Sudoku Generation and Difficulty Metrics

Introduction

We develop an algorithm to construct solvable Sudoku puzzles of varying difficulties with a unique solution. The algorithm begins by creating a completed Sudoku board, and then removes entries while knowing that it will remain solvable and has one sole solution. The boards produced by this algorithm represent only a small fraction of all possible filled-board enumerations. Many Sudoku generators employ a similar tactic and allow for a user-specific difficulty rating determining which puzzle is presented to them. Our goal is to define difficulty metrics applicable to our generated puzzles and extensible to boards that our algorithm fails to generate. Sudoku difficulties may be derived in a number of ways such as computer computation times, human computation times, or counting the logical steps required for various deduction methods. An algorithm produces a puzzle that must satisfy two initial requirements;

(i) The puzzle must be solvable, and
(ii) The puzzle must have a unique solution.

Terminology and Background Information

We assume the rules of Sudoku are widely known. For an explanation of the terminology which shall be used throughout this paper, see Figure 1 below.

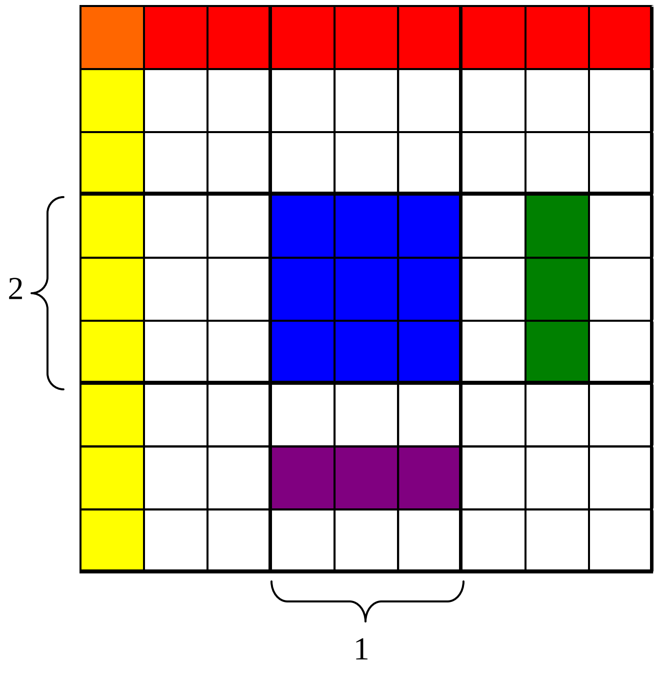

Figure 1 – Pillars (1) are made up of three adjacent columns (red) that form three blocks (blue). Ribbons (2) are three adjacent rows (yellow) that also form three distinct blocks. A minor column (green) is a single column of a block and a minor row (purple) is a single row of a block. An entry (orange) is just one element of the puzzle.

The calculation of the number of all possible board enumerations was published in 2005 and is 6,670,903,752,021,072,936,960[1]. The number of possible essentially different enumerations—neglecting group symmetry operations and relabeling—is estimated to be 5,472,730,538[2]. The lowest number of givens for a puzzle to remain solvable is still unknown, but the smallest irreducible puzzles constructed thus far have only 17 given entries.

Problem Analysis

Our puzzle generating algorithm must incorporate and accomplish a minimal number of tasks. First, unique boards must be produced. By unique, we mean,

$$\text{If } a_{ij} = a_{mn}\text{, for some } i \neq m \text{ and } j \neq n\text{, then } a_{in} \neq a_{mj}.$$

Second, entries are removed in a symmetric fashion for aesthetic purposes only—there is no reason for published puzzles to maintain any symmetry, it is just a logic rule for puzzle creation. Third, after any entry elimination, a solver must verify a solution still exists and is the same as the original board. Fourth, a difficulty index must be calculable for the generation of each board. And lastly, the algorithm must be correct, efficient, and complex enough to allow for a wide range of difficulties.

Assumptions

- Variations of Sudoku, like Samurai Sudoku, have also intrigued avid players. For this problem we assume the difficulty metrics will not require extensibility to any variations of Sudoku.
- We assume a puzzle solver does not know the board generation method since a simple generation of repeated permutations can easily be deduced if it were an initial search parameter in the solver.
- We assume the random number generation within an algorithm is actually random.
- We assume the probability of consecutively generating identical boards is essentially zero.

Algorithm A

Board Generation

Using a random seed matrix for block 1, we apply elementary row and column operations to create eight other distinct matrices to fill the remaining blocks (Appendix A). The order in which the permuted matrices are placed guarantees uniqueness. Consider a Sudoku puzzle composed of 9 blocks, U, where B_1 is the seed matrix and $B_2, \ldots, B_9$ are the permutations of B_1. For example;

$$U = \begin{bmatrix} B_1 & B_2 & B_3 \\ B_4 & B_5 & B_6 \\ B_7 & B_8 & B_9 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} B_1,\ B_4 = B_1 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}\text{, and } B_9 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} B_1 \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Randomization

The randomization procedure arbitrarily applies a set of operations—pillar, ribbon, row, and column permutations—to the assembled board. The frequency at which each operation occurs is randomly set within the algorithm.

Entry Elimination

Under the constraint that at least a specified number of givens, in this case 54, must remain on the board, the locations of elements are randomly selected, along with the location of their rotationally symmetric partner, for removal. Thus, if a_{ij} is the element selected for removal, then $a_{(10-i)(10-j)}$ is also removed. After each pair elimination, the algorithm executes a solver. If the entries removed have left the puzzle unsolvable under the scope of the solver, the entries are reinserted and a new pair is selected. If the solution found is not equivalent to the original board, implying multiple solutions exist, then this is a violation of our initial requirement for a unique solution and the entries are reinserted. This process continues until the number of givens is less than or equal to the specified amount.

The Solver and Difficulty Indexing

For algorithm A, the solver consists of the most elementary deduction processes commonly employed and suggested by training guides and players [3]. Naked-single candidate elimination examines the row and column of an unknown entry, if it can be deduced that only one possible value exists, that value is entered. Hidden-single candidate elimination examines the rows and columns passing through the block containing an unknown entry; if a value exists that is present in each row and each column outside of the block, that value is inserted. Here, they are treated as one process for the difficulty calculations. This is because they are the only logic methods that will evoke a single candidate for insertion—all other methods tirelessly deduce the number of possible candidates, until a single elimination procedure finds the one possible value. The lowest number of givens required to solve a puzzle generated by this algorithm observed thus far, using these processes alone, is 29.

The metric that is evaluated after each entry elimination cycle is of the form

$$D = m \cdot \prod_{k=1}^{p} \left(\omega_k\right), \text{ for } \omega_p = \frac{1}{I_p \cdot p!}, \text{ and if } I_x = 0 \text{ we let } \omega_x = 1.$$

The metric is a function of p, the assigned critical rating for a deduction method, and I_p, the number of iterations each method completes that successfully inserts a candidate entry. If an algorithm were to use a method other than single candidate elimination, it will narrow the possible values affixed to that entry. The assigned critical rating increases as deduction methods become more advanced and require more computing power. This equation produces a range of difficulty from $0 \le D \le 2$. The following calculations examine the upper bound and lower bound of D for algorithm A;

$$\text{If } m = 54, I_p = 27, \text{ and } p = 1, \text{ then } D = 54 \cdot \left(\frac{1}{27 \cdot 1!}\right) = 2.$$

If $m = 29$, $I_p = 52$, and $p = 1$, then $D = 29 \cdot \left(\frac{1}{52 \cdot 1!} \right) \approx .5576923$.

This would be the case for the least and most difficult Sudoku puzzles generated by algorithm A, respectively. We define $2 \geq D > 1.6$, $1.6 \geq D > 1.2$, $1.2 \geq D > 0.8$, and $0.8 \geq D > 0.5576923$ to be the ranges of very easy, easy, medium, and hard puzzles created, respectively. Below are four examples of Sudoku puzzles with different difficulty levels.

Very Easy - User specified $D = 2$ Actual $D \approx 1.893$

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

Easy - User specified $D = 1.4$ Actual $D \approx 1.382$

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

Medium - User specified $D = 1.1$ Actual $D = 1.025$

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

Hard - User specified $D = 0.8$ Actual $D \approx 0.761$

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

Strengths

By definition, our logical elimination scheme produces a puzzle with exactly one solution. The algorithm can produce a significantly large subset of all possible Sudoku boards.

Limitations

The scope of difficulty levels produced by this algorithm is very constricted. Puzzles of the hardest difficulty level are in fact not any more difficult compared to an easy level puzzle for the average Sudoku player.

Algorithm B – The Improved Algorithm A

Board Generation

Boards in algorithm B are generated in the same fashion as A.

Randomization

Randomization operations in A are used again. Additionally, we implement the operations of minor row and minor column permutation and board transposition. Operations are applied a random number of times.

Entry Elimination

Entry elimination follows the same method as the simple model. Rotationally symmetric entries are again removed. The solver is applied to the resultant puzzle to ensure that it is solvable and the solution matches the original board. The maximum number of givens remains the same.

The Solver and Difficulty Indexing

Our solver is modified to include a second method of logical deduction, which considers the possibilities for each entry. We use the same difficulty metric as in A, but our formula will now contain a second critical rating and iteration count for the additional deduction method. Thus, we notice an increase in the range of difficulties offered, $0.23\overline{63} \le D \le 2$. Therefore, the difficulty ranges for very easy, easy, medium, and hard become $2 \ge D > 1.6$, $1.6 \ge D > 1.2$, $1.2 \ge D > 0.8$, and $0.8 \ge D \ge 0.23\overline{63}$, respectively.

Strengths

The elimination procedure uses more than one deduction method. As a result, the range of difficulties is extended and harder Sudokus can be created.

Limitations

The average puzzle generation time for this algorithm increases. The algorithm accounts for just two deduction methods. This algorithm still produces boards of permuted seed matrices—this is a small fraction of all possible boards enumerated under a uniqueness constraint.

Algorithm Complexity

The following time complexities for both algorithms are based on our defined difficulty metric.

$$\text{Algorithm A} - O\left(\frac{1}{n}\right)$$

$$\text{Algorithm B} - O\left(\frac{1}{n!}\right)$$

Conclusion

The algorithms provide a large number of randomly generated, unique puzzles where a user can specify one of four difficulties that we offer. The algorithms are capable of making a significantly large subset of all possible Sudokus. We can extend algorithm B to further analyze any completed Sudoku boards. Given the logical processes needed to solve a particular Sudoku, we can compute a Difficulty Index. Including more logical methods, such as X-Wing, Nishio, Stochastic, and other optimization algorithms, one would be able to generate puzzles of much greater difficulty since our algorithm is based on the number of iterations per deduction method.

References

1. Felgenhauer, B., Jarvis, F. (2005) *Enumerating possible Sudoku grids*, available: http://www.shef.ac.uk~pm1afj/sudoku/sudoku.pdf.

2. Russell, E., Jarvis, F. (2006), *Mathematics of Sudoku II*, http://www.afjarvis.staff.shef.ac.uk/sudoku/russell_jarvis_spec2.pdf.

3. Yates, J. *Sudoku Strategy Guide-Techniques for Solving Sudoku Puzzles*, available: http://www.chessandpoker.com/sudoku-strategy-guide.html [accessed 15 February 2008].

Appendix A

This is some seed matrix with entries represented by some A_n, B_n, or C_n .

A1	B1	C1
A2	B2	C2
A3	B3	C3

If the Sudoku board is generated by the elementary permutations of this seed matrix, we have;

A1	B1	C1	A2	B2	C2	A3	B3	C3
A2	B2	C2	A3	B3	C3	A1	B1	C1
A3	B3	C3	A1	B1	C1	A2	B2	C2
B1	C1	A1	B2	C2	A2	B3	C3	A3
B2	C2	A2	B3	C3	A3	B1	C1	A1
B3	C3	A3	B1	C1	A1	B2	C2	A2
C1	A1	B1	C2	A2	B2	C3	A3	B3
C2	A2	B2	C3	A3	B3	C1	A1	B1
C3	A3	B3	C1	A1	B1	C2	A2	B2