Evolutionary Algorithm for Sudoku Puzzles

Report

# 1 Solution Space and Solution Representation

## Solution Space

-All columns, all rows and all little squares must contain each number between 0 and 9 exactly once.

-Incrementing: No double number in col, row or square. Terminate when all numbers filled in. Probably smaller search space

-All Fields where every number is filled in

## Solution Representation

Grid of numbers like the puzzle. No dots anymore -> all numbers

# 2 Fitness Function

Correctness combined with number of numbers already filled in.

# 3 Crossover Operator

# 4 Mutation Operator

# 5 Termination criterion

No empty space (dot) left.

1) Choose an appropriate solution space and solution representation.

2) Define an appropriate fitness function.

3) Define a crossover operator for the chosen representation.

4) Define a mutation operator for the chosen representation.

5) Choose an appropriate termination criterion.

6) Implement an evolutionary algorithm in Python following the pseudocode given in the previous section and the problem-specific components above.

7) Run experiments for the three Sudoku grids provided on the ELE page, for population sizes 10, 100, 1000, 10000. Each experiment (i.e., a specific combination of grid and population size) needs to be ran 5 times (each one with a different random seed) and average performance across runs considered. In total these amount to 3 x 4 x 5 = 60 runs.

8) For the analysis of results, answer the following questions: a. What population size was the best? b. What do you think is the reason for your findings in question 8.a? c. Which grid was the easiest and which the hardest to solve? d. What do you think might be the reason for your findings in question 8.c? e. What further experiments do you think it may be useful to do and why?