Generating Sudoku Puzzles

Isaac Bergman, Ben Lamkin, Cullen Ring, Jack Zickrick

April 28, 2023

Summary

Sudoku has attracted millions of players[1], and its presence is a staple of newspapers, so generating an appropriate puzzle for a wide range of users from the novice to the expert is worth investing time and research into. Therefore, we have developed a set of metrics to define a range of difficulties and wrote an algorithm to generate a puzzle for the desired difficulty. Our algorithm guarantees that each solution is unique, allowing for the enjoyable experience of methodically applying pure logic toward a solution.

With respect to the design of our algorithm, there were a few requirements for our puzzles. First, we would stick with only the basic rules of the typical 9x9 Sudoku puzzle. If we chose to add in more rule sets, then we would be making multiple generating algorithms and it would be less accessible to the general person. We also wanted our puzzles to all be unique. This way a set solution can be provided, and if someone is attempting to solve one, they can do so without worrying about working towards multiple solutions at once. We also wanted to be able to choose a difficulty, and have a puzzle matching that be generated.

Our algorithm starts with a Sudoku solution. It randomly removes a cell, and after each removal, attempts to solve it using solving techniques. If a solution is found, it will continue removing cells until no more cells could be removed, or a puzzle of desired difficulty was found.

We also developed metrics to rate the difficulty of a puzzle. We wanted to create the four ratings of Easy, Medium, Hard, and Expert. We decided to determine the appropriate rating using a combination of the initial cells in the puzzle, the difficulty of the hardest technique required, and the variety of techniques required. We created difficulty score based on an equally weighted average of the three components on a range of zero to ten.

However, satisfying these requirements came at a cost. Our algorithm does not guarantee that it will generate a puzzle of the desired difficulty, which factors more heavily at higher difficulties. We decided to accept this drawback as preferable to compromising our definition of difficulty.

Contents

1 Introduction **3**

2 Definitions **3**

3 Sudoku Difficulty **4**
3.1 Number Of Givens 4
3.2 Breadth Of Techniques 5
3.3 Depth Of Techniques 5
3.4 Technique Difficulty 5
3.5 Algorithm for Difficulty 6

4 Algorithm **7**
4.1 Uniqueness Guaranteed 7
4.2 Failure Rate 9

5 Implementation **10**
5.1 Algorithm Prototypes 10
5.2 Technique Difficulty Ratings 11
5.3 Puzzle Difficulty Ratings 11

6 Strengths And Weaknesses Of Our Model **12**

7 Future Work **13**

1 Introduction

Gas brings customers to gas stations, and in turn the station takes opportunity of the customers time to vend its own secondary goods and services. Likewise, Sudoku draws in millions of players to platforms like newspapers, magazines, and phones apps to generate similar market opportunity[1]. Thus at first glance the study of Sudoku is a problem of economics.

But a closer look reveals the beating heart of Sudoku to be a problem of mathematics as well. It asks whether a Sudoku can be solved as fast as its solution can be checked. To provide a definitive answer for this would not only qualify the researcher for an award of one million dollars, but could catapult this era into a new age of mathematics. We aim to take only a small step towards this, first by defining varying levels of difficulty for a Sudoku puzzle, and then by creating an algorithm that can generate puzzles of these different difficulties.

It is important to first lay out the terms of Sudoku. A solution to a Sudoku puzzle is a filled 9x9 grid such that no row, column, or 3x3 submatrix contains a duplicate value using the numbers 1-9. The puzzle begins partially filled, with many cells in the grid being blank; the task of the solver is to deduce the numbers needed to fill in the blanks to arrive at a valid solution.

With the groundwork for Sudoku in place, we now move to discuss our problem and our process. Our problem is to develop metrics to measure the difficulty of a Sudoku puzzle and to develop an algorithm that generates puzzles for four difficulty levels: Easy, Medium, Hard, and Expert.

2 Definitions

A Sudoku puzzle is considered unique if there is only one way to fill the empty cells to create a valid Sudoku puzzle. If there are multiple potential valid solutions for a particular puzzle, the puzzle is considered non-unique. Since non-unique puzzles require guesswork to solve, most players assume that a published puzzle will have a unique solution, and so, we shall aim to only produce unique puzzles.

The non-empty cells of the initial puzzle are referred to as givens or clues. We define a Sudoku puzzle to be minimal if no more givens can be removed from the puzzle while maintaining a unique solution. Note-worthily, the

smallest minimal Sudoku possible has been proven to be only 17 givens[2].

While solving a puzzle, it is often useful to track the remaining potential values for a cell. These are often referred to as candidates.

A solving technique is a sequence of logical steps in a Sudoku that often appear in a pattern. They can be used either to determine the exact value of a cell or to eliminate candidates from a cell or cells. For example, the "naked single" technique simply recognizes that if only candidate remains in a cell then that candidate must be the value of the cell.

3 Sudoku Difficulty

Any algorithm that generates puzzles of a particular difficulty must rely heavily on that definition of difficulty. After serious thought, we decided on using the number of givens, the breadth of techniques required, and the hardest technique required to solve the puzzle to quantify difficulty on a scale of zero to ten.

3.1 Number Of Givens

Givens are the most intuitive definition of difficulty available. Givens represent the amount of information the puzzles provide you at the start, and therefore, the more givens exist the less information the solver needs to deduce. Thus, as the number of givens decreases the difficulty tends to increase.

To quantify the number of givens into a score of zero to ten, we broke the range of possible number of givens into five equal segments, taking advantage of the fact that no Sudoku can exist with fewer than 17 givens[2]. We rate each segment with a constant score, linearly increasing the score as the number of givens decreases.

Table 1: Given's difficulty ranges.

Givens	
Range	Difficulty Rating
81-69	0
68-56	2.5
55-43	5
42-30	7.5
29-17	10

3.2 Breadth Of Techniques

Next, we considered how the breadth of techniques required might increase the difficulty of the puzzle. Given that the typical approach to solving a puzzle involves scanning the board repeatedly while trying to spot the next solving technique to apply, we recognized that this process requires having a large repertoire of solving techniques. So we decided to incorporate this as our second component of puzzle difficulty.

To quantify it, we decided to represent this as the percentage of techniques used in a given puzzle out of the total number of techniques the solver implementation knew, and then we scaled this value to be between 0 and 10.

3.3 Depth Of Techniques

Finally, given that when solving many a puzzle, the chief difficulty spike comes when the next solving technique is the most difficult technique to spot, we decided that this would constitute our final component of puzzle difficulty. To quantify this component, we represented it as the difficulty value of the most difficult technique used to solve the Sudoku.

3.4 Technique Difficulty

Technique difficulty is both highly subjective and highly qualitative, and our difficulty heuristic is dependent on an objective quantitative rating. Thus technique difficulty poses a challenge to determine puzzle difficulty.

We decided to approach this challenge by assuming that technique difficulty would be rated solely on how hard it is to both identify an opportunity

for the technique and apply it. We considered how much conceptual overhead existed based - for example - on the number of cells involved. We rated this on a scale of 0 - 10, using the planning poker strategy to come to a consensus and reduce the subjectivity in the rating.

Figure 1 represents the distribution of techniques for our implementation of our algorithm.

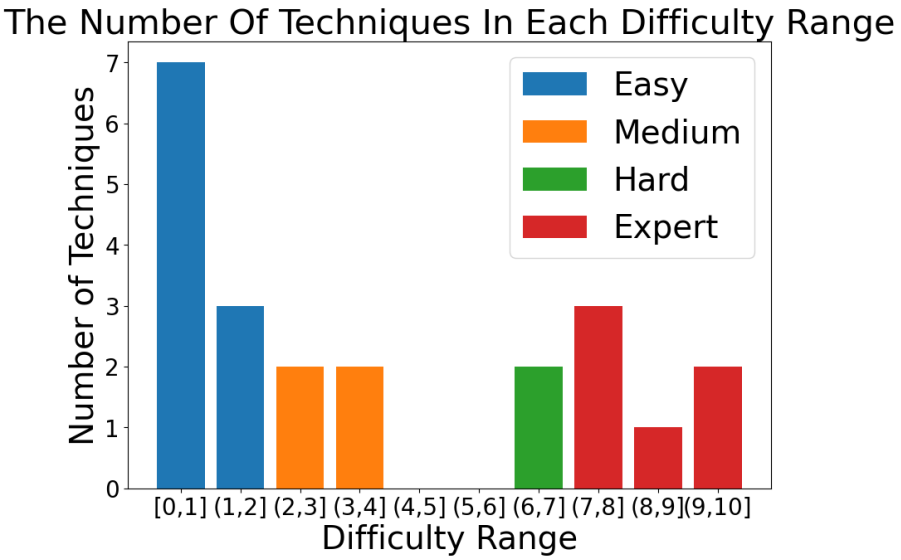

Figure 1: Histogram of Technique Difficulty

3.5 Algorithm for Difficulty

Finally, we turned our attention to how we needed to bring those components together into a final quantitative value for puzzle difficulty. We decided that none of these components deserved an unequal weighting, so they have been each assigned a equal weight of $\frac{1}{3}$. In equation 1, we show the final equation where P_{uzzle} represents the difficulty of the puzzle overall. B_{readth} represents the variety of difficulties required, M_{ax} the maximum technique difficulty, and G_{ivens} the number of givens in the puzzle as discussed in sections 3.2, 3.3, and 3.1 respectively.

$$P_{uzzle} = \frac{1}{3}B_{readth} + \frac{1}{3}M_{ax} + \frac{1}{3}G_{ivens} \tag{1}$$

4 Algorithm

Using all of these components, we finally propose an algorithm to generate Sudoku puzzle given the desired difficulty.

Algorithm to Generate a Puzzle

```
while not puzzle.is_of_desired_difficulty() and any
    untested_cells():
  cell = random.choose_cell(puzzle)
  cell.remove()

  solving_techniques = solver.solve(puzzle)

  if not puzzle.is_solved():
    cell.add()
  else
    givens -= 1
    puzzle.update_difficulty(max(solving_techniques), givens)
  endif
endwhile
```

This algorithm can work well with a generic solver. The solver can use any solving techniques that guarantee uniqueness, and the generator will adapt. The only restriction on the solver is that it applies easier techniques before harder techniques since frequently a harder technique can solve in lieu of the easier technique and thus distort the puzzle difficulty.

4.1 Uniqueness Guaranteed

Now it's worth considering how this algorithm guarantees uniqueness. Each solving technique brings with basic assumptions like the definition of a Sudoku puzzle, and at its core it consists of logical steps that force a conclusion. Thus there cannot exist any other permutation of the puzzle that violates that logic, much like a mathematical proof.

Consider for example the bi-value cell forcing chain technique. The technique starts with a bi-value cell, which is a cell that has exactly two candidates remaining. It considers the implications of the cell being the first value, and then the second. If in both cases the chain of implications result in a particular candidate being removed from a particular cell, that candidate can

be removed without knowing the solution to the bi-value cell.

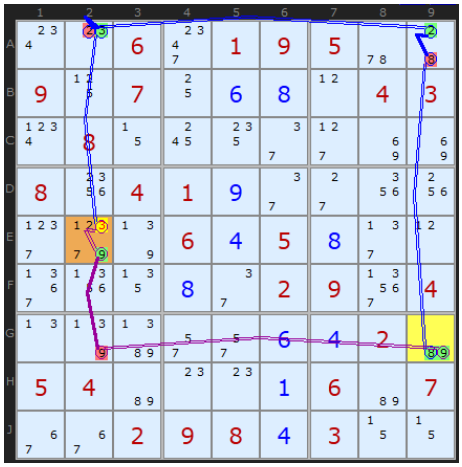

Figure 2: Example of a Bi-Value Cell Forcing Chain[4]

Consider cell G9 in Figure 2. It contains only the candidates 8 and 9, rendering the cell an option for the bi-value cell forcing chain technique. There are correspondingly two case scenarios to consider. If G9 is an 8, then A9 cannot be an 8, forcing it to be a 2. This forces A2 to be a 3, which correspondingly eliminates the candidate 3 from cell E2. Therefore, if G9 is an 8, then E2 cannot be a 3. If G9 is a 9, then G2 cannot be a 9. This leaves the only cell in column 2 with 9 as a candidate to be E2, which means it must be 9. Naturally, this prevents E2 from being a 3. Therefore, if G9 is a 9, then E2 cannot be a 3. Therefore, if G9 is an 8 or a 9, then E2 cannot be a 3. Therefore, in this board there can exist no solution where E2 being 3.

This illustrates how uses a sequence of rigorous techniques can turn into a mathematical proof that forces a puzzle into a particular solution. If you attempt to use the solver on a non-unique puzzle, then guesswork is required to arrive at a solution, which the solver will not employ. Therefore, using the solver on a non-unique puzzle results in a puzzle which has not been fully solved, which is rejected by our algorithm.

This implies that our solver can employ nearly any solving technique available, but there does exist a class of techniques that will not work. The techniques in this class assume that the Sudoku puzzle will result in a unique puzzle. If the solver uses these techniques, our algorithm will create a tautology establishing uniqueness, which will correspondingly fail to prove unique-

ness.

It has been hypothesized that there exists alternative but more complicated logical sequences that force identical outcomes but that do not rely on assuming uniqueness[5]. So far, this has not been rigorously proven.

4.2 Failure Rate

It is worth noting that this algorithm can fail in two ways. Firstly and less frequently, if by chance it happens to generate a puzzle of greater difficulty than desired, it has no mechanism for course correct, and it will continue to remove cells until it reaches a minimal Sudoku. However, this relies on choosing a cell whose presence makes the puzzle too easy and whose absence makes the puzzle too hard. This is a rare case scenario, but it is worth noting.

More importantly, it could choose the wrong cells to remove and reach a minimal Sudoku without achieving the desired difficulty. Naturally, this is an issue that predominantly affects the hard and expert difficulties.

While a poor choice of difficulty algorithm can cause the success rate to drop, after extensive trial and error we came to the conclusion that the core success rate is the success rate of the algorithm itself. Although it is possible to design a difficulty algorithm that forces a successful generation - such as the use of a givens threshold which rates a puzzle at the desired difficulty once passed, we decided that the use of such an algorithm would only serve to obscure the failure rate of the algorithm and to dilute the definition of difficulty.

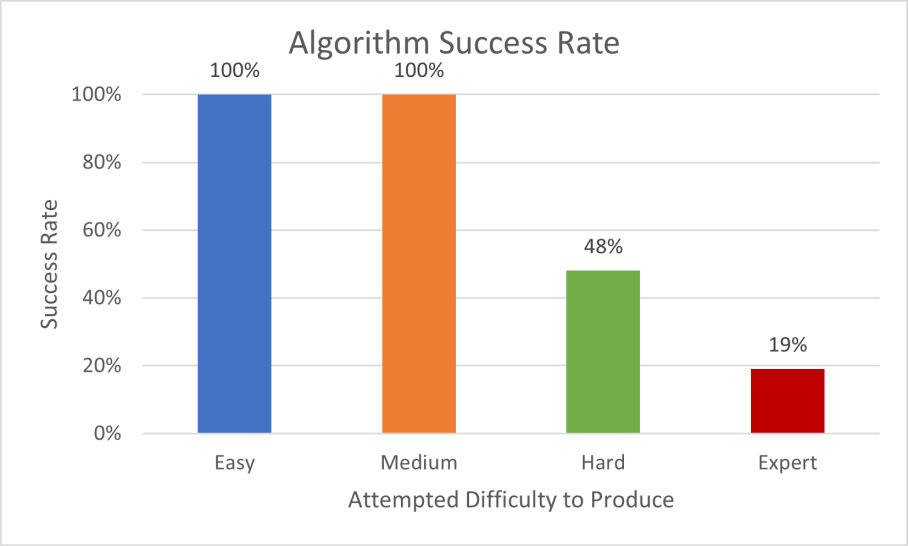

Figure 3: Success rate of the algorithm when asked to produce a Sudoku of a given difficulty

Therefore, Figure 3 represents how often our generation algorithm successfully creates a puzzle of the desired difficulty. The percentages were computed by attempting to generate one hundred puzzles of each difficulty and measuring the quantity of successes.

5 Implementation

5.1 Algorithm Prototypes

Given some of the interesting issues and considerations that arose in implementing the various algorithm prototypes as well as the final algorithm, we shall now discuss the lessons learned.

Our initial prototype similarly started with a Sudoku solution and likewise removed cells at random. It used a backtracking solver to verify that a unique solution existed. However, it offered little control of the generation process, and it provided us with few parameters to derive a definition of difficulty from. Therefore, we discarded it.

From there we experimented with a prototype that used solving techniques and started from an empty board. It would randomly choose a cell

and set its value to be one of its candidates, and then it would check for validity in a similar manner to our final approach. However, we discovered that it was highly susceptible to randomness that would forced it into a rut. Forcing it out of its rut would cause complexity to skyrocket, so we discarded this prototype.

We finally arrived at our final algorithm. We decided to outsource the implementation of the various techniques our solver used to a third-party[3], which introduced noteworthy challenges. For a while, our generator failed to produce unique puzzles despite the strong theoretical proof. The ultimate cause was not a logical fallacy in our proof, but it was rather a faulty implementation of the bi-value cell forcing chain-solving technique.

5.2 Technique Difficulty Ratings

To rate the difficulty of our chosen techniques, we considered each technique individual, estimated a difficulty value, and then discussed amongst ourselves where each value should belong. Table 2 compiles our results.

Table 2: Technique Difficulty Ratings

Technique	Rating	Technique	Rating
Naked Singles	0	Y-Wing	3
Naked Pairs	0.5	Jellyfish	4
Naked Triples	0.5	XYZ-wing	4
Hidden Singles	0.5	2-Cell Subset Exclusion	7
Hidden Pairs	0.5	3D Medusa	7
Naked Quads	1	Dual Medusa	7.25
Hidden Triples	1	3-cell Subset Exclusion	7.5
Hidden Quads	1.5	Dual unit Forcing Chain	8.25
Unit Intersection	1.5	Nishio Forcing Chain	9.5
X-Wing	2	Anti-Nishio Forcing Chain	10
Swordfish	3		

5.3 Puzzle Difficulty Ratings

To group puzzles in difficulty ranges, we broke the range of difficulty score into four groups. We ran the generator on large batches of puzzles and inspected the output. Based on puzzles, we adjusted the divisions and

iterated until we were satisfied by the ranges. Table 3 displays our final results.

Table 3: Puzzle Difficulty Ranges

Classification	Difficulty Interval
Easy	[1, 2.5]
Medium	(2.5, 4]
Hard	(4, 6]
Expert	(6, 10]

6 Strengths And Weaknesses Of Our Model

Overall, we feel that our model serves its purpose quite well. The majority of issues we had resulted from our implementation and choices we had to make along the way.

A strength of our algorithm is the balance it strikes between being easy to understand and staying true to what make Sudoku's difficult. There do exist simpler methods for creating puzzles but they often rely on removing the idea of a solving technique and relying totally on the number of givens. While a decent approximation of difficulty, we preferred to increase complexity to introduce better metrics of difficulty in order to be more confident in our assessment of the final difficulty of the puzzle. Nevertheless the process is still easy to model and to understand as demonstrated in Figure 4.

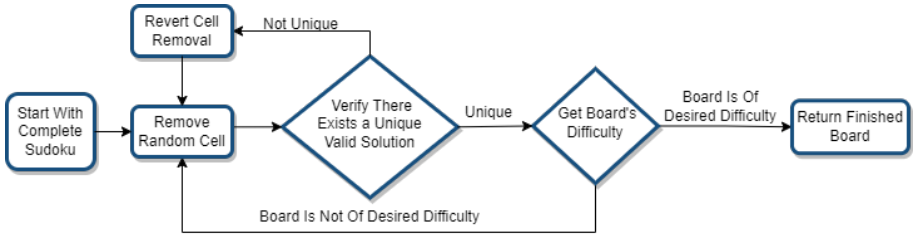

Figure 4: Algorithm Flowchart

Another strength of the algorithm is it's quite elegant guarantee of uniqueness by exercising the logic underlying the solving techniques used by our solver. This poses in contrast to many other generators which guarantee

with some variation of a brute-force solver that tests and invalidates other potential solutions.

On the other hand, one of the greatest weaknesses our model suffers from is the inherent subjectivity of difficulty. We rated techniques based solely on the challenge in finding and applying, but a thorough understanding is required to accurately gauge difficulty. We also reduced the subjectivity by group discussion, but it still remains present. Furthermore, our classification on how to partition and label is also highly subjective.

It also struggles to generate puzzles using a wide variety of techniques, relying on a stroke of good luck from the random number generator. This is likely the most significant cause of the algorithm's tendency to fail to generate higher difficulties.

7 Future Work

Future development should conduct studies of Sudoku players regarding the difficulty of puzzles and techniques. This would refine the model to significantly reduce the subjectivity currently present and serve as a better generator.

The algorithm used for difficulty score could also be investigated, as other components could be derived from the givens and techniques which could have meaningful impact on a difficulty score. The equal weighting scheme currently implemented may be worthwhile reconsidered. Since givens are more crucial for defining an easy puzzle but less important for expert puzzles, a variable weighting scheme could work well.

Our solver is also unable to harness uniqueness techniques, due to the tautology it creates. Future research may be able to prove that the uniqueness techniques can be solved with alternative techniques that do not assume uniqueness, and this would enable our solver to use the simpler uniqueness techniques.

Finally, it would worth iterating on our algorithm to either reduce or completely eliminate the potential for failure that it is prone to, especially when generating higher difficulties. This could require designing algorithms that can force the use of certain techniques or certain classifications of techniques, which would enable a highly customizable generation of Sudoku puzzles.

References

[1] Jon Hurdle. *Sudoku players hold first national championship*. URL: https://www.reuters.com/article/us-usa-sudoku/sudoku-players-hold-first-national-championship-idUSN2033062920071020.

[2] Gary McGuire, Bastian Tugemann, and Gilles Civario. *There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration*. URL: https://arxiv.org/pdf/1201.0749v2.pdf.

[3] Remy Oukaour. *Sudoku*. URL: https://github.com/roukaour/sudoku.

[4] Andrew Stuart. *SudokuWiki.org - Cell-Forcing Chains*. URL: https://www.sudokuwiki.org/Cell_Forcing_Chains.

[5] Sudopedia. *Uniqueness Controversy - Sudopedia*. URL: https://www.sudopedia.org/wiki/Uniqueness_Controversy.