COMPSCI 683 Project Proposal

Erin Song
esong@umass.edu

Harold Thidemann
hthidemann@umass.edu

Shubhang Vagvala
svagvala@umass.edu

Jiaqi Ye
jiaqiye@umass.edu

Abstract

Our project explores how to solve a Nurikabe puzzle of varying board sizes. The puzzle is a binary determination puzzle that is NP-complete, meaning that the larger the board size, and exhaustive search would take a long time. We approach this problem as a Constraint Satisfaction Problem (CSP) and use –(insert algorithm solution)– to solve it. We find that –(insert relevant results and conclusions)–.

1 Introduction

Nurikabe puzzles were developed in Japan and published in alogic puzzle magazine from Nikoli in 1991. The puzzle consists of a grid board where each cell can have one of three states. Each cell can be either black, white or contain a number. The rules are as follows:

1. You cannot fill in cells containing numbers.
2. A number tells the number of continuous white cells. Each area of white cells contains only one number in it and they are separated by black cells.
3. The black cells are linked to be a continuous wall.
4. Black cells cannot be linked to be 2×2 square or larger.

Figure 1 shows an example puzzle along with its solution.

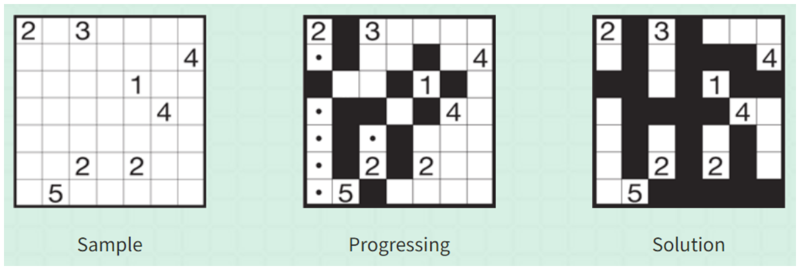

Figure 1: A Nurikabe Puzzle and its Solution

1.1 Our Contributions

We meet weekly to discuss the work we've done the past week and our goals for next week.

1. Erin Song: Developed a way to visually represent game boards, wrote the intro and rules of the game, and will develop a method to check if a solution is correct.

2. Harold Thidemann: Worked on code to generate data set of boards, will write the formal mathematical definitions and perform data analysis.
3. Shubhang Vagvala: Worked on code to make an interactive way to play Nurikabe and will do testing on the algorithm(s).
4. Jiaqi Ye: Worked on code to generate data set of boards, reviewed related papers, and will develop algorithm(s) for solving Nurikabe.

We plan on using the information from the related works section to develop an algorithm, or just picking one of them.

1.2 Related Work

Here is what we have found so far:

Johan Groenen explains how to solve all solvable puzzles and check for unique solvability using brute force algorithms, extended with backtracking and optimizations. The algorithm expands islands recursively, each time adding a cell from an island's neighborhood to itself, and then updating the neighborhood. If the island is full, the next island is expanded. If there is no next island, we check if there are no pools and only one stream (which would make it a solution). You can reduce the search time by not allowing expansion to cells that have been tried earlier.

- Can also check if there are pools in unreachable areas of the puzzle; if so, there's no solution and there is no need to keep checking it.
- There are two possible ways to generate puzzles. The first method is to generate all possible solutions, throw away all solutions containing pools and/or zero or multiple streams, create all corresponding puzzles by placing numbers in every way possible, and select all puzzles that appear only once in the generated set (which means they have only one solution). The second method is to generate all puzzles and check them for correctness. No two pivots can be neighbors and there is a maximum number of white cells, so it's possible to skip a lot of configurations.

Spencer Young et al. developed a genetic algorithm solver for Nurikabe which can solve 5x5 Nurikabe problems in approx. 1 second. The solver uses all the traditional properties of genetic algorithms, such as population, mating pool, elitism, mutation, and breeding.

Martyn Amos et al. made a Nurikabe solver using Ant Colony Optimization.

- At each iteration, a number of "ants" are given a local copy of the game board. Each ant constructs a solution.
- Each ant constructs a set of possible candidate cells starting from the "seed" of an island (a number on the grid). Candidate cells are initially the cells around the seed, and we prune cells that violate game constraints (remove any cell that cuts off the stream, remove any cell that is adjacent to an existing island, etc). Each ant will move on to the next "seed" until a complete solution is made.
- Then select the best performing ant ("according to a cost function which counts how many constraints are broken"), update the global pheromone matrix (future ants are then biased towards these "better" cells), and then "best value evaporation" operator is applied.

Paul Bass et al. took a scattered neighborly approach to solving Nurikabe. The method combines scatter search and variable neighorhood search.

- Overview: generate a set of diverse (potential / partial) solutions using scatter search, selects the fittest subset of solutions, searches the space around the solutions to find the final correct, solved puzzle using VNS made up of four neighborhoods.
- Neighborhoods: Surplus island swap, unifying walls, breaking up pools, and regenerating islands.
- Variable neighborhood search is basically hill-climbing but by hill-climbing different "neighborhoods" of possible solutions, which balances exploration with exploitation.

Nurikabe puzzles may also be encoded as SAT problems. Stephan T. Lavavej developed an algorithm that solves Nurikabe using a SAT solver.

2 Model and Preliminaries

3 Results

4 References

5 Additional Results