Playing Games and Solving Puzzles using AI

Sudoku Game

Project Plan

Dimitar Seraffimov

CS3821 - BSc Final Year Project

Supervised By: Professor Magnus Wahlstrom

Department of Computer Science

Royal Holloway, University of London

1. Abstract

"Sudoku" is a combinatorial, number-placement puzzle game featured for the first time in the 19th century, but the modern version only began to gain widespread popularity in 1986 when it was published by the Japanese puzzle company under the name Sudoku.

The objective of the game is to fill a 9 x 9 grid with digits so that each column, each row, and each of the nine 3 x 3 sub grids (that compose the grid) contains all the digits from 1 to 9. The puzzle provides the creator/generator with 6.671×1021 possible combinations, consisting of a partially completed 9 x 9 grid (which for a well-composed puzzle has a single solution).

The difficulty of the puzzle depends not only on how many numbers are provided, but also on the number placement of the given cell, respectively row and column. It is a challenging task to try to determine the hardness of Sudoku puzzles, which causes a lot of research to be done by computational scientists. There are typically fewer prescribed symbols in harder puzzles. Although there is no consensus on what makes a Sudoku puzzle difficult, either for a human or for an algorithm, the number of prescribed cells is not the only factor to consider.

Due to the wide variety of possible combinations, the game provides a wide range of possible puzzle variations, involving varying levels of difficulty and opportunities for the player to improve their skills, experiment with different approaches to solving the problem, and track the time they have spent solving them in the past.

2. Aims

The aims of the project are to understand simple AI algorithms and successfully implement them to solve a standard Sudoku puzzle [12]. The focus will be on researching the following AI algorithms and techniques:

- Backtracking algorithm [1]

An incremental approach to solving a problem involves building a solution candidate incrementally and backtracking (undoing) when a solution is invalid or unsatisfactory. The end result of implementing the algorithm will result in following these steps: starting with an empty puzzle and systematically fills in the cells while checking for violations of the Sudoku rules. If a violation is found, it backtracks to a previous step and explores different possibilities. This process continues until a valid solution is found or all possibilities are exhausted.

- Constraint Programming [2]

By emphasizing on defining constraints and relationships among variables, it finds a valid solutions to complex combinatorial problems. It involves specifying a set of variables, their domains, and constraints that need to be met. In order to find a valid solution or an optimal one, the solver systematically explores the search space of possible variable assignments while adhering to these constraints. There are existing libraries, like "python-constraint" [9] [10] that can be used to model Sudoku as a constraint satisfaction problem and find solutions systematically.

- Exact Cover [3]

An exact cover concept is a way of solving problems in which each item is covered exactly once by a subset of items selected from a larger set. This concept is used in algorithms such as Donald Knuth's famous "Exact Cover Problem" which seeks to find every possible exact covering of a given set of items. In order to solve this problem efficiently, Knuth developed the Algorithm X [4], often referred to as "Dancing Links." As a powerful tool for combinatorial problem-solving, it is renowned for its elegance and speed in identifying all solutions. The efficiency of Algorithm X is attributed to its use of the Dancing Links data structure, a

doubly linked list that enables rapid selection and deselection of rows and columns in a sparse matrix representation of the problem. This data structure, combined with recursive backtracking, enables Algorithm X to explore the solution space effectively while maintaining a minimal memory footprint.

- Stochastic Optimization Method [5]

Optimal solutions or approximate optimal solutions can be found using stochastic optimization algorithms in problems involving uncertainty or randomness. To explore the solution space, these methods leverage probabilistic elements and randomness, typically achieved by utilizing techniques such as random sampling, simulated annealing, genetic algorithms, or particle swarm optimization.

It is important to conduct a thorough research into each of the algorithms and techniques listed. Moreover, the spotlight will be on implementing these methods cohesively to enhance the speed and efficiency of the solving and solution verification. The aim of this approach is not only to accelerate the puzzle-solving but also to ensure the robustness and reliability of the solving software, offering users a seamless and satisfying experience.

3. Objectives

The objectives the project will strive to achieve for the end result of the program functionality, design and usability are essentially following the guidelines provided by the project description. I have decided to broaden the scope of the project to achieve slightly ambitious but realistic goals if the timeline is strictly followed and the program is developed, following to the Agile methodology.

The project will utilize Python as the primary programming language for implementing the Sudoku solver. The project's core objective will be to develop feature-rich capabilities efficiently by utilizing Python's simplicity and fundamentals. I'll be able to achieve a broader range of features with concise code using Python's inherent productivity, so I won't have to worry about excessive verbosity as with C or C++. Furthermore, Python's cross-platform compatibility will align with the goal of making the program accessible across different platforms.

While acknowledging that certain programming languages may outperform Python in specific tasks, I will remain committed to Python for its unparalleled flexibility and portability. Notably, the choice of the programming language should not significantly impact the performance analysis of different algorithms, as time complexity will remain algorithm-dependent rather than language-dependent. Furthermore, this will be a good opportunity for me to broaden my skills and knowledge in this programming language as I have mostly worked with Java.

The Sudoku solver will operate seamlessly through the command line, facilitating ease of use. The Python3 codebase will be developed within the Visual Studio Code (VS Code) editor, and the solver will require no additional external programs for execution. Furthermore, in the subsequent phases of the project, React Native [11] will be integrated to design and develop the frontend of the mobile application, thereby ensuring a comprehensive and user-friendly solution across multiple platforms.

Early Deliverables:

In the initial stage of the project, time will be dedicated on understanding in depth the algorithms and techniques that will be using. This will give me the foundation and knowledge needed to build the Proof of Concept programs, including a visually appealing graphical user interface (GUI) with interactive elements, and implementing essential data structures, such as a priority queue. These early deliverables will serve as tangible demonstration of my capabilities and dedication to creating a user-centred design interface and data management. Additionally, I will aim to delve into algorithmic exploration by solving the classic Eight Queens problem using a backtracking algorithm. This exercise will not only showcase my algorithmic understanding but also lay the groundwork for solving Sudoku puzzles effectively. The next step of the

project will be the creation of a proof of concept program capable of solving simplified Sudoku examples, which will offer a glimpse into my progress.

On the research side, I will compile informative reports on various facets of AI and puzzle-solving [8]. These reports will focus on design patterns relevant to AI and search algorithms, insights into Backtracking and recursion, and an exploration of Constraint satisfaction techniques. As I explore the complexities of user interface design, complexity analysis, and algorithms used by human solvers, I aim to build a solid foundation for the future phases of the project.

Final Deliverables:

The final objective of my project is to develop a mobile application for Sudoku-solving game, based on the early deliverables and research findings. The application will not only feature a visually engaging GUI but will also incorporate a splash screen and multiple user interaction screens to ensure an immersive user experience. It will possess the capability to generate, load, and solve puzzles, adhering to Sudoku rules and providing hints to users when needed. Crucially, the application will be designed to produce puzzles with unique correct solution, ensuring the utmost puzzle integrity.

Accompanying the software, there will be a comprehensive final report detailing the software engineering processes undertaken, algorithms and programming techniques employed, and achieved final outcome. It will provide an in-depth exploration of the algorithms and programming techniques applied throughout the project. The final report will be written using the Overleaf platform, ensuring a well-structured and professional display of the project's findings and results.

Beyond these objectives, the plan is to extend the project by scaling it into a mobile application that locally tracks user progress, including previous attempts, completion times, identified mistakes, used hints and puzzle difficulty categorization [7]. This additional functionality strives to elevate the user experience by enabling players to measure their progress and challenge themselves with puzzles of varying complexities, ranging from super easy to extreme difficulty levels.

In summary, the project objectives span the entire development lifecycle, from the creation of proof of concept programs and in-depth research during the early deliverables phase to the implementation of a fully-featured Sudoku mobile application with advanced user tracking capabilities. In addition to providing a user-centric and engaging interface, this extensive approach also ensures thorough exploration of Sudoku puzzle solving and artificial intelligence techniques.

4. Timeline

The timeline will divide the academic year into evenly distributed sprints which will help me keep focus on the algorithms research, proof of concept programs, GUI implementation and final deliverables. This approach will employ in the best way the Agile development methodology and ideally leave me with additional time to implement all additional features.

4.1 Term 1

Sprint 1 (Week 1-2): Research and Understanding

- Writing the project plan and sending it for a review.
- Research and comprehending on the fundamentals of Sudoku puzzle solving.

Sprint 2 (Week 3-4): Algorithm and Method Exploration

- Diving into the Backtracking algorithm, gaining a deep understanding of its principles.
- Exploring constraint programming techniques, with a focus on modelling Sudoku as a constraint satisfaction problem.
- Begin to research the Exact Cover concept and Donald Knuth's Algorithm X.
- Studying the Stochastic optimization methods applicable to Sudoku puzzle solving.

Sprint 3 (Week 5-8): Further Algorithm Exploration

- Explore constraint programming techniques extensively, with a primary focus on modelling Sudoku as a constraint satisfaction problem.
- Study Algorithm X (Dancing Links) in detail, understanding its data structure and recursive backtracking.
- Begin practical experimentation with the algorithms, laying the foundation for their integration into the Sudoku-solving application.

Sprint 4 (Week 9-12): Proof of Concept Programs

- Implement a visually appealing GUI for the Sudoku solver.
- Develop the basic structure of the Sudoku application.
- Begin coding the Backtracking algorithm within the application.
- Produce a preliminary report on AI design patterns and search algorithms.
- Finalising the presentation and preparing the application for the demo show.

4.2 Term 2

Sprint 5 (Week 13-16): Algorithm Integration

- Integrate the Backtracking algorithm into the application.
- Incorporate constraint programming techniques for enhanced solving capabilities.
- Implement Algorithm X (Dancing Links) for efficient exact cover problem solving.
- Research and experiment with stochastic optimization methods.

Sprint 6 (Week 17-20): Enhanced Functionality

- Enhance the GUI with interactive elements and user-friendly features.
- Integrate React Native for the frontend of the mobile application.
- Implement puzzle generation, loading, and saving functionalities.
- Begin user progress tracking, including previous attempts and completion times.
- Compile a report on Backtracking and recursion techniques.

Sprint 6 (Week 21-24): Testing and Optimization

- Compile a report on Backtracking and recursion techniques.
- Rigorously test the Sudoku solver and mobile application.
- Add functionality to identify and highlight user mistakes.
- Complete user progress tracking, including difficulty categorization.
- Compile a report on constraint satisfaction, particularly consistency techniques.

Sprint 7 (Week 25-28): Reporting, Documentation, and Final Touches

- Prepare and format the final report using the LaTeX/Overleaf platform.
- Document the entire software engineering process undertaken.
- Detail the design and implementation of algorithms and programming techniques.

5. Risks and Mitigations

Like all projects, certain risks are inherent, and while some may be beyond our control, it is crucial to address and manage them effectively. My project is no exception, and while it is an ambitious endeavour, the associated risks are manageable.

5.1 Poor Estimation of the Timetable

This could be a real problem and I have learned from my experience. Therefore, based on what I learned in my second-year module, Software Engineering - led by Prof. David Cohen, I have decided to use the Agile software development methodology. The main reasons why I selected it are because I have already had some experience and knowledge about it, moreover – the Agile methodology is based on iterative and incremental development, which is ideal for following the plan and making sure that I don't fall behind on the timeline.

5.2 Hardware Malfunction

There exists a risk of potential hardware failure during the academic year and completion of my project, which could lead to substantial data loss and disrupt or completely loose the progress made. The timing of hardware failures is unpredictable, but I have devised a robust mitigation strategy to safeguard against data loss. To address this risk, I will implement a rigorous version control system by utilizing the GitHub platform. This approach involves regularly committing my project code, ensuring that every code iteration is securely stored in the cloud.

Furthermore, I will extend this version control approach beyond code management. I will also employ Overleaf to maintain version control over my project report. By diligently following this strategy, all project-related work, including code and project documentation, will be securely stored in the cloud.

5.3 Uneven Balance Between Report/Code

Both the code development and the report writing aspects of the project are equally crucial, and I can potentially delay the progress of the other. Getting overly absorbed in coding might lead to a misalignment with the report's development, and vice versa. To address this potential risk, the timeline will be integrated with Agile software development methodology. This approach will assist in maintaining alignment with the project plan and ensuring the responsible allocation of time to both coding and reporting segments.

5.4 Testing and Conscious Experiments

In the development of the project, a key focus for me will be on adopting Test-Driven Development (TDD) principles. By adhering to the TDD [6] practices, my aim is to ensure the reliability and robustness of the Sudoku-solving algorithms, as well as the user interface components from the very beginning of the development process. This approach will allow me to systematically identify, and on-time rectify potential issues, enhance code maintainability, and deliver a high-quality, error-free mobile application that aligns with best practices in software engineering.

A lot of time can be consumed by experimenting with machine learning, GUI design, and code implementation, especially if a large amount of data is involved, the desired result is desired and new features are implemented. It is possible for experiments to be run and code to be written that ultimately add little to what is additional value or are wasted due to lack of robust code, time or project requirements.

Acronyms

AI - Artificial Intelligence.

TDD - Test-driven Development.

GUI - Graphical User Interface.

Glossary

GitHub - A cloud-hosted extension of a Git version control system with a web GUI.

Overleaf - An online LaTeX editor providing features such as history/version control and collaboration.

Python - A high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

React Native - An open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use the React framework along with native platform capabilities.

Proof of Concept - A small-scale demonstration that a product in development is likely to be successful in later stages of development.

References

[1] Civicioglu, P., 2013. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and computation*, *219*(15), pp.8121-8144.

Link to the webpage

[2] Brailsford, S.C., Potts, C.N. and Smith, B.M., 1999. Constraint satisfaction problems: Algorithms and applications. European journal of operational research, 119(3), pp.557-581.

Link to the webpage

[3] Chang, W.L. and Guo, M., 2003. Solving the set cover problem and the problem of exact cover by 3-sets in the Adleman–Lipton model. BioSystems, 72(3), pp.263-275.

Link to the webpage

[4] Knuth, D.E., 2000. Dancing links. arXiv preprint cs/0011047.

Link to the webpage

[5] Marti, K., 2008. Stochastic optimization methods (Vol. 2). Berlin: Springer.

Link to the webpage

[6] Anwer, F., Aftab, S., Waheed, U. and Muhammad, S.S., 2017. Agile software development models tdd, fdd, dsdm, and crystal methods: A survey. International journal of multidisciplinary sciences and engineering, 8(2), pp.1-10.

Link to the webpage

[7] Pelánek, R., 2011, May. Difficulty Rating of Sudoku Puzzles by a Computational Model. In FLAIRS Conference.

Link to webpage

[8] Yato, T. and Seta, T., 2003. Complexity and completeness of finding another solution and its application to puzzles. IEICE transactions on fundamentals of electronics, communications and computer sciences, 86(5), pp.1052-1060.

Link to the webpage

[9] Stephens, R., 2019. Essential algorithms: a practical approach to computer algorithms using Python and C. John Wiley & Sons.

Link to the webpage

[10] Bynum, M.L., Hackebeil, G.A., Hart, W.E., Laird, C.D., Nicholson, B.L., Siirola, J.D., Watson, J.P. and Woodruff, D.L., 2021. Pyomo-optimization modeling in python (Vol. 67). Berlin/Heidelberg, Germany: Springer.

Link to the webpage

[11] Eisenman, B., 2015. Learning react native: Building native mobile apps with JavaScript. " O'Reilly Media, Inc.".

Link to the webpage

[12] Davis, S., Henderson, M. and Smith, A., 2010. Modeling Sudoku Puzzles with Python. In practice, 7, p.3.

Link to the webpage