Sudoku as a Linear/Integer Programming Problem

Professor Naonori Kakimura
Operations Research

Lin Gengxian Shunji
08-144505
29th January 2015

Background

Sudoku is combinatorial number-placement game that was invented by the Japanese puzzle company Nikoli, and has seen become a popular puzzle game played by many (Hayes B, 2006). Players attempt to fill a square-grid with an integer per cell, with numbers falling within the length of the grid (e.g. numbers 1 – 4 for a 4x4 grid). Although the most popular version of the game involves a 9x9 grid, the same principle can be applied to various other square-grid sizes.

The general rules of the placement of numbers in Sudoku are as follows:

1) Each integer can only show up once in each row

2) Each integer can only show up once in each column

3) Each integer can only show up once in each sub-square, where the sub-square refers to a smaller square-division of the entire grid (e.g. a 9x9 board has 9 sub-squares of 3x3 size each).

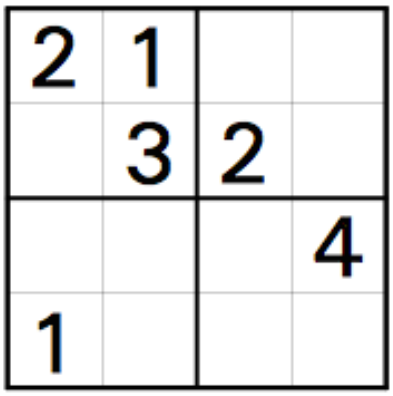
Finally, the initial Sudoku puzzle is also partially filled with numbers, which serve as "clues" for the puzzle. In the clues are well designed, this ensures that the puzzle has a unique set of solutions. The following is an example of a Sudoku problem, with its solution (taken from www.websudoku.com):

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

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

Modelling the Integer Programming Problem

The Sudoku problem can be modeled as an integer-programming problem. In the explanation below, we will assume that the general case is a Sudoku of square grid (m x n grid, where m = n). In order to formulate the problem, I will also use a simplified version, a 4x4 Sudoku problem, as an example (taken from www.sudoku-download.net).



Representing the grids and numbers (4x4 problem)

For the above problem (m = n = 4), we can visualize the Sudoku grid with its row and column indexes as follows:

r \ c	1	2	3	4
1	2	1		
2			2	
3				4
4	1			

In order to represent the numbers on the grid, we can represent each cell with a set of binary variables (which can have values 0 or 1), where the amount of variables per cell is equivalent to the length of the grid. For the general case, the number can take on a value of between 1 to m in each cell, and hence each cell needs m variables. In this case, we represent each cell with 4 binary variables. To represent the number on each cell, one of the 4 binary variables is set to 1, while the rest are set to 0, as follows:

4x4 case:

Number in Cell	var1	var2	var3	var4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Visualizing the clue variables in the Sudoku grid, we get the following:

r \ c	1	2	3	4
1	0 1 0 0	1 0 0 0	0	0
2	0	0	0 1 0 0	0
3	0	0	0	0 0 0 1
4	1 0 0 0	0	0	0

As for the cells without clues, we can set the 4 cell variables to contain all 0, as indicated above. In the general case, there will be a total of m^3 number of variables $x_1 \; to \; x_{m^3}$.

Representing the constraint given by the clues

We can model the initial constraints given by the partially filled Sudoku problem by fixing the binary values of the clue cells. In algebraic form, for the given 4x4 example above, the constraints given by the 6 clues will be as follows:

r	c	clue in cell	constraints
1	1	2	$x_1 = 0,\ x_2 = 1,\ x_3 = 0,\ x_4 = 0$
1	2	1	$x_5 = 1,\ x_6 = 0,\ x_7 = 0,\ x_8 = 0$
2	2	3	$x_{21} = 0,\ x_{22} = 0,\ x_{23} = 1,\ x_{24} = 0$
2	3	2	$x_{25} = 0,\ x_{26} = 1,\ x_{27} = 0,\ x_{28} = 0$
3	4	4	$x_{45} = 0,\ x_{46} = 0,\ x_{47} = 0,\ x_{48} = 1$
4	1	1	$x_{49} = 1,\ x_{50} = 0,\ x_{51} = 0,\ x_{52} = 0$

In algebraic form, this is given as $x_{(r-1)\times m^2 + (c-1)\times m + clue\ number} = 1$, for the given

clue numbers.

The above constraints can also be expressed in the matrix form **Ax** = **b**. **x** is assigned to the single-column matrix of all variables, with length m^3. **b** is assigned to a single-column matrix of binary values (0 or 1) fixed by the clues, with length $(no.of\ clues \times m)$, which in this case is 6 x 4 = 24. **A** is a matrix of dimension $(no.of\ clues \times m)$ rows by m^3 rows, and simply serves as a 'selector' to match the corresponding clue variables in **x** to their binary values in **b**. Thus for each row in **A**, there is only one value of 1 while the rest are 0s.

Representing the constraint given by row and column rules
Recall rule 1 and 2:
1) Each integer can only show up once in each row
2) Each integer can only show up once in each column

Since there are only m possible numbers and m cells, it is also helpful to note that each integer must show up once in each row/ column. Thus these constraints can be formulated by ensuring that in each row/ column, the binary values of variables corresponding to the same number sum up to be exactly 1. Rule 1 (row rule) gives m^2 constraints and likewise, Rule 2 (column rule) gives m^2 constraints, giving $2m^2$ additional constraints in total. One row constraint example and one column constraint example are given below:

type	index	number	constraint
row	2	3	$x_9 + x_{23} + x_{27} + x_{31} = 1$
column	4	1	$x_{13} + x_{29} + x_{45} + x_{61} = 1$

In algebraic form, the general forms of the row and column constraints can be expressed as follows:

Row constraints

$$\sum_{k=[(r-1)\times m]}^{m} x_{[k\times m]+number} = 1$$

Where r represents the row index (1 to m) and number represents the possible numbers (1 to m).

Column constraints

$$\sum_{k=1}^{m} x_{[(c-1)\times m]+number+(k-1)\times m^2} = 1$$

Where c represents the column index (1 to m) and number represents the possible numbers (1 to m)

In matrix form **Ax** = **b**, **x** remains the single-column matrix of all variables, while **b** is now the single-column matrix of 1s, of length $2m^2$. Hence A is the matrix of dimension $2m^2$ by m^3, with each row having m entries of 1 to 'select' the set of variables in x that sums up to 1.

Representing the constraint given by the sub-squares

Recall Rule 3:

3) Each integer can only show up once in each sub-square.

In a general m x m Sudoku grid where m is a perfect square, each sub-square has $\sqrt{m}$ by $\sqrt{m}$ cells, and is demarcated by the bolder lines (as seen in the given 4 x 4 example). There are a total of m sub-squares in entire grid. Using the same principle as the the row/ column rules, we need to sum up each set of variables within each sub-square that correspond to each number and equate them to 1s. As such, there are m^2 additional constraints given by the sub-squares rule.

In algebraic form, the general form of the sub-square constraints can be expressed as

follows:

$$\sum_{l=1}^{\sqrt{m}} \sum_{k=1}^{\sqrt{m}} x_{[(k-1)m+(l-1)\times m^2+(SS_r-1)\times m^{5/2}+(SS_c-1)\times m^{3/2}+number]} = 1$$

Where SS_r represents the sub-square row (1 to $\sqrt{m}$), SS_c represents the sub-square column (1 to $\sqrt{m}$) and number represents the possible numbers (1 to m). For the 4x4 problem given, the sub-square row and column indexes can be visualized as follows:

SS_r \ SS_c	1		2	
1	2	1		
			2	
2				4
	1			

As with the row/ column constraints, the sub-square constraints can be expressed in the matrix form **Ax = b,** with a similar principle. A now has dimensions m^2 by m^3, and **b** is of length m^2.

As an example, the sub-square constraints of the bottom right sub-square,(SS_r = 2, SS_c = 2) is given as follows:

number	constraint
1	$x_{41} + x_{45} + x_{57} + x_{61} = 1$
2	$x_{42} + x_{46} + x_{58} + x_{62} = 1$
3	$x_{43} + x_{47} + x_{59} + x_{63} = 1$
4	$x_{44} + x_{48} + x_{60} + x_{64} = 1$

Representing cell constraints

Finally, with m binary variables per cell, we have to impose the constraint that each cell can only take one number. This gives an additional m^2 constraints, and each cell constraint is expressed algebraically as follows:

$$\sum_{k=1}^{m} x_{k+(r-1)\times m^2+(c-1)\times m} = 1$$

Where r represents the row index (1 to m) and c represents the column index (1 to m).

In matrix form **Ax** = **b**, **A** now has dimensions m^2 by m^3, and **b** has a length of m^2.

Combining the constraints and modeling the problem in SCIP

Combining the clue, row /column, sub-square and cell constraints, in total there are $(no.of\ clues \times m) + 4m^2$ constraints. In the matrix form **Ax** = **b**, **A** will have the dimension $[(no.of\ clues \times m) + 4m^2]$ by m^3, **x** will be the single-column matrix of length m^3 and **b** will be the single-column matrix of length $[(no.of\ clues \times m) + 4m^2]$.

Since for this integer programming problem, the variables simply have to satisfy the strict constraints, there is no need to maximize or minimize any objective function. Hence the integer programming problem can be modeled as such:

max 0**x**

subject to:

Clue Constraints: $x_{(r-1)\times m^2+(c-1)\times m+clue\ number} = 1$ (for all clue numbers)

Row Constraints: $\sum_{k=[(r-1)\times m]}^{m} x_{[k\times m]+number} = 1$

Column Constraints: $\sum_{k=1}^{m} x_{[(c-1)\times m]+number+(k-1)\times m^2} = 1$

Sub-square Constraints:

$$\sum_{l=1}^{\sqrt{m}} \sum_{k=1}^{\sqrt{m}} x_{[(k-1)m+(l-1)\times m^2+(SS_r-1)\times m^{5/2}+(SS_c-1)\times m^{3/2}+number]} = 1$$

Cell Constraints: $\sum_{k=1}^{m} x_{k+(r-1)\times m^2+(c-1)\times m} = 1$

$x_{(1\ to\ m^3)} \in \{0,1\}$

Where r = (1 to m), c = (1 to m), SSr = (1 to $\sqrt{m}$), SSc = (1 to $\sqrt{m}$), number = (1 to m)

Implementing the model

I have created a ruby program, 'sudoku.rb', to implement the above linear programming model. The program is able to parse through a text file containing the partially filled Sudoku in grid form (clues), deriving the constraints and outputting the constraints to

the SCIP program. The returned results can then be output into a text file in grid form. More information on the program can be found in the documentation text.

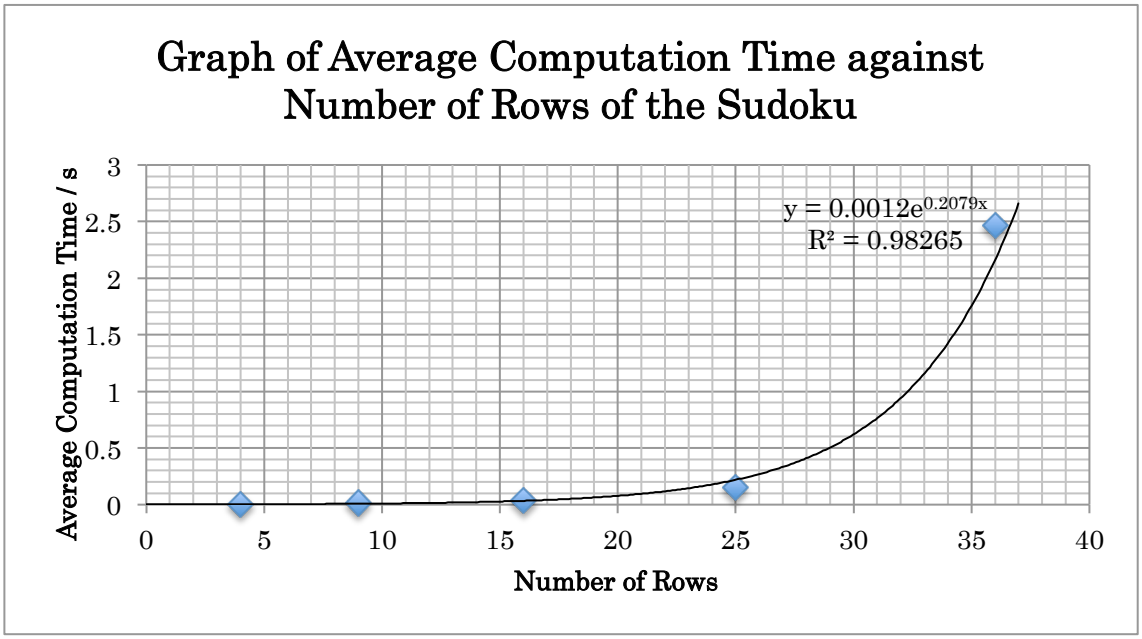
Using this model, I have attempted to solve several examples of Sudoku problems from www.websudoku.com, www.sudoku-download.net and http://www.menneske.no including nine 4 by 4 puzzles, twelve 9 by 9 problems, five 16 by 16 puzzles, two 25 by 25 puzzles and one 36 by 36 puzzle. The model is able to solve the given Sudoku puzzles,

LP relaxation

Furthermore, given sufficient clues, or for Sudoku's puzzles with one unique solution, it is not necessary to apply the integer constraint, and LP relaxation is able to produce the integer results. This is due to the constraint matrix **A** being a sparse matrix with only 0 and 1 values, giving it its unimodular property.

Computation Time

It is also interesting to see how computation time varies with increased dimensions of the Sudoku puzzle. Taking the average computation time (as given in SCIP) against the number of rows of the puzzle, I was able to obtain the following graph:



It appears that for the given set of Sudoku puzzles, the computation time follows an exponential rate of increase as the number of rows of the puzzle increases. This might be due to the fact that the constraints grow exponentially as the dimension increases, given that we have to specify m number of variables for each cell of the Sudoku. As such, this model might not be efficient enough to solve Sudoku of large dimensions.

Conclusion

It can be seen that the linear programming model as explained above is effective in solving a Sudoku puzzle with well-defined clues. Furthermore, the linear programming model can be modeled in a general way to handle any Sudoku puzzle with row m by m, where m is a perfect square. However, the linear programming model is by no means the only way to solve Sudoku puzzles, and there are many other algorithms that can be used in its place. Also, the computational time for the linear programming model of solving the Sudoku puzzle increases exponentially with increased dimension, making this model not efficient in solving higher dimension Sudoku puzzles.

References

Hanssen V. (n.d.) *Sudoku puzzles 6x6 (Su Doku).* Retrieved 28th January 2015, from: http://www.menneske.no/sudoku/6/eng/

Hayes B. (2006) Unwed Numbers – *The mathematics of Sudoku, a puzzle that boasts "No math required!"* American Scientist. Vol 94 – 1, Pg 12. Retrieved 23th January, from: http://www.americanscientist.org/issues/pub/2006/1/unwed-numbers

Maack A. (2015) *Sudoku-Download.* Retrieved 24th January 2015, from: http://www.sudoku-download.net

Shalom DR & Sriram S. (2014) Linear and Integer Programming. University of Colorado Boulder. Retrieved 23th January 2015, from: https://www.coursera.org/course/linearprogramming

Web Sudoku. (n.d.) *Web Sudoku*. Retrieved 24th January 2015, from: http://www.websudoku.com