# Solving Solitaire game with AI

SAM DAVID FREEMAN [SDF2]

# The problem

- The solitaire problem I have decided to tackle is Sudoku.
- The method I have chosen to solve Sudoku is Evolutionary algorithms.
- For the problem I needed to have, some form of encoding for the puzzle, a fitness function, mutation and a way to handle constraints.

# Aims of the project

- Take a Sudoku puzzle as an input
- Use EA with repair method constraints to generate a solution if there is one
- Use multi objective EA without a repair method to generate a solution if there is one
- Compare EA with repair method to multi objective EA

# Search space/ Encoding

Encoding for the algorithm will be represented using a integer encoding of 0-n, where n is the size of the puzzle.

This is stored in a n x n array, along with a second array which stores the initial positions of the puzzle.

# Fitness function

The fitness function used will evaluate the puzzles state will be quite simple, it will look at one thing, the number of filled spaces within the puzzle.

To deal with how simple the fitness function is here, the program will also use constraint handling.

# Constraint handling

For the handling of constraints, the program uses a repair method.

The repair method will have 3 constraints to get towards a feasible solution, which are the grid rule, or either of the line rule violations.

The program will find all of the constraint violations and randomly remove one, if this creates a feasible solution it moves on, if not it will keep trying to repair.

# Mutation

Mutation will be dealt with two steps:

- If there is an empty space, it will mutate a random empty space.

- Otherwise randomly change a space on the puzzle that is not an initialized value.

# Alternative method

There will be a similar program that works with multiple objective evolutionary algorithms, that will use no repair method and have two objectives.

These objectives are, number of spaces filled and the number of constraint violations.

# Comparison

The project will overall compares the two methods described based on, how fast they can solve the puzzles and how many puzzles each method gets stuck on, it will also consider and sort them into other factors, such as puzzle size, and human puzzle difficulty.