AI Report

Design

Solution Space and Representation

The solution space can be thought of as any Sudoku grid (that is any 9x9 grid of numbers), once we take into account the fixed values given by the file, the solution space is cut down in size. To cut down the solution space further I have constrained it to only grids that follow the row constraint of the game (*i.e.* a row contains each value only once).

As for representation, I decided the best way to represent a grid with my given constraint on solution space is a list of 9 lists, where each inner list contains the values of a row.

Fitness Function

Counting the number of errors in rows, columns and 3x3 squares gives an indicator of how "correct" a solution is. Given I have already applied the row constraint to the solutions, I have opted to count the errors in columns and squares only.

Crossover Operator

There were a few ways this could've been done:

- Take the first half of rows from one solution and the second half of rows from the other
- Same as above but take a random pivot row
- For each row probabilistically select which solution to take it from

These could of course also be applied to columns or squares as opposed to rows; with my representation rows made the most sense. I chose the third methodology as my crossover operator.

Mutation Operator

The mutation could be done in a myriad of ways. Ideally, it should maintain variance without losing much of the desirable properties of the solution. As such, I take each cell and swap it with a random (non-fixed) cell from the same row.

Initialisation

To initialise the population I take the set of numbers and assign a random permutation to each row, making sure to follow the row constraint of Sudoku.

Selection

Upon doing some research I found a few methods for selection.

Truncation: The simplest method, you sort the population by fitness and take the top x solutions.

Roulette Wheel: Give each solution a portion of the roulette wheel based on their fitness and pick a value. Begin summing their fitness scores and when the partial sum passes the value take that solution. This gives lower scoring solutions a non-zero chance of selection.

Tournament Selection: From the population select k solutions and take the best of them, repeat until new population is selected. Again giving a non-zero probability to lower scoring candidates.

Each method brings its own merits, in the end I opted for truncation due to its simplicity.

Termination

Termination is rather simple, when a solution registers 0 errors the problem is solved. However we don't want it to run forever so termination occurs when the error count is 0 or the generation limit is reached.

Questions

Question 1

The ideal population could be argued to be 100 or 1000. 100 gave a better fitness on the first two problems but 1000 gave a better solution on the third grid. 100, tended to be faster timewise but took more generations. I'd argue 100 still gave the best results.

Question 2

10 simply didn't have enough variance and whilst 10000 had plenty of variance the sheer amount of calculations required made it too slow to be of any real use. Thus 100 and 1000 gave a nice compromise.

Question 3

Judging by the fitness achieved by the EA, problem 2 was the easiest and problem 3 was the hardest.

Question 4

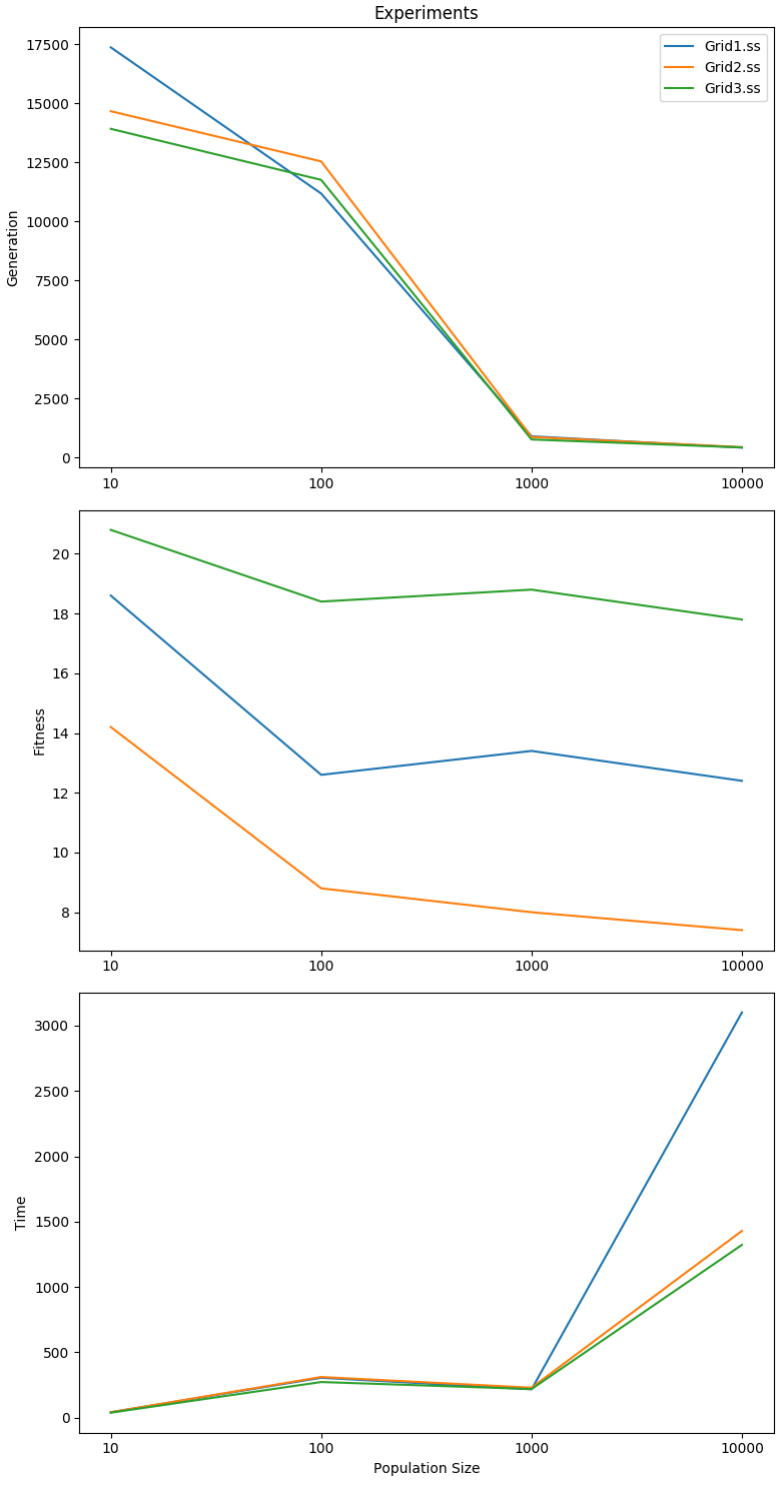
Grid 3 had more missing values and a less helpful spread of values. It had a row where no values were filled. Grid 2 had a nicer distribution and far fewer missing values.

Question 5

I think experimenting with the higher populations with a higher generation limit might give a good comparison. Also, running on an empty grid or yet more puzzles.

Experiments

Below you'll find graphs of my experiment findings. As expected from prior research, the results from using an EA were disappointing. An optimal solution was never found and they took a long time to run. Sudoku is much more easily solved via a backtracking algorithm or ACO.


Experiments
Grid1.ss
Grid2.ss
Grid3.ss
Generation
17500
15000
12500
10000
7500
5000
2500
0
10
100
1000
10000
Fitness
20
18
16
14
12
10
8
Time
3000
2500
2000
1500
1000
500
0
Population Size