1 Analysis Sudoku Maker/Solver

1.1 Introduction

A Sudoku puzzle is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 subgrids that compose the grid (also called ”boxes”, ”blocks”, ”regions”, or ”subsquares”) contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a unique solution. (reference: `https://en.wikipedia.org/wiki/Sudoku`)

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

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

1.2 Aims

The objective of this project is to create Sudoku puzzles with a specific *difficulty* for a user to solve with a GUI. To guide the user to a solution the program must be able to solve the puzzles in a systematic manner using a set of Sudoku *techniques*. This functionality will allow hints to be provided to the user (based on the appropriate puzzle solving *technique* and will ensure that the puzzle is solvable using a well defined set of techniques.

1.3 Background

I chose this task because I enjoy solving Sudoku puzzles and have found a lack of Sudoku applications that use sophisticated solving techniques and have a rich set of user features. I am particularly interested in puzzle solving strategies such as `http://www.sudokuwiki.org/Jelly_Fish_Strategy` and `http://www.sudokuwiki.org/WXYZ_Wing`. The user features of interested include creating Sudoku problems at various levels of difficulty, providing hints to solve the puzzle at any point in the game and other user support tools to make the puzzle solving experience more enjoyable.

1.4 Research

1.4.1 Similar Programs

`http://www.sudokuwiki.org/sudoku.htm`

This is an expert-level Sudoku problem maker/solver with a cluttered and confusing user interface. However, what sets this maker/solver apart is the level of puzzle solving techniques that can be selected to complete a Sudoku problem and the documentation provided. These puzzle solving techniques form the basis of my analysis of the difficulty level of Sudoku puzzles which is used in my solution.

`https://play.google.com/store/apps/details?id=com.brainium.sudoku.free&hl=en`

This is a Sudoku problem maker that is available from the Google play store. The special feature provided by this application is an intuitive help button. I plan to use a similar approach to providing help that operates at the different

levels of difficulty provided by my application.

Others

<List of urls> Many other Sudoku makers/solvers are available on the Internet. However, they have a similar set of basic features that allow a user to solve a limited set of Sudoku puzzles. The two applications described above provide distinct ideas which are used in my solution.

1.5 Anticipated Difficulties

`http://stackoverflow.com/a/7280623`

Unless P = NP, there is no polynomial-time algorithm for generating general Sudoku problems with exactly one solution. In his master's thesis, Takayuki Yato defined The Another Solution Problem (ASP), where the goal is, given a problem and some solution, to find a different solution to that problem or to show that none exists. Yato then defined ASP-completeness, problems for which it is difficult to find another solution, and showed that Sudoku is ASP-complete. Since he also proves that ASP-completeness implies NP-hardness, this means that if you allow for arbitrary-sized Sudoku boards, there is no polynomial-time algorithm to check if the puzzle you've generated has a unique solution (unless P = NP).

This discussion from stackoverflow suggests that the fastest way to generate a Sudoku puzzle (ie a partially completed grid that a user can solve) from scratch (ie starting with a blank grid) is by brute force (try every combination of values) as the puzzles are intractable. As this is a very time consuming process (there are $9\hat{8}1$ possible combinations although clearly many of these are invalid) a more efficient method of generating Sudoku puzzles was required. The method chosen to create new Sudoku puzzles is to work backwards from a completed Sudoku grid. These completed grids are created by a trial and error, back-tracking algorithm. Once completed grids are available individual tiles can be blanked out and the Sudoku Solver can be used to determine how difficult the puzzle has become. If the puzzle has become too difficult the algorithm can back-track one step and attempt to remove an additional tile to achieve the level of difficulty required. This process is continued until a puzzle with the appropriate set of characteristics (level of difficulty, number of remaining tiles) has been created.

`http://zhangroup.aporc.org/images/files/Paper_3485.pdf`

This paper describes how all the trivial Transformations of an individual Sudoku Grid can be computed. (Re-labelling, swapping rows, columns, rotations etc.). This information is used to dramatically reduce the number of Sudoku Grids that are hosted on the server and also increase the number of Sudoku puzzles that can be presented to a user to be solved. This is because a computationally trivial change to a Sudoku grid will look like a completely different puzzle to the un-tutored eye.

1.6 Objectives

1. To create a program that generates solvable Sudoku puzzles with the maximum number of blank tiles having only one solution.
2. To create Sudoku puzzles with a certain level of Difficulty (defined by the methods required to solve them)
3. To create an automated method of solving Sudoku puzzles.
4. To provide a simple and elegant graphical user interface (gui) that provides a rich set of tools to a user(s) to solve Sudoku puzzles.
5. To provide relevant and useful hints that will assist the user in solving the Sudoku puzzles.
6. To support the user importation of Sudoku puzzles from other sources (eg Times newspaper).
7. To create a database of completed Sudoku Grids which can be used as the basis for Sudoku puzzles.
8. To provide, maintain and display a high scores table (moves/mistakes/time taken) so that users can track their puzzle solving performance.

9. To support a wide selection of Sudoku puzzle solving techniques in puzzle formation and solution.

1.7 Proposed Solution

I decided to split the solution to these objectives into three distinct modules. These modules are the client application that contains the user interface and resides on the client machine; the Sudoku puzzle generator that resides on the server and the administration module that also resides on the server. The administration module manages the communication between the components, the high-score table and provides access to the relational database that stores the Sudoku puzzles, high-score information and other ancillary information.

The client application will contain the options menu and the main Sudoku playing area. The options menu will allow the user to set the features of puzzle playing experience. These features include the general Difficulty level of the puzzles presented for solving, the specific Sudoku techniques that will be required to solve puzzles presented to the user and an option to permit the user to enter a Sudoku puzzle of their own from another source. If this final option is selected, a blank Sudoku grid will be presented to the user so that the initial values of the external Sudoku puzzle can be entered.

Once the button to start play on the user interface has been pressed, the predefined values of the Sudoku puzzle will be locked in place and the user will be able to input Values into the Sudoku Grid. The user can write Dummy Values (which appear in a small font) into blank tiles to assist the user in solving the puzzle, ask for help (which will be a step-by-step guide on what Technique they should be used next) and a save and exit option. If the user chooses to save and exit, the next time the application is started, they will be asked whether they want to continue with the previous puzzle, if this option is selected the previous puzzle will be retrieved from the local storage and presented to the user.

The server will host the puzzle generator and the administration module. The administration module uses a MySQL database and uses a Raspberry PI as the hardware platform. This provides a cost effective database environment to support the server processing.

1.8 Key Words

Sudoku Grid

A 99 grid of tiles that can each hold a single digit between 1 and 9.

Sudoku puzzle/problem

A Sudoku puzzle/problem is a Sudoku Grid with minimal values pre-filled in to make it only have one solution for the user to solve

Completed puzzle/ Sudoku seed

A Completed puzzle/ Sudoku seed is a Sudoku puzzle/problem that has every single value filled in and obeying the Sudoku laws

Values (with reference to Tiles)

A Value is the Value that is stored within the tile

Dummy Values (with reference to Tiles)

A Dummy value is a Technique where the user lists all the possible Values for a Tile

Difficulty (with reference to Sudoku Grid)

Difficulty, in the scope of my project will define the level of Techniques that is needed to use to complete the puzzle

Techniques

A technique is a method of solving a Sudoku puzzle i.e. Dummy Values, BUG, X_Cycles, Unit_Forcing_Chains & Sword_Fish_Strategy

Transforming (a Sudoku Grid)

Transforming or Transformations of Sudoku Grids are aprocess, which converts completed Sudoku Gridinto a new one. I.e. a rotation of $90°$.

A uniqueSudoku seed (with reference to the SQL server)

A Sudoku seed that through no Transformation can result in an uploaded seed