

**GitHub:** <https://github.com/Dan20210677/CIFO-TSP-PROJECT>

**Project**

We coded 3 selection methods (fps, ranking and tournament), 4 mutation methods (swap, insertion, inversion, and displacement) and 3 crossover methods (cycle, ordered and single point (pmx)).

**Why did you choose the representation you did?**

This project aims to use Genetic Algorithm (GA) knowledge to solve an optimization problem. The problem chosen was the Travelling Salesman Problem since we thought it was the most applicable to the subject and the materials provided in the classes. This problem uses various types of selection, mutation, and crossover and with this in mind the group got the opportunity to implement and understand better all the different parts of genetic algorithms from start to finish.

We also considered the Asymmetric Travelling Salesman Problem, choosing two TSP, one small (16) and one larger (76), and two ATSP, also one small (17) and one larger (70), to understand the difference between them.

**Which configurations worked best together?**

There isn't a specific best configuration, it depends on the problem, mostly in terms of size, and it’s a probabilistic method, however, based on the results, the selections methods ranking, and tournament seem to work best, for crossover methods, cycle crossover has a slight advantage, the inversion and displacement mutation also got better results and elitism almost always gets superior performance.

**How many did you try, and how did you determine the "best" one?**

For the two smallest problems, we tried all different combinations of selection, mutation and crossover algorithms implemented as well as with and without elitism with a total of 72 combinations.

We ran each combination 100 times for 50 generations and created a table with the best outcome, the mean value and standard deviation. (Figure 1, Figure 2)

Just the best 8 combinations were chosen and tested on the two largest problems, running each combination 30 times for 3000 generations creating new tables. (Figure 3, Figure 4)

We kept the probabilities of mutation and crossover fixed (0.2 and 0.8 respectively) and the population size was kept at 100.

The best configurations were chosen based on the tables, but afterwards we can compare any two combinations any number of times we require for any number of generations by plotting the curve of evolution of the mean values of each generation. (Figure 5)

**How did you design the fitness function?**

Given that the main objective of the TSP/ATSP problem is to find a roundtrip of minimal total length visiting each node exactly once, we consider our fitness function as the total sum of all the distances between each two consecutive nodes of the path.

**Did you try using different fitness functions, to see the impact on your GA?**

The fitness function for TSP/ATSP is the length of trace, with the objective of minimizing such distance by finding the best path.

With this in mind, the only change to the fitness function would be to adapt it to a maximization problem.

**Do different operators affect the convergence of your GA?**

We were able to observe, mostly on the larger problems, that different operators affect the convergence of the GA, however, on the small problems there isn’t much of a difference.

Different operators have different speeds of convergence, and some operators are more prone to get stuck in local optimums, although all operators gave satisfactory results.

**Have you implemented elitism?**

We tried two types of elitism, just maintaining the best individual, and substituting the worst individual with the best one.

Although both gave good results, we decided to stick with the latter since it was the most used. We didn’t implement elitism with multiple individuals.

**Does the inclusion or exclusion of elitism impact your GA?**

Yes, the inclusion elitism impacts our GA in a positive way since when searching for the global optimum we converge to the result faster and such result is closer to the global optimum.

**Are your implementations abstract and work with both minimization and maximization?**

Yes.

**Do you get good results for the project you chose?**

For the small problems, the GA algorithm almost always achieved the global optimum in 50 generations, which is excellent. For the larger problems, even after 3000 generations, the algorithm did not return the global optimum, but returned somewhat good results (local optimums), however seeing the evolution of the population each generation, maybe with more generations the algorithm could reach global optimum.

Also, since one objective of this project was to compare different operators, on that point, good results were also obtained, getting to understand clear differences between each configuration.

**What could be improved?**

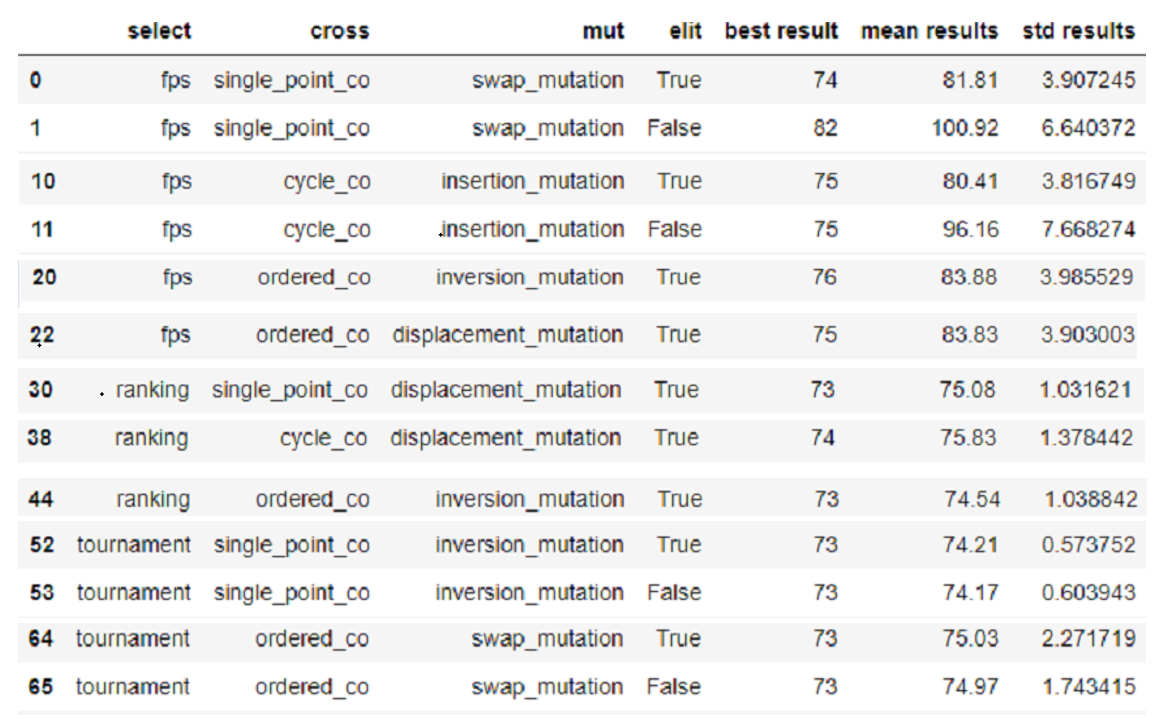
Although we got good results, we could have gotten even better if we increased the number of generations. This would have led to a slower computation, so we had in consideration the best ratio of number of generations and number of repetitions vs time. Taking this into consideration, with more powerful equipment this situation wouldn’t be as challenging, and we could increase the number of generations which would lead to a result closer to the global optimum.

However, our project is already getting very good results. We could have also implemented different types of crossovers, mutation, selection, and elitism in order to get different and hopefully better results.

Another worry we considered was the loss of diversity, which would cause the algorithm to get stuck in a local optimum, making increasing the number of generations ineffective. There are some techniques that could be implemented to prevent this problem and improve the algorithm.

**Discussion and interpretation of the results obtained**

Although we created tables with every possible combination of implementation of the operators for the small problems, for being too extended we decided to just show some of the most relevant combinations. The tables are shown below.

Graphical user interface

Description automatically generated with medium confidence

Figure 1- ATSP 17 (partial) table

Figure 2- TSP 16 (partial) table

Based on the results, we can clearly see the positive impact of elitism, mostly on the fps selection operator, and differences between each operator, with fps selection getting the worst results, and on average, single point crossover (pmx), swap and insertion mutation give slightly worst results, the difference being not significative.

Graphical user interface

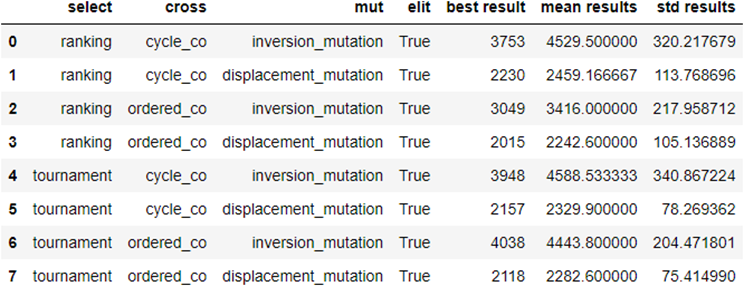
Description automatically generatedAfter analysing the results corresponding to the small problems, we decided to create the same tables just with the best types of combinations observed in the tables above with the purpose of simplifying it to the larger problems and with that get a better evaluation of the results. This is shown below.

Figure 4- TSP 76 table

Figure 3- ATSP 70 table

Based on the results, for the ATSP problem, inversion mutation always gave better results, and tournament selection performed slightly above ranking selection, and cycle crossover had worse results compared to ordered crossover. For the TSP problem, the mutations have inverted results, inversion presenting the best results, the rest being the same.

In case of doubt when comparing two combinations of operators, we coded a function to plot a statistical comparison along the evolutionary process. For example, running the ATSP 70 for 3000 generations 50 times each combination of operators, comparing the tournament and ranking selections and inversion and displacement mutations, fixing ordered crossover and elitism true (tournament and inversion in blue, ranking and displacement in red):

Chart

Description automatically generated

*Figure 5- comparison between combination of operators*

We implemented a function to plot the best individual of the population each generation when running the GA algorithm, to understand the behaviour of the population based on the combination of operators throughout the generations.

After studying the different combinations of operators, we decided to plot the evolution of the GA algorithm, for each of the problems, with one of the best combinations.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

Figure 9- ATSP 70 of the larger problems

Figure 8- ATSP 17 of the small problems

Figure 7- TSP 76 of the larger problems

Figure 6- TSP 16 of the small problems