Applying AI to Solve a Single Player Puzzle Game(Solitaire)

Project outline

Sam David Freeman (sdf2)

Degree Scheme: G400 Computer Science
Module: CS39440 Major Project
Supervisor: Thomas Jansen (thj10)

Date: 07/02/2023
Version: 1.1(Draft)

Project Description

The project is based around the solving solitaire games, and for this I have chosen to apply AI to solve the logic based puzzle Sudoku. Sudoku is a well established problem and I have decided to tackle the problem using genetic algorithms. My program will have two different sections to it, solving a Sudoku puzzle and generating Sudoku puzzles with Unique solutions.

The project will use the idea of kanban boards, to monitor and track the progress of the different stories that need to be completed throughout the project. The project will also make use of the agile idea of a heartbeat, which will help keep the pace of the project and allow for adaptation further along in the project, whilst also havings workable code earlier on in the development time.

Proposed Tasks

- **Investigation of genetic algorithms.** Further reading into how to apply genetic algorithms to the problem will need to be done. This will be done by looking at papers on google scholar and looking into papers referenced by the most useful papers found.
- **Setting up and maintaining version control system and issues board.** The project will use a git repository to hold all the files, and have a backup to any local files. This will be done using GitHub and will also make use of GitHubs issues and project board features, to add and keep track of tasks in the development of the project.
- **Spike work on application of genetic algorithm.** Before the theory behind the genetic algorithms can be applied to the problem of solving sudoku, some spike work will need to done first, in how to apply GA in the chosen programming language, which will either be Python or Java.
- **Development.** In the development of the project, genetic algorithms will be used, the algorithm will need to represent the sudoku in some way, by seperating each block into integers and use these as the different genes the algorithm works with. There will be two main subtasks in the development of the project with the final goal being the linking of these two tasks:
  - **Solving Sudoku.** The first part of the project will be the solving of sudoku, this will be done is several phases, with increase in difficulty. The increase in difficulty will be done by decreasing the number of given numbers, and increasing the size of the Sudoku, starting with a 4x4 puzzle and increasing in size to a 9x9. With hopefully by the end of the project being able to go past that to 16x16 or even 25x25. The exit conditions for the algorithm will be, either finding a suitable solution(an instance where all of the numbers are filled and does not exist in the same block or line as a number that is the same), or the program goes past the limit for a suitable number of generations.
  - **Generating unique Sudoku.** The second task in the development will taking on the task of generating unique sudoku puzzles for a user or the solver to complete. This will build on the solving of sudoku, but using a completely empty starting set, letting the genes mix and mutate over generations eventually creating a unique solved puzzle, then backtracking and removing random numbers from the puzzle until it reaches the required number for the selected difficulty. This program will start off by asking the user, the size and difficulty of the sudoku puzzle they want. Equally this program will start out by only working for only 4x4 but eventually getting to 16x16 or 25x25.
- **Project meetings.** Project will have weekly meetings with supervisor, where summaries of the meetings will be documented and added onto the GitHub, which can then be converted to issues if anything urgent is pointed out.

Project Deliverables

- **Sudoku solver.** A stand alone application that takes an Sudoku puzzle as an input and gives an output result of a filled in puzzle. This will include the code behind the application, and instructions on how to launch it. The target time for completing the solver is the 6th of March, which means it can hopefully be used to show progress at the Mid project demonstration.

- **Sudoku generator.** A stand alone application that can generate unique Sudoku puzzles of a given size and difficulty, this will then be possible to pipe into the solver program. This will include the code, runnable program and instructions on how to use the program. The target time for completing the generator is the 27th of March.
- **Requirements document.** A document will be produced listing the requirements for the project, listing out the goals for what will count as the project being complete, as well as some non functional requirements as stretch goals. This will later be integrated into the final report. The target time for completion of the requirements document is 20th of February.
- **Tests.** A set of tests will be created using the requirements, both unit and integration tests, which will be made using Junit or unittest depending on the language decided. There will also be a document produced alongside this, noting the results of the tests and how the different requirements have been met. The document will later be integrated into the final report, and the test code handed in alongside the technical work. The target time for the tests and testing document to be complete would be the 18th of April, however the tests themselves will be made alongside the programs.
- **Final report.**
- **Final demonstration.**

Initial Annotated Bibliography

T. Mantere and J. Koljonen, ”Solving, rating and generating Sudoku puzzles with GA,” 2007 IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 1382-1389, doi: 10.1109/CEC.2007.4424632.
Provides an in depth looks at methods required to solve and generate sudoku puzzles using GA. Will be used a future resource.

Geem, Z.W. (2007). Harmony Search Algorithm for Solving Sudoku. In: Apolloni, B., Howlett, R.J., Jain, L. (eds) Knowledge-Based Intelligent Information and Engineering Systems. KES 2007. Lecture Notes in Computer Science(), vol 4692. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74819-9_46
Quick look into harmony search algorithm, example of alternative meta-heuristic algorithms used to solve sudoku. Probably will not be used in the project.

Lewis, R. Metaheuristics can solve sudoku puzzles. J Heuristics 13, 387–401 (2007). https://doi.org/10.1007/s10732-007-9012-8 N. Musliu and F. Winter, ”A Hybrid Approach for the Sudoku Problem: Using Constraint Programming in Iterated Local Search,” in IEEE Intelligent Systems, vol. 32, no. 2, pp. 52-62, Mar.-Apr. 2017, doi: 10.1109/MIS.2017.29.
Paper on Constraint programming to solve sudoku. Example of alternative algorithms used to solve sudoku. Probably will not be used in the project.

Russell, S.J. and Norvig, P. (2014) “Beyond Classical Search,” in Artificial Intelligence: A modern approach. Harlow, Essex: Pearson.
Contains a subsection on genetic algorithms, helpful for implementing them in the project.

Sturm, Thomas. (2009). Sudoku and A.I... 2. 910-911.
Provides a brief summary of a proposed approach to solving sudoku with AI, quite general. Probably not useful further along than initial research

Mahima Dubey, Vijay Kumar, Manjit Kaur, Thanh-Phong Dao, ”A Systematic Review on Harmony Search Algorithm: Theory, Literature, and Applications”, Mathematical Problems in Engineering, vol. 2021, Article ID 5594267, 22 pages, 2021. https://doi.org/10.1155/2021/5594267
Provides further insight into harmony search algorithm as well as a general summary of meta-heuristic algorithms in the introduction. Probably will not be used past initial research.