Cracking the Sudoku:
A Deterministic Approach

David Martin

Erica Cross

Matt Alexander

Youngstown State University

Center for Undergraduate Research in Mathematics

Abstract

The model begins with the formulation of a Sudoku puzzle solving algorithm. The algorithm implements a hierarchy of four simple logical rules commonly used by humans. The difficulty of a puzzle is determined by recording the sophistication and relative frequency of the methods that were required to solve it. Four difficulty levels are established, each pertaining to a range of numerical values returned by the solving function. Like humans, the program begins solving each puzzle with the lowest level of logic necessary. When all lower methods have been exhausted, the next echelon of logic is implemented. After each step, the program returns to the lowest level of logic, so as to always use the lowest possible level of logic. The procedure loops until either the problem is completely solved or the logical techniques of the program are insufficient to make further progress.

The construction of a Sudoku puzzle begins with the generation of a solution. This is accomplished by means of a random number based function within the program. Then, working backwards from the solution, numbers are removed one by one, at random, until one of several conditions, such as a minimum difficulty rating and a minimum number of empty squares, has been met. Following each change in the grid, the difficulty is evaluated as the program solves the current puzzle. If the program cannot solve the current puzzle, then one of two possibilities must be true: either the puzzle does not have a unique solution, or the solution is beyond the grasp of the logical methods possessed by the solver. In either case, the last solvable puzzle is restored and the process continues.

Uniqueness is guaranteed due to the fact that the algorithm never guesses. If there is not sufficient information to draw further conclusions, such as the event that an arbitrary choice must be made (which must invariably occur for a puzzle with multiple solutions to be solved) the solver simply stops. For obvious reasons, puzzles lacking a unique solution are undesirable. Due to the fact that the logical techniques possessed by the program enable it to solve most commercial puzzles (for example, most evil puzzles from websudoku.com were solved successfully), the assumption was made that demand for puzzles requiring logic beyond the current grasp of the solver is low. Therefore, there is no need to distinguish between puzzles requiring very advanced logic and those lacking unique solutions.

Contents

1 Introduction

The development of an algorithm capable of constructing Sudoku puzzles of varying difficulty entails the preceding formation of a puzzle solving algorithm. Hence, the program (written in C++) contains a function which attempts to generate the solution to a given puzzle. Four simple logical rules encompass the reasoning necessary to solve most commercially available Sudoku puzzles. Each sequential technique is more logically complex than the previous method. The varying complexity among the methods establishes a somewhat natural system by which the difficulty of a Sudoku puzzle can be rated. Each technique is given a weight proportional to its complexity; then, difficulty is determined by a weighted average of the methods used. Our algorithm places each puzzle in one of four categories which we identify as Easy, Medium, Hard, and Very Hard. The lowest level of logic is the most fundamental method used by our program (and humans!) in an attempt to solve a Sudoku puzzle. In the circumstance that a level of reasoning can no longer be used, the next level of logic is prompted. A successful attempt at this new level is followed by a regression back to the lowest level of logic employed. A failed attempt at the new stage initiates a further advancement in logic. The procedure loops until the problem is completely solved or no more progress can be made. Consistency is guaranteed by the use of our check function. This procedure verifies that each row, column, and box contains each of the digits 1 to 9 without duplication. Under the circumstance that the techniques are inadequate to solve a Sudoku puzzle, the progression of the loop will terminate.

Our algorithm constructs Sudoku puzzles in a somewhat "backward" manner. First, a completed Sudoku is formulated using a simple random-number-based function, similar to many "brute force" methods of solving the puzzles. Before the puzzle generation begins, the user enters restrictions such as desired difficulty level and the maximum number of cells that can be given. Creating a puzzle begins by randomly eliminating digits from one cell at a time. The elimination process continues until the conditions specified are met. After each removal, the program attempts to solve the existing puzzle. A Sudoku puzzle cannot be solved in one of two scenarios. The first case is exhibited when the puzzle no longer has a unique solution. The algorithm is deterministic, and only draws conclusions that follow directly from the current state of the puzzle. In such a case, because an arbitrary decision must be made in order to choose a particular solution, the algorithm simply terminates. The other situation occurs when the logical methods available to the solver are not sufficient. In either circumstance the program restores the last solvable puzzle and resumes the process.

Due to the undesirable nature of both ambiguous puzzles and puzzles that require guessing, the algorithm averts exploiting the guessing technique. If there exists a unique solution for a given puzzle, then failure to solve implies that the puzzle requires logical methods higher than those written into the program. This characteristic is appropriate as demand is low for Sudoku puzzles requiring extremely sophisticated logical methods. Thus our algorithm does not distinguish between puzzles with no solution and those requiring more advanced logic.

2 Definitions

A comprehension of the terminology describing Sudoku puzzles is fundamental to understanding our analysis of the problem and algorithm. The following represent the principal vocabulary utilized:

Cell: A location on a Sudoku grid identified by the intersection of a row and a column, which must contain a single digit.

Row: A horizontal alignment of 9 cells in the Sudoku grid.

Column: A vertical alignment of 9 cells in the Sudoku grid.

Box: One of the nine 3×3 square groups of cells which together comprise the Sudoku grid.

Group: A row, column, or box on the Sudoku grid which must contain each digit from 1-9 exactly once.

Given: A cell whose answer is provided at the beginning of the puzzle.

Candidate: A possible solution for a cell that was not given.

Method: The technique used to eliminate candidates as possibilities and solve cells.

Unique: The puzzle is considered to be unique when it can only be solved in one possible manner and therefore has a unique solution.

Difficulty: The level of skill needed to solve the puzzle based on the perceived complexity and frequency of the methods required to solve the puzzle.

3 Assumptions

1. We will work only with the classic Sudoku grid consisting of a 9×9 square matrix.
2. Guessing, while a form of logic, is not a deterministic method. Demand is low for Sudoku puzzles which require nondeterministic logic.
3. Demand is low for puzzles requiring extremely complicated logical methods.
4. The difficulty of a puzzle can be calculated as a function of the sophistication and frequency of the logical methods demanded.
5. The ordering of a given set of puzzles by difficulty will be the same for the program as for humans.

The first assumption restricts the dimensions of the Sudoku puzzles that will be considered. It is reasonable to use the 9×9 matrix as it is the most common type of the Sudoku puzzle. For the second assumption, note that all puzzles from WebSudoku.com can be solved without guessing (or so they claim). Justification for the third assumption results from the fact that the present algorithm solves all Easy, Medium, Hard, and some Very Hard puzzles. The fourth assumption is taken to be obvious. The final assumption is based on the fact that because the solver uses the same techniques employed by humans.

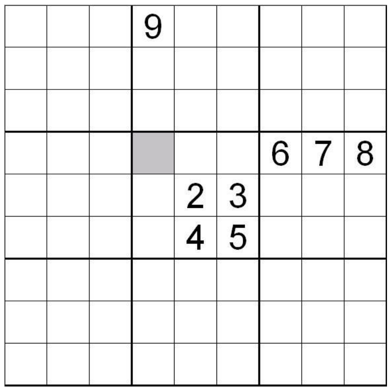
4 Model Design

4.1 The Solver

The program is based on simple logical rules, utilizing many of the same methods employed by humans. Like humans, the program begins solving each puzzle with the lowest level of logic necessary. When all lower methods have been exhausted, the next echelon of logic is implemented. After each step, the program returns to the lowest level of logic, so as to always use the lowest possible level of logic. The procedure loops until either the problem is completely solved or the logical techniques of the program are insufficient to make further progress. The following techniques are included in the algorithm.

I. Naked Single: a cell for which there exists a unique candidate based on the circumstance that its groups contain all the other digits (Davis, 2007)

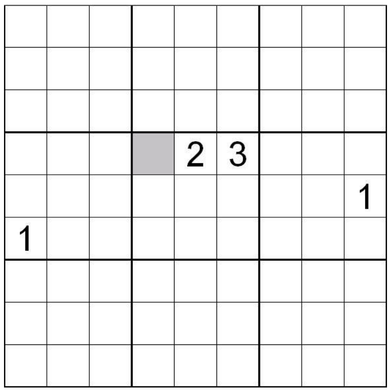
Figure 1: Naked Single



Clearly, 1 is the only candidate for the shaded cell.

II. Hidden Single: a cell for which there exists a unique candidate based on the constraint that no other cell in one of its groups can be that number (Davis, 2007)

Figure 2: Hidden Single

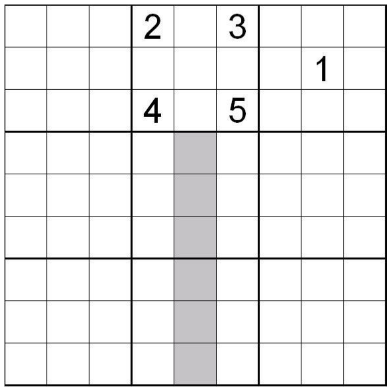


The shaded cell must obviously be a 1.

III. Locked Candidate (Davis, 2007):

A method of elimination for which a number within a box is restricted to a specific row or column and therefore can be excluded from the remaining cells in the corresponding row or column outside of the selected box

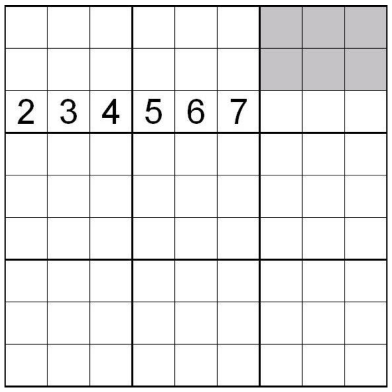
Figure 3: Locked Candidate (Box)



None of the shaded cells can be a 1.

A method of elimination for which a number within a row or column is restricted to a specific box and therefore can be excluded from the remaining cells within the box.

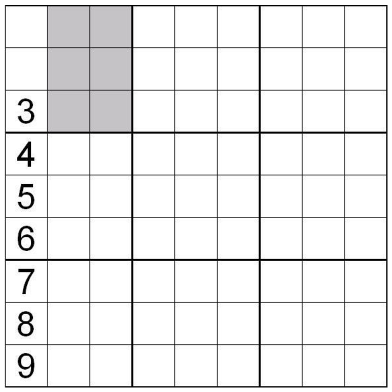
Figure 4: Locked Candidate (Rows and Columns)



Again, none of the shaded cells can be a 1.

IV. Naked Pairs: This method of elimination pertains to the situation in which two numbers are candidates in exactly two cells of a given group. Consequently, those two numbers are eliminated as candidates in all other cells within the group (Davis, 2007).

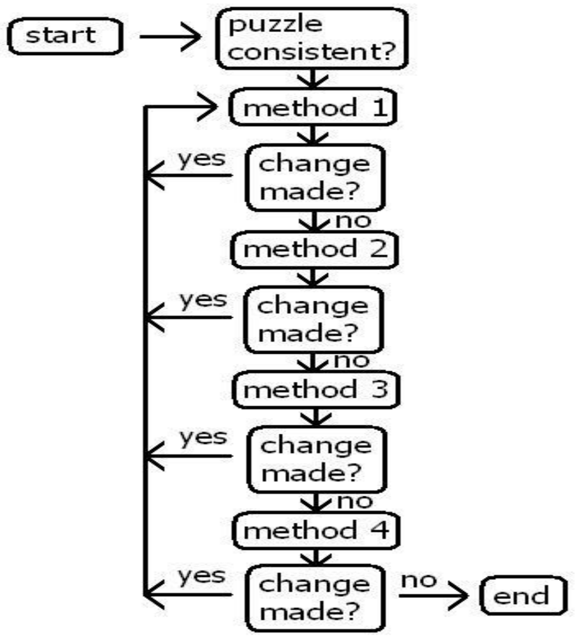
Figure 5: Naked Pairs



None of the shaded cells may contain either a 1 or 2.

The algorithm is represented by the following diagram:

Figure 6: Puzzle Solver



4.1.1 Difficulty

The algorithm is based on many techniques commonly employed by humans. Therefore, one can reasonably propose that for a given set of puzzles, the ordering by difficulty will be about the same for the program as for humans. Hence, the difficulty rating produced by the program is of practical value in commercial applications.

As the solver works on a puzzle, it keeps track of the number of times it uses each level of logic. Let n_i be the number of times technique i was used. The difficulty rating can then be calculated by means of a simple formula

$$D(n_1, n_2, n_3, n_4) = \frac{\sum w_i n_i}{\sum n_i},$$

where w_i is a difficulty weight assigned to each method. Naturally, weight should increase with the complexity of the logic used in a technique. Clearly, as the proportion of changes made by a method increases, the difficulty value approaches the weight assigned to that technique. In practical application, higher methods are used extremely rarely. Therefore, seemingly disproportionately high weights should be assigned to the higher methods for them to have an appreciable effect on difficulty. The choice of these values is somewhat arbitrary, and small changes are not likely to have an appreciable effect on the ordering by difficulty of a set of puzzles. For the current purpose, the following values were chosen: $w_1 = 1, w_2 = 3, w_3 = 9, w_4 = 27$. In application, these values provide a nice spectrum ranging from 1 (only the first level of logic is used) to about 4 (the higher levels are used frequently). One of the hardest puzzles generated by the program required the use of techniques one, two, three, and four 42, 11, 10, and 2 times, respectively. Hence, the corresponding difficulty rating was calculated to be

$$D = \frac{42 \cdot 1 + 11 \cdot 3 + 10 \cdot 9 + 2 \cdot 27}{42 + 11 + 10 + 2} \approx 3.37.$$

Difficulty categories can then be determined by partitioning the interval [1,4] into any four subintervals. Reasonable subintervals were determined to be: Easy=[1,1.5), Medium=[1.5,2), Hard=[2,3), Very Hard=[3,4]. A typical Easy puzzle with a rating of 1.25 requires use of the second level of logic seven times. A typical Medium puzzle with a rating of 1.7 requires the use of the second level of logic 17 times and the third level once. A typical Hard puzzle with a rating of 2.5 requires the use of the second level of logic 17 times, of the third level 4 times, and of the fourth level once. The aforementioned puzzle, whose rating of 3.37 renders it Very Hard, required the use of the second method 11 times, of the third method 10 times, and of the fourth method twice.

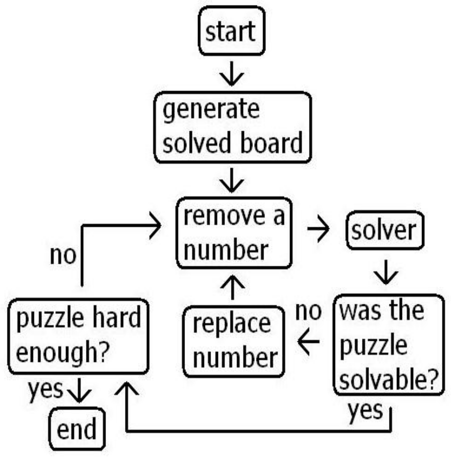
4.2 The Puzzle Creator

Rather than starting with an empty grid and adding numbers, the program begins with a completed Sudoku, produced by a random-number-based function within the program. The advantage of the latter method over the former is obvious; rather than starting with a puzzle lacking a unique solution and hoping to stumble upon one with a unique solution, the program begins with a puzzle having a unique solution and maintains the uniqueness.

Once a completed Sudoku has been created, the puzzle is developed by working backwards from the solution, removing numbers one by one (at random) until one of several conditions has been met. These conditions include a minimum difficulty rating (to ensure that the puzzle is hard enough) and a minimum number of empty squares (to ensure that the puzzle is far from complete). Following each change in the grid, the difficulty is evaluated as the program solves the current puzzle. If the program cannot solve the current puzzle, then one of two possibilities must be true: either the puzzle does not have a unique solution, or the solution is beyond the grasp of the logical methods possessed by the algorithm. In either case, the last solvable puzzle is restored and the process continues.

In theory, there may occur a situation in which removing any given number will not yield a puzzle that is solvable (by the algorithm) and has a unique solution. In such a case, the puzzle creator has reached a “dead end”, and cannot make further progress toward a higher difficulty rating. To overcome this obstacle the program, rather than generating a single puzzle as close as possible to a given difficulty rating, generates 1000 puzzles and sorts them in order of increasing difficulty. In this manner, the program produces a virtual continuum of difficulties ranging from 1.0 to whatever difficulty was requested (within the limits of the program, which cannot produce puzzles that are harder than about 4).

Figure 7: Puzzle Creator



4.2.1 Uniqueness

Uniqueness is guaranteed due to the fact that the algorithm never guesses. If there is not sufficient information to draw further conclusions, such as the event that an arbitrary choice must be made (which must invariably occur for a puzzle with multiple solutions to be solved) the solver simply terminates. For obvious reasons, puzzles lacking a unique solution are undesirable.

4.2.2 Difficulty

Due to the fact that the logical techniques possessed by the program enable it to solve most commercial puzzles, the assumption was made that demand for puzzles requiring logic beyond the current grasp of the solver is low. Therefore, there is no need to distinguish between puzzles requiring very advanced logic and those lacking unique solutions. In either case, the change which resulted in the status should be reversed.

5 Model Testing (Relevance of the Difficulty Rating)

In order to determine the relevance of the algorithm to real world Sudoku puzzles, the program was set loose on a set of 48 randomly selected puzzles from three popular websites. Four puzzles were selected from each of four difficulty categories for each source.

The following table summarizes relevant results.

Label Awarded by Source	Websudoku	Puzzles	Sudoku Puzzles Online
Easy	Easy	Easy	Easy
	Easy	Easy	Easy
	Easy	Easy	Easy
	Easy	Easy	Easy
Medium	Easy	Easy	Easy
	Easy	Medium	Easy
	Easy	Medium	Easy
	Easy	Medium	Medium
Hard	Easy	Easy	Medium
	Easy	Medium	Medium
	Medium	Hard	Easy
	Medium	Hard	Medium
Very Hard	Hard	Not Solved	Hard
	Very Hard	Not Solved	Very Hard
	Not Solved	Not Solved	Not Solved
	Not Solved	Not Solved	Not Solved

(a)

		Websudoku	Puzzles	Sudoku Puzzles Online
Easy	Average Difficulty	1.0	1.1	1.0
	Proportion Solved	4 / 4	4 / 4	4 / 4
Medium	Average Difficulty	1.2	1.6	1.3
	Proportion Solved	4 / 4	4 / 4	4 / 4
Hard	Average Difficulty	1.6	1.9	1.7
	Proportion Solved	4 / 4	4 / 4	4 / 4
Very Hard	Average Difficulty	3.5	4.0	3.5
	Proportion Solved	2 / 4	0 / 4	2 / 4

(b)

Figure 8: Tests

All puzzles labeled by the source as Easy, Medium, and Hard (or an equivalent word) were solved successfully and rated. On the other hand, some but not all of the Very Hard puzzles were solved successfully and rated. Those beyond the grasp of the program were simply given a rating of 4.0, the maximum rating allowable under the current scale. Although the algorithm was able to crack exactly one half of the Very Hard puzzles attempted from both Websudoku.com and SudokuPuzzlesOnline.com, none of the Very Hard puzzles obtained from Puzzles.com were solved.

Under the suggested partition ([1,1.5),[1.5,2),[2,3),[3,4]), all of the puzzles labeled by the source as Easy (or something equivalent) were awarded the same rating by the program. Agreement was excellent with Puzzles.com, with which the program agreed on 13 of the 16 puzzles tested (the four puzzles from that source that were given a rating of Very Hard were not successfully solved by the algorithm, and received a difficulty rating of 4.0 by default). The three puzzles for which the algorithm and Puzzles.com disagreed were all given a lower difficulty rating by the program. The program successfully solved four Very Hard puzzles from the other two sources but was apparently not too impressed, awarding half of those solved a mere Hard rating. In the Medium and Hard categories, puzzles from Websudoku.com and SudokuPuzzlesOnline.com were consistently awarded lower difficulty ratings than those suggested by the source.

6 Model Analysis

6.1 Complexity Analysis

6.1.1 The Solver

The puzzle solving algorithm is surprisingly short, relying on a simple topology connecting a heirarchy of just four logical methods (see Figure 6). At first glance, one might suspect that most puzzles would be beyond the scope of the four logical operations available to the solver. However, as seen above, the algorithm does not meet its match until it is presented with puzzles labeled as Very Hard.

6.1.2 The Puzzle Creator

At the start of the process, a mysterious procedure randomly creates a solved puzzle. This procedure is not complicated, and can be summarized as follows. Going from the top of the puzzle to the bottom, and from the left to the right, a random number is inserted into each empty box. When a number is inserted, the program checks for consistency. If the addition of the number has contradicted a constraint of the Sudoku puzzle, then another number is attempted. After a fixed number of attempts have failed at a given cell (No number, in fact, may be possible if there remain no candidates for a given cell.), the program removes both of the numbers involved in the last contradiction. This allows the program to dismantle relationships that make a puzzle unsolvable. The process loops until the puzzle is both consistent and complete (no empty spaces).

The rest of the puzzle creation process is largely the inverse of the above procedure, only rather than adding numbers and checking for consistency and completeness, the program removes numbers and checks for solvability and uniqueness (which are equivalent for reasons discussed above), as well as constraints pertaining to difficulty rating and number of givens.

6.2 Sensitivity Analysis

The primary source of arbitrarity in the model is the method by which difficulty ratings are established. The current method requires the user to assign to each logical technique a weight proportional to its complexity and contribution to the overall difficulty of the puzzle. Weights of 1, 3, 9, and 27 were assigned to the first, second, third, and fourth levels of logic, respectively. It turns out that the exact values of these numbers is relatively unimportant, evidenced by the fact that two additional sets of weights produced exactly the same ordering by difficulty of a set of 8 typical puzzles created by the program. The following table summarizes the relative independence of the ordering on weight values.

LOGIC LEVEL	WEIGHTING SYSTEM			
	4^N	3^N	2^N	2N+1
1	1	1	1	1
2	4	3	2	3
3	16	9	4	5
4	64	27	8	7

(a)

PUZZLE	4^N	3^N	2^N	2N+1
A	1.00	1.00	1.00	1.00
B	1.47	1.32	1.16	1.16
C	1.96	1.61	1.29	1.27
D	2.66	2.00	1.45	**1.40**
E	3.52	2.29	1.48	**1.35**
F	3.87	2.61	1.66	**1.51**
G	5.38	3.06	1.70	**1.46**
H	5.78	3.28	1.78	1.52

(b)

Figure 9: Sensitivity Analysis

Note that although all three of the exponential weight systems produce the same difficulty rating, the linear system does not. Because the featured system, which uses the weights of 1, 3, 9, and 27, agrees so well with Puzzles.com, it seems safe to say that the exponential weighting system makes much more sense (at least with the current hierarchy of logical techniques) than the linear.

6.3 Shortcomings of the Model

- We assume four levels of logic are sufficient to solve any Sudoku puzzle while in actuality other techniques for solving exist.
- The model lacks the capacity to generate the solution to some evil puzzles featured on websudoku.com using logical methodology due to the absence of more complex

methods within the program.

- Our model reports an error for Sudoku puzzles which either have no solution or multiple solutions but it does not differentiate between the two.

6.4 Strengths of the Model

- Our model considers the fact that once a higher level of logic is used and a cell is filled, a human will return to attempting a solution with the simplest method of logic and therefore so does our program.
- Utilizing a functional program, we were able to construct and evaluate the difficulty of a thousand Sudoku puzzles in a matter of minutes.
- The program uses deterministic logic in each method featured in the program and does not resort to guessing.
- The code can be easily expanded to include more advanced levels of logic such as naked triplets and quads, x-wings and swordfish, or coloring.
- The code could also be easily modified to do other types of Sudoku puzzles such as a 16x16 grid and other rules for the puzzle.

7 Conclusion

In spite of the seemingly small scope of the four logical operations available to the solver, the algorithm was able to solve all Easy, Medium, and Hard puzzles obtained from three popular internet sources. About 1/3 of the Very Hard puzzles obtained from those sources were also solved correctly. Therefore, it is clear that a small set of logical rules is sufficient to solve nearly all commercially available Sudoku puzzles. In order to expand the scope of the solver, the overall complexity of the algorithm need not be increased. Simply adding another logical technique to the loop can increase the Sudoku solving power of the algorithm. A mere two or three additional methods would probably suffice to enable the program to solve all commercially available puzzles that do not require guessing.

8 Just For Fun

The following puzzle, created by the program and given a difficulty rating of 3.52 (Very Hard), requires the use of methods one, two, three, and four 45, 12, 9, and 3 times, respectively. Have fun!

Figure 10: Difficulty 3.52

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

References

Davis, Tom. *"The Mathematics of Sudoku"*. 21 Nov 2007. 15 Feb 2008
http://www.geometer.org/mathcircles/sudoku.pdf.

www.websudoku.com

www.sudokupuzzlesonline.com

www.puzzles.com