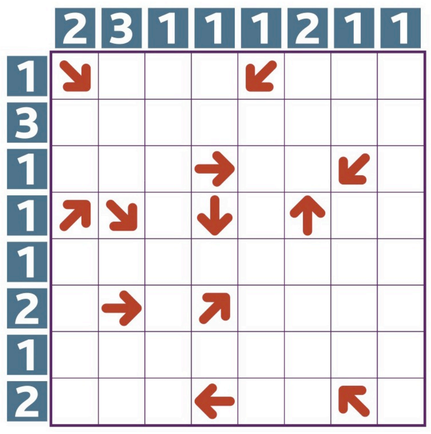
Evolutionary Algorithm for Generation of Entertaining Shinro Logic Puzzles

David Oranchak, doranchak@gmail.com http://oranchak.com

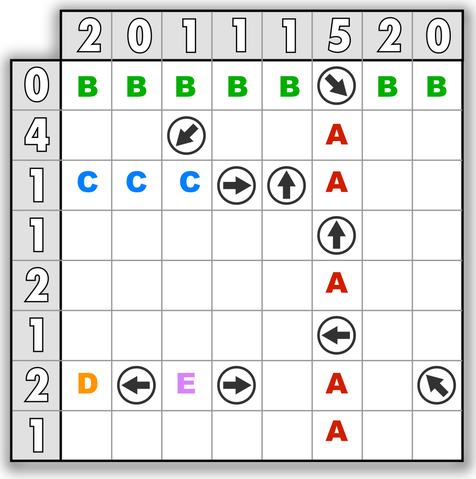
INTRODUCTION

Shinro puzzles are fun logic puzzles that first appeared in Japanese puzzle magazines, and enjoyed renewed popularity when an airline company adapted the puzzles for their inflight magazine.



Original puzzle as it appeared in Spirit magazine by Southwest Airlines

Each puzzle has twelve hidden stones. Arrows point to some of the hidden stones. Numbers along the side and top indicate how many stones are in each row and column. Armed with these clues, you must deduce the location of all stones.



Sample puzzle demonstrating easy logical deductions

A Must be stones, since the count is 5 for this column, and there are only 5 free positions.

B Cannot be stones, since the count is 0 for this row.

C Cannot be stones, since placing one here makes the right-facing arrow unsatisfiable.

D Must be a stone because the arrow points to its only possible position.

E Cannot be a stone, otherwise we cannot satisfy both the adjacent left- and right-facing arrows.

PROBLEM: HOW TO GENERATE THE PUZZLES

Clearly, constructing the puzzles by hand is very time-consuming. There are very many valid puzzle configurations. Completely randomized puzzles are not very much fun, because they can be too easy to solve, lack aesthetic qualities, and seem too repetitive or boring. How can we create entertaining puzzles?

DESIRED PUZZLE QUALITIES

- Solution validity.
- No brute force or guessing allowed.
- High number of steps in the solution.
- Specific difficulty level.
- Stones and/or arrows form interesting patterns and shapes.

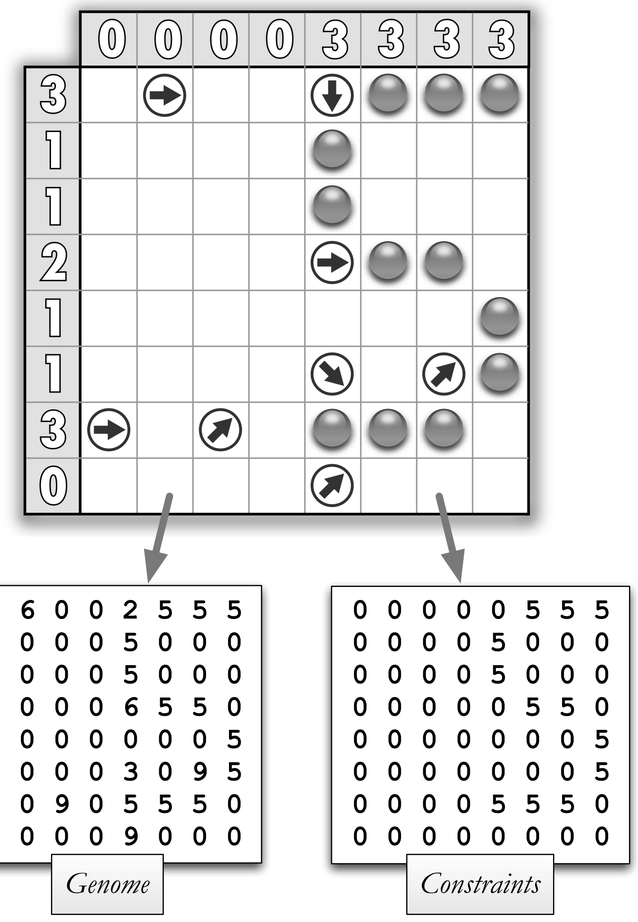
SOLUTION: GENETIC ALGORITHM

We can evolve puzzles using a genetic algorithm that optimizes for the desired puzzle characteristics.

To measure solution validity, solutions steps, and difficulty level, we built a Shinro puzzle solver. The solver uses logical deduction rules to solve the puzzles.

What makes a puzzle invalid?

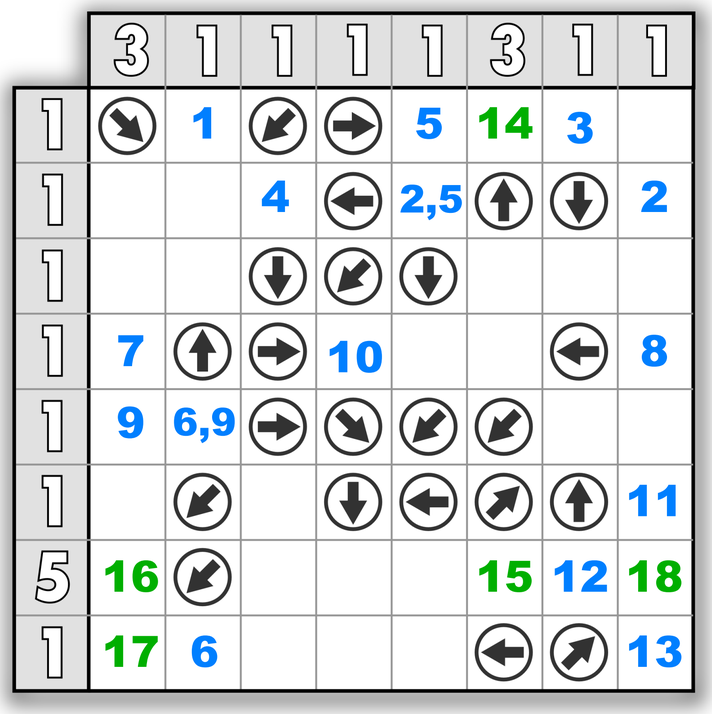
- The puzzle does not have exactly 12 stones.
- The puzzle has an arrow that does not point to a stone.
- The solver cannot reach exactly one unique solution.
- The solver reaches a solution but it is incomplete, or does not match the stone locations.
- Brute force or guessing is necessary to solve the puzzle.
- The puzzle can be solved too quickly with too few steps.



Constraints are optionally applied to enforce shape/pattern configurations

ALGORITHM DETAILS

- Population size: 10
- Mutation operators:
  - mutate a random position in the genome
  - swap randomly selected positions
  - add random arrow
  - remove random arrow
  - add random stone
  - remove random stone
- No crossover
- Termination criteria: 5000 generations elapse without improvement
- Fitness function runs solver algorithm to determine number of steps in solution
- Factors in fitness function:
  - Minimize puzzle errors / violations
  - Maximize number of steps in solution
- User controls:
  - Target difficulty: {easy, medium, hard}
  - Lower bound on acceptable number of solution steps
  - Optional shape/pattern constraint
  - Optional on/off switch for symmetry



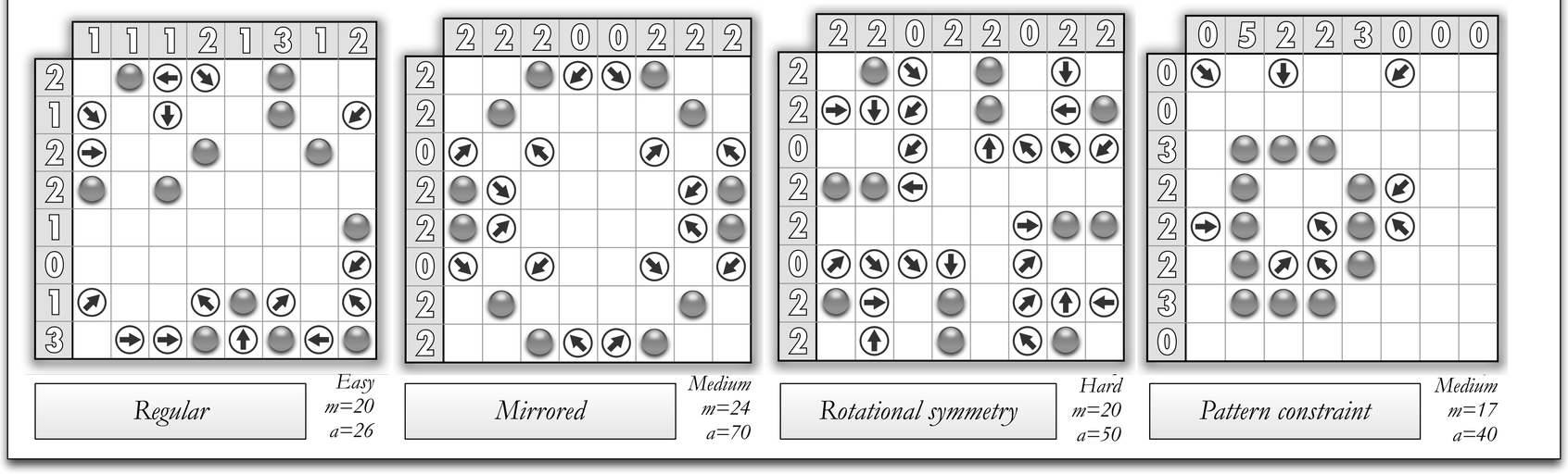
Green: Stone can be placed
Blue: Stone cannot be placed

Easy m=37 a=298

m: # *of moves* **a:** *sum of all available moves for all steps*

Sample puzzle evolved to maximize number of solution steps. Unfortunate side-effect: Too many available moves at each step. There are 18 available first moves. The puzzle is too easy to solve.

Sample evolved puzzles, after improving fitness function to minimize sum of available moves (a) for all solution steps:



RESULTS / CONCLUSIONS

- Puzzles are more interesting to play when fewer available moves are present at each solution step.
- Algorithm produced great variety of puzzle configurations.
- Production of symmetric and constraint-bound puzzles was much faster after introducting symmetry- and constraint-aware mutation operators.

FURTHER STUDY

- Solver selects moves greedily (easiest first). Investigate effect of other selection modes.
- Identification of logical deductions the solver does not yet implement. How would they affect fitness evaluation?
- Investigate other metrics:
  - Arrow density
  - Clustering
  - Balance among types of moves in the puzzle solution