Computational Intelligence for Optimization Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Sudoku Solver using Genetic Algorithms

Group Jorge Palma

Gonçalo Cardoso, 20230588

Sara Jardim, 20230497

Sofia Afonso, 20230519

May, 2024

https://github.com/sofiarafonso/Sudoku-Solver

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

1. Introduction

The present report emerges from the Computational Intelligence for Optimization course of the Master's Degree in Data Science and Advanced Analytics at Nova IMS. It was proposed to evolve solutions to a Sudoku problem by the usage of Genetic Algorithms (GAs). In order to do so, it is necessary to represent the problem, design a fitness function and genetic operators, and define the type of optimization problem as well as the selection, crossover and mutation operators to apply.

2. Problem Definition

Sudoku is a number placement problem that relies on logic. The goal is to fill a 9x9 grid with numbers so that all of the digits from 1 to 9 are included in every row, column, and sub grid (the nine 3x3 sub grids, which will be referred as box). A proper Sudoku puzzle has a unique solution that can be reached logically. The main goal is to resolve this problem using GAs.

2.1. Representation

It was decided to represent the sudoku in a 9x9 matrix, where each array represents a row of the sudoku problem. This matrix has a total of 81 values, which each value can be any number from 1 to 9, and where the empty values in the initial sudoku, (values to be filled by evolved solutions) are represented with zero. It was represented 3 initial sudokus with different levels of difficulty (Easy, Medium and Hard) taken from *sudoku.com*.

3. Fitness Function

The Fitness Function was defined around a minimization problem. The *get_fitness* function checks for all duplicate values in rows, columns, and boxes. For each duplicate it adds 1 point in which the goal is to have 0 points (no duplicates neither in rows, columns, and boxes). Since the crossover operators do not affect the initial fixed values, the mutation operators were made in a way to not change these values. For this reason, the existence of duplicates is the only way of penalizing.

It was decided to only test this idea of fitness function. Considering that the only metric to be applied was the duplicates, it seemed redundant to implement other fitness functions.

4. Selection Methods

Three selection methods were tested: **Tournament Selection** (with size 5), **Fitness Proportionate Selection** and **Ranking Selection**.

In terms of results, the fitness proportionate selection method showed a better performance when compared to the others, by a huge difference. This makes sense when considering that it associates a probability of selection with each individual and that probability is proportional to its fitness score, while ranking selection is not sensitive to differences in fitness and that tournament choose a completely random set of individuals from each will be chosen the best.

5. Genetic Operators

For every operator tested, it was guaranteed that fixed and predetermined elements may not be swapped out of position, for both crossover and mutation operators.

5.1. Crossover

Three crossover operators were explored: **Single-Point Crossover**, **Two-Point Crossover**, **Uniform Crossover**. In single-point crossover, each parent was split in half to generate each child, and in two-point crossover the two parents were split in three, each child is 2/3 a parent and 1/3 the other. In uniform crossover, each value of the matrix is randomly assigned to a child.

All operators showed better results at the presence of the insertion mutation with exception of the two-point crossover that had slightly better results with the presence of the swap mutation in the same column.

5.2. Mutation

Two mutation operators were tested: **Swap Mutation** and **Insertion Mutation**. For swap mutation, it was applied two different approaches, swap values in the same row and in the same column.

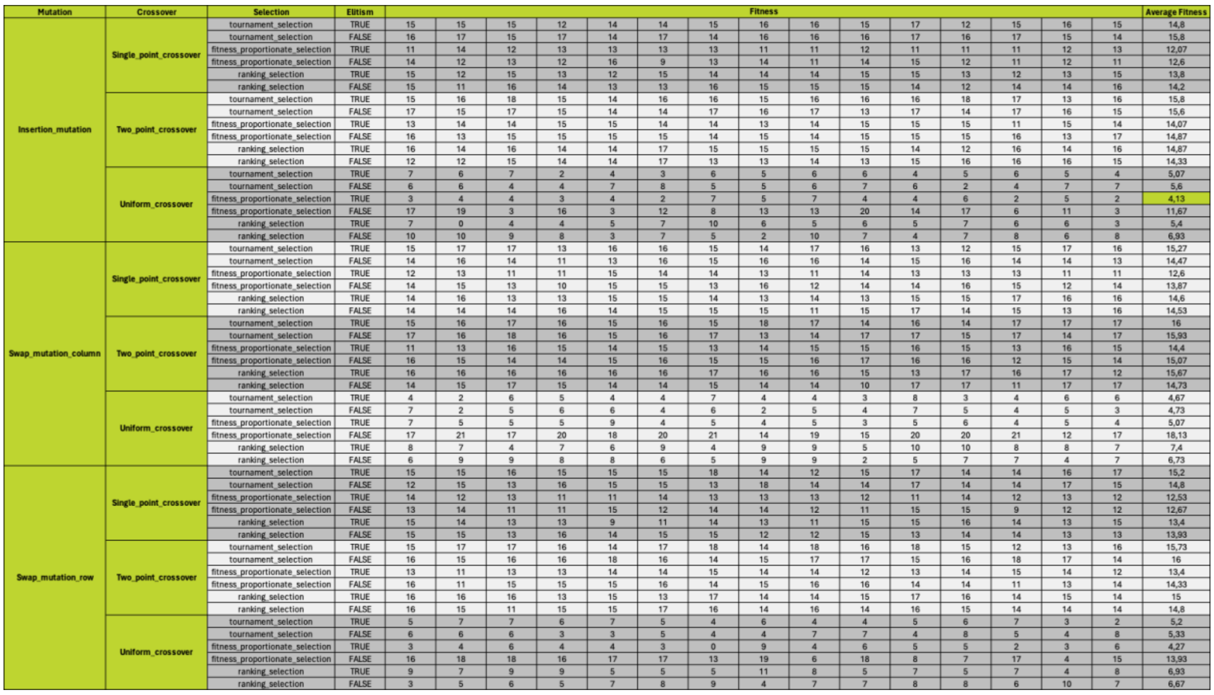
The insertion mutation showed a better behavior compared to all others, independently of the operators applied for crossover.

6. Comparison between Models

It was implemented all possible combinations of mutation (probability of 0.2), crossover (probability of 0.8) and selection methods, with and without elitism, in order to obtain the best GA. These combinations were obtained through a benchmark of 15 runs for each combination where the population was 500 in size and 200 generations, as it can be seen in the Table 1.

The best total average was obtained with insertion mutation, uniform crossover, fitness proportionate selection and with elitism, and has an average fitness score of 4.13. The implementation of elitism impacted positively the GA with a huge difference when selecting with the fitness proportionate method (11.67 average fitness in the chosen combination without, and 4.13 with elitism) whereas in the other selection methods the difference was minimal.

Besides the average finesses obtained, it should be also considered the convergence during the evolution of populations. It could be concluded that the two-point crossover had tendency to not converge, due to its oscillation, independently of the crossover operator. This makes sense considering the operator is deteriorating the sudoku sequential solution. Although the insertion mutation was the one with more tendency to converge, when combined with the two-point crossover, it was not the case. Also, it can be noted that the algorithms with swap mutation column had tendency to get stuck in a certain fitness value.



Mutation	Crossover	Selection	Elitism	Fitness															Average Fitness
Insertion_mutation	Single_point_crossover	tournament_selection	TRUE	15	15	15	12	14	14	15	16	16	15	17	12	15	16	15	14,8
		tournament_selection	FALSE	16	17	15	17	14	17	14	16	16	16	17	16	17	15	14	15,8
		fitness_proportionate_selection	TRUE	11	14	12	13	13	13	13	11	11	12	11	11	11	12	13	12,07
		fitness_proportionate_selection	FALSE	14	12	13	12	16	9	13	14	11	14	15	12	11	12	11	12,6
		ranking_selection	TRUE	15	12	15	13	12	15	14	14	14	15	15	13	12	13	15	13,8
		ranking_selection	FALSE	15	11	16	14	13	13	16	15	15	15	14	12	14	14	16	14,2
	Two_point_crossover	tournament_selection	TRUE	15	16	18	15	14	16	16	15	16	16	16	18	17	13	16	15,8
		tournament_selection	FALSE	17	15	17	15	14	14	17	16	17	13	17	14	17	16	15	15,6
		fitness_proportionate_selection	TRUE	13	14	14	15	15	14	14	13	14	15	15	15	11	15	14	14,07
		fitness_proportionate_selection	FALSE	16	13	15	15	15	15	14	15	14	15	15	15	16	13	17	14,87
		ranking_selection	TRUE	16	14	16	14	14	17	15	15	15	15	14	12	16	14	16	14,87
		ranking_selection	FALSE	12	12	15	14	14	17	13	13	14	13	15	16	16	16	15	14,33
	Uniform_crossover	tournament_selection	TRUE	7	6	7	2	4	3	6	5	6	6	4	5	6	5	4	5,07
		tournament_selection	FALSE	6	6	4	4	7	8	5	5	6	7	6	2	4	7	7	5,6
		fitness_proportionate_selection	TRUE	3	4	4	3	4	2	7	5	7	4	4	6	2	5	2	4,13
		fitness_proportionate_selection	FALSE	17	19	3	16	3	12	8	13	13	20	14	17	6	11	3	11,67
		ranking_selection	TRUE	7	0	4	4	5	7	10	6	5	6	5	7	6	6	3	5,4
		ranking_selection	FALSE	10	10	9	8	3	7	5	2	10	7	4	7	8	6	8	6,93
Swap_mutation_column	Single_point_crossover	tournament_selection	TRUE	15	17	17	13	16	16	15	14	17	16	13	12	15	17	16	15,27
		tournament_selection	FALSE	14	16	14	11	13	16	15	16	16	14	15	16	14	14	13	14,47
		fitness_proportionate_selection	TRUE	12	13	11	11	15	14	14	13	11	14	13	13	13	11	11	12,6
		fitness_proportionate_selection	FALSE	14	15	13	10	15	15	13	16	12	14	14	16	15	12	14	13,87
		ranking_selection	TRUE	14	16	13	13	15	15	14	13	14	13	15	15	17	16	16	14,6
		ranking_selection	FALSE	14	14	14	16	14	15	15	15	11	15	17	14	15	13	16	14,53
	Two_point_crossover	tournament_selection	TRUE	15	16	17	16	15	16	15	18	17	14	16	14	17	17	17	16
		tournament_selection	FALSE	17	16	18	16	15	16	17	13	14	17	17	15	17	14	17	15,93
		fitness_proportionate_selection	TRUE	11	13	16	15	14	15	13	14	15	15	16	15	13	16	15	14,4
		fitness_proportionate_selection	FALSE	16	15	14	14	15	16	15	15	16	17	16	16	12	15	14	15,07
		ranking_selection	TRUE	16	16	16	16	16	16	17	16	16	15	13	17	16	17	12	15,67
		ranking_selection	FALSE	14	15	17	15	14	14	15	14	14	10	17	17	11	17	17	14,73
	Uniform_crossover	tournament_selection	TRUE	4	2	6	5	4	4	7	4	4	3	8	3	4	6	6	4,67
		tournament_selection	FALSE	7	2	5	6	6	4	6	2	5	4	7	5	4	5	3	4,73
		fitness_proportionate_selection	TRUE	7	5	5	5	9	4	5	4	5	3	5	6	4	5	4	5,07
		fitness_proportionate_selection	FALSE	17	21	17	20	18	20	21	14	19	15	20	20	21	12	17	18,13
		ranking_selection	TRUE	8	7	4	7	6	9	4	9	9	5	10	10	8	8	7	7,4
		ranking_selection	FALSE	6	9	9	8	8	6	5	9	9	2	5	7	7	4	7	6,73
Swap_mutation_row	Single_point_crossover	tournament_selection	TRUE	15	15	16	15	15	15	18	14	12	15	17	14	14	16	17	15,2
		tournament_selection	FALSE	12	15	13	16	15	15	13	18	14	14	17	14	14	17	15	14,8
		fitness_proportionate_selection	TRUE	14	12	13	11	11	14	13	13	13	12	11	14	12	13	12	12,53
		fitness_proportionate_selection	FALSE	13	14	11	11	15	12	14	14	12	11	15	15	9	12	12	12,67
		ranking_selection	TRUE	15	14	13	13	9	11	14	13	11	15	15	16	14	13	15	13,4
		ranking_selection	FALSE	15	15	13	16	14	15	15	12	12	15	13	14	14	13	13	13,93
	Two_point_crossover	tournament_selection	TRUE	15	17	17	16	14	17	18	14	18	16	18	15	12	13	16	15,73
		tournament_selection	FALSE	16	15	16	16	18	16	14	15	17	17	15	16	18	17	14	16
		fitness_proportionate_selection	TRUE	13	11	13	13	14	14	15	14	14	12	13	14	15	14	12	13,4
		fitness_proportionate_selection	FALSE	16	11	15	15	15	16	14	15	16	16	14	14	11	13	14	14,33
		ranking_selection	TRUE	16	16	16	13	15	13	17	14	14	15	17	16	14	15	14	15
		ranking_selection	FALSE	16	15	11	15	15	17	16	14	16	14	16	15	14	14	14	14,8
	Uniform_crossover	tournament_selection	TRUE	5	7	7	6	7	5	4	6	4	4	5	6	7	3	2	5,2
		tournament_selection	FALSE	6	6	6	3	3	5	4	4	7	7	4	8	5	4	8	5,33
		fitness_proportionate_selection	TRUE	3	4	6	4	4	3	0	9	4	6	5	5	2	3	6	4,27
		fitness_proportionate_selection	FALSE	16	18	18	16	17	17	13	19	6	18	8	7	17	4	15	13,93
		ranking_selection	TRUE	9	7	9	9	5	5	5	11	8	5	7	5	7	4	8	6,93
		ranking_selection	FALSE	3	5	6	5	7	8	9	4	7	7	8	8	6	10	7	6,67

Table 1 - Fitnesses for Benchmark GAs

7. Best Model Application

Considering the previous mentioned best combination of genetic operators and selection method, it was needed to finetune the algorithm, specifically, the probabilities of crossover and mutation.

It was tested, for crossover, probabilities between 0.7 and 0.95 with an increment of 0.05, and for each one, the probabilities of 0.01, 0.02, 0.05, 0.1, 0.15, 0.2 for mutation were applied. These probabilities were set as a low value for the mutation considering the nature of the sudoku problem, it would only prejudice the solution that the GA is trying to find. Moreover, to guarantee the genetic continuation, the crossover probabilities must be relatively high.

Considering this variety, it was implemented a grid search to find the best probabilities. The fitness of 5 runs of populations with size 1000 and 400 generations for each tuple of probabilities can be seen in Table 2. The best configuration ended up being set at 0.85 crossover probability and 0.2 mutation probability, considering it has achieved the lowest average fitness result, since it is a minimization problem and that the goal is to have fitness equal to 0.

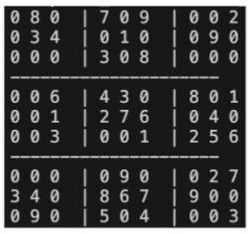
Crossover Probability	Mutation Probablity	Fitness					Average Fitness
0,7	0,01	4	5	4	2	5	4
0,7	0,02	5	5	5	2	3	4
0,7	0,05	5	2	3	2	4	3,2
0,7	0,1	4	6	2	0	4	3,2
0,7	0,15	3	3	0	2	3	2,2
0,7	0,2	3	3	4	3	4	3,4
0,75	0,01	3	4	4	4	6	4,2
0,75	0,02	4	4	3	3	6	4
0,75	0,05	4	5	6	5	4	4,8
0,75	0,1	3	3	4	7	5	4,4
0,75	0,15	3	2	3	2	6	3,2
0,75	0,2	5	0	3	6	5	3,8
0,8	0,01	4	6	6	4	5	5
0,8	0,02	4	3	2	4	3	3,2
0,8	0,05	4	3	0	3	0	2
0,8	0,1	5	3	3	5	3	3,8
0,8	0,15	3	2	3	2	2	2,4
0,8	0,2	3	3	2	4	2	2,8
0,85	0,01	2	5	4	4	2	3,4
0,85	0,02	4	3	3	3	3	3,2
0,85	0,05	4	3	2	3	4	3,2
0,85	0,1	2	4	4	4	2	3,2
0,85	0,15	4	3	3	3	0	2,6
0,85	0,2	0	0	3	3	3	1,8
0,9	0,01	4	4	2	5	2	3,4
0,9	0,02	6	3	4	2	4	3,8
0,9	0,05	4	2	2	2	2	2,4
0,9	0,1	6	6	3	3	3	4,2
0,9	0,15	3	4	2	2	3	2,8
0,9	0,2	2	3	2	3	2	2,4
0,95	0,01	6	4	2	4	5	4,2
0,95	0,02	3	4	6	5	4	4,4
0,95	0,05	4	2	5	2	4	3,4
0,95	0,1	3	6	3	3	3	3,6
0,95	0,15	4	4	2	4	3	3,4
0,95	0,2	7	2	3	4	6	4,4

Table 2 – Fitnesses of Probabilities' Finetuning

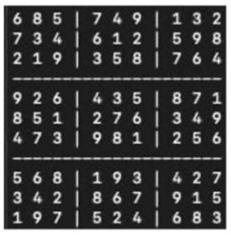
When applying the GA to all levels of the sudoku problem, with the increase of the difficulty, the population size and the number of generations were also increased.

- Easy level: population size 1000; generations 400; fitness 0.
- Medium level: population size 2000; generations 400; fitness 3.
- Hard level: population size 2000; generations 600; fitness 15.

Easy Sudoku Problem and GA Solution

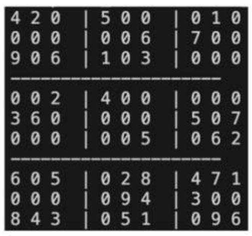


```
0 8 0  | 7 0 9  | 0 0 2
0 3 4  | 0 1 0  | 0 9 0
0 0 0  | 3 0 8  | 0 0 0
---------------------
0 0 6  | 4 3 0  | 8 0 1
0 0 1  | 2 7 6  | 0 4 0
0 0 3  | 0 0 1  | 2 5 6
---------------------
0 0 0  | 0 9 0  | 0 2 7
3 4 0  | 8 6 7  | 9 0 0
0 9 0  | 5 0 4  | 0 0 3
```

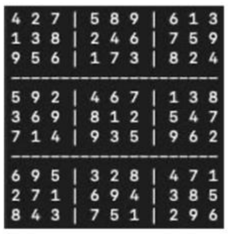


```
6 8 5 | 7 4 9 | 1 3 2
7 3 4 | 6 1 2 | 5 9 8
2 1 9 | 3 5 8 | 7 6 4
---------------------
9 2 6 | 4 3 5 | 8 7 1
8 5 1 | 2 7 6 | 3 4 9
4 7 3 | 9 8 1 | 2 5 6
---------------------
5 6 8 | 1 9 3 | 4 2 7
3 4 2 | 8 6 7 | 9 1 5
1 9 7 | 5 2 4 | 6 8 3
```

Medium Sudoku Problem and GA Solution

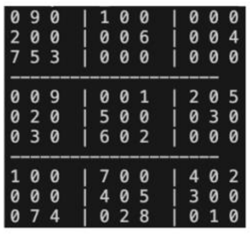


```
4 2 0  | 5 0 0  | 0 1 0
0 0 0  | 0 0 6  | 7 0 0
9 0 6  | 1 0 3  | 0 0 0
---------------------
0 0 2  | 4 0 0  | 0 0 0
3 6 0  | 0 0 0  | 5 0 7
0 0 0  | 0 0 5  | 0 6 2
---------------------
6 0 5  | 0 2 8  | 4 7 1
0 0 0  | 0 9 4  | 3 0 0
8 4 3  | 0 5 1  | 0 9 6
```

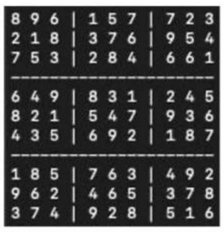


```
4 2 7 | 5 8 9 | 6 1 3
1 3 8 | 2 4 6 | 7 5 9
9 5 6 | 1 7 3 | 8 2 4
---------------------
5 9 2 | 4 6 7 | 1 3 8
3 6 9 | 8 1 2 | 5 4 7
7 1 4 | 9 3 5 | 9 6 2
---------------------
6 9 5 | 3 2 8 | 4 7 1
2 7 1 | 6 9 4 | 3 8 5
8 4 3 | 7 5 1 | 2 9 6
```

Hard Sudoku Problem and GA Solution



```
0 9 0  | 1 0 0  | 0 0 0
2 0 0  | 0 0 6  | 0 0 4
7 5 3  | 0 0 0  | 0 0 0
---------------------
0 0 9  | 0 0 1  | 2 0 5
0 2 0  | 5 0 0  | 0 3 0
0 3 0  | 6 0 2  | 0 0 0
---------------------
1 0 0  | 7 0 0  | 4 0 2
0 0 0  | 4 0 5  | 3 0 0
0 7 4  | 0 2 8  | 0 1 0
```



```
8 9 6 | 1 5 7 | 7 2 3
2 1 8 | 3 7 6 | 9 5 4
7 5 3 | 2 8 4 | 6 6 1
---------------------
6 4 9 | 8 3 1 | 2 4 5
8 2 1 | 5 4 7 | 9 3 6
4 3 5 | 6 9 2 | 1 8 7
---------------------
1 8 5 | 7 6 3 | 4 9 2
9 6 2 | 4 6 5 | 3 7 8
3 7 4 | 9 2 8 | 5 1 6
```

It is important to note that the GA may have not achieved the solution for the medium and hard problems due to it was only considered the initial easy level sudoku problem for the benchmark and finetuning.

8. Conclusion

The final results were considered to be good, considering the GA was able to solve the easy level, however, it did not meet expectations. The results could have been better if there would have been time to test more different genetic operators and combinations, which was highlighted by the GA not being able to solve the medium and hard level.

Besides this, it would have been interesting to implement a restart heuristic, where the GA is restarted with a new set of random solution strings if it gets stuck in a local minimum and, for example, fitness sharing to counteract the early convergence of some GAs. Furthermore, if more time were available, a specific finetuning could have been applied to each level of the problem, as well as adjusted the population size and number of generations to each combination of benchmark to search for more convergency.

9. Division of Labor

The structure of the code was firstly developed by Sofia. Gonçalo and Sara worked to improve its functionality and efficiency, implementing extra genetic operators, always considering the library from the practical classes. Regarding the report writing, code execution, and analysis, it was equally divided by the tree of us.

10. References

Weiss, J. M. (2009, April). Genetic algorithms and sudoku. In Midwest Instruction and Computing Symposium (MICS 2009) (pp. 1-9).

Vanneschi, L., & Silva, S. (2023). *Lectures on Intelligent Systems*. Springer.