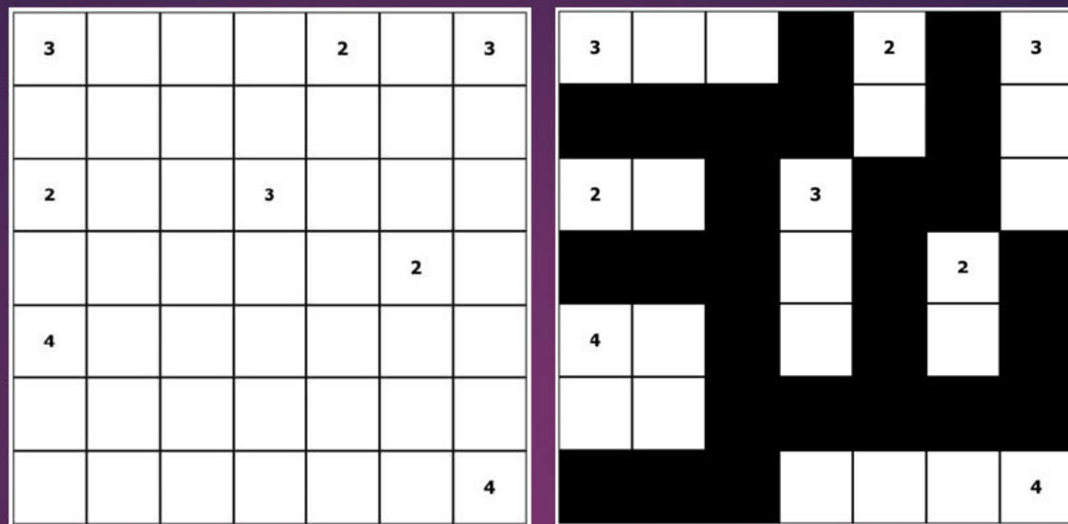


A.I. Solver for Games

A Genetic Algorithm for solving Nurikabe

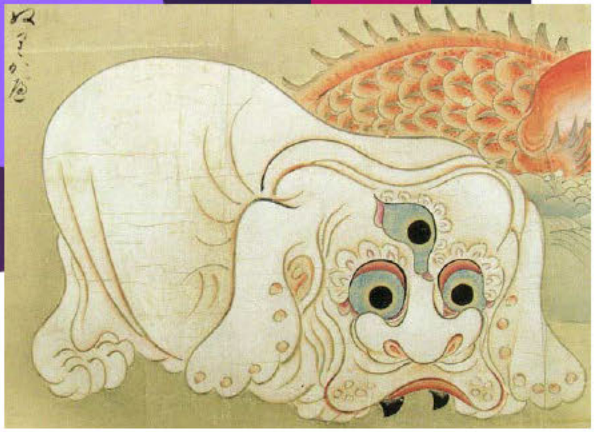
[redacted]

[redacted]@STU.MMU.AC.UK

Structure of Talk

1. Introduction to the project
2. Aims and objectives
3. Previous work
4. My approach
5. System design
6. Implementation framework
7. Fitness Function
8. The Nurikabe engine
9. Evaluation
10. Personal reflection

Introduction to the Project



- Nurikabe is a Japanese puzzle game, created in 1991.
- 'The Nurikabe' is a spirit from Japanese folklore, a wall which impedes or misdirects travelers at night.
- Trying to go around is pointless as it extends forever, which is represented in a correct solution to the puzzle game.
- There exists no automated A.I. solver for Nurikabe, so I will be researching the viability of using one, namely a genetic algorithm.
- Nurikabe has multiple objectives, making it a more difficult puzzle.

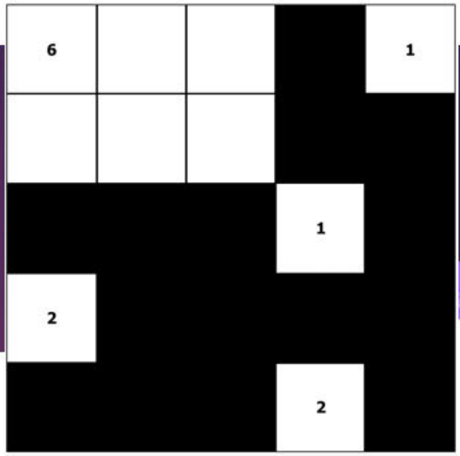
Aims and Objectives

- **Aim**
  - Research and implement an automated solver for Nurikabe, using a genetic algorithm which can deal with the multiple objectives of the puzzle.
- **Objectives**
  - Perform research in the area.
  - Define a product plan for the solver; languages, platform, engine, and genetic algorithm.
  - Develop the engine, implement the genetic algorithm and run tests.
  - Evaluate the results of testing, to conclude on the viability of using a genetic algorithm to solve Nurikabe.

Previous Work

- A number of puzzle games have been solved using artificial intelligence, and extensive work has gone into investigating the viability of different techniques.
- Genetic algorithms have been used in the past to play the puzzle-platformer game Lemmings.
  - A chromosome is defined as a script of moves for the individual lemmings in a game, and the fitness function evaluates strength based on distance travelled, lemmings saved, time remaining etc.
- The Zen Puzzle Garden game has also been solved using genetic algorithms, by a previous student of MMU.
  - A chromosome dictates a starting location and a script of moves, the goal being to walk a monk around a garden board, in order to rake every square whilst dealing with hazards and obstructions on the way.

My Approach



- A Nurikabe chromosome represents a set of moves, to be made relative to each island on the board. A gene represents a single move:
  - Up = 001, Down = 010, Left = 011, Right = 100
- Starting with the first number on the board, the moves will be encoded until the number's island has been satisfied, and then it will move to the next island.
- This process will repeat until each number on the board has its island satisfied, and a fitness function will evaluate the board's strength.
- The genetic algorithm is used to generate populations of random solutions, evaluate them and create new populations by evolving the strongest solutions of the last, until a solution is found or a plateau in fitness value is reached.

System Design

The overall design of the genetic algorithm is as follows:

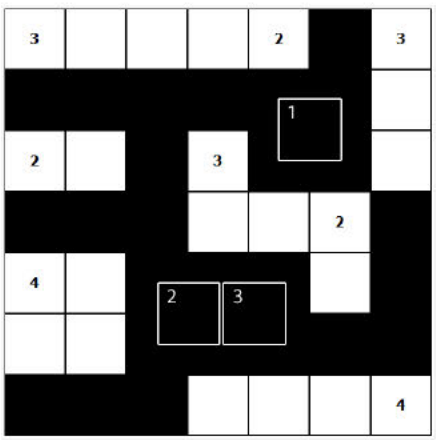
1. Generate a population of random solutions to a given Nurikabe board.
2. Encode each solution to the puzzle board.
3. Evaluate the strength of each solution using a fitness function.
4. Utilize selection and alteration techniques resulting in a new generation, adding new random solutions to the population if required.
5. Return to step 2 and repeat until a solution is found with a fitness of 100%, or until the increase in fitness throughout each generation reaches a plateau.
6. Display the best chromosome to the user in a GUI, printing results to file.

Implementation Framework

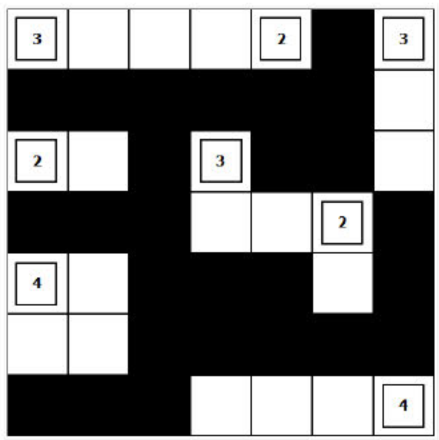
- Language: Java
- IDE: Eclipse
- The genetic algorithm library Jenetics will be used, as it provides the appropriate containers and functions required to build and run the genetic algorithm.
- The main benefit of Jenetics is that it allows abstract chromosome classes, as the Nurikabe genes and chromosomes aren't compatible with usual integer/binary implementations.

Fitness Function

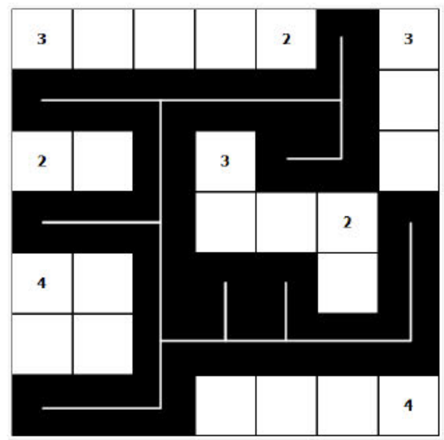
- The most important part of encoding a problem space into a genetic algorithm.
- Returns a value which represents the strength of a Nurikabe solution, from 0 to 100.
- Calculated using values of the multiple objectives of the puzzle.



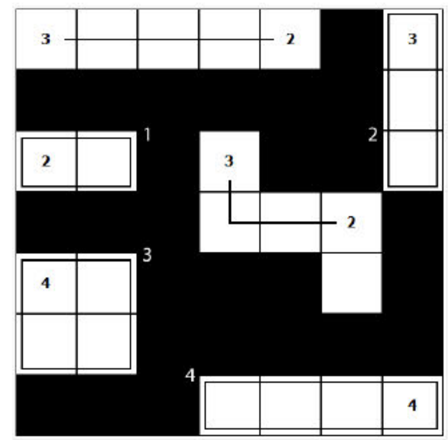
No. Black 2x2s: 3



No. Expected Islands: 8



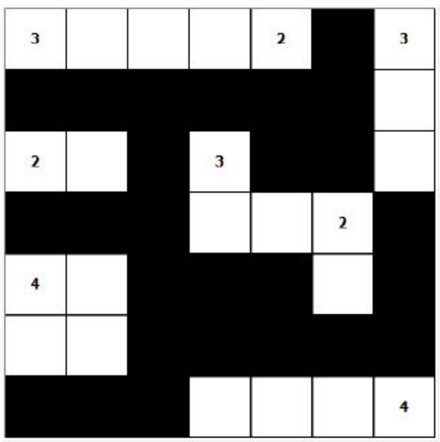
No. Ocean Areas: 1



No. Satisfied Islands: 4

Fitness Function

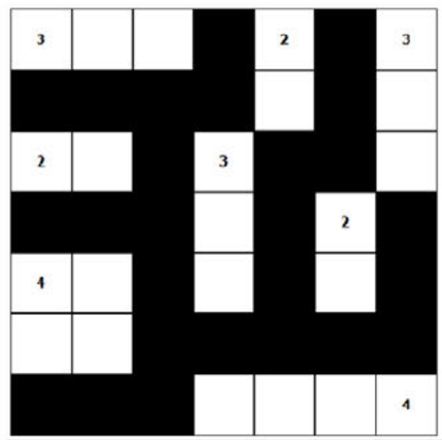
- Having a representative value for each solution is required for the genetic algorithm to understand the relative strength of each chromosome, for selection and alteration.



No. Black 2x2s: 3
No. Expected Islands: 8
No. Ocean Areas: 1
No. Satisfied Islands: 4

A = (4 / 8) * 100
B = 1 – (3 / (3 + 50))
C = 1/ 1

Fitness = (A* B) * C
Fitness = 48.07%



No. Black 2x2s: 0
No. Expected Islands: 8
No. Ocean Areas: 1
No. Satisfied Islands: 8

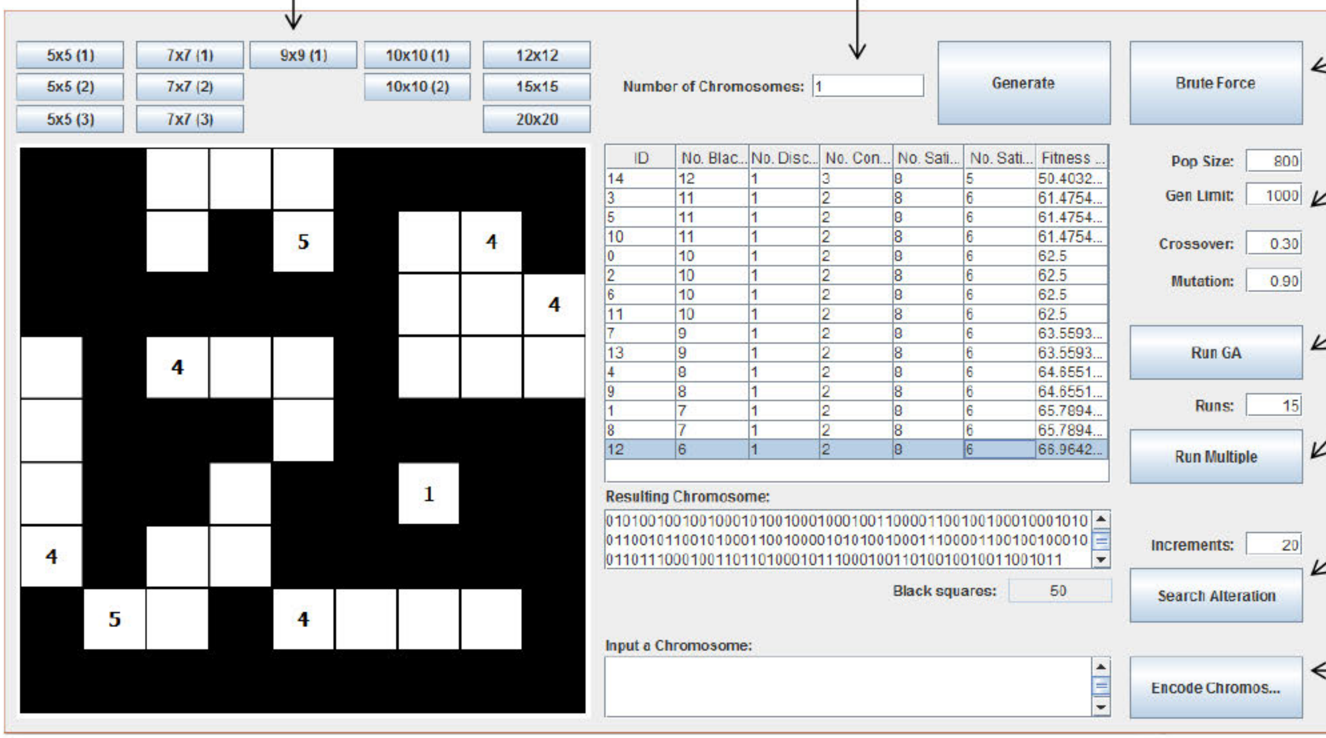
A = (8 / 8) * 100
B = 1 – (0 / (0 + 50))
C = 1/ 1

Fitness = (A* B) * C
Fitness = 100%

The Nurikabe Engine

A number of different puzzles for testing

Generate Random Chromosomes



Brute force to provide contrast to GA

Configure GA engine parameters

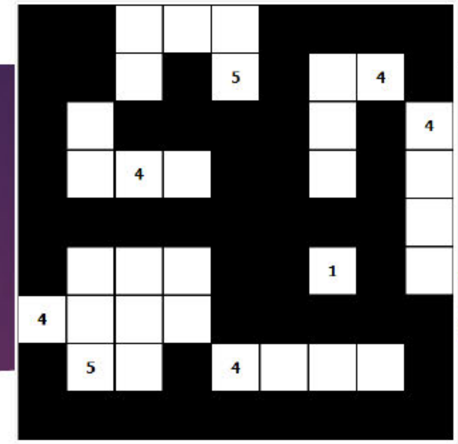
Run a fast, single instance

Run multiple GA runs, printing results to a text file in the results folder

Search levels of alteration, running multiple GAs, printing results to file

Encode a chromosome back into a puzzle, useful for evaluating test results

Evaluation



- The genetic algorithm is successful in solving small puzzles, 5x5s and 7x7s.
- The genetic algorithm is able to rarely solve a 9x9 puzzle, and the increasing complexity of Nurikabe is reflected in the fitness values on larger puzzles.
- The best chromosome of any genetic algorithm run is typically simple to fix by eye, suggesting that a pruning function could fix these solutions.
- Future work will involve:
  - Variations of the representation and encoding of Nurikabe solutions.
  - The extension of library classes to write alteration and selection functions which are more specific to Nurikabe chromosomes.
  - Highlighting other puzzles or real world problems that have a similar path based nature to Nurikabe, perhaps tailoring the genetic algorithm to be more of an API with multiple uses.

Personal Reflection

- The final year project has taught me a considerable amount about how to perform thorough research.
- In order for a project to be successful, it needs a strong background and plan, so the targets are always clear.
- I am satisfied with the result of my work on Nurikabe, and also optimistic about continuing to investigate the potential of using genetic algorithms to solve problems.
- This project has served as great preparation for my PhD, as my writing, research, development and presentation skills will be essential throughout.
- As the most hands on, independent and challenging piece of work I have done it has been the most enjoyable part of my undergraduate degree.