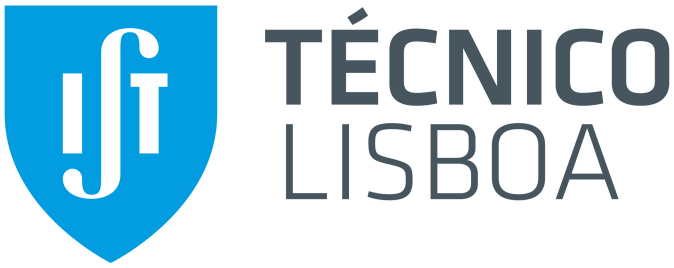

Instituto Superior Técnico

MEEC

Applied Computational Intelligence

Project 2 – EAs for Single and Multi-Objective Optimization

Work done by:	*Number:*
Tiago Almeida	96328
Rodrigo Contreiras	90183

Group 35

4th of November 2022

Introduction

The goal of this second project was to employ evolutionary computation to solve a variant of the traveling salesman problem. In our case, a **Genetic Algorithm** was used.

Moreover there where two objectives to it. The first was a single objective problem and the second was a multi-objective problem. The "singleobjective.py" and "TESTME.py" files pertain to the single objective, and the "multiobjective.py" file pertains to the multi-objective. All of them are explained on the "README.txt" and on the comments inside the code.

For the single objective problem, we advise utilizing the **"TESTME.py"** file for code analysis, since "singleobjective.py" is only used to create all the graphs for the delivery it's more complicated and takes about 40 minutes to run, while this file is simpler and much faster. The path taken by the salesman at the conclusion is also shown in the **"TESTME.py"** at the end.

Representation chosen of each problem

Single Objective

In order to create our population for the **Genetic Algorithm**, on the generic individual creation, we generated at random n numbers of customers (depending on the number of customers that the salesman had to travel to) to be added to one member of the population, without repetition. Then we also have two heuristics that can be used to create one member of the population. One works the same way as the one described in the project description. The other that we created and the one that achieved the best results, looked for the customer that was closest to its present location at each step and headed there.

```
[1, 0, 4, 9, 6, 5, 8, 2, 3, 7]
```

Figure 1: Path for salesman created randomly for a population of 10 customers. Value 0 = Customer 1, value 1 = Customer 2, ...

```
[8, 2, 1, 0, 9, 3, 6, 7, 5, 4]
```

Figure 2: Path for salesman created by the heuristic 1 for a population of 10 customers. Value 0 = Customer 1, value 1 = Customer 2, ...

```
[6, 4, 3, 7, 9, 5, 0, 8, 2, 1]
```

Figure 3: Path for salesman created by the heuristic 2 for a population of 10 customers. Value 0 = Customer 1, value 1 = Customer 2, ...

The other important aspect of the code is the evaluation functions used, responsible for deciding which was the best path. To do this, we start by retrieving the distance from the warehouse to the first individual of a population and the number of orders it carried to that point. Between each individual, we added the distance from both points and the number of orders the salesman had to take from the previous to the next customer. If the number of orders from the previous to the next customer would be above the maximum Truck Capacity, then we assumed that the distance the salesman had to travel, was now from this previous customer, back to the warehouse, where he would get the remaining orders, from which he would finally move to the next customer. At this stage, the new max number of orders he would be able to carry was $MaxTruckCapacity - OrdersToNextCustomer$.

Regarding the function used for the mating and mutation part of this algorithm, we decided to use the "tools.cxOrdered" and "tools.mutShuffleIndexes()", respectively. The first made sure that when doing the crossover between two individuals, the customers visited wouldn't be repeated, and the last would do the same but this time by shuffling the indexes as the mutation technique. Then to select the offspring that would integrate the next population, we used the tournament technique. For all variables tuning (probability for mutation, the probability for crossover, number of tournaments, ...) a trial and error method was applied to get the best parameters for each of the project directives.

Lastly, all the values for the minimum, median, and standard deviation values for all populations, as well as the best individual achieved, were stored for plotting purposes in the end.

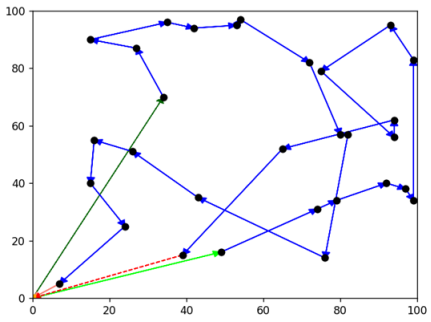

Figure 4: Best individual for a population of 30 with no heuristic. The dark green arrow represents the first time the salesman leaves the warehouse, the dark red last time he goes back to the warehouse.

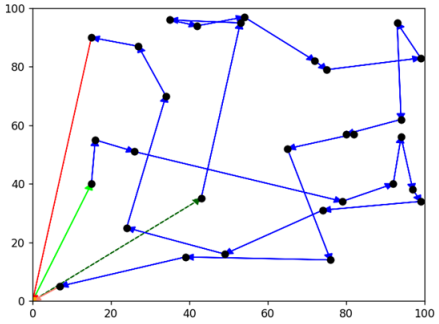

Figure 5: Best individual for a population of 30 with heuristic 2. The dark green arrow represents the first time the salesman leaves the warehouse, the dark red last time he goes back to the warehouse

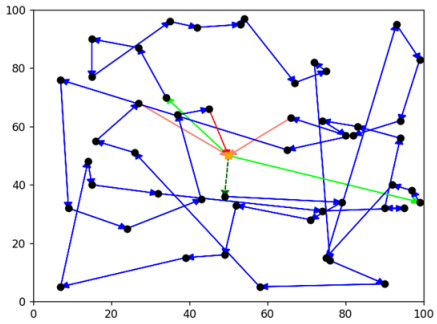

Figure 6: Best individual for a population of 50 with no heuristic. The dark green arrow represents the first time the salesman leaves the warehouse, the dark red last time he goes back to the warehouse.

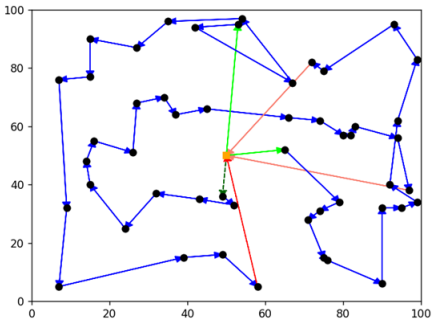

Figure 7: Best individual for a population of 50 with heuristic 1. The dark green arrow represents the first time the salesman leaves the warehouse, the dark red last time he goes back to the warehouse.

Multi-Objective

The multi-objective problem, comprises all the same deductions and functions that we used for the single objective, but now with the difference that the evaluation function, had to minimize two functions. The first is the same as the one as before, regarding the distance the salesman traveled, and the other, correlates the distance traveled between two customers and the amount of orders the later customer needed.

To see the results, we also introduce some code to calculate the pareto front and the hypervolume for each generation, as shown in section **3.2.1**.

Results

Here we present the results obtained and requested for the project. In the final section, we will analyze these results.

Section 2.2.1 - Single Objective

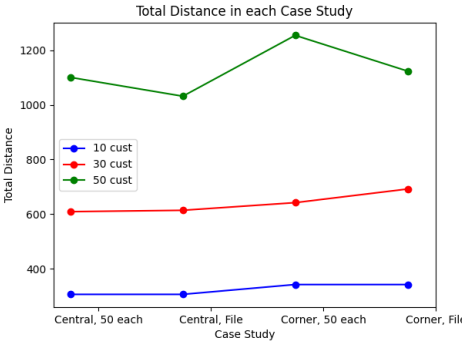

Figure 8: Plot of the total distance of the best run out of 30 for each case study.

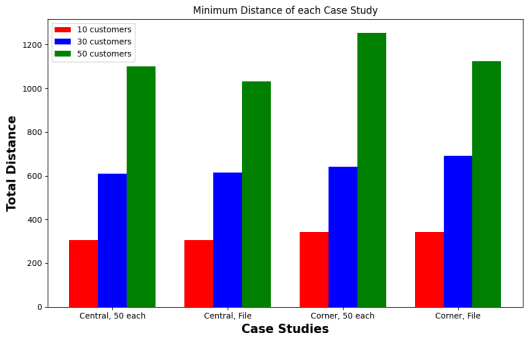

Figure 9: Bar plot of the total distance of the best run out of 30 for each case study.

	WHCentral_OrdFile		WHCentral_Ord50		WHCorner_OrdFile		WHCorner_Ord50	
#Customers	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	319,933	9,571	319,933	9,571	342	19,735	342	19,735
30	707,733	43,991	722,2	43,098	784,066	47,114	841,533	59,225
50	1321,233	77,596	1389,6	73,074	1321,233	77,596	1389,6	73,074

Figure 10: Mean and Standard Deviation of 30 runs

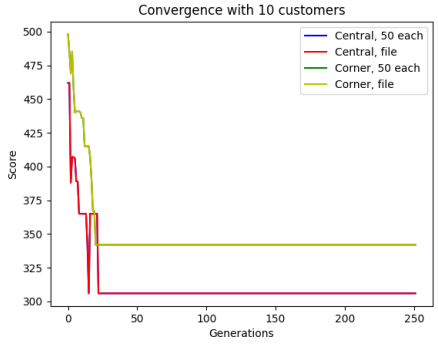

Figure 11: Convergence of the best run of each case study, with 10 customers.

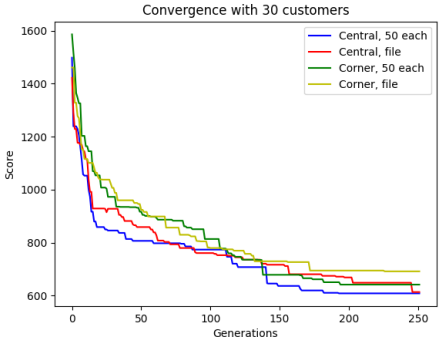

Figure 12: Convergence of the best run of each case study, with 30 customers.

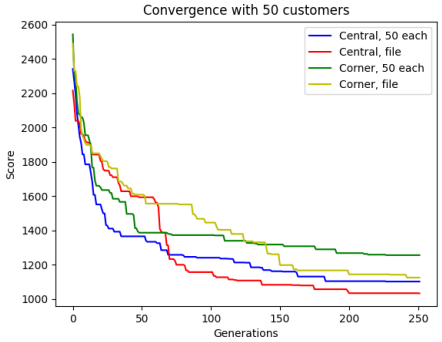

Figure 13: Convergence of the best run of each case study, with 50 customers

Section 2.3.1 - Heuristic

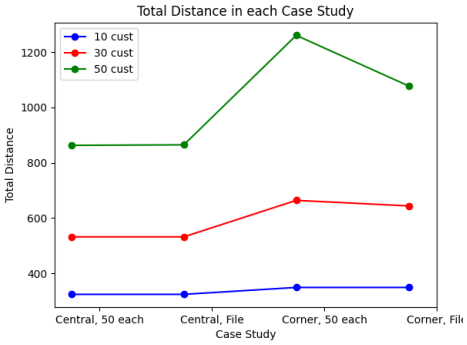

Figure 14: Plot of the total distance of the best run out of 30 for each case study.

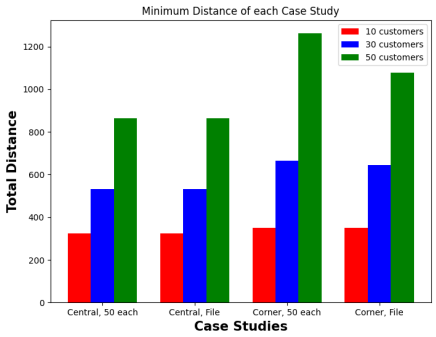

Figure 15: Bar plot of the total distance of the best run out of 30 for each case study.

	WHCentral_OrdFile		WHCentral_Ord50		WHCorner_OrdFile		WHCorner_Ord50	
#Customers	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	326,3	1,269	326,3	1,269	380,633	31,468	380,633	31,468
30	558,967	19,104	557,7	18,812	737,333	54,327	728,433	52,297
50	906,2	24,035	914,933	28,59	1314,1	78,839	1353,567	65,265

Figure 16: Mean and Standard Deviation of 30 runs

Section 3.2.1 - Multi Objective

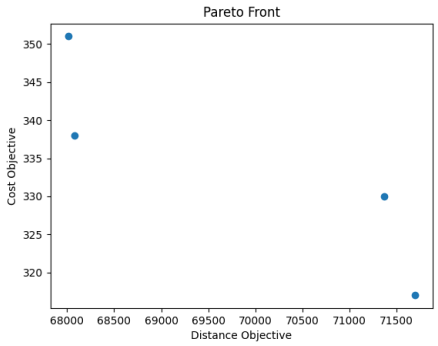

Figure 17: Pareto Front of the multiobjective problem with 10 customers.

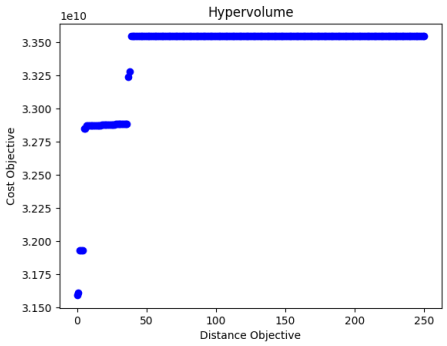

Figure 18: Hypervolume of the multiobjective problem with 10 customers.

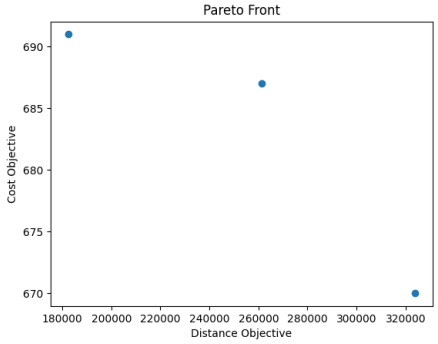

Figure 19: Pareto Front of the multiobjective problem with 30 customers.

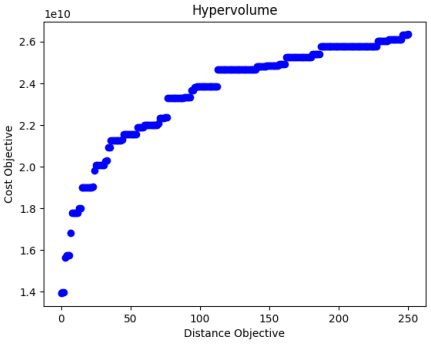

Figure 20: Hypervolume of the multiobjective problem with 30 customers.

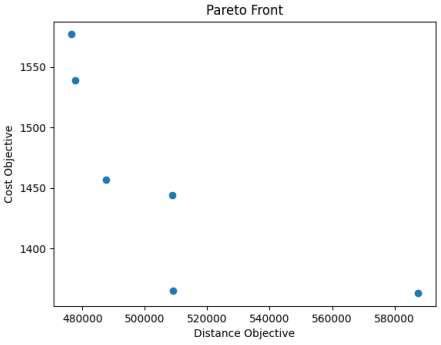

Figure 21: Pareto Front of the multiobjective problem with 50 customers.

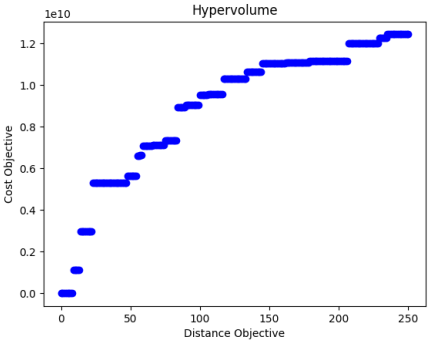

Figure 22: Hypervolume of the multiobjective problem with 50 customers.

	Min Cost		Min Dist	
#Customers	Dist	Cost	Dist	Cost
10	68 010	351	71 700	317
30	182 330	691	323 960	670
50	476 370	1 577	587 620	1 363

Figure 23: Scores of each objective

Conclusions

After completing this project, we can make a few assumptions. First, even if they are designed to be faster than more accurate, like in this project where the population was set to 40 and the value for the generations was set to 250, this **Genetic Algorithm**, one of several Evolutionary algorithms, can still produce excellent results.

Additionally, we discovered that heuristics functions, when designed appropriately, can really aid the algorithm in producing results that are significantly superior to those obtained when the population is generated at random. They also have the potential to speed up the convergence to the best individual.

Regarding the tuning of the **Genetic Algorithm** and the crossover and modifying functions used have a significant impact on the outcomes, and by changing one, all others need to be changed accordingly.

Lastly, for figure 10 and 16, the values of the mean and std, when the number of customers is 10, for the file WHCentral_OrdFile is the same as WHCentral_Ord50 and the file WHCorner_OrdFile is the same as WHCorner_Ord50, because in this cases, the salesman doesn't need to return to the warehouse, since he is always carrying less than 1000 orders.