Business Management Dynamics
Vol.5, No.3, Sep 2015, pp.01-11

A Genetic Spreadsheet Model for Solving Sudoku Problems

Kathryn W. Ernstberger[1] and M.A. Venkataramanan[2]

Abstract

Sudoku puzzles are both a source of entertainment for all ages and an algorithmic design challenge for mathematical modelers. Prior spreadsheet based optimization models for solving the Sudoku problem using Excel require the Premium Solver add-in, have complex constraint sets, and/or have limited success yielding solutions. This paper describes an alternate spreadsheet model formulation that uses relatively few decision variables, streamlined constraint sets, and can be solved using Evolutionary Solver which is included in the standard version of Excel. This approach for solving Sudoku problems is thus readily available for those involved in teaching and learning management science.

Key words: Spreadsheet, Sudoku, Genetic Search

Available online
www.bmdynamics.com
ISSN: 2047-7031

INTRODUCTION

While the precursor to Sudoku, Latin Squares, has been around for centuries, the current form of Sudoku was introduced in 1979 by Howard Garns, an architect from Indiana (Delahaye, 2006). Today, a regular supply of new puzzles can be found online, in newspapers, magazines, and books, and even downloaded as an app from the iTunes or Windows stores. Players range from grade-schoolers sharpening logical thinking skills to Alzheimer's patients trying to slow the progression of their illness (ABC News, 2012).

The most common version of the puzzle is a 9 X 9 matrix, composed of nine 3 X 3 submatrix blocks. The goal is organize nine sets of the integers 1 to 9 in such a way that each integer occurs exactly once in each row, column, and 3 X 3 submatrix block. Some entries are fixed at the start and the player has to fill in the rest by deduction. The puzzles are typically categorized from easy to insane, depending on the degree of difficulty. An example of the starting point of a 9X9 puzzle and its solution is shown in figure 1 (Friesen, et. al., 2013).

Several binary integer linear program (BILP) formulations have been developed for the Sudoku problem. Chond (2005) used the Xpress-Mosel programming language, Koch (2005) generated the BILP using the algebraic modeling language ZIMPL, and Bartlett, et. al. (2008) used MATLAB. Weiss and Rasmussen (2007) offered a spreadsheet based formulation of the BILP, using VBA code for creating some of the constraints and used Excel's Solver add-in for solving. Rasmussen and Weiss (2007) and Frisen, et. al. (2013) proposed spreadsheet based integer programming model (non-binary) formulations that are solved using Excel's Premium Solver add-in. We present an alternate spreadsheet model formulation that can be solved using Evolutionary Solver in the standard version of Excel.

[1] *Indiana University Southeast, School of Business*
E-mail: kernst@ius.edu
[2] *Indiana University, Kelley School of Business*
E-mail: venkatar@indiana.edu

Figure 1: Sudoku Problem and Solution

	A	B	C	D	E	F	G	H	I	J
1		Original Puzzle								
2		1	2	3	4	5	6	7	8	9
3	1	***6***	0	0	0	***4***	0	0	***1***	0
4	2	0	***1***	0	0	0	0	0	0	***3***
5	3	0	0	***2***	0	0	***8***	0	***4***	0
6	4	0	***2***	0	0	0	0	0	0	***4***
7	5	0	0	***7***	***3***	***8***	***2***	***6***	0	0
8	6	***5***	0	0	0	0	0	0	***2***	0
9	7	0	***9***	0	***5***	0	0	***1***	0	0
10	8	***4***	0	0	0	0	0	0	***7***	0
11	9	0	***5***	0	0	***9***	0	0	0	***2***
12										
13		Solution								
14		1	2	3	4	5	6	7	8	9
15	1	***6***	3	5	2	***4***	9	8	***1***	7
16	2	8	***1***	4	7	5	6	2	9	***3***
17	3	9	7	***2***	1	3	***8***	5	***4***	6
18	4	3	***2***	6	9	1	5	7	8	***4***
19	5	1	4	***7***	***3***	***8***	***2***	***6***	5	9
20	6	***5***	8	9	4	6	7	3	***2***	1
21	7	2	***9***	3	***5***	7	4	***1***	6	8
22	8	***4***	6	1	8	2	3	9	***7***	5
23	9	7	***5***	8	6	***9***	1	4	3	***2***

ORIGINAL MODELS

The Frisen, et. al. (2013) formulation and the first Rasmussen and Weiss (2007) formulation rely on the use of "alldifferent" constraints (requires that all decision variables in the range are unique) and "if then" logic. Three sets of decision variables are needed so the "alldifferent" constraint can be imposed on rows, columns, and submatrix blocks. Due to the number of decision variables, solution of a 9X9 Sudoku puzzle using their models requires the use of Premium Solver, which is a separately purchased add-in. The models have no objective function, since no value is optimized.

Rasmussen and Weiss (2007) proposed a second formulation that reduced the number of decision variables to one-third of that required by their first formulation and replaced the "alldifferent" constraints with numerous COUNTIF functions that counted how many times each integer was used in each row, column, and submatrix block. Since each integer should appear only once in each row, column, and submatrix block, the objective function minimized the number of violations. The model was solved using both Evolutionary Solver and OptQuest Solver, with limited success using the former.

ALTERNATE GENETIC SEARCH MODEL

Similar to the models described in the prior section, the alternate model proposed in this work is spreadsheet based, uses the "alldifferent" constraints to ensure that the entries in each submatrix are unique, and imposes penalties in the objective function for violations. However, to ensure that row and column entries are unique, this model uses the RANK function in conjunction with constraints on the sum of the ranks. This model is also able to use one-third the number of decision variables as is required with the Rasmussen and Weiss (2007) first formulation and with the Frisen, et. al. (2013) formulation. Evolutionary Solver, which uses genetic algorithms, can be used with the standard version of Excel to solve this model, thus making it readily available for all of those involved in teaching and learning management science.

Genetic algorithms are heuristic methods that intelligently search the feasible region of an optimization model for a solution. If the algorithm runs long enough, it will find the optimal solution, but if the model is overly complex, the time required could be prohibitive. A genetic algorithm (GA) performs best when there are few constraints other than those imposing bounds on the decision variables. However, the objective function can be as complex as necessary. The latter is the key advantage of using GA over optimization methods when models include differentiable functions, dichotomous regions, concavity or convexity (Winston and Albright, 2016). To solve an optimization problem, the GA randomly generates a set of possible solutions, commonly referred to as the population. Each solution is then evaluated based on feasibility and the objective function value. The best solutions are retained and bred to generate the next generation of the population. Mutations are introduced on a probabilistic basis to avoid local optima. The process of creating new generations continues until either optimality is reached or further improvement cannot be achieved. Genetic search has been successfully implemented in business, engineering, and health care fields (Walter, 2003; Gen & Cheng, 2000; Smith & Cagnoni, 2011). Due to the widespread applicability of this technique, Frontline Systems recently added Evolutionary Solver with genetic search capability, formerly available only in Premium Solver, to the standard version of Solver in Excel.

The steps of the proposed genetic spreadsheet model for solving the Sudoku problem are provided below and an example follows.

1. Enter a 9X9 Sudoku problem in the corresponding 9 rows and 9 columns in Excel.
2. Copy the above matrix to a new location in the spreadsheet. These are the 81 decision variables.
3. Use the rank function to determine the rank of the decision variables in each row. The rank of a number is its size relative to other values in a list. If each decision variable in the row is unique, the sum of the ranks will be 45. (Note that the rank must be an integer from 1 to 9, with no duplicates/ties.)
4. Use the rank function to determine the rank of the decision variables in each column. If each decision variable in the column is unique, the sum of the ranks will be 45.
5. Using if-then logic, if there is a starting fixed value for a cell, check to see if the decision variable for that cell is equal to the corresponding starting fixed value. If not, flag it. Do this for all cells.
6. In genetic problems, constraints are usually incorporated in the objective by penalizing the objective if the constraint logic is violated. For this model, there are three types of penalties that contribute to the objective function – a penalty when the decision variable does not match the fixed starting value, a penalty for each duplicate in a row, and a penalty for each duplicate in a column. The objective function is then the sum of all penalty costs and thus should be minimized.
7. Impose an "alldifferent" constraint on each of the nine 3X3 submatrix blocks. This ensures that the set of integers 1 through 9 appears exactly once in each submatrix block.
8. Solve using Evolutionary Solver.

At optimality, the objective function is zero, indicating that all constraint logic is satisfied. If the solution procedure stops before it reaches zero, it should be started again.

The genetic spreadsheet Sudoku model will be demonstrated using the same problem that Frisen, et. al. (2013) used and was already shown here in figure 1. Screenshots that illustrate the steps are provided.

Steps 1 and 2 – Figure 2 shows the original problem and the necessary 81 decision variables.

Figure 2: Original Problem and Decision Variables

	A	B	C	D	E	F	G	H	I	J
1		Original Puzzle								
2		1	2	3	4	5	6	7	8	9
3	1	***6***	0	0	0	***4***	0	0	***1***	0
4	2	0	***1***	0	0	0	0	0	0	***3***
5	3	0	0	***2***	0	0	***8***	0	***4***	0
6	4	0	***2***	0	0	0	0	0	0	***4***
7	5	0	0	***7***	***3***	***8***	***2***	***6***	0	0
8	6	***5***	0	0	0	0	0	0	***2***	0
9	7	0	***9***	0	***5***	0	0	***1***	0	0
10	8	***4***	0	0	0	0	0	0	***7***	0
11	9	0	***5***	0	0	***9***	0	0	0	***2***

To illustrate how the process works, an infeasible, sample solution is shown in figure 3 and evaluated as the process moves toward optimality.

Figure 3: Problem with Sample Decision Variables

	A	B	C	D	E	F	G	H	I	J
1										
13		Sample Solution								
14		1	2	3	4	5	6	7	8	9
15	1	***6***	3	5	1	***4***	9	8	***1***	7
16	2	9	***1***	4	2	5	6	2	9	***3***
17	3	8	7	***2***	7	3	***8***	5	***4***	6
18	4	3	4	6	5	9	1	7	8	***4***
19	5	9	2	***7***	***3***	***8***	***2***	***6***	5	9
20	6	***5***	8	1	6	7	7	3	***2***	1
21	7	2	***9***	3	***5***	7	4	***1***	6	8
22	8	***4***	6	1	8	2	3	9	2	4
23	9	7	***5***	8	6	***9***	1	5	3	7

Steps 3 and 4 – Rank the decision variables in each row and each columns. The RANK.EQ function in Excel assigns a rank to the cell that is relative to the values in the given range. If there are ties (which would not occur at optimality), the same rank is assigned to the cells containing the common values. At optimality, the sum of the ranks for a 9X9 puzzle should be 45. Figure 4 shows the ranks for this suboptimal solution and figure 5 shows the formulas.

Figure 4: Rank Orders of Rows and Columns

	A	B	C	D	E	F	G	H	I	J	K
1											
24											
25	Determining Column Rank										
26		1	2	3	4	5	6	7	8	9	45
27	1	5	7	4	9	7	1	2	9	3	47
28	2	1	9	5	8	6	4	8	1	8	50
29	3	3	3	7	2	8	2	5	5	5	40
30	4	8	6	3	5	1	8	3	2	6	42
31	5	1	8	2	7	3	7	4	4	1	37
32	6	6	2	8	3	4	3	7	7	9	49
33	7	9	1	6	5	4	5	9	3	2	44
34	8	7	4	8	1	9	6	1	7	6	49
35	9	4	5	1	3	1	8	5	6	3	36
36	45	44	45	44	43	43	44	44	44	43	Sum
37	Determining Row Rank										
38		1	2	3	4	5	6	7	8	9	45
39	1	4	7	5	8	6	1	2	8	3	44
40	2	1	9	5	7	4	3	7	1	6	43
41	3	1	3	9	3	8	1	6	7	5	43
42	4	8	6	4	5	1	9	3	2	6	44
43	5	1	8	4	7	3	8	5	6	1	43
44	6	5	1	8	4	2	2	6	7	8	43
45	7	8	1	7	5	3	6	9	4	2	45
46	8	4	3	9	2	7	6	1	7	4	43
47	9	3	6	2	5	1	9	6	8	3	43
48	45	35	44	53	46	35	45	45	50	38	Sum

Figure 5: Formulas for Rank Orders of Rows and Columns

	A	B	C	D	E	F
1						
24						
25	Determining Column Rank					
26		1	2	3	4	5
27	1	=RANK.EQ(B15,B$15:B$23)	=RANK.EQ(C15,C$15:C$23)	=RANK.EQ(D15,D$15:D$23)	=RANK.EQ(E15,E$15:E$23)	=RANK.EQ(F15,F$15:F$23)
28	2	=RANK.EQ(B16,B$15:B$23)	=RANK.EQ(C16,C$15:C$23)	=RANK.EQ(D16,D$15:D$23)	=RANK.EQ(E16,E$15:E$23)	=RANK.EQ(F16,F$15:F$23)
29	3	=RANK.EQ(B17,B$15:B$23)	=RANK.EQ(C17,C$15:C$23)	=RANK.EQ(D17,D$15:D$23)	=RANK.EQ(E17,E$15:E$23)	=RANK.EQ(F17,F$15:F$23)
30	4	=RANK.EQ(B18,B$15:B$23)	=RANK.EQ(C18,C$15:C$23)	=RANK.EQ(D18,D$15:D$23)	=RANK.EQ(E18,E$15:E$23)	=RANK.EQ(F18,F$15:F$23)
31	5	=RANK.EQ(B19,B$15:B$23)	=RANK.EQ(C19,C$15:C$23)	=RANK.EQ(D19,D$15:D$23)	=RANK.EQ(E19,E$15:E$23)	=RANK.EQ(F19,F$15:F$23)
32	6	=RANK.EQ(B20,B$15:B$23)	=RANK.EQ(C20,C$15:C$23)	=RANK.EQ(D20,D$15:D$23)	=RANK.EQ(E20,E$15:E$23)	=RANK.EQ(F20,F$15:F$23)
33	7	=RANK.EQ(B21,B$15:B$23)	=RANK.EQ(C21,C$15:C$23)	=RANK.EQ(D21,D$15:D$23)	=RANK.EQ(E21,E$15:E$23)	=RANK.EQ(F21,F$15:F$23)
34	8	=RANK.EQ(B22,B$15:B$23)	=RANK.EQ(C22,C$15:C$23)	=RANK.EQ(D22,D$15:D$23)	=RANK.EQ(E22,E$15:E$23)	=RANK.EQ(F22,F$15:F$23)
35	9	=RANK.EQ(B23,B$15:B$23)	=RANK.EQ(C23,C$15:C$23)	=RANK.EQ(D23,D$15:D$23)	=RANK.EQ(E23,E$15:E$23)	=RANK.EQ(F23,F$15:F$23)
36	=SUM(A27:A35)	=SUM(B27:B35)	=SUM(C27:C35)	=SUM(D27:D35)	=SUM(E27:E35)	=SUM(F27:F35)

	G	H	I	J	K
26	6	7	8	9	=SUM(B26:J26)
27	=RANK.EQ(G15,G$15:G$23)	=RANK.EQ(H15,H$15:H$23)	=RANK.EQ(I15,I$15:I$23)	=RANK.EQ(J15,J$15:J$23)	=SUM(B27:J27)
28	=RANK.EQ(G16,G$15:G$23)	=RANK.EQ(H16,H$15:H$23)	=RANK.EQ(I16,I$15:I$23)	=RANK.EQ(J16,J$15:J$23)	=SUM(B28:J28)
29	=RANK.EQ(G17,G$15:G$23)	=RANK.EQ(H17,H$15:H$23)	=RANK.EQ(I17,I$15:I$23)	=RANK.EQ(J17,J$15:J$23)	=SUM(B29:J29)
30	=RANK.EQ(G18,G$15:G$23)	=RANK.EQ(H18,H$15:H$23)	=RANK.EQ(I18,I$15:I$23)	=RANK.EQ(J18,J$15:J$23)	=SUM(B30:J30)
31	=RANK.EQ(G19,G$15:G$23)	=RANK.EQ(H19,H$15:H$23)	=RANK.EQ(I19,I$15:I$23)	=RANK.EQ(J19,J$15:J$23)	=SUM(B31:J31)
32	=RANK.EQ(G20,G$15:G$23)	=RANK.EQ(H20,H$15:H$23)	=RANK.EQ(I20,I$15:I$23)	=RANK.EQ(J20,J$15:J$23)	=SUM(B32:J32)
33	=RANK.EQ(G21,G$15:G$23)	=RANK.EQ(H21,H$15:H$23)	=RANK.EQ(I21,I$15:I$23)	=RANK.EQ(J21,J$15:J$23)	=SUM(B33:J33)
34	=RANK.EQ(G22,G$15:G$23)	=RANK.EQ(H22,H$15:H$23)	=RANK.EQ(I22,I$15:I$23)	=RANK.EQ(J22,J$15:J$23)	=SUM(B34:J34)
35	=RANK.EQ(G23,G$15:G$23)	=RANK.EQ(H23,H$15:H$23)	=RANK.EQ(I23,I$15:I$23)	=RANK.EQ(J23,J$15:J$23)	=SUM(B35:J35)
36	=SUM(G27:G35)	=SUM(H27:H35)	=SUM(I27:I35)	=SUM(J27:J35)	Sum

	A	B	C	D	E	F
37	Determining Row Rank					
38		1	2	3	4	5
39	1	=RANK.EQ(B15,$B15:$J15)	=RANK.EQ(C15,$B15:$J15)	=RANK.EQ(D15,$B15:$J15)	=RANK.EQ(E15,$B15:$J15)	=RANK.EQ(F15,$B15:$J15)
40	2	=RANK.EQ(B16,$B16:$J16)	=RANK.EQ(C16,$B16:$J16)	=RANK.EQ(D16,$B16:$J16)	=RANK.EQ(E16,$B16:$J16)	=RANK.EQ(F16,$B16:$J16)
41	3	=RANK.EQ(B17,$B17:$J17)	=RANK.EQ(C17,$B17:$J17)	=RANK.EQ(D17,$B17:$J17)	=RANK.EQ(E17,$B17:$J17)	=RANK.EQ(F17,$B17:$J17)
42	4	=RANK.EQ(B18,$B18:$J18)	=RANK.EQ(C18,$B18:$J18)	=RANK.EQ(D18,$B18:$J18)	=RANK.EQ(E18,$B18:$J18)	=RANK.EQ(F18,$B18:$J18)
43	5	=RANK.EQ(B19,$B19:$J19)	=RANK.EQ(C19,$B19:$J19)	=RANK.EQ(D19,$B19:$J19)	=RANK.EQ(E19,$B19:$J19)	=RANK.EQ(F19,$B19:$J19)
44	6	=RANK.EQ(B20,$B20:$J20)	=RANK.EQ(C20,$B20:$J20)	=RANK.EQ(D20,$B20:$J20)	=RANK.EQ(E20,$B20:$J20)	=RANK.EQ(F20,$B20:$J20)
45	7	=RANK.EQ(B21,$B21:$J21)	=RANK.EQ(C21,$B21:$J21)	=RANK.EQ(D21,$B21:$J21)	=RANK.EQ(E21,$B21:$J21)	=RANK.EQ(F21,$B21:$J21)
46	8	=RANK.EQ(B22,$B22:$J22)	=RANK.EQ(C22,$B22:$J22)	=RANK.EQ(D22,$B22:$J22)	=RANK.EQ(E22,$B22:$J22)	=RANK.EQ(F22,$B22:$J22)
47	9	=RANK.EQ(B23,$B23:$J23)	=RANK.EQ(C23,$B23:$J23)	=RANK.EQ(D23,$B23:$J23)	=RANK.EQ(E23,$B23:$J23)	=RANK.EQ(F23,$B23:$J23)
48	=SUM(A39:A47)	=SUM(B39:B47)	=SUM(C39:C47)	=SUM(D39:D47)	=SUM(E39:E47)	=SUM(F39:F47)

	G	H	I	J	K
38	6	7	8	9	=SUM(B38:J38)
39	=RANK.EQ(G15,$B15:$J15)	=RANK.EQ(H15,$B15:$J15)	=RANK.EQ(I15,$B15:$J15)	=RANK.EQ(J15,$B15:$J15)	=SUM(B39:J39)
40	=RANK.EQ(G16,$B16:$J16)	=RANK.EQ(H16,$B16:$J16)	=RANK.EQ(I16,$B16:$J16)	=RANK.EQ(J16,$B16:$J16)	=SUM(B40:J40)
41	=RANK.EQ(G17,$B17:$J17)	=RANK.EQ(H17,$B17:$J17)	=RANK.EQ(I17,$B17:$J17)	=RANK.EQ(J17,$B17:$J17)	=SUM(B41:J41)
42	=RANK.EQ(G18,$B18:$J18)	=RANK.EQ(H18,$B18:$J18)	=RANK.EQ(I18,$B18:$J18)	=RANK.EQ(J18,$B18:$J18)	=SUM(B42:J42)
43	=RANK.EQ(G19,$B19:$J19)	=RANK.EQ(H19,$B19:$J19)	=RANK.EQ(I19,$B19:$J19)	=RANK.EQ(J19,$B19:$J19)	=SUM(B43:J43)
44	=RANK.EQ(G20,$B20:$J20)	=RANK.EQ(H20,$B20:$J20)	=RANK.EQ(I20,$B20:$J20)	=RANK.EQ(J20,$B20:$J20)	=SUM(B44:J44)
45	=RANK.EQ(G21,$B21:$J21)	=RANK.EQ(H21,$B21:$J21)	=RANK.EQ(I21,$B21:$J21)	=RANK.EQ(J21,$B21:$J21)	=SUM(B45:J45)
46	=RANK.EQ(G22,$B22:$J22)	=RANK.EQ(H22,$B22:$J22)	=RANK.EQ(I22,$B22:$J22)	=RANK.EQ(J22,$B22:$J22)	=SUM(B46:J46)
47	=RANK.EQ(G23,$B23:$J23)	=RANK.EQ(H23,$B23:$J23)	=RANK.EQ(I23,$B23:$J23)	=RANK.EQ(J23,$B23:$J23)	=SUM(B47:J47)
48	=SUM(G39:G47)	=SUM(H39:H47)	=SUM(I39:I47)	=SUM(J39:J47)	Sum

Step 5 – Figure 6 shows the decision variables that are flagged because they do not match the given starting entries. Figure 7 shows the formulas that are used to identify the mismatches.

Figure 6: Decision Variables that Do Not Match Initial Entries

	A	B	C	D	E	F	G	H	I	J
1										
49	Decision Variables that Do Not Match the Initial Entries									
50		1	2	3	4	5	6	7	8	9
51	1	0	0	0	0	0	0	0	0	0
52	2	0	0	0	0	0	0	0	0	0
53	3	0	0	0	0	0	0	0	0	0
54	4	0	1	0	0	0	0	0	0	0
55	5	0	0	0	0	0	0	0	0	0
56	6	0	0	0	0	0	0	0	0	0
57	7	0	0	0	0	0	0	0	0	0
58	8	0	0	0	0	0	0	0	1	0
59	9	0	0	0	0	0	0	0	0	1

Figure 7: Formulas for Flagging Decision Variables that Do Not Match Initial Entries

	A	B	C	D	E
1					
49	Decision Variables that Do Not Match the Initial Entries				
50		1	2	3	4
51	1	=IF(B3=0,0,IF(B15=B3,0,1))	=IF(C3=0,0,IF(C15=C3,0,1))	=IF(D3=0,0,IF(D15=D3,0,1))	=IF(E3=0,0,IF(E15=E3,0,1))
52	2	=IF(B4=0,0,IF(B16=B4,0,1))	=IF(C4=0,0,IF(C16=C4,0,1))	=IF(D4=0,0,IF(D16=D4,0,1))	=IF(E4=0,0,IF(E16=E4,0,1))
53	3	=IF(B5=0,0,IF(B17=B5,0,1))	=IF(C5=0,0,IF(C17=C5,0,1))	=IF(D5=0,0,IF(D17=D5,0,1))	=IF(E5=0,0,IF(E17=E5,0,1))
54	4	=IF(B6=0,0,IF(B18=B6,0,1))	=IF(C6=0,0,IF(C18=C6,0,1))	=IF(D6=0,0,IF(D18=D6,0,1))	=IF(E6=0,0,IF(E18=E6,0,1))
55	5	=IF(B7=0,0,IF(B19=B7,0,1))	=IF(C7=0,0,IF(C19=C7,0,1))	=IF(D7=0,0,IF(D19=D7,0,1))	=IF(E7=0,0,IF(E19=E7,0,1))
56	6	=IF(B8=0,0,IF(B20=B8,0,1))	=IF(C8=0,0,IF(C20=C8,0,1))	=IF(D8=0,0,IF(D20=D8,0,1))	=IF(E8=0,0,IF(E20=E8,0,1))
57	7	=IF(B9=0,0,IF(B21=B9,0,1))	=IF(C9=0,0,IF(C21=C9,0,1))	=IF(D9=0,0,IF(D21=D9,0,1))	=IF(E9=0,0,IF(E21=E9,0,1))
58	8	=IF(B10=0,0,IF(B22=B10,0,1))	=IF(C10=0,0,IF(C22=C10,0,1))	=IF(D10=0,0,IF(D22=D10,0,1))	=IF(E10=0,0,IF(E22=E10,0,1))
59	9	=IF(B11=0,0,IF(B23=B11,0,1))	=IF(C11=0,0,IF(C23=C11,0,1))	=IF(D11=0,0,IF(D23=D11,0,1))	=IF(E11=0,0,IF(E23=E11,0,1))

	F	G	H	I	J
50	5	6	7	8	9
51	=IF(F3=0,0,IF(F15=F3,0,1))	=IF(G3=0,0,IF(G15=G3,0,1))	=IF(H3=0,0,IF(H15=H3,0,1))	=IF(I3=0,0,IF(I15=I3,0,1))	=IF(J3=0,0,IF(J15=J3,0,1))
52	=IF(F4=0,0,IF(F16=F4,0,1))	=IF(G4=0,0,IF(G16=G4,0,1))	=IF(H4=0,0,IF(H16=H4,0,1))	=IF(I4=0,0,IF(I16=I4,0,1))	=IF(J4=0,0,IF(J16=J4,0,1))
53	=IF(F5=0,0,IF(F17=F5,0,1))	=IF(G5=0,0,IF(G17=G5,0,1))	=IF(H5=0,0,IF(H17=H5,0,1))	=IF(I5=0,0,IF(I17=I5,0,1))	=IF(J5=0,0,IF(J17=J5,0,1))
54	=IF(F6=0,0,IF(F18=F6,0,1))	=IF(G6=0,0,IF(G18=G6,0,1))	=IF(H6=0,0,IF(H18=H6,0,1))	=IF(I6=0,0,IF(I18=I6,0,1))	=IF(J6=0,0,IF(J18=J6,0,1))
55	=IF(F7=0,0,IF(F19=F7,0,1))	=IF(G7=0,0,IF(G19=G7,0,1))	=IF(H7=0,0,IF(H19=H7,0,1))	=IF(I7=0,0,IF(I19=I7,0,1))	=IF(J7=0,0,IF(J19=J7,0,1))
56	=IF(F8=0,0,IF(F20=F8,0,1))	=IF(G8=0,0,IF(G20=G8,0,1))	=IF(H8=0,0,IF(H20=H8,0,1))	=IF(I8=0,0,IF(I20=I8,0,1))	=IF(J8=0,0,IF(J20=J8,0,1))
57	=IF(F9=0,0,IF(F21=F9,0,1))	=IF(G9=0,0,IF(G21=G9,0,1))	=IF(H9=0,0,IF(H21=H9,0,1))	=IF(I9=0,0,IF(I21=I9,0,1))	=IF(J9=0,0,IF(J21=J9,0,1))
58	=IF(F10=0,0,IF(F22=F10,0,1))	=IF(G10=0,0,IF(G22=G10,0,1))	=IF(H10=0,0,IF(H22=H10,0,1))	=IF(I10=0,0,IF(I22=I10,0,1))	=IF(J10=0,0,IF(J22=J10,0,1))
59	=IF(F11=0,0,IF(F23=F11,0,1))	=IF(G11=0,0,IF(G23=G11,0,1))	=IF(H11=0,0,IF(H23=H11,0,1))	=IF(I11=0,0,IF(I23=I11,0,1))	=IF(J11=0,0,IF(J23=J11,0,1))

Step 6 – Compute penalties for each row that has a duplicate rank, for each column that has a duplicate rank, and for decision variables not matching the fixed initial entry. A penalty of 1000 is assigned for each violation. Figure 8 shows the results and figure 9 shows the formulas.

Figure 8: Penalties

	K	L	M
1			
2		Penalty/violation	1000
3			
4		Column rank violation	8
5		Row rank violation	8
6		Original value violations	3
7		Sum of violations	19
8		Total penalty	19000

Figure 9: Formulas for Penalty Calculations

	K	L	M
1			
2		Penalty/violation	1000
3			
4		Column rank violation	=A35-COUNTIF(B36:J36,"=45")
5		Row rank violation	=J38-COUNTIF(K39:K47,"=45")
6		Original value violations	=SUM(B51:J59)
7		Sum of violations	=SUM(M4:M6)
8		Total penalty	=M7*M2

Step 7 – Figure 8 shows the "alldifferent" constraints imposed on each submatrix block as part of the Solver settings. Note also that the objective function is the total penalty and that the solving method is Evolutionary Solver.

Figure 8: Solver Settings

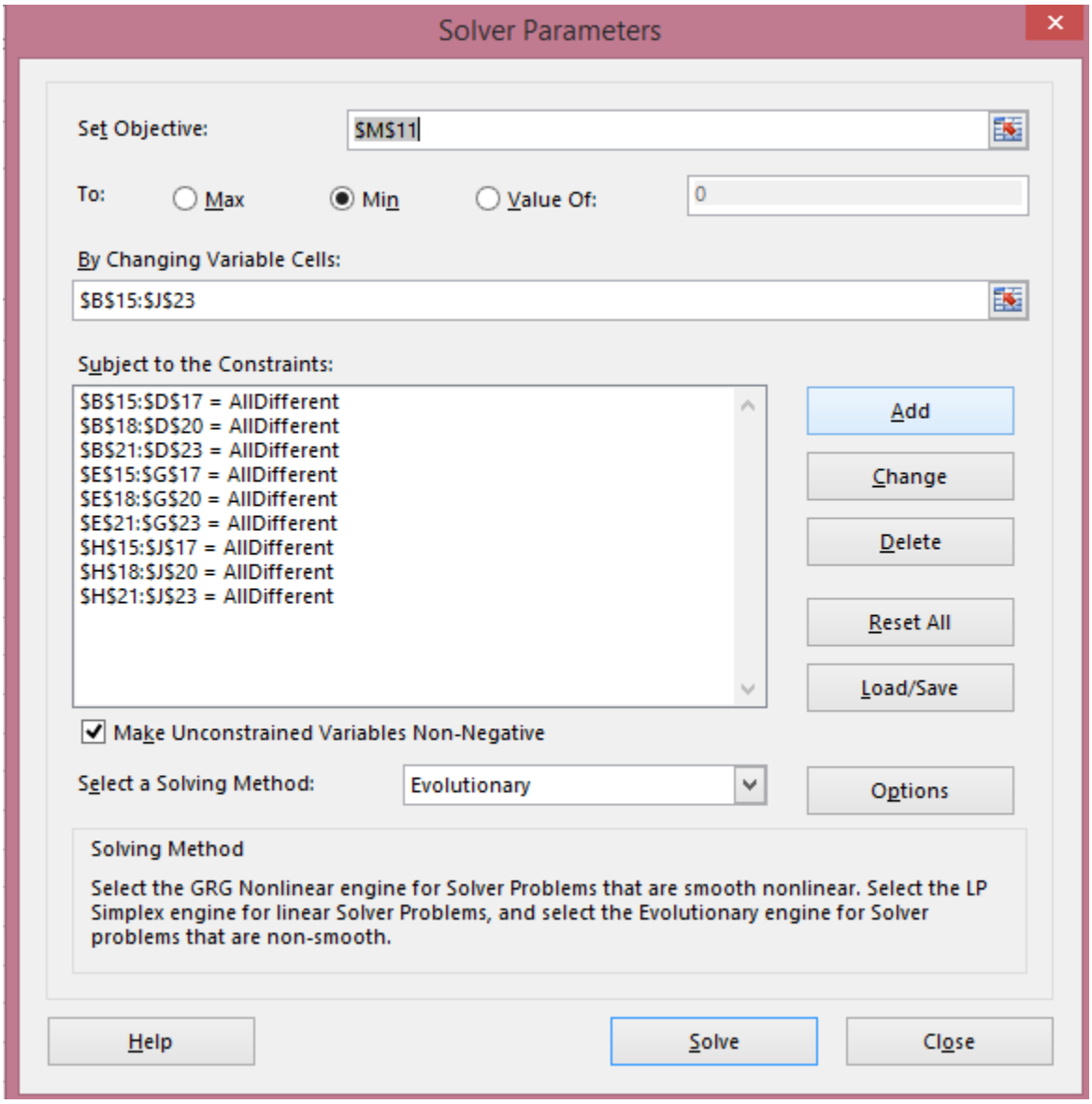
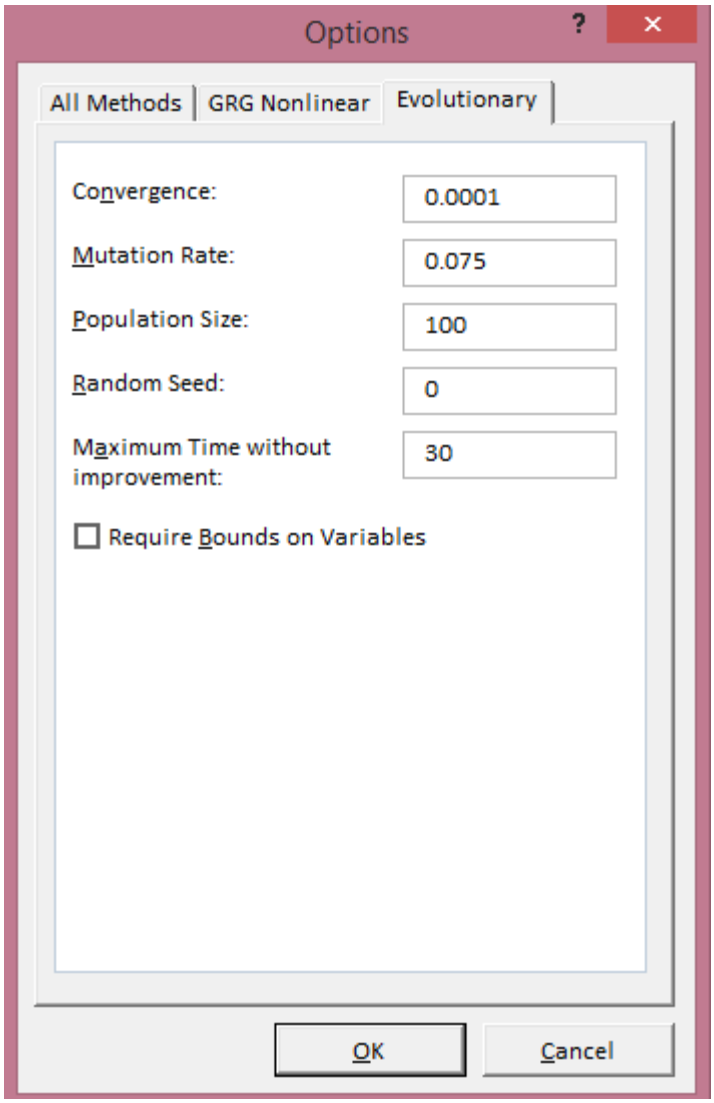


Figure 9 shows the default settings for Evolutionary Solver options.

Figure 9: Evolutionary Solver Options



Step 8 – Solve using Evolutionary Solver. If the analysis stops with an objective function value (i.e. total penalty) greater than zero, restart either with the same settings or with alternate values for the mutation rate and/or population size to aid the convergence process. A higher mutation rate generates a more diverse set of possible solutions, while a larger population size generates more possible solutions.

SUMMARY

For most people, the popular Sudoku game entertains. For mathematical modelers though, it challenges us to think of innovative and efficient ways to model and solve the puzzle. The proposed Sudoku model requires fewer variables than most of its IP and BILP predecessors and introduces the use of the RANK function to help ensure unique entries in each row and column. It uses a genetic search approach and can be implemented in the standard version of Excel, making this a teaching tool available to a wide audience at minimal cost.

REFERENCES

Halazun, H. (2012, January 23). "Brain Games May Help Thwart Alzheimer's: Study", Retrieved July 30, 2015 from http://abcnews.go.com/blogs/health/2012/01/23/brain-games-may-help-thwart-alzheimers-study

Bartlett, A., Chartier, T., Langville, A., & Rankin, T. (2008). "An Integer Programming Model for the Sudoku Problem", *Journal of Online Mathematics and its Applications*, Vol. 8.

Chond, M. (2005). "Classroom Exercises in IP Modeling: Sudoku and the Log Pile", *INFORMS Transactions on Education*, Vol. 5 No. 2, pp. 77-79.

Delahaye, J. (2006). "The Science behind SUDOKU", *Scientific American*, Vol. *294* No. 6, pp. 80-87.

Friesen, D., Harmel, B., & Patterson, M. C. (2013). "A Spreadsheet Optimization Model for Solving Sudoku Problems". *Business Management Dynamics*, Vol. 2 No. 9, pp 15-22.

Gen, M., & Cheng, R. (2000). *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons.

Goldberg, D.E. (1989). *Genetic Algorithms in Search. Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Koch, T. (2005). "Rapid Mathematical Programming or How to Solve Sudoku Puzzles in a Few Seconds", *Operations Research Proceedings*, Konrad-Zuse-Zentrum fur Informationstechnik Berlin, ZIB Report 05-51.

Rasmussen, R. & Weiss, H. (2007). "Advanced Lessons on the Craft of Optimization Modeling Based on Modeling Sudoku in Excel", *INFORMS Transactions on Education*, Vol. 7 No. 3, pp. 228-237.

Smith, S. L., & Cagnoni, S. (Eds.). (2011). *Genetic and Evolutionary Computation: Medical Applications*. John Wiley & Sons.

Walter, C. (2003). "Digital Darwin", *Technology Review*, Vol. 106 No. 9, p. 24.

Weiss, H. & Rasmussen, R. (2007). "Lessons from Modeling Sudoku in Excel", *INFORMS Transactions on Education*, Vol. 7 No. 2, pp. 178-184.