TÉCNICO LISBOA

Applied Computational Intelligence

MEEC

Second Project Report

Authors:

Diogo José Pereira Araújo (93906) diogoparaujo@tecnico.ulisboa.pt
Diogo Rafael Esteves Ribeiro (102277) diogo.esteves.ribeiro@tecnico.ulisboa.pt

Group 2

2022/2023 – First Semester, P1

1 Introduction

This paper aims to solve a variant of the Traveling Salesman Problem (TSP) using Evolutionary Algorithms. First, a Single-Objective Optimization Problem with the goal of minimizing the distance traveled will be solved. Then, the problem will be viewed as a Multi-Objective Optimization one, where there is also a need to minimize the cost of transporting the products.

2 Single-Objective Optimization Problem (SOOP)

2.1 Problem Formulation and EA Set Up

Given that the problem in question is a variant of the TSP problem, it was decided to represent the candidate solutions as an array of size equal to the number of customers. Each of the elements in the array corresponds to the ID of a given customer.

In the table 1 there are illustrated all the parameters and operators defined in the EA implemented. It is important to note that all the operators used in the algorithm are from the deap python library[1].

Given that the salesman can only visit each customer once and only once (hard restriction of the problem), the crossover implemented was the *cxOrdered*.

Taking into account that the individuals are arrays of integers, the mutation operator that made the most sense to implement was the $mutShuffleIndexes$.

Considering that there was a restriction in the number of evaluations, the group decided that the selection operator should be a strong one. Therefore the $selTournament$ was used.

Extensive testing was used to choose the remaining parameters in an attempt to improve the algorithm's performance. They will be analyzed in greater detail in section 2.2.1.

Given its simplicity, the group decided to implement the *eaSimple* function from the deap library. In figure 1 it is possible to see a representation of the EA approach.

#Cust.	#Gen.	#Pop.	Crossover op.	Mutation op. (indqb)	Selection op.	CXPB	MUTPB
10	40	250	Ordered	ShuffleIndexes (0.05)	Tournament ($tournsize = 4$)	0.7	0.7
30	100	100	Ordered	ShuffleIndexes (0.05)	Tournament ($tournsize = 12$)	0.8	0.8
50	250	40	Ordered	ShuffleIndexes (0.05)	Tournament ($tournsize = 22$)	0.8	0.8

Table 1: Parameters specified in the SOOP for each study case.

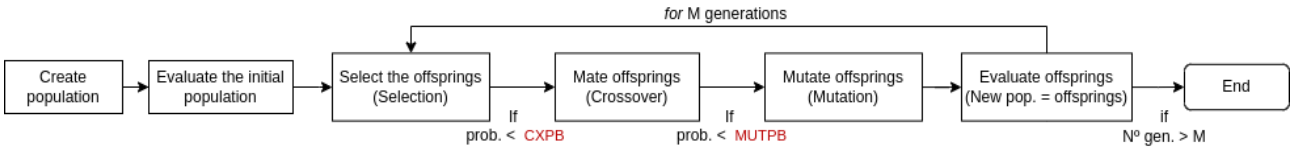


Figure 1: Brief graphical description of the EA approach for the SOOP.

2.2 Solving the Optimization Problem

The problem to be solve is to minimize the distance traveled. The best solution will be the one that minimizes the sum of the distances between each customer i and $i+1$. If the number of orders of the $i+1$ customer exceeded the actual capacity of the truck, the salesman has to take into account that he needs to go from customer i to the warehouse, fill the truck ($capacity = 1000$), and then go from the warehouse to the $i+1$ customer. It is also mandatory to add the distance between the warehouse and the first element of the solution, and the distance between warehouse and the last one.

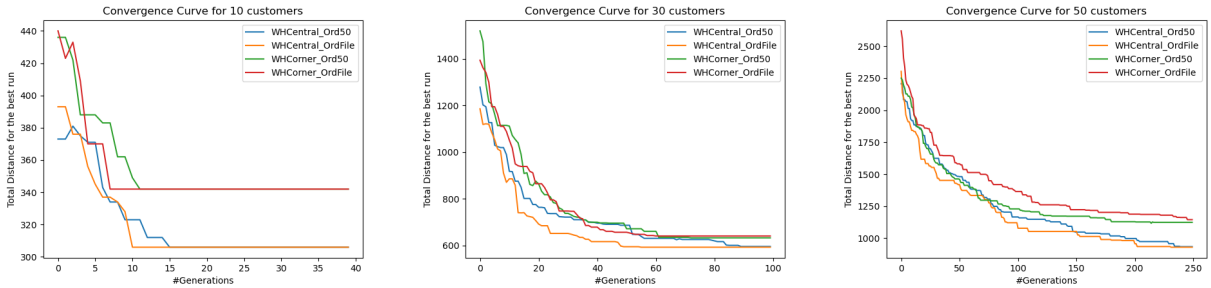
[1] https://deap.readthedocs.io/en/master/

2.2.1 Results

The results obtained are illustrated in the table 2 and figure 2. Given that the search space is $N!$ ($N = \#Cust.$), the group concluded that for larger number of customers it would be more beneficial to increase the number of generations in conjunction with applying a stronger selection (by increasing the size of each tournament). This makes sense, because given that the problem is non-convex, if the search space increases it becomes much harder to converge to the right solution (that is why for a larger number of customers the STD is high).

# Customers	WHCent-File		WHCent-50		WHCorn-File		WHCorn-50	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	309.8	3.37	309.3	4.33	348.9	8.61	349.0	6.98
30	669.8	46.25	667.2	43.97	749.4	50.61	750.4	47.60
50	1126.8	72.12	1108.4	68.09	1252.7	70.44	1202.8	60.48

Table 2: SOOP results with 30 runs (with different random seed) for each case.



(a) (b) (c)

Figure 2: SOOP convergence curve for: (a) 10 customers, (b) 30 customers, (c) 50 customers.

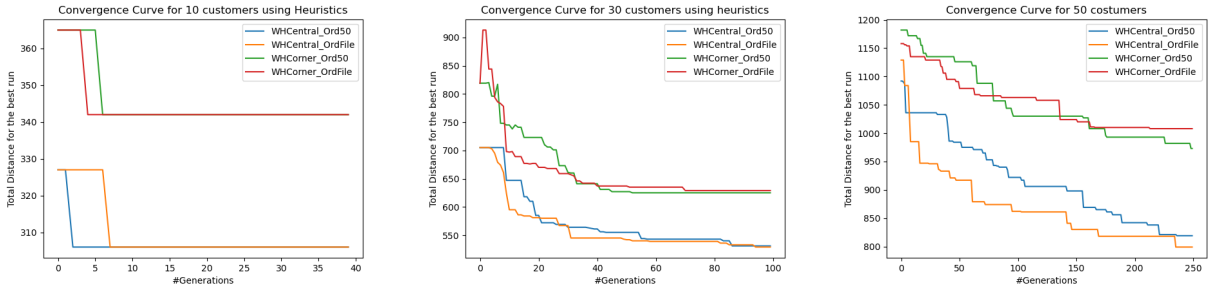
2.3 Using Heuristics

2.3.1 Results

The results obtained are illustrated in the table 3 and figure 3. Comparing the tables 2 and 3, it is clear that inserting a heuristically created solution into our initial population improves the results tremendously. This was expected, given that if a good solution to the problem is inserted in the initial population, the probability of the algorithm to converging to the right solution increases.

# Customers	WHCent-File		WHCent-50		WHCorn-File		WHCorn-50	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	312.9	4.86	311.4	3.93	343.9	4.70	344.4	4.83
30	546.6	15.77	553.6	18.15	681.9	19.77	678.1	28.57
50	868.5	36.37	865.4	30.24	1050.7	19.66	1037.1	26.32

Table 3: SOOP results using heuristics with 30 runs (with different random seed) for each case.



(a) (b) (c)

Figure 3: SOOP convergence curve (using heuristics) for: (a) 10 customers, (b) 30 customers, (c) 50 customers.

3 Multi-Objective Optimization Problem (MOOP)

3.1 Problem Formulation and EA Set Up

As mentioned earlier, now the problem has two objectives to minimize. The new objective is to minimize the cost of transporting the products.

The group followed exactly the same steps as in the SOOP. However, given that this is a new problem, there needed to be some changes. This changes will be explained in this section.

In the table 4 are presented the operators and parameters implemented. Unlike the SOOP, when choosing the parameters and the operators, the group had to take into account that here is a trade-off between better distance or better cost.

The group decided that the mutation operation would be always applied to every individual, i.e., MUTPB= 1.

Given that this is an MOOP, it is important to compute metrics like the crowding distance. In this way, after testing several times, the selection operator chosen was the selNSGA2.

In this case the group did not use the eaSimple(), but decided to implement from scratch using toolboxs from the DEAP library.

#Cust.	#Gen.	#Pop.	Crossover op.	Mutation op.	Selection op.	CXPB
10	39	252	cxOrdered	ShuffleIndexes (0.05)	selNSGA2	0.8
30	357	28	cxOrdered	ShuffleIndexes (0.05)	selNSGA2	0.8
50	250	40	cxOrdered	ShuffleIndexes (0.01)	selNSGA2	0.95

Table 4: Parameters specified in the MOOP for each study case.

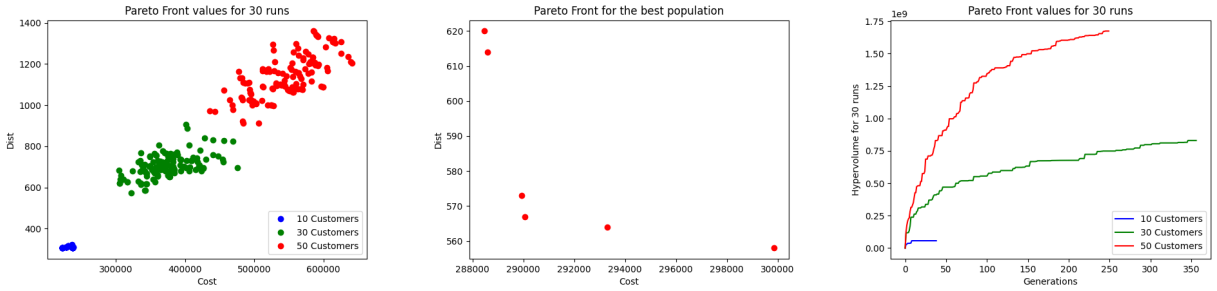
3.2 Solving the Optimization Problem

For this problem, our goal is to minimize both objectives regarding to the constraints presented in 2.2. The new objective (cost of transporting products) was calculated by doing the multiplication between the products in the truck and the travelled distance by the truck.

3.2.1 Results

The results obtained are illustrated in the table 5 and figure 4. The group represented the Pareto values for 30 runs in (a) aiming to understand if the algorithm is robust or not. We can conclude that as the number of clients increases, the more difficult it is for the algorithm to converge to the optimal values, due to the higher variance. In (b) is shown only an example of a

Pareto front for 30 customers and in (c) it is illustrated an example of the hypervolume for each case study (the greater the number of customers the greater the hypervolume and the longer it takes to establish). It is noteworthy that (b) and (c) were obtained by choosing the 'best population' among our candidates. The best population is the one that minimizes the sum of the normalized cost and distance. Regarding the table, we can conclude that is in accordance with the figures shown.



(a) (b) (c)

Figure 4: MOOP: (a) Set of Pareto values over 30 runs for all cases (#Customers = 10, 30, 50.), (b) Pareto front for 30 customers, (c) Hypervolume curves for all the study cases.

#Customers	Min Cost		Min Dist	
	Dist	Cost	Dist	Cost
10	306.0	222120.0	306.0	222120.0
30	620.0	288450.0	558.0	299870.0
50	1021.0	438780.0	928.0	445130.0

Table 5: MOOP results with 30 runs (with different random seed) for each case.

4 Some extra conclusions

In SOOP, the group concluded that there was not much difference between the case where each customer had 50 orders, or each one had its specific number of orders. The results were practically the same, due to the fact that in both situations the truck had to go to the warehouse the same number of times (e.g. for 30 customers, in both cases, the truck only needed to go to the warehouse once).

In both table 2 and 3 it is obvious that the minimum distance in the situations where the warehouse is in the corner are higher than those were the corner is in the center. This can be justified by fact that the warehouse in the corner has an optimal solution higher than when the warehouse is in the center.

During the development of the project, the group noticed that with higher values of CXPB and MUTPB, the mean of the thirty runs would improve, however the best solution would converge slowly. Due to the fact that there are more mutations and crossovers, in the early generations some solutions might oscillate, not converging as fast.