Sudoku via Optimization

Robert J. Vanderbei

2019 October 21

INFORMS
Seattle WA

http://www.princeton.edu/∼rvdb

NY Times, Hard Puzzle, Oct. 11 2019

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

Rules:

1. Each row must contain the numbers $1, 2, \ldots, 9$.
2. Each column must contain the numbers $1, 2, \ldots, 9$.
3. Each 3×3 block must contain the numbers $1, 2, \ldots, 9$.

NY Times, Hard Puzzle, Oct. 11 2019

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

$\Rightarrow$

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

Rules:

1. Each row must contain the numbers $1, 2, \ldots, 9$.
2. Each column must contain the numbers $1, 2, \ldots, 9$.
3. Each 3×3 block must contain the numbers $1, 2, \ldots, 9$.

An Optimization Approach

Variables: $x_{i,j,k} = \begin{cases} 1 & \text{if number } k \text{ is in row } i, \text{ column } j, \\ 0 & \text{otherwise.} \end{cases}$

An Optimization Approach

Variables:

$$x_{i,j,k} = \begin{cases} 1 & \text{if number } k \text{ is in row } i, \text{ column } j, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints:

- Each cell must contain a number:

$$\sum_{k} x_{i,j,k} = 1 \qquad \text{for } i = 1, \ldots, 9, \quad j = 1, \ldots, 9$$

- Each row must contain each number:

$$\sum_{j} x_{i,j,k} = 1 \qquad \text{for } i = 1, \ldots, 9, \quad k = 1, \ldots, 9$$

- Each column must contain each number:

$$\sum_{i} x_{i,j,k} = 1 \qquad \text{for } j = 1, \ldots, 9, \quad k = 1, \ldots, 9$$

- Each 3×3 block must contain each number:

$$\sum_{i=3i_0-2}^{3i_0} \sum_{j=3j_0-2}^{3j_0} x_{i,j,k} = 1 \qquad \text{for } i_0 = 1, \ldots, 3, \quad j_0 = 1, \ldots, 3$$

An Optimization Approach – Continued

The *gray* cells have given values that we're not allowed to change. Let's denote those values by $P_{i,j}$.

One Last Constraint Set:

- For each "gray" cell:

$$x_{i,j,P_{i,j}} = 1$$

An Optimization Approach – Continued

The *gray* cells have given values that we're not allowed to change. Let's denote those values by $P_{i,j}$.

One Last Constraint Set:

- For each "gray" cell:

$$x_{i,j,P_{i,j}} = 1$$

Objective Function:

- We don't need an objective function. Any feasible solution will do.

AMPL Model

Here's the code written in AMPL:

```
param P{1..9,1..9} default 0, integer, >= 0, <= 9;


var x {i in 1..9, j in 1..9, k in 1..9} >= 0, <= 1, integer;


minimize zero: 0;


s.t. val_sum {i in 1..9, j in 1..9}: sum {k in 1..9} x[i,j,k] = 1;
s.t. row_sum {i in 1..9, k in 1..9}: sum {j in 1..9} x[i,j,k] = 1;
s.t. col_sum {j in 1..9, k in 1..9}: sum {i in 1..9} x[i,j,k] = 1;
s.t. box_sum {i0 in 1..3, j0 in 1..3, k in 1..9}:
          sum {i in 3*i0-2..3*i0, j in 3*j0-2..3*j0} x[i,j,k] = 1;

s.t. fixed_vals {i in 1..9, j in 1..9: P[i,j] <> 0}: x[i,j,P[i,j]] = 1;
```

AMPL Data and Solve

```
param P:  1  2  3   4  5  6   7  8  9 :=

1         .  .  9   7  .  .   .  .  3
2         .  .  .   9  .  .   1  .  .
3         .  .  .   3  .  6   .  .  8

4         9  .  6   .  4  .   .  .  .
5         2  .  3   .  .  5   .  .  6
6         .  .  .   .  .  .   .  5  7

7         .  3  .   .  .  2   .  8  5
8         8  .  .   .  .  .   .  .  .
9         1  .  .   .  .  .   .  .  .  ;

solve;

printf "\n";
for {i in 1..9} {
    if (i mod 3) = 1 then {printf "\n ";} else {printf " ";}
    for {j in 1..9} {
        if (j mod 3) = 1 then {printf "   ";} else {printf " ";}
        printf " %1d", round(sum {k in 1..9} k*x[i,j,k]);
    }
    printf "\n";
}
printf "\n";
```

AMPL Output

```
LOQO 7.03:
ignoring integrality of 198 variables
variables: non-neg      0,  free        0,  bdd        198,  total      198
constraints: eq       212,  ineq        0,  ranged       0,  total      212
nonzeros:    A        792,  Q           0
OPTIMAL SOLUTION FOUND

Times (seconds):
Input =  0.000791
Solve =  0.00512
Output = 0.000257

LOQO 7.03: optimal solution (15 iterations}
primal objective 0
  dual objective -2.288607166e-08


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

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

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

Integer vs. Linear

Note that the solver used was LOQO.

This solver ignores *integer* constraints.

The solver found the correct answer.

Why?

Because the problem has a unique integer solution.

And that implies that the linear relaxation has the same unique solution.

For creators of sudoku problems, the problem must have a unique solution.

Hence, it is always possible to solve sudoku problems as linear optimization problems.

Creating Sudoku Problems

- Put the nine numbers $1, 2, \ldots, 9$ at nine randomly chosen cells in the grid. Declare these cells as *fixed*.
- Solve the sudoku problem enforcing the integrality of the $x_{i,j,k}$'s and using a random objective function:

$$\text{minimize} \sum_{i=1}^{9} \sum_{j=1}^{9} \sum_{k=1}^{9} c_{i,j,k} x_{i,j,k}$$

  where the $c_{i,j,k}$'s are i.i.d. random variables.
- Solve the sudoku problem again enforcing the integrality of the $x_{i,j,k}$'s but using the "opposite" of the above random objective function:

$$\text{maximize} \sum_{i=1}^{9} \sum_{j=1}^{9} \sum_{k=1}^{9} c_{i,j,k} x_{i,j,k}$$

- If the two solutions are the same, then we are *done*.
- If the two solutions differ, then pick one of the not-yet-fixed cells where the two solutions differ and fix the value to that given by the first (or second) solution.
- While not done, repeat the previous four steps.