An Integer Programming Model for the Sudoku Problem

Andrew C. Bartlett[*] Amy N. Langville[†]

March 18, 2006

Abstract

Sudoku is the new 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. 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, pronounced suh-DOE-koo, is a logic-based puzzle that first appeared in the U.S. in 1979. 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 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, which his company Pappocom then sold and now continues to sell to various publication outlets. In the last year, Sudoku's fame spread to the U.S., where the puzzles now appear alongside the long-time staple of linguistic games, the crossword puzzle, in national newspapers, such as *The Los Angeles Times* and *The Washington Post.* The word Sudoku is a Japanese abbreviation for the phrase, "suji wa dokushin ni kagiru," which translates as the "the digits must remain single."

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 determines the game's level of difficulty. Figure 1 is an example of a 9×9 Sudoku puzzle.

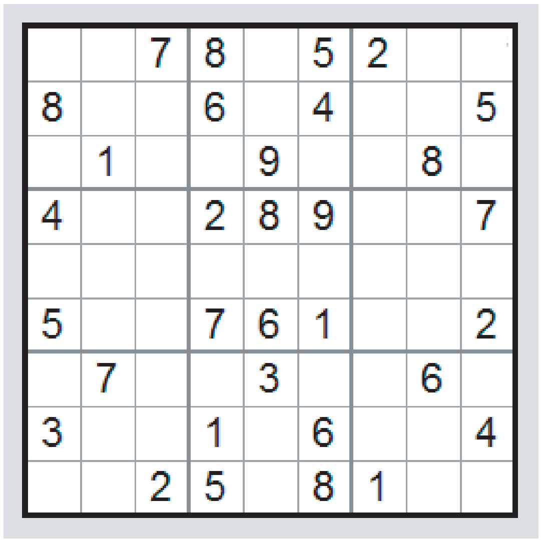
This puzzle idea can accommodate games of other sizes. Of course, a 4×4 puzzle would be easier and a 16×16 puzzle, harder. In general, any $n \times n$ game can be created, where $n = m^2$ and m is any positive integer. In fact, the 25×25 puzzle is sometimes called Samurai Sudoku, because it is much more challenging and time-consuming. And there are numerous other variants of the game; see [2].

Sudoku puzzles elicit the following two interesting mathematical questions:

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

[*]Department of Mathematics, College of Charleston, Charleston, SC, USA, `acbartle@edisto.cofc.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.



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

Figure 1: An example Sudoku puzzle

2 Question 1: Solving the Puzzle

2.1 The Mathematical Model

An assignment for a graduate class in Optimization at the College of Charleston challenged students to mathematically model a Sudoku puzzle. As a result, we formulated the following binary integer linear program (BILP) for general $n \times n$ puzzles.

Define:

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

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

This could also be formulated as an integer linear program (ILP), letting the integer variables take on values from 1 to 9. However, in order to use the available class software (Matlab's Optimization Toolbox), we used a BILP, and therefore, Matlab's `bintprog` command. 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). A more difficult and interesting problem, which we briefly mention in Section 3, is to find *all* feasible solutions, without knowledge of the givens in the opening Sudoku matrix (i.e., solving the satisfiability problem without the objective and constraint (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, so 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

Consider once again the 9×9 Sudoku puzzle of Figure 1. To solve this, and any other $n \times n$ Sudoku puzzle, we created a Matlab program called `sudoku.m`, which executes the above BILP model. This program can be downloaded from `http://math.cofc.edu/~langvillea/Sudoku/sudoku.m`. It requires that the user input the elements, called givens, provided at the start of the puzzle. The matrix of givens for our example is

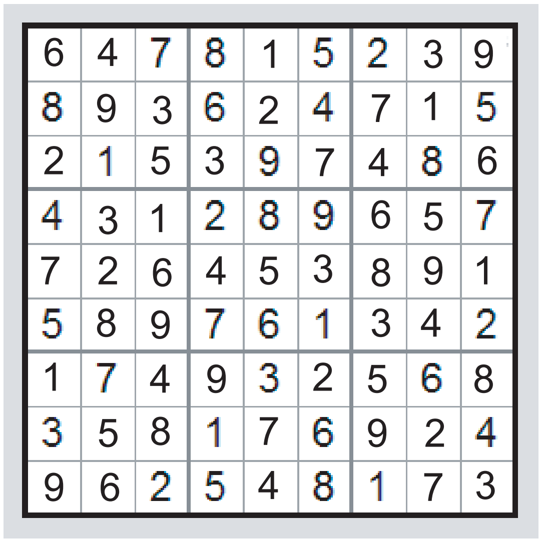
$$\texttt{Givens} = \begin{pmatrix} 1 & 3 & 7 \\ 1 & 4 & 8 \\ 1 & 6 & 5 \\ 1 & 7 & 2 \\ 2 & 1 & 8 \\ 2 & 4 & 6 \\ & \vdots & \\ 9 & 3 & 2 \\ 9 & 4 & 5 \\ 9 & 6 & 8 \\ 9 & 7 & 1 \end{pmatrix},$$

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`. On a Mac G5 with dual 2.7 GHz processors and 8 GB of memory, our program found the feasible solution, presented in Figure 2.2, in 16.08 seconds.

2.3 The Integer Programming Technique

Matlab's `bintprog` uses the classic branch and bound method to determine the optimal solution. In this case, 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. (When we asked Matlab to find at least one feasible solution to the related problem, which ignores constraint (5), the procedure terminated because the maximum number of iterations for the LP relaxation (which we increased to half a million) was exhausted before a solution was found.) It's easy to understand why each given element of the matrix helps the branch and bound procedure. 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)



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

Figure 2: Solution for example Sudoku puzzle

- $x_{1j7} = 0, \quad \forall j \neq 3$ (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 3.1), 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. Originally, we assumed each Sudoku puzzle has one unique solution since usually an answer is provided with each puzzle. However, after playing a few games, we discovered that uniqueness is not a requirement for puzzle creators. Some puzzles have multiple solutions. In such cases, our solution was correct but distinct from the solution provided by the puzzle creators.

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

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

Exercise 3*: Find all solutions to the Sudoku puzzle below. (Hint: there are 6.)*

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

Challenge Competition*: 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 [6], which implies NP-completeness and demonstrates the challenge and complexity of the all solutions problem.)*

Challenge Competition*: Often the givens provided create "forces." These are entries which can be determined from the givens in the initial matrix. Identifying these forces would greatly help the BILP, as in effect the number of givens would increase. Write an algorithm that locates forces 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 Competition*: Although it is a fun application for Optimization classes, 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*: Experiment with some 4×4 puzzles. How do you know when your solution is unique?*

Exercise 5*: 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? State and prove a theorem (or theorems) that summarizes your findings. For example, "any 4×4 puzzle whose 5 givens satisfy . . . has a unique solution."*

3 Question 2: Creating the Puzzles

The Sudoku class assignment also asked: how do you think Sudoku puzzles are created? We originally assumed that puzzle creators are interested in creating puzzles with unique solutions because an answer is usually provided with each puzzle. However, as mentioned above, after playing a few games, we stumbled across a few puzzles that had multiple solutions. In such cases, our solution was correct, but distinct, from the solution provided by the puzzle creators.

3.1 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 and deserve the title of "Sudoku matrix"? 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)?

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 $6,670,903,752,021,072,936,960 \approx 6.67 \times 10^{21}$ [1]. 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. This figure has been confirmed independently by numerous others [5]. Later Russell and Jarvis [4] 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, we proposed to stop as soon as we found one. With a full Sudoku matrix in hand, we could then simply omit entries to create a puzzle. But the question was how to do the omitting. Specifically, we posed 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 led us 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 35396 distinct such puzzles [3].) To date, no puzzle with 16 or less givens, which in turn produces a unique solution, has been discovered.
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. For this reason, question two quickly becomes untractable. There are too many factors to consider at once.
3. We provide several answers to this question in the next section.

3.2 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 3.1 *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 3.1 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 3.2 *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

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

where $\mathbf{E}_i$ and $\mathbf{F}_i$ are elementary matrices representing permutations of 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 $\bar{\mathbf{S}}$

$$\mathbf{e}_i^T \left(\mathbf{E}_1\mathbf{S}_{11}\mathbf{F}_1 \quad \mathbf{E}_1\mathbf{S}_{12}\mathbf{F}_2 \right) = \mathbf{e}_i^T \mathbf{E}_1 \left(\mathbf{S}_{11}\mathbf{F}_1 \quad \mathbf{S}_{12}\mathbf{F}_2 \right).$$

Consider the last expression in the above equation. The matrix $(\mathbf{S}_{11}\mathbf{F}_1 \quad \mathbf{S}_{12}\mathbf{F}_2)$ 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 6*: 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 3.3 *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 7*: How many new Sudoku matrices can be created from an original matrix when relabeling is used?*

CHALLENGE COMPETITION*: Can you think of other ways to create new puzzles from one original Sudoku matrix? The longest list wins.*

4 Conclusion

This paper examined the popular Sudoku puzzles from two angles: puzzle solution and puzzle creation. The first half 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. 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 vary the level of difficulty of the games. Since it's common practice among puzzle creators to not worry about the uniqueness of a puzzle's solution, these simple techniques are enough to enable us to compete with today's puzzle producers, and could garner us thousands of dollars in revenue too. Answers to the exercises are provided below, and we hope students attempt and enjoy the challenge competitions.

References

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

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

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

[4] 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`

[5] "Sudoku." *The Wikipedia Encyclopedia*. `http://en.wikipedia.org/wiki/Sudoku`.

[6] TakayukiYato 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 4.1 *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. $\square$

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

3.

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

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

4. 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. 5 givens, yes; 4 givens, yes; 3 givens, no. When placing givens, position is more important than the value of the given. Some possible theorems related to this topic are below.

   Theorem 4.2 *Every* 4×4 *Sudoku puzzle whose 4 givens satisfy the condition that each given appears in its own row, own column, own submatrix, and has its own integer value has a unique solution.*

   There are $(16 \cdot 8 \cdot 4 \cdot 1)(4!) = 12,288$ puzzles of size 4×4 of this type.

   Theorem 4.3 *Every* 4×4 *Sudoku puzzle with only 3 givens has multiple solutions.*

6. 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 3.2 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!

7. Theorem 3.3 creates $n!$-1 new Sudoku matrices. For $n = 9$, that's 362,879 new puzzles! Combining the three theorems in Section 3.2 with strategies for varying the number and location of givens creates an even larger number of puzzles.