An integer programming model for the sudoku problem

Article · April 2008

CITATIONS
30

READS
2,229

4 authors, including:

Amy Langville
College of Charleston
46 PUBLICATIONS **4,079** CITATIONS

SEE PROFILE

An Integer Programming Model for the Sudoku Problem

Andrew C. Bartlett* Timothy P. Chartier† Amy N. Langville‡
Timothy D. Rankin§

May 3, 2008

Abstract

Sudoku is the recent craze in logic puzzles. Players must fill in an $n \times n$ matrix, which contains some given entries, so that each row, column, and $m \times m$ submatrix contains each integer 1 through n exactly once. Two issues associated with these puzzles interest us mathematically: puzzle solution and puzzle creation. A Sudoku puzzle can be solved by creating a feasibility problem where the goal is to find at least one feasible solution to the puzzle. We present a binary integer linear program to solve this feasibility problem. Further, such an approach is extended to variations on the traditional Sudoku puzzle. In addition, we speculate as to how Sudoku puzzles are created, and provide several theorems for generating many new puzzles from one given original puzzle. Exercises and challenges problems that use principles from optimization, combinatorics, linear algebra, and computer science are presented for students.

1 Introduction

Sudoku is a logic-based puzzle that first appeared in the U.S. under the title "Number Place" in 1979 in the magazine *Dell Pencil Puzzles & Word Games* [6]. The game was designed by Howard Garns, an architect who, upon retirement, turned to puzzle creation. In the 1980s, the game grew in popularity in Japan and was renamed by publisher Nikoli to "suji wa dokushin ni kagiru," which translates as the "the digits must remain single." This was eventually shortened to "sudoku" or "single number."

By 1997 an entrepreneur named Wayne Gould saw the financial potential available in the game. Gould spent six years refining his computer program so that it could quickly generate puzzles of varying levels of difficulty. In November 2004, Gould convinced *The Times* of London to print a puzzle. From there the popularity of the puzzle spread until now they commonly appear in a wide variety of newspapers and magazines. Interestingly, Gould does not charge newspapers for his puzzles, but they must include the Web address `http://www.sudoku.com` where his Sudoku program can be downloaded (for a free trial version and a fee for permanent use).

Sudoku most commonly appears in its 9×9 matrix form. The rules are simple: fill in the matrix so that every row, column, and 3×3 submatrix contains the digits 1 through 9 exactly once. Each puzzle appears with a certain number of *givens*. The number and location of these determine the game's level of difficulty. Figure 1 is an example of a 9×9 Sudoku puzzle.

*Department of Mathematics, College of Charleston, Charleston, SC, USA, `abartle1@uga.edu`
†Department of Mathematics, Davidson College, Davidson, NC, USA, `tichartier@davidson.edu.`
‡Department of Mathematics, College of Charleston, Charleston, SC, USA, `langvillea@cofc.edu.` This work was supported in part by the National Science Foundation under NSF grant CAREER-0546622.
§Department of Mathematics, Davidson College, Davidson, NC, USA, `timothy.rankin@duke.edu.`

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

Figure 1: An example Sudoku puzzle

This puzzle idea can accommodate games of other sizes. Of course, a 4×4 puzzle would be easier and a 16×16 puzzle harder. In general, any $n \times n$ game can be created, where $n = m^2$ and m is any positive integer. There are numerous other variants of the game; see [9, 2, 8].

Sudoku puzzles elicit the following two interesting mathematical questions:

- How can these puzzles be solved mathematically?
- What mathematical techniques can be used to create these puzzles?

In the following sections, we explore these questions. We present a binary integer linear program to solve this feasibility problem. Further, such an approach is extended to variations on the traditional Sudoku puzzle. In addition, we speculate as to how Sudoku puzzles are created, and provide several theorems for generating many new puzzles from one given original puzzle. Exercises and challenge problems that use principles from optimization, combinatorics, linear algebra, and computer science are presented for students. Answers to the exercises are contained at the conclusion of the article.

2 Question 1: Solving the Puzzle

2.1 The Mathematical Model

Can we solve Sudoku puzzles mathematically? That is, can a mathematical process find the solution for us? There are two approaches that we could take to this end:

Solution 1 Write a computer program to execute the logic that a person uses to solve a Sudoku puzzle.

Issue What if the programmer has yet to learn some level of logic that solves more difficult problems? This type of solution can necessitate a certain level of proficiency at solving such puzzles.

Solution 2 Formulate the problem in a way such that a mathematical algorithm can be applied to find the exact solution, if it exists.

Let us mathematically model the Sudoku puzzle found in Figure 1 as a linear program. More specifically, we will formulate a binary integer program (BILP) for general $n \times n$ puzzles. Once the program is developed to solve the BILP for Figure 1, it can be easily adapted to solve *any* Sudoku puzzle.

To begin, we define our decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if element } (i,j) \text{ of the } n \times n \text{ Sudoku matrix contains the integer } k \\ 0, & \text{otherwise.} \end{cases}$$

When the values of the decision variables are determined, we will know whether each integer k ($1 \leq k \leq 9$) appears in each element (i,j) of the $n \times n$ Sudoku matrix. That is, the solution to the corresponding Sudoku puzzle will be defined.

We now turn to the ever important objective function and set of constraints. Notice how the constraints require only a knowledge of the rules of Sudoku puzzles (with the addition of one implicit constraint which is constraint (4) below) and do not require a proficiency with the logic necessary to solve such puzzles by hand. A BILP formulation suitable for Sudoku puzzles is as follows:

$$\begin{aligned}
\min \quad & \mathbf{0}^T\mathbf{x} \\
s.t. \quad & \sum_{i=1}^{n} x_{ijk} = 1, \quad j{=}1{:}n,\ k{=}1{:}n \quad \text{(only one } k \text{ in each column)} && (1)\\
& \sum_{j=1}^{n} x_{ijk} = 1, \quad i{=}1{:}n,\ k{=}1{:}n \quad \text{(only one } k \text{ in each row)} && (2)\\
& \sum_{j=mq-m+1}^{mq} \sum_{i=mp-m+1}^{mp} x_{ijk} = 1, \quad k{=}1{:}n,\ p{=}1{:}m,\ q{=}1{:}m \quad \text{(only one } k \text{ in each submatrix)} && (3)\\
& \sum_{k=1}^{n} x_{ijk} = 1 \quad i{=}1{:}n,\ j{=}1{:}n \quad \text{(every position in matrix must be filled)} && (4)\\
& x_{ijk} = 1 \quad \forall\,(i,j,k) \in G \quad \text{(given elements } G \text{ in matrix are set ``on'')} && (5)\\
& x_{ijk} \in \{0,1\} && (6)
\end{aligned}$$

In this article, Matlab's `bintprog` command will solve this BILP. (Note that the `bintprog` command is available with Matlab's Optimization Toolbox.) Other software could also solve this BILP; examples include Excel [15], Xpress-Mosel [3], and SAS [4].

The Sudoku problem is actually a *satisfiability* problem or feasibility problem (also known as a constraint programming problem). The goal is to find at least one feasible solution satisfying constraints (1)-(5). In theory, because this is a satisfiability problem, no objective function is needed. However, in order to use Matlab's `bintprog` command, an objective function is required. We chose $\mathbf{0}^T$ as the vector of objective function coefficients.

EXERCISE 1*: Prove or disprove the following statement. Any uniform vector would work as the objective function of coefficients in the above BILP.*

2.2 Testing the Model in Matlab

Executing the above BILP model, a Matlab program named `sudoku.m` can solve this puzzle. This program can be downloaded from `http://aristotle.davidson.edu/chartier/sudoku/sudoku.m`. We will see that the code can be easily adapted to solve other puzzles. The program requires that the user input the elements, called givens, provided at the start of the puzzle. The matrix of givens for our example

is

$$\texttt{Givens} = \begin{pmatrix} 1 & 8 & 2 \\ 2 & 2 & 2 \\ 2 & 7 & 5 \\ 3 & 3 & 7 \\ 3 & 6 & 3 \\ 3 & 7 & 4 \\ & \vdots & \\ 8 & 3 & 9 \\ 8 & 8 & 1 \\ 9 & 2 & 1 \end{pmatrix},$$

where each line in this matrix gives the row index, column index, and integer value of each element appearing in the initial Sudoku matrix. The m-file `sudoku.m` calls the built-in Matlab function, `bintprog` which is available through the Optimization Toolbox. On a Mac G5 with dual 2.7 GHz processors and 8 GB of memory, our program found the feasible solution, presented in Figure 2, in 16.08 seconds. Admittedly, this approach will generally be slower than a technique that executes the logical steps a person uses to solve a Sudoku puzzle. One such implementation is available at [14] and is based on the Mathematica implementation by Simons [12].

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

Figure 2: Solution for example Sudoku puzzle

2.3 The Integer Programming Technique

Matlab's `bintprog` uses the classic branch and bound method to determine the optimal solution. The branch and bound method identifies a solution to a BILP by solving a series of LP-relaxation problems, by replacing (or relaxing) the binary constraint $x_{ijk} = 1$ or 0 to the weaker constraint $0 \leq x_{ijk} \leq 1$. Initially, all the variables adhere to the relaxed constraint $0 \leq x_{ijk} \leq 1$. At each branching step, the branch and bound method chooses a variable x_j whose current value for the LP-relaxation problem is not integral and adds a constraint $x_j = 0$ for one branch and $x_j = 1$ for another branch. For our problem, a branch of the search tree is discontinued if the LP-relaxation problem at the current node is infeasible. In our BILP formulation, the bounding of a branch and bound algorithm will not play a role given that our objective function equals 0 for any feasible solution. For additional information on the branch and bound algorithm, a reader is encouraged to reference the MATLAB documentation for the `bintprog` function.

Given the structure of a branch and bound algorithm, each additional constraint enforced by a given element of the opening Sudoku matrix greatly reduces the search space of the branch and bound procedure

and makes the problem tractable. It's easy to understand why each given element of the matrix helps the branch and bound procedure. For example, because the (1,3) element in our 9×9 Sudoku example from Section 1 is 7, this means that $x_{137} = 1$, which implies that

- $x_{13k} = 0, \quad \forall k \neq 7$ (there can be no other integer in the (1,3) position)
- $x_{i37} = 0, \quad \forall i \neq 1$ (there can be no other 7 in the third column)
- $x_{1j7} = 0, \quad \forall j \neq 3$ (there can be no other 7 in the first row)
- $x_{217} = 0, x_{227} = 0, x_{317} = 0, x_{327} = 0$ (there can be no other 7 in the (1,1) submatrix)

Thus, for each given element, at most 29 (in general, $3(n-1)+(m-1)^2+1)$) of the $n^3 = 729$ decision variables are determined.

EXERCISE 2*: Why can't we say 29 exactly?*

Without considering the constraints, the total search space for our model is 2^{n^3} because each decision variable can take on two values, and there are n^3 decision variables. For $n = 9$, $2^{729} \approx 2.82 \times 10^{219}$. Fortunately, constraint (5) supplies many givens (at least 17, see Section 4.1), each of which drastically reduces the search space. Suppose only 9 givens are provided, where each given appears in its own row, column, and submatrix (so that there is no double counting), then $9 * 29 = 261$ of the 729 decision variables are determined, leaving a search space of $2^{468} \approx 7.62 \times 10^{140}$. Of course, considering the remaining givens further reduces the search space, and makes the problem tractable for the IP technique.

2.4 One vs. Many Solutions

Our BILP finds at least one feasible solution, if it exists. While Sudoku puzzles are defined as having a unique solution, the authors occasionally discovered puzzles in which this constraint was not met. Such faulty puzzles have multiple solutions. In such cases, we discovered that our solution while correct was distinct from the solution provided by the puzzle creators.

Example The $n = 4$ Sudoku puzzle below has two solutions.

$$\text{Puzzle} = \left(\begin{array}{cc|cc} & 2 & & \\ 3 & & & \\ \hline & & & \\ & 3 & & \end{array}\right), \quad \text{Solution} = \left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \\ \hline 2 & 1 & 4 & 3 \\ 4 & 3 & 1 & 2 \end{array}\right) \text{ or } \left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ \hline 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \end{array}\right).$$

EXERCISE 3*: Find all solutions to the Sudoku puzzle below. (Hint: there are 3.)*

$$\text{Puzzle} = \left(\begin{array}{cc|cc} & 2 & & 4 \\ 3 & & & \\ \hline & & & 4 \\ & & & \end{array}\right).$$

CHALLENGE*: Create a program to find all solutions to a Sudoku puzzle. One way to do this is to modify the* `sudoku.m` *file to find all optimal solutions to our BILP. (Yato and Seta have shown that this problem is something they define as ASP-complete [16], which implies NP-completeness and demonstrates the challenge and complexity of the all solutions problem.)*

CHALLENGE*: Generally, the values of a group of cells in the puzzle can be determined from the givens in the initial matrix using only very simple logic. Identifying such values essentially increases the number of*

givens for the puzzle and generally aids in the efficiency of the BILP. Write an algorithm that determines the values of cells from the givens (again using only simply logic on the placement of the givens) and insert this as a preprocessing step in the `sudoku.m` *file so that the matrix of* `Givens` *is appended. Compare the computation time required with and without this preprocessing step.*

CHALLENGE*: Although it is a fun application for classes that study optimization, it is not necessary to use a BILP or even optimization techniques to solve Sudoku puzzles. Can you use other mathematical or computational techniques to solve these puzzles? Write your own algorithm.*

EXERCISE 4*: Can you create a* 4×4 *puzzle with a unique solution using only 5 givens? 4 givens? 3 givens? Which is more important, the position or the value of the givens? Formulate a proposition that summarizes your findings. For example, "any* 4×4 *puzzle whose 5 givens satisfy...has a unique solution."*

CHALLENGE*: Prove the proposition(s) you formulated in Exercise 4.*

3 Solving Variations on Sudoku Puzzles

Given the popularity of Sudoku puzzles, variations on the traditional puzzle have emerged. One colorful, extensive source of puzzles is authored by Riley and Taalman in [9]. In this section, we formulate a BILP for each of several puzzles that are contained in this book. Each puzzle contains the same set of constraints on a solution as a traditional Sudoku but then also imposes one or more additional constraints. The authors thank Riley and Taalman for creating the puzzles in this paper.

All constraints needed for a traditional Sudoku puzzle would also be required for any puzzle which has the same set (plus additional) constraints as a traditional puzzle. Therefore, we are faced with creating the additional constraints that capture the additional requirements for the puzzle under consideration. This will be the goal for each puzzle below.

3.1 Sudoku X

The "Sudoku X" puzzle is like the standard puzzle, with an extra requirement: the two long diagonals of the board must also contain each digit from 1 to 9 exactly once. Thus, any solution to the Sudoku X puzzle is also a solution to the standard Sudoku puzzle, but the converse is not the case. An example of a Sudoku X puzzle is given in Figure 3, where each long diagonal contains a dotted green line.

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

Figure 3: An example Sudoku X puzzle

The only additional requirements of a Sudoku X puzzle are that the two long diagonals have exactly one of each digit. This results in 18 additional constraints (two diagonals with nine digits each).

To capture the requirement for the positive diagonal, we add the nine constraints

$$\sum_{r=1}^{9} x_{rrk} = 1, \;\; k = 1:9.$$

That is, each number on the positive diagonal appears exactly once. Similarly, for the anti-diagonal, the following set of nine constraints are added

$$\sum_{r=1}^{9} x_{r(10-r)k} = 1, \;\; k = 1:9.$$

As a result of these two additional constraints, every digit must appear exactly once on each diagonal.

3.2 Four Square Sudoku

The "Four Square" Sudoku is again a standard puzzle but with an extra requirement. There are four shaded 3×3 regions on the Sudoku board, and in addition to the requirements of the standard Sudoku, each shaded region must also contain each digit from 1 to 9 exactly once. A sample Four Squares puzzle is given in Figure 4.

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

Figure 4: An example Four Square Sudoku puzzle

In addition to the standard Sudoku constraints, the Four Square Sudoku puzzle requires that each shaded 3×3 square contains each digit exactly once. This requirement results in 36 constraints on top of the standard constraints (4 squares, 9 digits). The constraints themselves look similar to the other constraints on the 3×3 subgrids:

$$\sum_{r=i}^{i+2} \sum_{c=j}^{j+2} x_{rck} = 1, \;\; i = 2, 6; \;\; j = 2, 6; \;\; \text{and} \;\; k = 1:9$$

Again, the choices of i and j determine the shaded square in question.

3.3 Four Pyramids Sudoku

Like the Four Square variant, the "Four Pyramids" Sudoku is a standard Sudoku puzzle with four shaded regions, where one must ensure that each shaded region contains exactly one of each digit from 1

to 9 (in addition to satisfying the requirements of the standard Sudoku). As one can see in the example below, the shaded regions are shaped somewhat like pyramids, giving this variant its name.

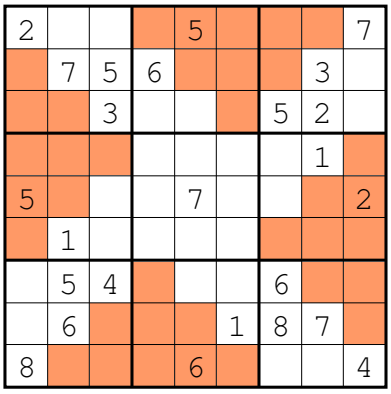


Figure 5: An example Four Pyramids Sudoku puzzle

Like the previous variant, the Four Pyramids variant adds 36 additional constraints to the standard ones. Each of the four pyramid-shaped shaded regions must contain each digit exactly once:

$$\sum_{r=1}^{3}\sum_{c=3+r}^{9-r} x_{rck} = 1,\ k = 1:9$$

$$\sum_{c=1}^{3}\sum_{r=1+c}^{7-c} x_{rck} = 1,\ k = 1:9$$

$$\sum_{r=7}^{9}\sum_{c=11-r}^{r-3} x_{rck} = 1,\ k = 1:9$$

$$\sum_{c=7}^{9}\sum_{r=13-c}^{c-1} x_{rck} = 1,\ k = 1:9$$

The four sums correspond to the four shaded regions in counterclockwise order, starting with the top one. For example, in the first sum, when $r = 1$, c varies from 4 to 8, when $r = 2$, c goes from 5 to 7, and when $r = 3$, c only takes the value of 6.

3.4 Other variations

Other variants on a traditional Sudoku puzzle exist. We present two here as they will be used in the exercises below.

Position Sudoku For each element in a "Position Sudoku" tableau, not only does one need to take into account the 3×3 subgrid within which the element lies but also the element's position within the 3×3 subgrid. In addition to satisfying the standard Sudoku requirements, a solution to the Position Sudoku must be such that exactly one of each digit from 1 to 9 is contained in the top left squares of all nine 3×3 subgrids, exactly one of each digit is contained in the top middle squares of all subgrids, and so forth. In the sample puzzle found in Figure 6 (a), the set of squares on the board sharing any given color must contain each digit exactly once.

Three Magic Sudoku Unlike the previous variants, in a "Three Magic Sudoku," the actual value of the digits is important, not just the fact that they are distinct. In a solution to a Three Magic Sudoku puzzle, along with satisfying the requirements of the standard Sudoku, each shaded 3×3 subgrid must also be a "magic square," where the three numbers in each column and in each row add up to the same number. The puzzle depicted in Figure 6 (b) is an example of a Three Magic Sudoku, but it is possible to create puzzles involving any number of magic squares, including the possibility of all nine subgrids being magic squares!

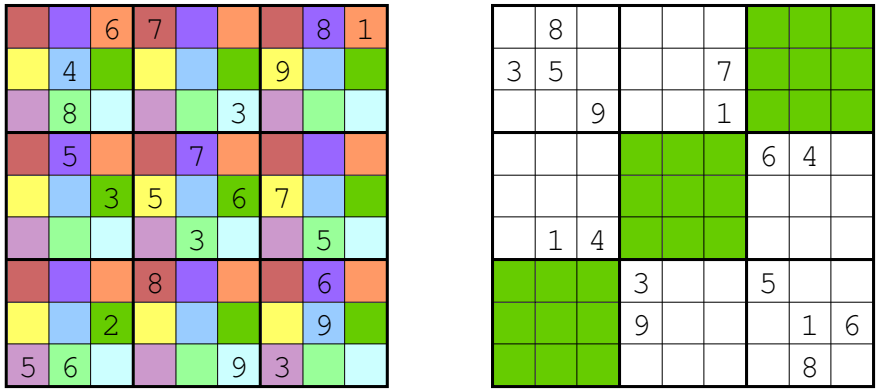


Figure 6: Example Position Sudoku (a) and Three Magic Sudoku (b) puzzles

EXERCISE 5*: Formulate a BILP to solve the Position Sudoku puzzle in Figure 6 (a). Further, adapt the Matlab code* `sudoku.m` *to solve a Position Sudoku.*

EXERCISE 6*: Formulate a BILP to solve the Three Magic Sudoku puzzle in Figure 6 (b). Further, adapt the Matlab code* `sudoku.m` *to solve a Three Magic Sudoku.*

CHALLENGE*: Formulate a BILP for another variation of the traditional Sudoku puzzle. You can find such variations in [9] or online sources such as [2, 8].*

WARNING*: Solving some of the puzzles as BILPs in Matlab may result in long solution times. You may wish to add a preprocessing step that identifies entries which can be determined with very simple logic from the givens in the initial matrix.*

4 Question 2: Creating the Puzzles

We now turn to the question: how might Sudoku puzzles be created? Should our Sudoku puzzle have a unique solution? According to [7], the answer is yes. A simple search on your favorite search engine will also show that many web sites related to puzzle design agree. However, armed with the BILP from this paper, one can occasionally verify, for instance with the Matlab program `sudoku.m`, that not all Sudoku puzzles that are posted have unique solutions. One can stumble across such examples when the BILP's computed solution is distinct from the solution provided by the puzzle creators.

4.1 Creating Puzzles by Brute Force

A first approach to creating Sudoku puzzles is to use brute force. One simple idea fills each element of the 9×9 matrix with a randomly chosen integer from 1 to 9, then checks to see if the resulting matrix satisfies the three Sudoku properties concerning rows, columns, and submatrices. This approach creates

$9^{81} \approx 1.97 \times 10^{77}$ different matrices that require checking. Just how many of these would satisfy the Sudoku properties? In other words, how many feasible 9×9 Sudoku matrices are there (i.e., matrices satisfying constraints (1)-(4) of the BILP of Section 2.1)?

First, note that a 9×9 Latin square consists of sets of the numbers 1 to 9 arranged in such a way that no row or column contains the same number twice. Therefore, every Sudoku puzzle is a special case of a 9×9 Latin square of which there are $5,524,751,496,156,892,842,531,225,600 \approx 5.525 \times 10^{27}$ [1]. How many of these Latin squares are Sudoku matrices?

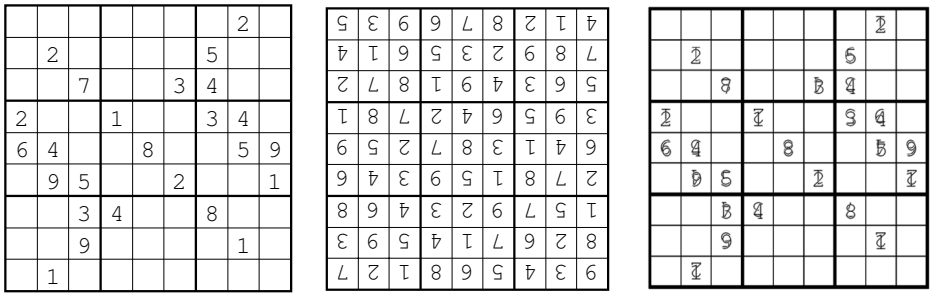
The answer to this question was provided by Felgenhauer and Jarvis in 2005. The number of Sudoku matrices for the standard 9×9 game was calculated to be $6,670,903,752,021,072,936,960 \approx 6.67 \times 10^{21}$ [5]. This number is equal to $9! \times 72^2 \times 2^7 \times 27,704,267,971$, the last factor being prime. The result was derived through logic and brute force computation. (Note, only about .00012% of 9×9 Latin squares are valid Sudoku puzzles.) Later Russell and Jarvis [11] showed that when symmetries were taken into account, there were, of course, many fewer solutions; 5,472,730,538 to be exact.

Given that Sudoku matrices can be created by a brute force technique, assume we stop as soon as we find one. With a full Sudoku matrix in hand, we could then simply omit entries to create a puzzle. At which point, the question becomes how to do the omitting so that a proper Sudoku puzzle results. Specifically, we pose the following mathematical questions.

1. What is the minimum number of givens required to create a puzzle with a unique solution?
2. How does this relate to the positions of the givens?
3. Given a puzzle with a unique solution, is there a way to create other related puzzles with unique solutions?

A literature review leads to some answers.

1. In general, the problem of the minimum number of givens required to create a puzzle with a unique solution is unsolved—at least, theoretically. Experimentally, however, much progress has been made. Algebraic graph theorist Gordon Royle has created a solvable 9×9 puzzle with as few as 17 givens. (In fact, he has found 35,396 distinct such puzzles [10].) To date, no puzzle with 16 or less givens, which in turn produces a unique solution, has been discovered. Note that Nikoli, the Japanese publisher mentioned in Section 1, added two conditions to the puzzles. First, no more than 30 numbers could be given at the start with more numbers generally leading to an easier puzzle. The second rule will be addressed below in the comment on the second question.



(a) (b) (c)

Figure 7: The publisher Nikoli set the rule that Sudoku puzzles should have symmetry as demonstrated in the images above. The puzzle in (a) is the same puzzle as Figure 1. The image in (b) is a 180° rotation of the puzzle in (a). Finally, in (c) we see the images in (a) and (b) placed on top of each other.

2. This question is very difficult to answer. Clearly, it is not solely the number of givens that determines a puzzle's difficulty. The position and values of the givens must also be considered. This question also leads to Nikoli's second rule for puzzles, where the first was addressed above. According to this rule, the pattern of digits must be symmetric. We see such symmetry in Figure 1 where the matrix in (a) is the puzzle from Figure 1 and b) is the same puzzle rotated 180^o. The image in Figure 1 (c) is Figure 1 (a) and (b) placed on top of each other. Note that the open spaces are the same for all three images. While not all Sudoku puzzles adhere to this constraint, many do. The reader may wish to verify that all the puzzles contained in this paper adhere to such symmetry. For these reasons, question two extends beyond the scope of this paper.

3. We provide several answers to this question in the next section.

4.2 Creating New Puzzles from Old Puzzles

Our goal in this section is to create as many new Sudoku puzzles $\bar{\mathbf{S}}$ from one original puzzle $\mathbf{S}$. A new Sudoku matrix is simply one such that $\bar{\mathbf{S}} - \mathbf{S} \neq \mathbf{0}$.

Definition 1 *A square matrix* $\mathbf{S}$ *is a* ***Sudoku matrix****, if the following four conditions hold.*

1. *The order of the matrix* n *is such that* $n = m^2$*, where* m *is any positive integer.*
2. *Every row in* $\mathbf{S}$ *uses the integers 1 through* n *exactly once.*
3. *Every column in* $\mathbf{S}$ *uses the integers 1 through* n *exactly once.*
4. *Every submatrix in* $\mathbf{S}$ *uses the integers 1 through* n *exactly once.*

Theorem 4.1 *If* $\mathbf{S}$ *is a Sudoku matrix, then* $\mathbf{S}^T$ *is also a Sudoku matrix.*

Proof. Transposition interchanges the rows and columns of a matrix, so that $[\mathbf{S}^T]_{ij} = [\mathbf{S}]_{ji}$. Since, $\mathbf{S}$ is $n \times n$, $\mathbf{S}^T$ is $n \times n$, and clearly Property 1 of the Sudoku matrix definition is satisfied. Property 2 (3) is satisfied for $\mathbf{S}^T$ by virtue of the fact that Property 3 (2) is satisfied by $\mathbf{S}$. It remains to show that $\mathbf{S}^T$ satisfies Property 4. Without loss of generality (w.l.o.g.) , let $m = 2$. Then in block form, where each block is 2×2,

$$\mathbf{S} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{pmatrix}, \quad \text{and} \quad \mathbf{S}^T = \begin{pmatrix} \mathbf{S}_{11}^T & \mathbf{S}_{21}^T \\ \mathbf{S}_{12}^T & \mathbf{S}_{22}^T \end{pmatrix}.$$

Each submatrix of $\mathbf{S}^T$ satisfies Property 4 because each submatrix of $\mathbf{S}^T$ is created from a submatrix of $\mathbf{S}$, which by assumption satisfies Property 4. □

Theorem 4.1 allows just one way to create a new Sudoku matrix from an old one. Our next theorem does much better, creating many new puzzles. Exactly how many is left as an exercise.

Theorem 4.2 *If* $\mathbf{S}$ *is a Sudoku matrix, then a new Sudoku matrix* $\bar{\mathbf{S}}$ *can be created by block reordering the rows and columns of* $\mathbf{S}$ *(i.e., only rows and columns within the same submatrix block can be permuted).*

Proof. Without loss of generality, let $m = 2$. Then the permissible block row and column reordering can be described by the matrix operation

$$\bar{\mathbf{S}} = \begin{pmatrix} \mathbf{E}_1 & \\ & \mathbf{E}_2 \end{pmatrix} \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{F}_1 & \\ & \mathbf{F}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{E}_1\mathbf{S}_{11}\mathbf{F}_1 & \mathbf{E}_1\mathbf{S}_{12}\mathbf{F}_2 \\ \mathbf{E}_2\mathbf{S}_{21}\mathbf{F}_1 & \mathbf{E}_2\mathbf{S}_{22}\mathbf{F}_2 \end{pmatrix}, \qquad (7)$$

where $\mathbf{E}_i$ and $\mathbf{F}_i$ are permutation matrices that result in elementary row (or column) interchanges applied to the 2×2 identity matrix. $\bar{\mathbf{S}}$ satisfies Property 1 because block reordering does not affect the size of the matrix. To show $\bar{\mathbf{S}}$ satisfies Property 2, w.l.o.g. consider the ith row of the first block row of $\bar{\mathbf{S}}$

$$\mathbf{e}_i^T \begin{pmatrix} \mathbf{E}_1\mathbf{S}_{11}\mathbf{F}_1 & \mathbf{E}_1\mathbf{S}_{12}\mathbf{F}_2 \end{pmatrix} = \mathbf{e}_i^T\mathbf{E}_1 \begin{pmatrix} \mathbf{S}_{11}\mathbf{F}_1 & \mathbf{S}_{12}\mathbf{F}_2 \end{pmatrix}.$$

Consider the last expression in the above equation. The matrix $\begin{pmatrix} \mathbf{S}_{11}\mathbf{F}_1 & \mathbf{S}_{12}\mathbf{F}_2 \end{pmatrix}$ is the first block row of $\mathbf{S}$ with its columns permuted, and $\mathbf{e}_i^T\mathbf{E}_1$ corresponds to one particular row in that matrix. Because the rows of $\mathbf{S}$ satisfy Property 2, then the particular row in question in $\bar{\mathbf{S}}$ satisfies Property 2. (Permutation of a vector does not affect Properties 2 or 3.) By considering the transpose, the same argument holds for Property 3.

It remains to show that $\bar{\mathbf{S}}$ satisfies Property 4. The *block* reordering restriction is required to insure that $\bar{\mathbf{S}}$ satisfies Property 4; cross-block reorderings destroy Property 4. From Equation (7), it is easy to see that each submatrix of $\bar{\mathbf{S}}$ satisfies Property 4 since it is a simple row and column permutation of the corresponding submatrix of $\mathbf{S}$, which satisfies Property 4 by assumption. □

EXERCISE 7*: How many new Sudoku matrices can be created from an original matrix when block reordering is used? (Hint: Try some* 4×4 *examples with the allowable reordering described above and look for counting patterns.)*

Theorem 4.3 *If* $\mathbf{S}$ *is a Sudoku matrix, then a new Sudoku matrix* $\bar{\mathbf{S}}$ *can be created by relabeling the integers in* $\mathbf{S}$*. That is, there is a one-to-one mapping between the integers* $\alpha = (1, \ldots, n)$ *used to create* $\mathbf{S}$ *and the permutation* β *of these same integers used to create* $\bar{\mathbf{S}}$*.*

Proof. (By Contradiction.) Assume $\alpha \neq \beta$, but $\bar{\mathbf{S}}$ is not a new Sudoku matrix. Assume w.l.o.g. that $\bar{\mathbf{S}}$ fails Property 2. Therefore, there exists a row in $\bar{\mathbf{S}}$ that repeats an integer. Thus, the mapping of integers from α to β is not one-to-one, which contradicts the one-to-one mapping requirement. Therefore, $\bar{\mathbf{S}}$ must be a new Sudoku matrix. □

Example The $n = 4$ Sudoku matrix $\mathbf{S}$ is used by create a new matrix $\bar{\mathbf{S}}$ under the permutation $\beta = (4 \quad 1 \quad 3 \quad 2)$ of $\alpha = (1 \quad 2 \quad 3 \quad 4)$.

$$\mathbf{S} = \left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ \hline 3 & 1 & 4 & 2 \\ 2 & 4 & 1 & 3 \end{array}\right) \quad \text{and} \quad \bar{\mathbf{S}} = \left(\begin{array}{cc|cc} 4 & 1 & 3 & 2 \\ 2 & 3 & 1 & 4 \\ \hline 3 & 4 & 2 & 1 \\ 1 & 2 & 4 & 3 \end{array}\right).$$

EXERCISE 8*: How many new Sudoku matrices can be created from an original matrix when relabeling is used?*

CHALLENGE*: Can you think of other ways to create new puzzles from one original Sudoku matrix?*

5 Conclusion

This paper examined the popular Sudoku puzzles from two angles: puzzle solution and puzzle creation. The first portion of the paper presented a binary integer programming formulation that solves any $n \times n$ Sudoku puzzle. A Matlab m-file, which executes a branch and bound solution method, is available for download. Further, such an approach was extended to variations on the traditional Sudoku puzzle. The second half of the paper presented theorems for creating new Sudoku puzzles. We discovered that, starting with one Sudoku puzzle, we can easily produce a daily calendar of Sudoku puzzles (enough for the entire next century!). By adding or removing givens, we can also vary the level of difficulty of the games. Answers to the exercises are provided below, and we hope students attempt and enjoy the challenge competitions.

Acknowledgement

Gratitude is expressed to Philip Riley and Laura Taalman for creating the puzzles in this paper. For more puzzles see their book Color Sudoku [9] or visit `http://brainfreezepuzzles.com`.

References

[1] Stanley Bammel and Jermome Rothstein. The number of 9×9 Latin squares, *Discrete Mathematics*. 11 (1975), 93–95.

[2] Brainfreeze Puzzles. `http://www.brainfreezepuzzles.com`.

[3] Martin Chlond. Classroom Exercises in IP Modeling: Sudoku and The Log Pile, *INFORMS Transactions on Education* , Vol. 5, No 2, January 2005.

[4] Richard DeVenezia, John Garlach, Larry Hoyle, Talbot Katz, and Rick Langston. SAS® and Sudoku. *SAS Global Forum 2007*. `http://www2.sas.com/proceedings/forum2007/011-2007.pdf`.

[5] Bertram Felgenhauer and Frazer Jarvis. There are 6670903752021072936960 Sudoku grids. `http://www.afjarvis.staff.shef.ac.uk/sudoku/`

[6] Howard Garns. Number Place. *Dell Pencil Puzzles & Word Games*. No. 16, May p. 6, 1979.

[7] Peter Gordon and Frank Longo. *Mensa Guide to Solving Sudoku: Hundreds of Puzzles Plus Techniques to Help You Crack Them All*. Sterling. 2006.

[8] Ed Pegg Jr. Sudoku Variations. *MAA Online*. Sept. 6, 2005. `http://www.maa.org/editorial/mathgames/mathgames_09_05_05.html`.

[9] Philip Riley and Laura Taalman. *Color Sudoku*. Sterling. 2007.

[10] Gordon Royle. Minimum Sudoku. `http://www.csse.uwa.edu.au/~gordon/sudokumin.php`.

[11] Ed Russell and Frazer Jarvis. There are 5472730538 essentially different Sudoku grids . . . and the Sudoku symmetry group. `http://www.afjarvis.staff.shef.ac.uk/sudoku/sudgroup.html`

[12] Fred Simons. Solving a Sudoku Puzzle with Mathematica. *Mathematica in Education and Research*. Vol. 10, No 4, 2005, 1 - 24.

[13] Laura Taalman. Taking Sudoku Seriously. *Math Horizons*. September 2007, 5 - 9.

[14] "Sudoku Game" from The Wolfram Demonstrations Project `http://demonstrations.wolfram.com/SudokuGame`. Contributed by: Bruce Torrence.

[15] Howard Weiss and Rasmus Rasmussen. Lessons from Modeling Sudoku in Excel. *INFORMS Transactions on Education*, Vol. 7, No. 2, January 2007.

[16] Takayuki Yato and Takahiro Seta. Complexity and Completeness of Finding Another Solution and Its Application to Puzzles. *Information Processing Society of Japan (IPSJ)*. SIG Notes 2002-AL-87-2, IPSJ, 2002.

Answers

1. You could first experiment with different objective function vectors in the `bintprog` line of the m-file `sudoku.m`. Change the objective function vector from `zeros(n^3,1)` to `ones(n^3,1)`, and other uniform vectors. You'd find that these other objective functions all give the same solution. After these experiments, you're ready to state a theorem and attempt the proof.

   Theorem 5.1 *If* $\mathbf{x}^*$ *is the optimal solution for the BILP in Section 2.1 with objective function given by* $\min \mathbf{0}^T\mathbf{x}$*, then* $\mathbf{x}^*$ *is also the optimal solution for the related problem that has the same constraints but different objective function given by* $\min \alpha\, \mathbf{e}^T\mathbf{x}$*, where* $\mathbf{e}$ *is the vector of all ones and* α *is a scalar.*

Proof. (By Contradiction.) Assume $\mathbf{x}^*$ is an optimal, and therefore feasible, solution for the original BILP of Section 2.1 (called Problem A), but $\mathbf{x}^*$ is not an optimal solution for the modified BILP with the new objective (called Problem B). Then there exists a feasible solution $\mathbf{y}$ such that $\alpha\,\mathbf{e}^T\mathbf{y} < \alpha\,\mathbf{e}^T\mathbf{x}^*$, which implies $\mathbf{e}^T(\mathbf{y} - \mathbf{x}^*) < 0$. Because $\mathbf{x}^*$ and $\mathbf{y}$ are binary vectors made up of only 0s and 1s, this means that $\mathbf{x}^*$ must have more nonzero elements than $\mathbf{y}$. Careful examination of the constraint set, which is the same for both BILPs, shows that a feasible solution must contain exactly n^2 elements that are 1. Since $\mathbf{y}$ is a feasible solution with exactly n^2 elements equal to 1, then $\mathbf{x}^*$ must have $n^2 + 1$ elements equal to 1, which implies $\mathbf{x}^*$ is not feasible. This contradicts our initial assumption that $\mathbf{x}^*$ is a feasible optimal solution to Problem A. □

2. We can't say 29 exactly, due to potential double counting as subsequent givens are considered.

3. The solutions are:

$$\left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \\ \hline 2 & 1 & 4 & 3 \\ 4 & 3 & 1 & 2 \end{array}\right), \quad \left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ \hline 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \end{array}\right), \text{ and } \quad \left(\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ \hline 2 & 3 & 4 & 1 \\ 4 & 1 & 2 & 3 \end{array}\right).$$

4. One can create 4×4 Sudoku puzzles with 5, and 4 givens that have unique solutions. However, it is not possible to formulate a puzzle of this size with a unique solution that has only 3 givens. When placing givens, position is generally more important than the value of the given. A possible theorem related to this topic is:

   Theorem 5.2 *The minimum number of givens in a* 4×4 *Sudoku puzzle is 4.*

   A proof of this theorem is contained in [13]. This article also counts the number of 4×4 puzzles. The process of enumerating such puzzles is designed to supply the reader with an overview of how Felgenhauer and Jarvis counted the number of 9×9 Sudoku boards (see Section 4.1). Further, the article poses a variety of open questions in this field of recreational mathematics.

   If, when filling in a puzzle, no options appear, then the puzzle has a unique solution. This is easy to determine for the 4×4 case, but very difficult to determine during the course of a 9×9 game.

5. Position Sudoku comes with the extra requirement that all corresponding locations within the 3×3 subgrids contain each digit exactly once. Since there are 9 possible locations and 9 possible digits, this leads to $9^2 = 81$ constraints in addition to the standard ones. The constraints are as follows:

$$\sum_{a=0}^{2}\sum_{b=0}^{2} x_{(3a+i)(3b+j)k} = 1, \;\; i = 1,2,3; \;\; j = 1,2,3; \;\; \text{and} \;\; k = 1:9$$

   Here the choices of i and j determine the particular position in question of all the subgrids; for example, $i = j = 1$ corresponds to the top left corner of each subgrid.

6. The constraints for Three Magic Sudoku are the most complicated of the ones covered in this paper since the value of each digit matters in addition to simply how many of each there are. In each shaded square, each row and each column must sum to the same amount. (For this particular puzzle, diagonals are not required to sum to the same number as well, though requiring this is an option.) It is not hard to see that the "magic number" to which each row and column must sum is 15. If we sum all the rows and all the columns of the 3×3 square together, each number in the square is added twice, once for its row and once for its column. The sum of the natural numbers from 1 to 9 is 45, so the combined sum of all the rows and all the columns must be $2 * 45 = 90$. Since there are three rows and three columns, dividing 90 by 6 will yield the sum of each row and each column, 15.

With the system of decision variables in use, $x_{rck} = 1$ if and only if k is the digit that appears in the box in row r and column c. It follows that $\sum_{k=1}^{9} kx_{rck}$ will produce the value of the digit in square (r, c): each summand will equal zero except for the one with the correct k, and then its value will be $k * 1 = k$, the digit appearing in the square. Knowing this fact and that each row and column in the shaded squares must add to 15, we can formulate the constraints as follows (for the top right magic square).

Row sums:

$$\sum_{c=7}^{9} \sum_{k=1}^{9} kx_{rck} = 15,\ r = 1:3$$

Column sums:

$$\sum_{r=1}^{3} \sum_{k=1}^{9} kx_{rck} = 15,\ c = 7:9$$

The constraints for the other two magic squares are done similarly, with r and c both ranging from 4 to 6 for the middle square, and r ranging from 7 to 9 and c ranging from 1 to 3 for the lower left square. There are six constraints for each magic square, and there are three magic squares, leading to 18 constraints on top of the standard ones. (It is possible to reduce these constraints down to 5 for each square instead of 6, by setting the second column sum equal to the first column sum, the third column sum equal to the first column sum, the first row sum equal to the first column sum, etc. However, doing this would most likely complicate the constraints enough to more than cancel out any gain in efficiency from having fewer constraints.)

7. There are $m!$ possibilities to permute an $m \times m$ identity matrix and m blocks that can be permuted. Counting both row and column permutations, this gives $[(m!)^m]^2$ different Sudoku matrices. Thus, Theorem 4.2 leaves $[(m!)^m]^2 - 1$ new Sudoku matrices that can be created from the original matrix. For $m = 3$, that's 46,655 different puzzles!

8. Theorem 4.3 creates $n!$-1 new Sudoku matrices. For $n = 9$, that's 362,879 new puzzles! Combining the three theorems in Section 4.2 with strategies for varying the number and location of givens creates an even larger number of puzzles.