full unit project - A sudoku solver using constraint satisfaction- project plan

Mark Harrison 29/09/2017

Supervisor Peter Robinson

Mark Harrison 2017

**Abstract of the Project Plan**

This plan presents an overview of a project on constraint satisfaction and how it can be used to solve Sudoku puzzles. The goal of the project is to produce proof of concept programs, reports and an application which can solve Sudoku puzzles.

Constraint satisfaction is a leading technology in AI for solving search problems. It can be seen as a related set of questions which each need to be answered. We cannot answer a question until the answers for the others are known. This model is used in game playing, scheduling and staff rostering.

The project will consist of 7 proof of concept programs and 10 reports. Proof of concept programs are necessary in order to prove the viability of an idea, feedback from the process can lead to better software or a change in design/direction. The initial reports I write will allow me to gather information in a structured way and help me to complete the final report. I have introduced further reports and proof of concept programs in order to facilitate a project which will go above and beyond the basic requirements of the project.

My proof of concept programs and reports will cover these topics;

* Algorithms and the various types used in constraint satisfaction
* Complexity and big O notation
* GUI building
* Design patterns
* Software engineering methodologies
* Professional issues for software designers/developers

The report will hopefully show that I have spent the time and effort to produce an exceptional piece of work. This will be invaluable to me when I use the report during interviews to show that I can;

* Produce work which is of a high standard
* Organise and plan the time I have to maximum efficiency
* Meet deadlines
* Use software design methodologies to best effect
* Regularly review/reflect and make changes as necessary
* Understand which knowledge sources to use

I chose this specific project because I have a strong interest in logical programming and how AI uses this to solve problems. This area of study will help me to perform better at work and understand how constraints on my work can be analysed efficiently and produce improved working practices.

I am hoping to continue my studies by undertaking a further year at Royal Holloway. Looking at the modules necessary to complete the extra year, I can see that writing another project is mandatory and worth 50% of the overall mark. Therefore, this project will prepare me for this and give some experience when it comes to writing another report.

**Proof of concept programs**

Core

1. Simple colourful GUI with a button – this will show I can program a basic GUI with limited functionality.
2. Data structures and priority Queues – this will show I can program and understand how the data structures will be used to hold and manipulate data.
3. Game tree for tic-tac-toe, how to prune search trees, estimating tree size – this will show I can understand basic logic, and optimise searching, also get an understanding of the size of a problem.
4. Eight queens using backtracking – This will show I can reason about techniques to solve constraint satisfaction problems of which there are many solutions.

Extensions

1. Mobile version of the GUI with button – I need to construct a basic GUI with a button using the Android SDK.
2. A simple application which can benchmark Sudoku puzzles – I need to write a program which can accept Sudoku puzzles and complete them with an accurate recording of the time taken to complete the puzzle.
3. An application which can accept different types of Sudoku Puzzles – A program which completes Griddler puzzles or Killer Sudoku, extending the original program.

**Reports**

Core

1. Design patterns – understanding design patterns is fundamental to producing easily maintainable applications and solve common design issues with OOP.
2. The use of layout managers for resizing GUI applications – So I can port to other devices and for different sized screens.
3. Constraint satisfaction, particularly consistency techniques – Covers the fundamentals of constraint satisfaction
4. Techniques used by human Sudoku solvers – Comparison between how we program to solve a problem and how humans do it will provide interesting findings which we can draw conclusions from.
5. User interface design for the Sudoku solver – How I used
6. Complexity, NP hardness, Big O notation – Show I understand the algorithms, running time and NP hardness.
7. Professional Issues – A compulsory part of the project to understand the impact my software has on moral and social issues.

Extensions

1. Techniques to solve CSPs- To understand that there are multiple ways in which CSP’s are solved and how some are better at certain problems.
2. Algorithms in CPSs – Covering GAC, Forward checking, DVO and backtracking.
3. Software design methodologies – Help me understand which methodology would best suit this project given the constraints.

**Timeline**

**First Term**

**15th September 2017** – Start Project Plan – reading

**20th September** - Meeting with Mentor to discuss project timeline, and issues/questions related to the project

**27th September 2017** – Meeting with Mentor to discuss the project plan, amendments, comments. Review project plan before it is submitted on Friday - send rough project plan to supervisor ahead of meeting on the 28th of September.

**28th September 2017** – Meeting with supervisor to discuss project plan and get advice on organising work to obtain a good grade.

**29th September 2017** – Project plan to be submitted by 23:59.

**9th October 2017** – Send reports and questions to supervisor in advance of meeting on the 11th of September, start data structures and priority queues program. Start Techniques to solve CSP’s report, Design patterns and Constraint satisfaction report.

**11th October 2017** – Meeting with supervisor to discuss current progress and ask questions. Start GUI with button program, Mobile version of GUI with button program. I think these two programs should be started together as the aims are closely related.

**16th October 2017** – Start Game tree for tic-tac-toe proof of concept program, user interface design report.

**23rd October 2017** – Send reports and questions to supervisor in advance of meeting on the 25th of October. Start Eight Queens using backtracking proof of concept program.

**25th October 2017** – Meeting with supervisor to discuss current progress and ask questions.

**6th November 2017** –email supervisor in advance with questions, start simple application which can benchmark Sudoku puzzles proof of concept program and an application which can accept different types of Sudoku Puzzles programs proof of concept program. These proof of concept programs need to be started after the game tree and eight queens programs as these have the basic logic that will be necessary to complete the later programs.

**8th November 2017** – Meeting with supervisor to discuss current progress and ask questions. Start Software design methodologies report and the use of layout managers report.

**13th November 2017** – Start Eight Queens using lookahead/forward checking proof of concept program.

**20th November 2017** - Send reports and questions to supervisor in advance of meeting on 22nd September. Start complexity, NP hardness and big O report.

**22nd November 2017** - Meeting with supervisor to discuss current progress and ask questions

**1st December 2017** – Interim programs and reports to be submitted

**3rd – 7th December 2017** – Interim review Viva

**Second Term**

**8th January 2018** – Send reports and questions to supervisor in advance of meeting on the 10th of January, start professional issues report and techniques used by human Sudoku solvers report.

**10th January 2018** – Meeting with supervisor to discuss current progress and ask questions.

**22nd January 2018** – Send reports and questions to supervisor in advance of meeting on the 24th of January 2018.

**24th January 2018** – Meeting with supervisor to discuss current progress and ask questions.

**12th February 2018** – Send draft report to supervisor for feedback

**16th February 2018** – Full unit draft report to be sent to Supervisor

**23rd March 2018** – Full unit final programs and report to be submitted

**Bibliography**

1. Ktiml.mff.cuni.cz. (2017). *Guide to Constraint Programming*. [online] Available at: <http://ktiml.mff.cuni.cz/~bartak/constraints/>

Useful introduction to material I had not previously studied, well setup, easy to follow and keep track of where I was. This is valuable as I need a solid base of knowledge in order to understand and reason about more advanced/complicated texts.

1. Constraintsolving.com. (2017). *CP-Tutorial « Constraint Solving*. [online] Available at: <http://www.constraintsolving.com/tutorials/cp-tutorial>

Nice basic examples on constraint satisfaction and problem fundamental concepts and categories. A Short text but again giving me basic solid information.

1. Ktiml.mff.cuni.cz. (2017). *Guide to Constraint Programming*. [online] Available at: <http://ktiml.mff.cuni.cz/~bartak/constraints/>

Fundamental concepts explained, the 8 Queens problem broken down and explained using multiple strategies to solve the CSP.

1. Daniel Hunter Frost (1997). Algorithms and heuristics for constraint satisfaction problems. [online] Available at: <http://www.ics.uci.edu/~csp/R69.pdf>

Read the abstract which combines separate ways of solving CSPs and will be useful for programming the algorithms.

1. Edward, T. (1993). Fundamentals of constraint satisfaction. London: Academic.

Recommended reading for the subject area. Explains the basic concept with simple understandable examples, searching, solving, arc consistency and graphs are also covered. Will be my ‘go to’ source of information during the project.

1. Cohen, D. (Unknown). CS2800: Software Engineering. [online] Available at <https://moodle.royalholloway.ac.uk>

Useful refresher in software design methodologies and design patterns, also code smells and how to avoid them.

1. BCS, the chartered institute for IT (2015). Code of conduct for BCS members. [online]

Available at: <https://moodle1617.royalholloway.ac.uk>

Expected behaviour when programming and the impact this has on the image of the profession.

1. Refsnes data(1999-2017). XML tutorial. [online] Available at <https://www.w3schools.com/Xml/>

Useful refresher and reference for XML

1. Horstman, C. (2010). Java for everyone. USA: John Wiley and sons.

Java reference book

1. Websudoku(2017). Websudoku. [online] Available at <http://www.websudoku.com/>

Resource for test Sudoku puzzles.

1. Dailykillersudoku(2017). Rules. [online] Available at <http://www.dailykillersudoku.com/main/rules/>

Tips on how to complete difficult puzzles and rules for killer Sudoku

1. M, Whalstrom(2016). CS2860 Algorithms and complexity. [online] Available at <https://moodle.royalholloway.ac.uk>

Course notes from my second year of study at Royal Holloway, covers big O notation, Trees, and complexity. They will be useful to complete the report on Big O.

**Risk Assessment**

I have experience in programming in Java and as Java can be run on multiple platforms including mobiles it would seem the appropriate language to use. However, GUI programming using Java seems cumbersome and awkward, I will have to learn how to use the layout manager which will mean that I will need to allow extra time to complete the programming to compensate. It is possible I will have to program using another language which provides a friendlier way to use GUI’s such as Python if I cannot successfully program a useable GUI. I will start reading my notes from the first-year games module in which we used Python to program simple arcade style games in case I need to change the programming language. These notes are no longer available online however, I have hardcopies to refer to.

The amount of time required to complete the project, considering I intend to implement the ‘extensions’, is considerable and I may be taking on too much work given the time constraints I have. I am confident that the project will be completed with everything listed in the reports and programs section. However, if I do fall behind my schedule, I will concentrate only on the core elements.

I may not have done sufficient reading or my sources may not have the required information in which case I will have to search for alternatives. This is likely to happen and I will have to do further reading throughout the development of the project.

The data structures suggested from the brief may be too difficult to program and subsequently the program may fail to run correctly. I will research other data structures which may be used to perform in the same or similar capacity.

The order in which I have planned to complete certain reports and proof of concept programs may need to be adjusted as I progress through the project. Again, this is likely to happen as I come up against unexpected technical and/or theoretical obstacles. I have tried my best to plan out the work in an organised/logical manner to negate this.