Third Year Project 2013/2014

Full Unit - The Project Plan

A Sudoku Solver Using Constraint Satisfaction (& Other Techniques)

Thomas Paul Clarke

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Dave Cohen

Abstract

I have very little experience with constraint satisfaction and the maths involved has always been one of my weaknesses. Therefore, within the computer science course I have always wanted an opportunity to put this to a practical use so I can learn in the most effective way possible. By using it myself, I will be discovering an easier method of resolution, as well as learning more about its reason, potential and limits within the programs I will create.

I am expecting to achieve a program that will have a user interface that is very strong and attractive. There will need to be a way for the user to complete a Sudoku puzzle and also gaming options to either solve their own puzzle or a randomly generated one. This will allow the user to solve a puzzle independently on the program, whilst also using the program to complete it. This will teach me about creating a complicated GUI and let me handle a way to store and save all of the data that is put into the Sudoku. There are many ideas for this project, but my conclusive aim it to be able to port the program to an android phone and put it on the store so that it will be used.

Description of project reports

Term 1:

User Interface design for a Sudoku solver

This report will be made during the design stage of the project, where there will be many different ideas and ways to present the application. This will specify the detail of the user interface and what the user will need.

The report will consist of the drawings and ideas that will have been explored in terms of what the project's inputs and outputs are. It will allow a clear direction for the rest of the project and identify what needs to be created in order for the application to function.

The user interface design will be completed on paper and presented in the form of technical drawings. There will be several versions that will allow me to choose and adapt the interface design to what myself and the users need. Drafts made in the GUI program will be completed using third party software on Eclipse.

As an extension to this I will port the code to a mobile android device also allowing me to develop a different and more refined interface just for a mobile device. Conclusively, I will be designing two user interfaces that include the same functions simultaneously.

The features I will need to consider:

Computer GUI	**Mobile Device GUI**
Contain all information	Easy to navigate
Attractive design	Few touches as possible
Resizing	Slim layout
Splash screen	Small file size
Display all options	Sizes of screens

The use of Layout Managers for resizable GUI applications:

This report will be made when making the GUI and will have to be linked to the design stage as well. There needs to be ways to allow the program to be resized and user friendly at the same time. Therefore, the buttons and user components all need to be big enough to allow the user to see them but not to continue being as large when the window is reduced. To complete this, a layout manager is used, and Java uses the Swing and AWT classes, which provide layout managers in many forms. This will allow the user to have a range of button formats, which, for example, I will expect to be using the GroupLayout to achieve. Therefore, my text and other objects will be enabled together and kept in the same shape. Conclusively meaning that they will stay together and repel from moving closer to other options.

If a layout manager is not used, this could cause problems when the window is resized, as it will use the buttons relative positions instead of a unit based position.

Relevant literature
http://docs.oracle.com/javase/tutorial/uiswing/layout/using.html

Design Patterns

This will be done during the design stages when the UML's and the Class diagrams are made. This is due to the level of thought that needs to be given to the whole structure of the project. Especially how currently known design patterns might be used within the project to help and provide a more stable structure.

There are basic patterns that will allow my program to function more effectively. Ones I might consider are:

Model View Controller pattern: This allows the there to be 3 classes, one of which has the view (GUI) that the user will use and will display the outputs, the second will be the model which represents an object that is passed between the classes, the controller is in the middle and manages the object and the view controlling the data and the inputs.

Singleton pattern: This involves a class that creates one object every time it is initialised, which can be used when there is a restriction of how many objects need to be made. This could become useful with creating the correct number of blocks for the Sudoku.

Composite pattern: Allowing me to treat a group of objects in a similar way by creating a list of the objects, which allows them to be added or removed. Potentially this will provide very beneficial when there needs to be lots of groups within the program needing to be handled in similar ways.

Relevant literature
<http://www.tutorialspoint.com/design_pattern/design_pattern_quick_guide.htm>

Techniques used by human Sudoku solvers

The puzzle solving techniques will explore what the program will need to solve the Sudoku's. To which algorithms will be used and incorporated into my program to use this technique.

Sudoku puzzles are premade and solved using a variety of methods. These methods actually determine the difficulty level due to the knowledge of the algorithms to solve the puzzle. Humans use many techniques that can aid in solving the puzzle by finding which number the cell in question is can be found using these matching's:Horizontal, Vertical, Horizontal and vertical simultaneously, Completing rows and columns, All other numbers have been used in the connected cells, Elimination and using the possibilities, if they provide a certainty for the current cell.

Relevant literature
http://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/techniques
http://www.stolaf.edu/people/hansonr/sudoku/explain.htm
http://www.decabit.com/Sudoku/Techniques
http://www.sudoku-solutions.com/background.php

Constraint Satisfaction, particularly consistency techniques

This report will be explaining about the different kinds of constraints involved within the puzzle and how they can be manipulated to perform better.

The meaning of constraint satisfaction is a way of finding a solution to a problem by following a set of rules or constraints that all of the variables must abide to. Allowing there to only be one or a set of correct answers.

This relates to the Sudoku puzzle because the way it works is using restrictions on what each cell needs to satisfy e.g. each row has to have the numbers 1-9 as well as each block has 1-9, meaning that already there are two restrictions that could allow the program to check if a cell could be a number.

However when there are more complex puzzles the solver will need to be able to filter out which cells need which algorithms to solve them. This can be done using consistency techniques that are designed to slim down the speed of finding the cell by disregarding the incorrect ways immediately before wasting time on them.

These are the techniques I will be focusing on:
Node Consistency
Arc Consistency
K-consistency (Path Consistency)

Relevant literature
http://ktiml.mff.cuni.cz/~bartak/constraints/consistent.html
http://artint.info/html/ArtInt_79.html

Complexity, NP hardness and the Big O notation

As this is one of the most complex reports it will require the most background reading and understanding. The literature below will help my understanding and it can also help me with parts that I find most challenging. Therefore it will help me with the coherency and efficiency of the algorithms.

Relevant literature:
Big O notation:
Big Java, International Students Version by Cay S Horstmann
Time and Space Designing Efficient Programs by Adrian Johnstone and Elizabeth Scott
CS2860 Algorithms and Complexity, I 2011 By Anders Yeo

NP hardness:
Algorithms and Complexity 2, CS2870 By Gregory Gutin February 16, 2013

Description of Project Concept Programs

Simple colourful GUI with an active button:

This program will prove that I am able to create and code a working and correctly designed user interface that fulfils all of the necessary specifications that were set out. This will contain the simple buttons and forms the program will need.

I will be using the swing library that will give me a much easier way to create the interface and maybe some third party software to make the process more user friendly.

Data structures including, for instance, dynamic trees and priority queue, populated with large random data sets:

The program will consist of ways to exercise and use data structures that will need to be used in the final version by using dynamic trees to filter and sort information. For example, a priority queue, which is a way of storing data the data in a heap.

Relevant literature

http://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html

Game tree for Tic-Tac-Toe

This considers a two-player game where there are very limited options. This uses a game tree, which is a way of displaying every option that a user could possibly use, from the highest, which displays the first option at the head and below branches into the other options of choices where to put a token. It is used to judge what the next user, the computer user should do next. By having all of the possibilities it means that one can be chosen at random and this will seemingly give a human vs computer experience.

Relevant literature

<http://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html>

Eight Queens using Backtracking

Using the backtracking algorithm, in this case, means placing a queen in a column and checking for clashes with previously placed queens. In that column if a row is found without a clash the queens mark this row/column as part of a solution, if not found go back and return false.

```
1) Start in the leftmost column
2) If all queens are placed return true
3) Try all rows in the current column.  Do following for every
tried row.
```

```
a) If the queen can be placed safely in this row then mark
this [row, column] as part of the solution and recursively
check if placing queen here leads to a solution.
b) If placing queen in [row, column] leads to a solution then
return true.
 c) If placing queen doesn't lead to a solution then unmark
this [row, column] (Backtrack) and go to step (a) to try other
rows.
3) If all rows have been tried and nothing worked, return
false to trigger backtracking.
```

I have taken the above example from the following website. This is to support and strengthen my knowledge and background research to my final year project.

<http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/> last accessed: 20/09/2013

Timeline

Date	Deliverable	Explanation
Term 1		
WEEK 1 (23-29/9/13)		
10/9/13	Start project plan	
WEEK 2 (30-6/10/13)		
3/10/13	Write specification And layout reports	
4/10/13 2pm	Finish Project Plan	
5/10/13	Detailed Specification Written	
5/10/13	Design and plan the GUI	What are the inputs and outputs for the system
WEEK 3 (7-13/10/13)		
8/10/13	Program 1: Simple colourful GUI with an active button	
8/10/13	Report 1: User Interface design for a Sudoku solver	
WEEK 4 (14-20/10/13)		
17/10/13	Finish Program 1	
17/10/13	Finish Report 1	
18/10/13	Report 2: The use of Layout Managers for resizable GUI applications solvers.	
17/10/13	Program 2: data structures including, for instance, dynamic trees and priority queue, populated with large random data sets.	
WEEK 5 (21-27/10/13)		
27/10/13	Finish Program 2	
27/10/13	Finish Report 2	
	If feeling up to date start main project design	
WEEK 6 (28-3/11/13)		
28/10/13	Program 3: Game tree for Tic-Tac-Toe	
28/10/13	Reports 3: Design Patterns.	
WEEK 7 (4-10/11/13)		
10/11/13	Finish Program 3	
10/11/13	Finish Report 3	
WEEK 8 (11-17/11/13)		
11/11/13	Program 4: Eight Queens using Backtracking	
13/11/13	Reports 4: Constraint Satisfaction, particularly consistency techniques.	
WEEK 9 (18-24/12/13)		
24/11/13	Finish Program 4	
24/11/13	Finish Report 4	
WEEK 10 (25-1/12/13)		
25/11/13	Report: Complexity, NP hardness and the big O notation	
25/11/13	Report: Techniques used by human Sudoku	

25/11/13	Catch-up Week finish needed	
25/11/13	Start final project if possible	
1/12/13	Finalise reports & programs	
WEEK 11 (2-8/12/13)		
4/12/13 2pm	Term 1 Reports deadline	
7/12/13	Prep for review	
WEEK 12 (9-13/12/13)		
9-13/12/13	December Review Viva	
Holiday	Final project coding and design	
	Outline of report	
Term 2 13/1/14		
14/1/14	Continue final program design	
14/1/14	High-Level Design	
22/1/14	Final report draft start	
1/2/14	First Prototype Program	
22/2/14	Draft Report Finish	
10/3/14	Finish program coding	**Final Deliverables** • The program must have a full object-oriented design • A full implementation life cycle using modern software engineering principles • The program will have a splash screen and two other user interaction screen. • The program will have a Graphical User Interface that can be used to generate, load, save, solve and help with Sudoku solving. • The report will describe the software engineering process involved in generating your software and describe interesting programming techniques and data structures used (or able to be used) on the project.
12/3/14	Extra extensions	**Suggested Extensions** Porting the program to a mobile device Solving and benchmarking public domain sets of puzzles using XML based files to store and load Sudoku puzzles **Ideas of extensions:** Save results of time to database to have a leader board (Store puzzles as xml) Allow random Sudoku puzzles playable by user Solve puzzles user puts in Solve in different ways Using different algorithms
13/3/14	Review notes from Algorithms and complexity for final reports	
15/3/14	Testing of code	
20/3/14	Finish report and proof read	
20/3/14	Finalise code and create working file	
22/3/14	Specification of Testing Procedures	
26/3/14 2PM	Final Report	
26/3/14 2PM	Final Project Program	
27-28/3/14	Project Demo Day	

Risks Associated With Project

The risks that are associated within a project like this are quite generic with normal programming projects:
Risk: Time – the length of the project timeline was not enough to allow the project to be completed, or rushed.
Prevention: To have a timeline with set dates and objectives to keep the project stay on track and not fall behind.
Risk: Scope – The ideas for the project outgrew what was actually feasible

Prevention: Set the objectives and to keep to them, allowing the project to remain in the pre-set objectives and milestones.
Risk: Loss of work if there is some work that is lost due to a technical fault
Prevention: Manual backups of both projects and code will be made to hard drives to provide backups, as well as using SVN that will allow the data to be stored and this allows a backup to be made of the whole project so it cannot all be lost, also allowing working copies of code to be retrieved.
Risk: misdirection of work – Focusing on the wrong things e.g. The GUI instead of the main algorithms
Prevention: The timeline shows a clear layout that if stuck to should prevent this by knowing when to move on and when to make sure something is finished.
Risk: The end user not liking the product
Prevention: Work closely with the end user and agree on the goals and ideas at the start, also setting regular meetings to allow them to follow the project as well as see what is needed to be changed.
Risk: The program not working
Prevention: Throughout the project there will be many working parts of the code and there will be many versions of a working program that will show many functions if not all functions can be made to work.