Initial Plan - 'Killer Sudoku' Solver

CM3203 - One Semester Individual Project

Thomas Petty - 1119707

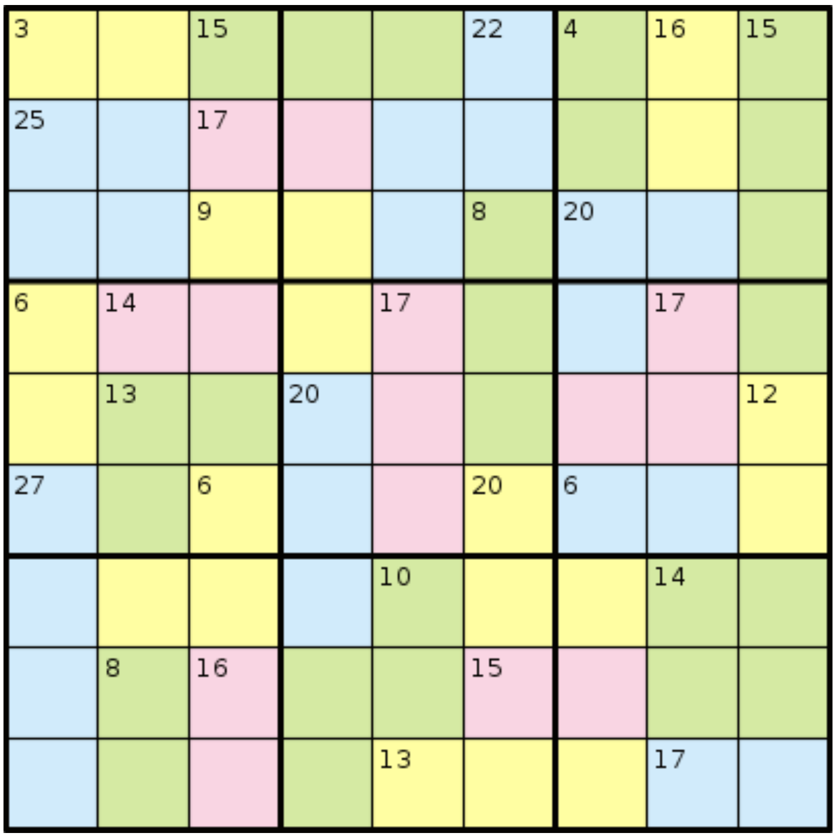
Supervisor: Ralph Martin

Moderator: Richard Booth

January 2016

Project Description

The aim of this project is to write a piece of software which is capable of solving a 'Killer Sudoku' puzzle. A Killer Sudoku takes the format of a traditional Sudoku puzzle, with an *n* x *n* grid of cells which contains *n* squares, called cages, each containing *n* cells. Each row, column and cage must contain each number between 1 and *n*. In traditional Sudoku, some of the cells in the initial puzzle already contain values, but in Killer Sudoku no such cells exist, and instead additional cages are defined, which must contain distinct values which sum to a given value. An example puzzle is given below.



In this example, the coloured areas represent cages, with the number in the cage representing the sum of values it contains. In this project, I will create software to create and save Killer Sudoko puzzles, and then extend that software to be capable to solving such a puzzle. I will implement a number of different solving strategies in my solution, and I will perform a series of experiments using different combinations or orderings of strategies to determine which combinations are the most effective or run fastest.

Project Aims and Objectives

The following bullet points represent the primary objectives of the project, and what should be achieved to satisfy these objectives.

- **Create a single piece of software which allows for the creation and resolution of Killer Sudoku puzzles.**
    - A graphical user interface should be provided to allow the user to easily input a puzzle.
    - A component should exist which can be enabled/disabled by the user, which updates the same GUI with the steps in a solution as it is being solved.
    - A model-view-controller architecture should be used, and the solving component should use a recursive algorithm, and a cache of sub-solutions to improve efficiency.
    - Facilities should be provided for the saving and loading of puzzles.
    - Multiple puzzle solving techniques and strategies should be implemented for the solver.
    - The software should support 4x4 grid puzzles as well as 9x9 grid puzzles, to allow to easier debugging/testing.
    - An object oriented programming approach should be used, to allow for the easy implementation and removal of different components.
- **Carry out experiments to analyse the efficiency and effectiveness of different solving strategies.**
    - Compile a test suite of puzzles, ranging in difficulty from easy to very difficult.
    - Perform tests which analyse the software's ability to solve puzzles of varying difficulty when using different combinations or orders of solving strategies.
    - Collate experiment results to show which strategies are most effective, and in which situations to software performed best, in terms of time taken to solve puzzles.

Work Plan

Below is the project timeline, which outlines the development process for the software, and how I will go about achieving the project objectives defined previously. A weekly supervisor meeting will be scheduled, to review the progress of the project and ensure that work has remained on track and within the scope of the project.

Week 1: 25th - 31st January 2016

- Meet with supervisor to discuss objectives, scope of project, rough work plan.
- Write Initial Plan: outlining project objective and work plan
- Deliver Initial Plan: 31st January.
- Background research: research killer sudoku puzzles; research solving strategies; attempt some killer sudoku puzzles myself.

Week 2-3: 1st - 14th February 2016

- Design main classes and methods for implementation.
- Design graphical user interface for creation of puzzles.
- Devise method of storing a puzzle in a file.
- Implement the GUI design, and the facility to save/load puzzles from files. Initially, implement interface for puzzles using a 4x4 grid.
- Test the software to ensure that puzzles and being saved/loaded correctly, and that changes made to a loaded puzzle are reflected in the file after it is saved.
- Modify implementation to support different grid sizes, with 9x9 as the default.
- Retest to ensure functionality has not be impaired.
- Compile test suite of puzzles of varying difficultly for use in testing and performance experiments.

Week 4-7: 15th February - 13th March 2016

- Meet with supervisor to discuss implementation of problem solving strategies (week 4): which strategies to implement first; how progress should be fed back to the GUI.
- Start by implementing rules to solve regular sudoku puzzles, then modify implementation to solve killer sudoku puzzles.
- Iteratively implement different problem solving strategies using object oriented programming.
- For each strategy, test it's ability to function as the only strategy (when possible), and in combination with other strategies.
- Implement functionality which allows the solver to display it's progress on the GUI.
- Test GUI progress feedback by comparing the progress displayed with an alternative form of progress feedback (e.g. command-line trace). Unit testing may also be used to test certain rules.

Week 8: 14th - 20th March 2016

- Buffer week: allow for any development overrun.
- Meet with supervisor to discuss work during easter break, outline for performance experiments.
- Formalise and document plan for experiments.
- If time available, carry out in-depth debugging of code to identify and remedy potential faults.

Easter Break: 21st March - 10th April 2016

- Conduct performance experiments, analysing and recording the times taken for different combinations of solving strategies to solve puzzles or varying difficulty.
- Document results of experiments, and write formal evaluation the results.

Week 9: 11th-17th April 2016

- Buffer week: allow for any overrun of experiments or evaluation.

- Meet with supervisor to discuss experiment results: what else can be analysed; further experiments which could be performed.

Week 10-12: 18th April - 6th May 2016

- Write Final Report: showing my solution to the problem; details of my design, implementation and testing; my evaluation of the experiments performed etc.
- Deliver Final Report and Project VIVA: 6th May 2016

References

"Killersudoku color" by Toon Spin (Toon81) - Own work. Licensed under Public Domain via Commons - https://commons.wikimedia.org/wiki/File:Killersudoku_color.svg#/media/File:Killersudoku_color.svg