Computational Intelligence for Optimization

Project Report

Hiromi Nakashima (m20201025@novaims.unl.pt)
Lucas Correa (m20211006@novaims.unl.pt)

The code and results can be found in this public repository:https://github.com/hnakashima96/CIFO_2022.

2nd Semester 2021/2022

NOVA IMS Information Management School

1. Introduction and Problem Definition

This report shows the application of Genetic Algorithms (GA) to find solutions to the Sudoku game. Sudoku is a puzzle made of a 9x9 grid where the goal is to fill all the 81 spaces with one of the numbers in the range 1-9 in a way where there are no duplicate numbers in any row, column or 3x3 (non overlapping) sub grids. There are different levels of difficulty in the Sudoku measured by the number of spaces already filled in the beginning of the game.

GA is a robust optimization algorithm which is based in Darwin's Theory of evolution on the idea of natural selection. The main concept is the improvement of the population through different variables that affect the evolution of the species: selection, reproduction, crossover, mutation, elitism etc.

This report will be divided into 3 parts, starting with Introduction and Problem Definition, where it will describe the representation choice of basic organisms in the GA, definition of the fitness function, genetic operators used and the methodology used to analyze the algorithm efficiency. Afterwards, the results of several combinations of different genetic operators will be discussed in order to define the best approach for the problem in the Results section. Finally, there is the conclusion.

The development of the project was split into structural problems (Sudoku problem), library of genetic algorithms and development of solutions (which merge the two mentioned parts). The first one was mostly made by Hiromi Nakashima, the second one by Lucas Correa and the last part was created together, just like this report. It is relevant to point out that all steps were revised by each other.

1.1. Representation choice

In order to address the Sudoku problem with GA, the first step is to decide how to represent an individual in the Sudoku problem. For this case, each individual will be a sudoku with a random not optimum solution. The only condition to initialize an individual is that each row cannot have duplicates numbers. This constraint was necessary in order to avoid 2 types of sudokus in the initialization: (i) ones with the final solution and (ii) sudokus that have more than 9 times the same number in the whole sudoku. In the first unwanted case, there would be no problem to solve and in the second it would add a whole new layer of complexity to the problem where it might take longer to converge into a solution.

1.2. Fitness function

Given the concept of an individual, the second step was the decision of how to classify each individual as more "adapted" than others. In other words, what is the criteria for an individual to be better than the other.

Since the goal of the Sudoku is to not have duplicate numbers in rows, columns or sub grids, the first option is to consider an individual with less duplicates better than the one with more and ultimately, the individual with no duplicates would be the best one. In this sense, this would be a minimization problem where the fitness function would be assigned by the sum of duplicates for each row, column and 3x3 sub-grid and the optimum solution is the resulting total value of 0.

Another possible approach to determine the fitness function would be to design the optimization problem in the opposite way. Instead of looking for the duplicates, look for the unique numbers of elements in the row, column and 3x3 sub-grid. This would then be a maximization problem where the higher sum of unique in the defined dimensions is 243, which is the final solution.

1.3. Genetic Operators (G.O.)

In order to achieve the optimum solution for any of the abstractions described before, one needs to define which genetic operators to use for the task. The decision of the G.O. has a direct affect in the performance and % of optimum solutions found as it will be seen in the next sections. For now, it will be defined the operator used:

For selection, 3 types of operators: fitness proportion selection (roulette wheel), rank selection and tournament selection with 5 individuals for the tournament choice. For the crossover were utilized *single point* crossover, applied in all grid, *parallel mapped crossover (pmx)* and *cycled crossover*, applied on lines of the grid. The last two crossovers were selected for its capability to solve the position problem of the standar crossover and to avoid duplicate values along rows of a Sudoku during the evolution process.

During all the experiment it were applied two types of mutation operators, the *swap mutation* where two numbers of a row in a given individual would swap and the *permutation mutation* where it takes a random row in a Sudoku and mix all the changeable (not given in the initialization of the puzzle) values in one of all the possible permutation with those numbers. The first approach of mutation is more subtle while the second is a drastic mutation.

Finally, one very comum operator used is the elitism where by selecting the best individuals of a population it was expected to converge to a faster optimum solution. For this matter, there were two possible values for it: 0 or no elitism, and 0.3 as 30% of the current population to be selected. The reasoning of a lower elitim value is due to the fast convergence in Sudoku problems and fast loss of diversity in the population, which could be enlarged by higher values of elitism.

1.4. Analysis

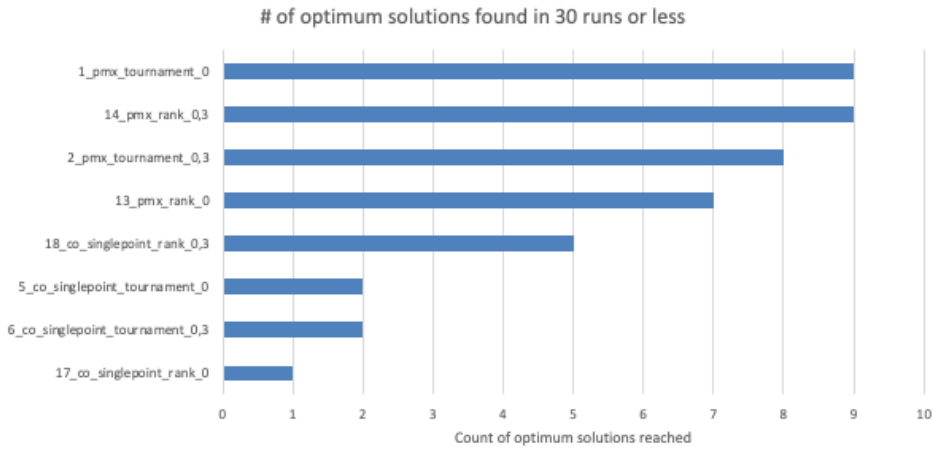
In order to analyze the results of the algorithm, two types of analysis were made: iterations and computational effort. The iterations analysis considered a population of 2500 individuals and up to 30 iterations in a maximization and minimization problem, limiting every iteration to 40 generations. If the algorithm found the optimum fitness before the 40th generation then the algorithm would stop. For this first analysis, it was considered the number of optimum solutions achieved in all the 30 iterations, the average number of best fit and average time taken to finish an iteration. Within the possibilities of selection, crossover and elitism, in table 1 there are all permutations of the genetic operators that were compared.

The combinations which could reach the perfect fitness were filtered and did a computational effort comparison with a population of 5000 individuals and 20 iterations. The results section will show the criterias to be considered the best combinations. The 20 iterations have the same understanding of the comparison mentioned before. In table 2 present the table for this comparison:

2. Results

2.1 - Maximization Results - Population Size: 2500

As mentioned in the previous section, the first metric to be considered is the number of optimum solutions found. By this metric, more than half of the combinations are cutted off and only 8 remain as the ones that reached the optimum solution at some point. In the graph 1 it is presented the number of optimum solutions found by combination.

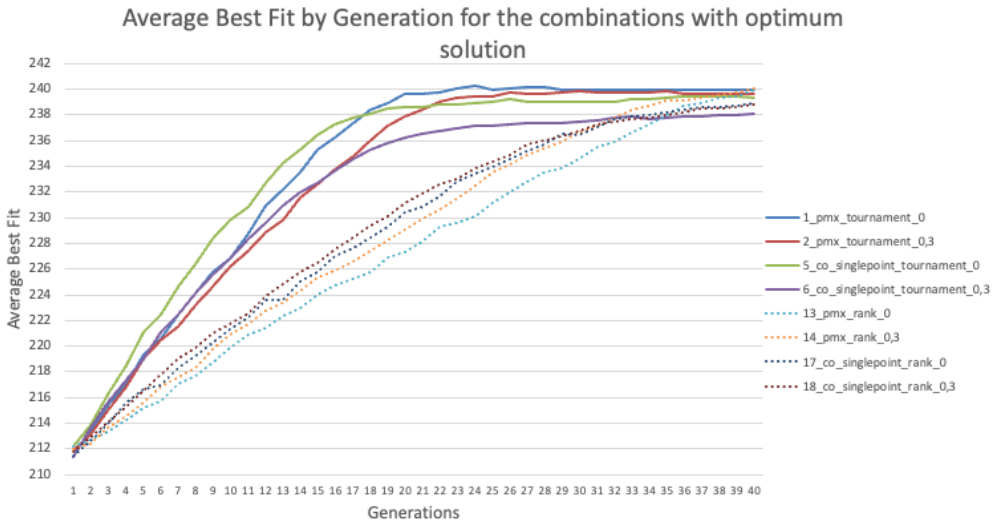


Graph 1 - Number of optimum solutions per combination of G.O in maximization.

It is possible to notice that any combination involving the *roulette wheel* selection or the *cycled crossover* did not reach a final solution for the Sudoku considered and thus could be seen as not good approaches to address this problem.

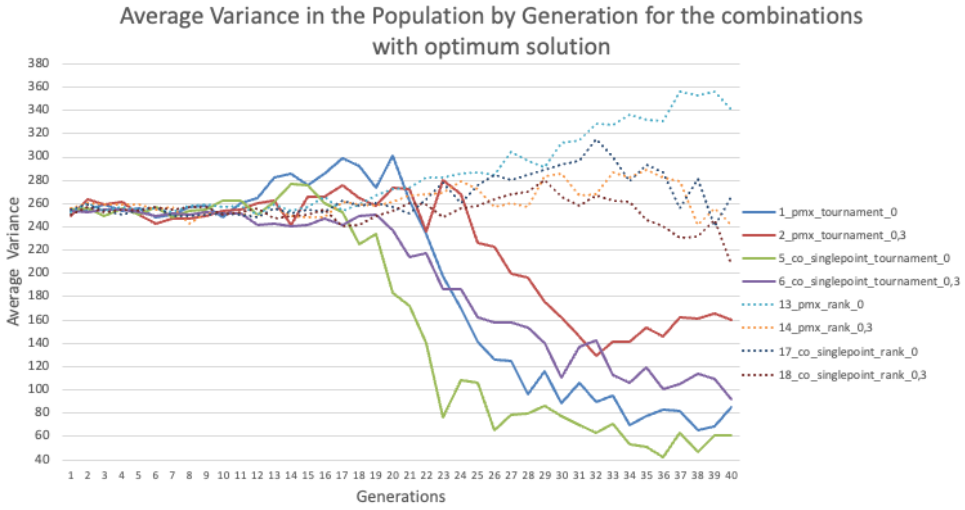
Another interesting point is regarding the effect of elitism in each of the combinations that had passed the first cut off point. Apparently, for the tournament selection the *elitism* has the effect of reaching better fitness in a slower pace, whereas for the *rank selection* is the opposite. This is probably due to the fact that with *elitism*, the probabilities in the rank selection are better distributed, once there are fewer individuals to share the total probability of being picked. While in the tournament selection elitism is just a faster way to reduce the variety of the population making it harder to cross over and come up with a good combination of genome.

To confirm that assumption, the computation of the population variance was done. This metric was used to track early convergence due to lack of diversity and can be used now also to understand the impact of *elitism* in different combinations of genetic operators. As can be seen in the graph 3, the average variance within the population (measured by the hamming distance of each Sudoku in the population) is higher for the rank selections than for the tournament selections. The tournament selection, with *elitism*, shows a higher value of variance.



Graph 2 - Average Best Fit by generation

Even though at first this suggests that *elitism* helps *tournament's* to be more diverse and thus avoid a local optima by fast convergence, this behavior is in fact happening because this selection operator reaches the optimum solution earlier than without elitism and thus had a lower variance in the end.



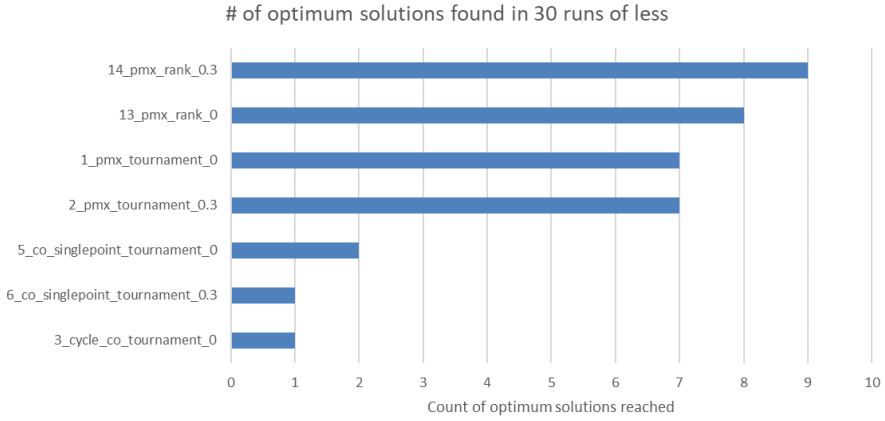
Graph 3 - Average Variance in the Population

2.2 - Minimization Results - Population Size: 2500

Following the same approach of maximization analysis, the first analysis is to check which combinations reached the optimum solution. Given this analysis, only 7 combinations could solve the sudoku. As noticed in graph 4; 5 of the 7 combinations that reached the solutions were using *tournament* as selection method, although the ones which used *rank selection* reached more times to the optimum solution.

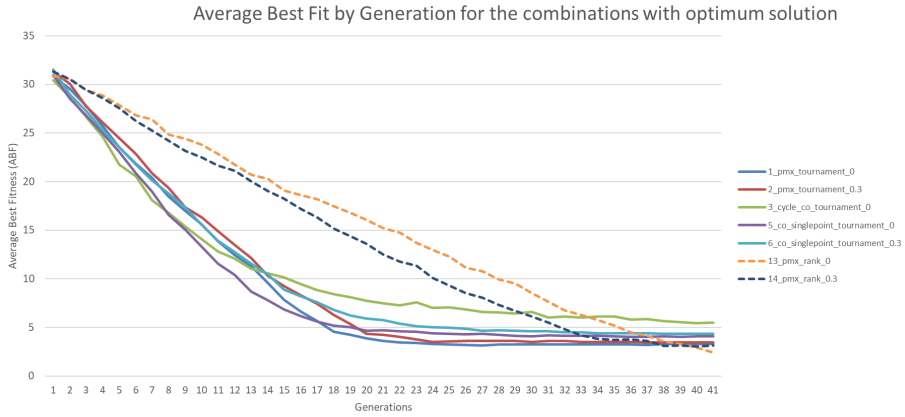
As in the maximization analysis, the combinations with *roulette wheel* have not solved the problem given the parameters determined. Meanwhile, *cycle crossover* could return once the optimum solution.

From use of *elitism* point of view, it had a similar behavior as maximization. Most of the combinations which reached the optimum solution without this parameter, kept the times they returned the solution or improved like the combinations 13 and 14.



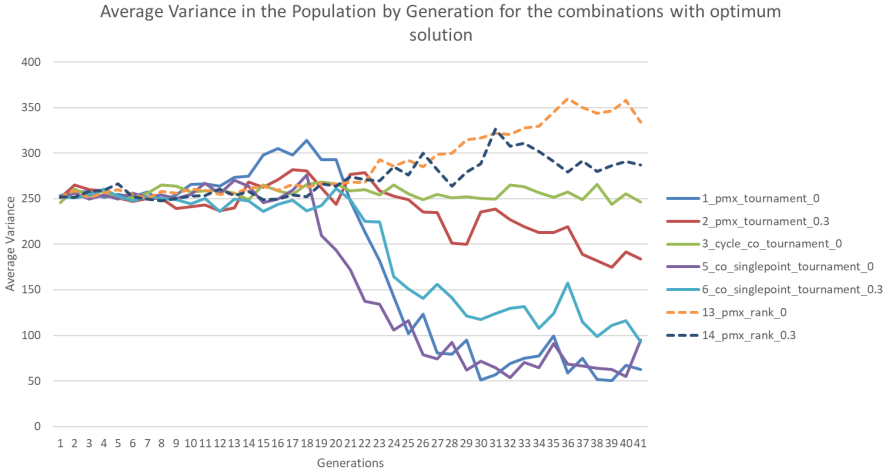
Graph 4 - Number of optimum solutions per combination of G.O in minimization.

In Graph 5, compared to the results on the graph 2, the combinations using rank selection take more generations to reach optimum solutions. The combinations using tournament, reached a "plato" at the same generation as in the maximization problem, approximately of the 20th generation.



Graph 5 - Average Best Fit by generation in minimization problem.

As in the maximization problem, the graph 6 shows that variance increases on the combinations with *rank selection*. *Elitism* usually restinged the variation of the population, except for the pmx_tournament which increased the average variation.

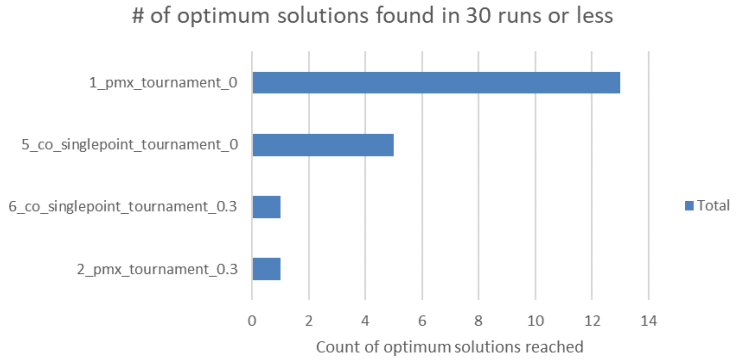


Graph 6 - Average variance in population by generation in minimization problem.

The highlight of this result plot was the combination 3 that used cycle crossover. It kept an average variance almost the same for all generations.

2.3 - Maximizations solutions - Population Size: 5000

This analysis was made to identify the comparison of performance and computational effort. In this case, the population size was doubled and divided by two the number of generations. Compared with the combinations only 4 of 8 reached the optimum solution (Graph 7).



Graph 7 - Number of optimum solutions per combination of G.O in maximization with population size of 5000 individuals.

It is possible to notice that the increase of population improved the times that combinations of pmx_tournament and co_singlepoint_ tournament without elitism reached the optimum solution. Meanwhile, the same combinations with elitism decreased, but it affected the pmx_tournament more.

Table 1 - Mean of time (in seconds) to reach to the optimum solution

Combination	Pop. Size = 2500 (time - s)	Pop. Size = 5000 (time - s)
1_pmx_tournament_0	20.63	33.97
2_pmx_tournament_0.3	15.82	25.19
5_co_singlepoint_tournament_0	14.22	21.43
6_co_singlepoint_tournament_0.3	11.41	16

The main goal of this analysis is to evaluate if the computational effort is improved by the increase of the population. Table 1 shows that the increase of the population takes more time to reach the solutions for all combinations.

3. Conclusion

After this project was possible to conclude the following statements:

- In all analysis, the parallel mapped crossover was the crossover with best performance. Meanwhile, the tournament selection had good results just like the rank selection.
- Elitism has not shown to be a relevant improvement for this type of problem, and sometimes could even be considered a constraint.
- The increase of the population is not guaranteed to reach an optimum solution faster, but in some cases as shown in section 2.3, graph 7, it improved to reach the result easily.

4. Anexs

Table 1 - Combinations of G.O.

Combinations	Selection	Crossover	Elitism
1	rank	co_singlepoint	0
2	rank	cycle_co	0
3	rank	pmx	0
4	roulette	co_singlepoint	0
5	roulette	cycle_co	0
6	roulette	pmx	0
7	tournament	co_singlepoint	0
8	tournament	cycle_co	0
9	tournament	pmx	0
10	rank	co_singlepoint	0,3
11	rank	cycle_co	0,3
12	rank	pmx	0,3
13	roulette	co_singlepoint	0,3
14	roulette	cycle_co	0,3
15	roulette	pmx	0,3
16	tournament	co_singlepoint	0,3
17	tournament	cycle_co	0,3
18	tournament	pmx	0,3