Applied Computational Intelligence

1st Semester 2022/23

Project 2: Evolutionary algorithms for Single and Multi-Objective Optimization

Authors:

Duarte Venâncio Leão Ribeiro da Silva, Nº 93243
Gonçalo Da Silva Dias Moura de Mesquita , Nº 94196

November 4, 2022

Contents

1 Introduction

In this project our main objective is to use evolutionary algorithms to solve a derivation of the Traveling Salesman Problem. The first part is relative to a single objective optimization problem, regarding the traveled distance with a limitation of 1000 orders. Whereas in the second part, we solve a multi objective optimization problem, taking into account the traveled distance with a limitation of 1000 orders and the cost of transporting the products.

2 Single Objective Optimization Problem

2.1 Algorithm

In this section we developed an evolutionary algorithm to solve the single objective problem for four cases. We can observe in figure 1 a simplified structure of our algorithm.

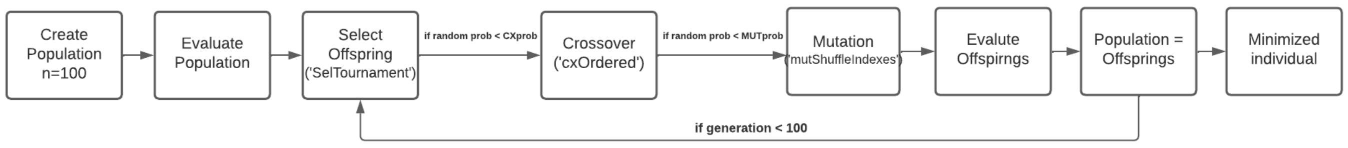


Figure 1: Algorithm structure of SOOP

The evolutionary operators that were used for mutation, crossover and selection were, respectively, *mutShuffleIndexes*, *cxOrdered*, *selTournament* from the library *DEAP*. In order to get better results we changed the probabilities of mutation and crossover to adjust better for each case. Due to the different number of individuals in each case, we also adapt the number of individuals the tournaments for each case. We developed an evaluate function that returns the total distance traveled by the truck. Taking into account the 1000 limit order, every time the number of orders in the truck is less than the order of the next client, the truck goes back to the warehouse.

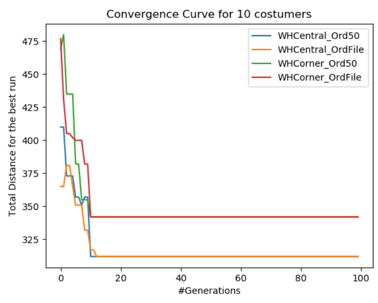
We also, for comparison, developed an heuristic that would give one candidate solution to be included in the first population. Our heuristic would get all the cities that were located in the left and in the right side of the map. The cities that were located in the left side were sorted by lower to higher value of the vertical axis. The cities that were located in the right side were sorted by higher to lower value of the vertical axis.

2.2 Results

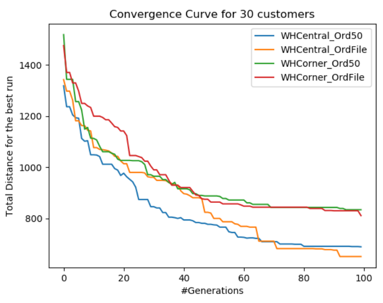
In Table 1 we can observe the results obtained for 10, 30 and 50 customers for different locations of the warehouse and different number of orders. In figure 2 we represent the convergence curve for each one of those cases.

	WHCentral-OrdFile		WHCentral-Ord50		WHCorner-OrdFile		WHCorner-Ord50	
#Customers	std	mean	std	mean	std	mean	std	mean
10	8.48	313.03	8.421	315.93	17.74	355.3	18.043	357.4
30	49.12	713.83	50.42	699.53	43.87	788.03	49.19	762.01
50	63.45	1196.66	66.01	1134.43	61.30	1366.46	80.49	1311.93

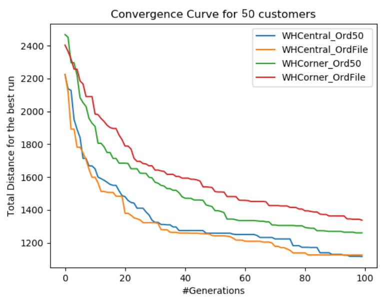
Table 1: SOOP results with 30 runs



(a)



(b)



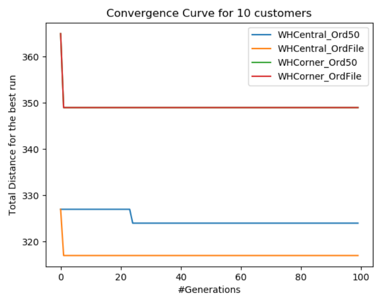
(c)

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

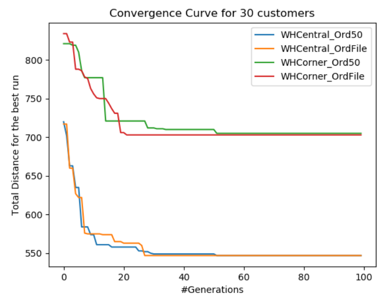
In Table 2 we represent the results that were obtain by the heuristic for all the cases mention above. In figure 3 we represent the convergence curve for each one of those cases.

	WHCentral-OrdFile		WHCentral-Ord50		WHCorner-OrdFile		WHCorner-Ord50	
#Customers	std	mean	std	mean	std	mean	std	mean
10	3.41	324.71	3.51	324.73	9.31	359.0	8.90	357.9
30	16.70	564.63	15.45	563.9	13.55	714.43	18.26	707.6
50	31.40	920.56	35.70	907.53	40.84	1121.16	33.14	1126.1

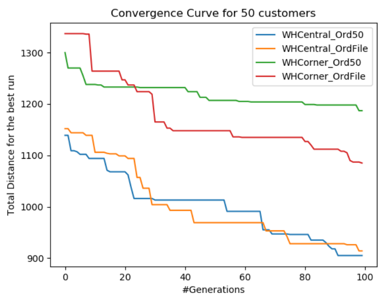
Table 2: SOOP results with 30 runs and heuristic



(a)



(b)



(c)

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

In conclusion, the results were good without the use of the heuristic, however the heuristic really improved the results for a higher number of customers, especially when the warehouse is located in the center of the map. In the majority of the cases, when the costumers orders are fixed the algorithm converges to a better minimum. The opposite happens when the warehouse is located in the corner, the minimum gets worse due to being further way from the customers than when it is located in the center. We also notice that when the number of customers grows the convergence time of the algorithm also grows. To be able to improve even more the results we increase the number individuals per tournament as the number of customers increases, because when the number of customers grows the number of possible combinations also grows.

3 Multi Objective Optimization Problem

3.1 Algorithm

In this section we developed an evolutionary algorithm to solve the multi objective optimization problem for three cases. In figure 4 we can observe a simplified structure of the algorithm developed for this problem.

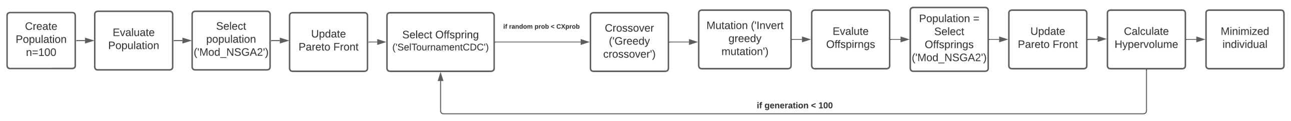


Figure 4: Algorithm structure of MOOP

The evolutionary operators that were used for crossover, mutation and selection were, respectively, *Greedy_crossover*, *Greedy_invert_mutation*, *Mod_NSGA2* and *selTournamentCDC*. The first three operators were either implemented from scratch or built upon already existing functions from the library *DEAP*, whereas the last one is the original function from the *DEAP* library. Besides the *selTournamentCDC* operator, which was chosen in in order to get a more diversified offspring, the operators were developed to adapt to the problem in question in order to yield better results. A more thorough description of what each operator does is presented in the comments of the script *MOOP.py*. Lastly the evaluate function was built upon the one from the SOOP where now it also returns the total cost of traveling with the orders, taking into account the constraints and the way to calculate the traveling cost given the project outline.

3.2 Results

The results obtained with the algorithm presented in the previous subsection are shown in table 3 and figures 5 and 6, where all cases had 100 generations with a population of 100. Here the Pareto curve is relative to the Pareto fronts obtained for each of the runs across the 3 cases. Whereas the hypervolume evolution curve is relative to a fixed seed for the 3 cases.

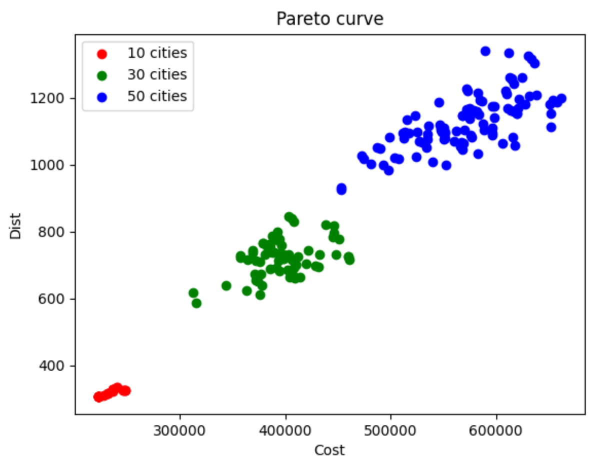


Figure 5: Pareto curves across 30 runs

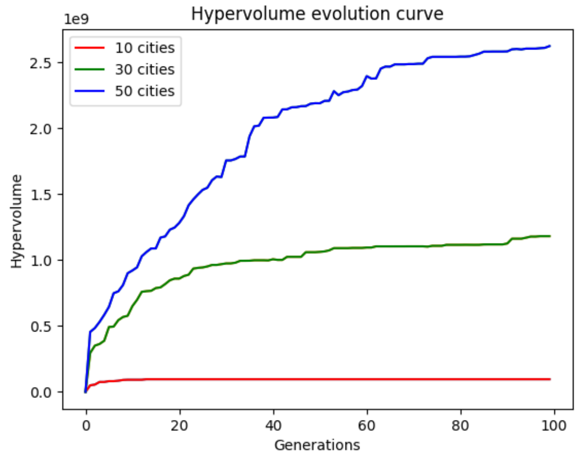


Figure 6: Hypervolume evolution of a random run

As it can be verified in figure 5 when the numbers of costumers grows the Pareto solutions become more and more sensitive to the initial population which makes it harder to converge to the same Pareto front. This happens because, since the search space is given by $n!$, as the number of costumers grows the number of possible candidate solutions of the first population has bigger a probability to the diverge from the population initialized with a different random seed. And since the problem is not convex when the search space grows the harder it is to converge to the same solutions with different initial populations. Therefore, it results in more dispersed Pareto front curves.

As for the hypervolume curves, figure 6 shows that as the number of costumers grows the more generations it needs to converge to a Pareto curve. This occurs for similar reasons as the ones above, since more costumers imply more possible solutions, the more diverse each population can become making it harder to converge to a Pareto front in the same number of generations.

	Min Cost		Min Dist	
#Customers	Dist	Cost	Dist	Cost
10	306	222120	306	222120
30	617	312820	588	314990
50	931	452860	927	453020

Table 3: Multi objective results with 30 runs

Table 3 shows the best individuals across all 30 Pareto fronts, when considering the minimization of each objective.

In conclusion, when comparing the results shown in table 3 and figure 5 we can see that as the number of costumers gets bigger the best individuals deviate more from the average of all the Pareto solutions from each case, showing once again the sensitivity to the initial population and the variance growth of the Pareto solutions.