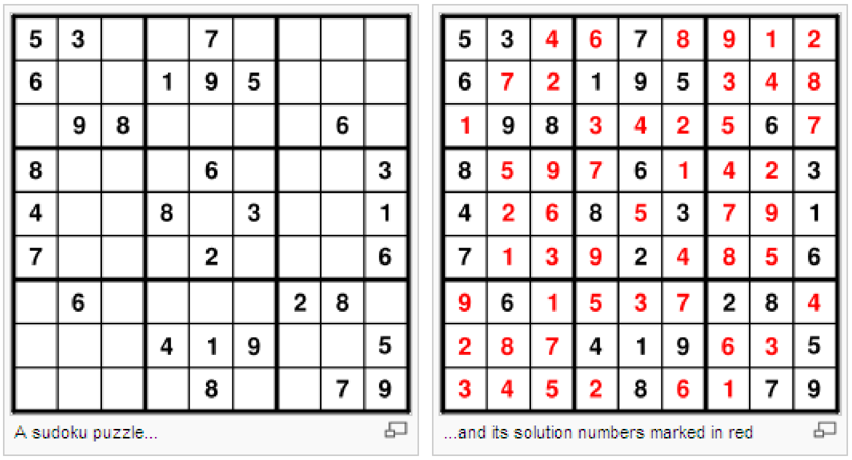

Figure 1: An example of Sudoku puzzle with its solution.

To summarize some general rules for Sudoku problem[?], each integer(from 1 to 9) should only show up once :

- in each 3x3 subgrids
- in each row
- in each column
- in each space
- no "clues" should be replaced

2 4x4 Small Sudoku

Before we dive into 9x9 standard Sudoku puzzle, we can started with a small Sudoku problem to find pattern we can use for our LP formation. An example of 4x4 Sudoku as shown in Figure 2.

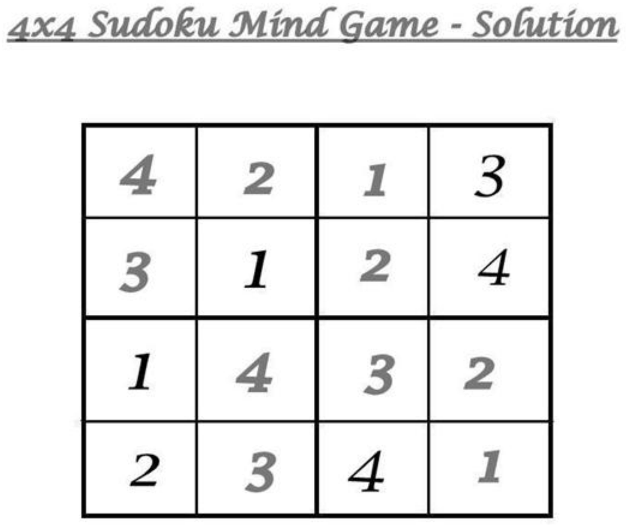

Figure 2: An example of Sudoku puzzle with its solution.

First let us define each digits in the Sudoku for easy expiation.

Table 1: Define each digits as an variable

x_{11}	x_{12}	x_{13}	x_{14}
x_{21}	x_{22}	x_{23}	x_{24}
x_{31}	x_{32}	x_{33}	x_{34}
x_{41}	x_{42}	x_{43}	x_{44}

We can observe that for each column, each row, and each 2x2 subgrid, the summation is $1 + 2 + 3 + 4 = 10$, we can write them in equations.

$$\begin{aligned}
&Rows: \\
&\quad x_{11} + x_{12} + x_{13} + x_{14} = 10 \\
&\quad x_{21} + x_{22} + x_{23} + x_{24} = 10 \\
&\quad x_{31} + x_{32} + x_{33} + x_{34} = 10 \\
&\quad x_{41} + x_{42} + x_{43} + x_{44} = 10 \\
&Columns: \\
&\quad x_{11} + x_{21} + x_{31} + x_{41} = 10 \\
&\quad x_{12} + x_{22} + x_{32} + x_{42} = 10 \\
&\quad x_{13} + x_{23} + x_{33} + x_{43} = 10 \\
&\quad x_{14} + x_{24} + x_{34} + x_{44} = 10 \\
&Subgrids: \\
&\quad x_{11} + x_{12} + x_{21} + x_{22} = 10 \\
&\quad x_{13} + x_{14} + x_{23} + x_{24} = 10 \\
&\quad x_{31} + x_{32} + x_{41} + x_{42} = 10 \\
&\quad x_{33} + x_{34} + x_{43} + x_{44} = 10
\end{aligned} \tag{1}$$

For each x_{ij} it contains 4 possibilities $x_{ij} \in 1, 2, 3, 4$. However, it is hard to solve integer problem in LP. Therefore, we are going to turn that $x_{ij} \in 0, 1$, this converts integer problem to binary problem, which we can use 1-norm to solve this. For example, now in order to represent the first digits, it becomes:

$$x_{11} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

Thus, for 4x4 Sudoku puzzle, we have $4 * 4 = 16$ digits, $4 * 4 * 4 = 64$ unknowns. Now we can write our LP as following:

$$\begin{array}{ll}
\text{minimize} & \|x\|_1 \\
\text{subject to} & \sum_{row} x = 10 \\
& \sum_{column} x = 10 \\
& \sum_{subgrids} x = 10 \\
& \sum_{x} x = 1
\end{array} \tag{3}$$

3 9x9 Standard Sudoku

Now, we can simply extend previous analysis to 9x9 standard Sudoku puzzles. First, let us visualize the 9x9 grid:

Table 2: Visualize 9x9 grid.

x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}
x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}	x_{39}
x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	x_{49}
x_{51}	x_{52}	x_{53}	x_{54}	x_{55}	x_{56}	x_{57}	x_{58}	x_{59}
x_{61}	x_{62}	x_{63}	x_{64}	x_{65}	x_{66}	x_{67}	x_{68}	x_{69}
x_{71}	x_{72}	x_{73}	x_{74}	x_{75}	x_{76}	x_{77}	x_{78}	x_{79}
x_{81}	x_{82}	x_{83}	x_{84}	x_{85}	x_{86}	x_{87}	x_{88}	x_{89}
x_{91}	x_{92}	x_{93}	x_{94}	x_{95}	x_{96}	x_{97}	x_{98}	x_{99}

For our row, column, and subgrids contains:

$$\begin{aligned}
&Rows: \\
&\qquad x_{11}+x_{12}+\ldots+x_{18}+x_{19}=45 \\
&\qquad x_{21}+x_{22}+\ldots+x_{28}+x_{29}=45 \\
&\qquad \vdots \\
&\qquad x_{91}+x_{92}+\ldots+x_{98}+x_{99}=45 \\
&Columns: \\
&\qquad x_{11}+x_{21}+\ldots+x_{81}+x_{91}=45 \\
&\qquad x_{12}+x_{22}+\ldots+x_{82}+x_{92}=45 \\
&\qquad \vdots \\
&\qquad x_{19}+x_{29}+\ldots+x_{89}+x_{99}=45 \\
&Subgrids: \\
&\qquad x_{11}+x_{12}+\ldots+x_{32}+x_{33}=45 \\
&\qquad \vdots \\
&\qquad x_{77}+x_{78}+\ldots+x_{98}+x_{99}=45
\end{aligned} \tag{4}$$

And we can define our LP as:

$$\begin{array}{ll}
\text{minimize} & \|x\|_1 \\
\text{subject to} & \sum_{row} x = 45 \\
& \sum_{column} x = 45 \\
& \sum_{subgrids} x = 45 \\
& \sum_{x} x = 1
\end{array} \tag{5}$$

4 Numerical Solution

Dr. Ruben has provided us some unsolved Sudoku puzzles at different difficult for us to test our solver. Here are some results:

4.0.1 Medium Level

Unsolved Sudoku Problem:

Table 3: Medium Level

0	5	0	0	2	0	3	7	0
0	3	0	9	4	0	0	0	1
0	0	0	7	0	0	0	0	0
0	0	5	8	0	0	9	2	0
3	0	0	0	0	0	0	0	5
0	7	8	0	0	9	1	0	0
0	0	0	0	0	2	0	0	0
8	0	0	0	7	6	0	5	0
0	2	1	0	8	0	0	6	0

Solving time 0.856612s and the solution is:

Table 4: Medium Level Solution

1	5	9	6	2	8	3	7	4
7	3	2	9	4	5	6	8	1
6	8	4	7	3	1	5	9	2
4	1	5	8	6	3	9	2	7
3	9	6	2	1	7	8	4	5
2	7	8	4	5	9	1	3	6
5	6	7	3	9	2	4	1	8
8	4	3	1	7	6	2	5	9
9	2	1	5	8	4	7	6	3

4.0.2 Evil Level

Unsolved Sudoku Problem:

Table 5: Evil Level

0	9	0	4	0	8	5	0	0
0	0	0	0	0	0	0	0	6
2	0	1	0	7	0	9	0	0
5	0	0	0	8	0	0	0	7
0	0	7	9	0	4	1	0	0
8	0	0	0	2	0	0	0	9
0	0	2	0	3	0	4	0	5
4	0	0	0	0	0	0	0	0
0	0	5	8	0	7	0	9	0

Solving time 0.849454s and the solution is:

Table 6: Evil Level Solution

6	9	3	4	1	8	5	7	2
7	5	4	3	9	2	8	1	6
2	8	1	5	7	6	9	3	4
5	4	9	1	8	3	6	2	7
3	2	7	9	6	4	1	5	8
8	1	6	7	2	5	3	4	9
9	7	2	6	3	1	4	8	5
4	3	8	2	5	9	7	6	1
1	6	5	8	4	7	2	9	3

5 Conclusion

In conclusion, we presented the LP formulation for solving Sudoku puzzles. It was shown through 4x4 small Sudoku puzzle and 9x9 standard Sudoku puzzle. It was interesting that we can solving some extremely difficult Sudoku within a second. Further constraints in the future could be added, such as constrain one digit can present once in each row, column and subgrids.

References

[1] Shalom Ruben. Lecture notes optimal design, April 2018.

6 Appendix

6.1 MATLAB Code

```
function [sudoku, x] = Sudoku_Zhao(inputMatrix)
% MCEN 5152
% Optimal Design
% Project #3: Sudoku
% Hanwen Zhao
% MEID: 650-703
% inputMatrix = [8 0 0 0 0 0 0 0 0;
%                0 0 3 6 0 0 0 0 0;
%                0 7 0 0 9 0 2 0 0;
%                0 5 0 0 0 7 0 0 0;
%                0 0 0 0 4 5 7 0 0;
%                0 0 0 1 0 0 0 3 0;
%                0 0 1 0 0 0 0 6 8;
%                0 0 8 5 0 0 0 1 0;
%                0 9 0 0 0 0 4 0 0]
N = 9; NN = N*N; NNN = N*N*N;
```

```
% allocate memory for row contrains matrix
A_row = zeros(NN, NNN);
for n = 1:9
    counter = 0;
    for i = 1:N
        for j = 1:N
            A_row(i+(n-1)*N,n+counter*N) = 1;
            counter = counter + 1;
        end
    end
end
% create matrix for column constains
A_col = repmat(eye(NN),1,9);
% allocate memory for subgrid constrains
A_sub = zeros(NN,NNN);
for k = 1:N
    for i = 1:N
        for j = 1:sqrt(N)
            A_sub(i+(k-1)*N,(1+floor((i-1)/3)*243)+(i-(floor((i
                -1)/3)*3+1))*N*3+(j-1)*N+k-1) = 1;
            A_sub(i+(k-1)*N,(1+floor((i-1)/3)*243)+(i-(floor((i
                -1)/3)*3+1))*N*3+NN+(j-1)*N+k-1) = 1;
            A_sub(i+(k-1)*N,(1+floor((i-1)/3)*243)+(i-(floor((i
                -1)/3)*3+1))*N*3+2*NN+(j-1)*N+k-1) = 1;
        end
    end
end
% calcualte total number of clues
numClue = sum(sum(inputMatrix>0));
% allocate memory for clue matrix
A_clue = zeros(numClue, NNN);
counter = 1;
for i =1:N
    for j = 1:N
        if inputMatrix(i,j)> 0
            A_clue(counter,(i-1)*NN+N*(j-1)+inputMatrix(i,j)) =
                1;
            counter = counter + 1;
        end
    end
end
% allocate memory for uniqueness (each binary should only exits
  one)
A_uni = zeros(NN,NNN);
for i = 1:NN
    for j = 1:N
```

```matlab
        A_uni(i,j+(i-1)*N) = 1;
    end
end
% start building final LP
A = [A_row; A_col; A_sub; A_clue; A_uni; -A_sub; -A_clue; -A_uni];
b = [ones(NN,1); ones(NN,1); ones(NN,1); ones(numClue,1); ones(NN,1); -ones(NN,1); -ones(numClue,1); -ones(NN,1)];
f = ones(NNN,1);
x = linprog(f, A, b, [], [], zeros(NNN,1), ones(NNN,1));
% convert binary solution to decimal
counter = 1;
sol = zeros(NN,1);
for i=1:NN
    for j = 1:N
        if x((i-1)*N+j,1)~=0
            sol(counter,1)=j;
            counter = counter + 1;
        end
    end
end
% display final matrix
sudoku = zeros(N,N);
counter = 1;
for i = 1:N
    for j = 1:N
        sudoku(i,j) = sol(counter,1);
        counter = counter + 1;
    end
end
sudoku
end
```