Algorithms to Determine the Difficulty of Sudoku Puzzles: Project Specification

Matthew Bullock - 1801903

1 Problem

1.1 Context

This project is focused on sudokus, a popular type of logic-based puzzle. Sudoku puzzles consist of a grid of 81 cells grouped into 9 boxes. The aim is to insert a number from 1 to 9 into each cell such that the following three conditions hold:

1. Each number can only appear once in each row
2. Each number can only appear once in each column
3. Each number can only appear once in each box

An initial sudoku grid consists of some cells that have numbers in and some that have been left empty for the player to fill in. For a set sudoku puzzle to be valid it must be able to be solved using only logical reasoning and must give a unique solution. An example of an initial sudoku grid is below.

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

The sudoku puzzle is completed when every cell has been filled in and is valid according to the 3 conditions stated above. Below is the completed version of the previous sudoku puzzle.

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

1.2 Sudoku Difficulty

The difficulty of sudoku puzzles are usually rated as one of easy, medium, hard, and expert. There is no set definition for what gives a sudoku a certain difficulty rating, but there are 3 commonly used methods of determining this.

1. Most complex technique required
2. Number of given cells
3. Time taken to solve

Each of these methods has limitations, which are explained below.

1.2.1 Most complex technique required

There are many techniques that might be required to solve a sudoku, which are explained fully in `http://www.sudokuoftheday.com/techniques` [1]. Some of these techniques need more cells and conditions to be used than others. These techniques are known as "more complex" than the others. The idea behind this method is that more complex techniques will be harder to spot and perform, making the puzzle harder.
One limitation of this is that it doesn't consider how many different techniques

are used. A puzzle requiring one complex technique repeatedly may be relatively easy, as the player would likely become more adept at spotting and using this technique. However, a puzzle using a variety of slightly less complex techniques may be harder overall, as each technique may only be possible at a specific point of solving the sudoku. The player would have no knowledge to guess which technique is possible at which point in the solve, so would likely have to check if every technique is possible, making it harder to spot the one correct technique.
Also, it would likely be easier to spot if a complex technique is required in a puzzle with few numbers filled in than in a puzzle with many numbers, as those extra numbers could distract the player from the required complex technique. Using this method, both these situations would be given the same difficulty rating, but are likely of different difficulties.

1.2.2 Number of given cells

This method bases the difficulty on how many cells are filled in in the initial sudoku, with fewer filled cells giving a harder puzzle.
This method has one key limitation, it doesn't consider the process of solving of the sudoku, it only considers how many numbers need to be found. A sudoku with 40 initially filled cells may require some very difficult techniques to be solved, whereas a sudoku with 20 initially filled cells may be able to be solved using one very simple technique repeatedly.

1.2.3 Time taken to solve

This method focuses on the average time the average player would take to solve the given sudoku puzzle.
The main limitation with this is that it requires the puzzle to be completed multiple times before a difficulty rating can be given, and this still might not represent the time the average player would take.

The aim of this project is to explore different methods for determining the difficulty of sudoku puzzles in order to remove or reduce the impact of these limitations.

2 Objectives

The project can be broken down into 2 key parts:

2.1 Creating algorithms to determine the difficulty of a sudoku

The aim is to develop multiple algorithms that will take a sudoku as input and output the difficulty rating of this sudoku. This difficulty rating will either be

one of easy, medium, hard, and expert, or will be a number that represents the difficulty of the sudoku. The algorithms will need to be implemented in code in order to test them, and the code will have to run in a reasonable time for this testing to be feasible.

2.2 Comparing the results of these algorithms to perceived difficulty

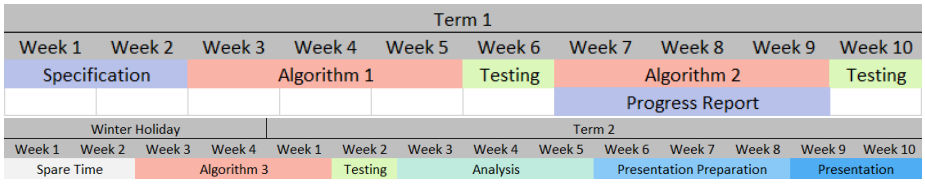
In order to compare the results to the human perceived difficulty of the sudoku, data from humans will need to be acquired. This data will then need to be processed and compared to the output of the algorithms, in order to see how well the algorithms determine the difficulty of each sudoku. The results of this will be analysed for each algorithm, including the currently used ones stated above, and comparisons will be drawn between them.

3 Methodology

Each algorithm developed will use a different approach to the problem in order to explore the problem more fully. The algorithms will first be planned out in pseudocode and then converted to program code for testing. During testing, the algorithms will be run using certain test sudokus and comparisons will be drawn with human test data to judge how well the algorithms determine the difficulty.

4 Timetable

This timetable is designed for developing 3 different algorithms, if the workload is too much or too little the timetable can be adjusted accordingly.



5 Resources

Some of the algorithms may require code for a sudoku solver to work. The chosen solver is by Kermalis[2] and is free to access via GitHub. This code may be adapted or used when required. This sudoku solver is written in C#, so further additions or adaptations will also be written in C# using Visual Studio.
Ideally, the data will be accessed online and will be publicly available. If this data is not sufficient, permission will be asked to have access to data that isn't

publicly available or data will be collected first hand via an interactive website. The files for the project will be stored in a git repository using GitHub. This is so that the contents are consistent across machines and previous versions can be accessed if required.

6 Legal, social, ethical and professional issues

The use of the sudoku solver by Kermalis is allowed provided it is acknowledged that the original code is by Kermalis and also any adapted code is clearly stated as such. Both these will be fulfilled, so no issues will arise from the use of this software.
If the data collected is anonymous and publicly available, there will be no ethical issues with using it. However, if this is not possible, an ethical review may be required for permission to use the data, which will be strictly anonymous.

7 References

[1] Sudoku Techniques, `http://www.sudokuoftheday.com/techniques`, Accessed September 2020

[2] Sudoku Solver by Kermalis, `https://github.com/Kermalis/SudokuSolver`, Accessed September 2020