EcoPioneer

VRP Project Report

COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

Repository

Flavia Motta, 20230574
Flavio Magalhães, 20230571
Mafalda Paço, 20220619

1. Project definition

The problem we are solving is a classic *Vehicle Routing Problem* (VRP) with multiple vehicles and a depot where they depart and return to. The locations must not be visited more than once. The goal of this project is to optimize the vehicle routes of the vehicles in the fleet, to visit every location, starting and ending the route at the depot, while minimizing the overall distance travelled and the amount of vehicles used. The choice of minimizing the number of cars used is based on a sustainability concern. This project is based on data from Google developers [6].

The **fitness function** we'll be optimizing, through minimization, is the total distance travelled. The **search space** consists of all the combinations of locations visited by vehicle. You can find the project repository here.

2. Implementation

2.1 Representation

The individual is represented by a list of lists – each inner list represents the route of a vehicle. For a 4-vehicle problem our individual will be a list of up to 4 routes. The amount of vehicles allocated for the day is chosen randomly between 1 and however many vehicles the fleet has – in our case 4. A route is a list of locations, in order of visit. The depot is omitted.

[[16, 1, 3], [8, 5, 11, 7, 14, 2], [12, 9, 15, 6], [ ]]

2.2 Fitness Function

Our fitness function returns the total distance of the route. To do this we sum the distance from the depot to the first location, the first location to the second, and so on until the last location back to the depot. There is a penalty added for each vehicle after the first, in order to incentivize using less vehicles and create a greener fleet, and for each location not visited.

2.3 Evolution

We adapted the *evolve* function to better suit our needs. We altered the way we applied **elitism** - elitism tends to improve the fitness of the generations, by saving the best value and, therefore, guaranteeing that the fitness will never decrease through the generations. With the goal of avoiding local optimums, we altered our elitism implementation to save not only the best individual, but an *x* number of elites. The intuition behind this is that while the first elite might be the best at a certain stage of the evolution, it might lock the development to a local optimum. With more options, we should get a higher chance at the best fitness available.

With this goal in mind, we also implemented a **plateau tolerance**, that will change our parameters after *n* generations without improvement. This technique was brought to our attention by a student who farms and used this grafting technique on his plants. By using biomimicry, and taking inspiration from grafting, we are able to get faster evolutions. Grafting combines two plants to get the characteristics of both. When this logic is applied to our problem what we get is once the plateau threshold is reached, the values on our crossover and mutation rate are altered. The goal is to not waste generations on still evolution. The mutation rate is multiplied by 1.1, and the crossover is multiplied by 0.15. Although we ideally want the mutation probability low, and the crossover probability high, that's how they start, so this is the alteration we're able to do.

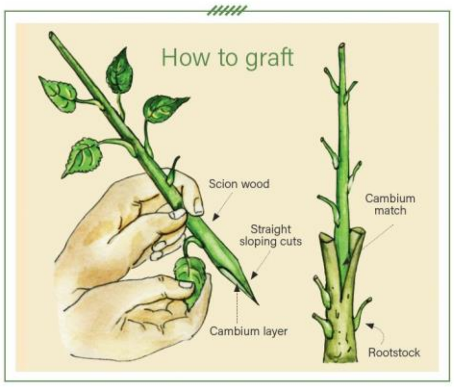


Figure 1 : Grafting tutorial

2.4 Selection

Selection is the choice of individuals to which apply the genetic operators, i.e. selecting the individuals that will generate offspring for the next generation. It needs to strike a balance – too-strong and the highly fit individuals will take over, reducing diversity; too-weak and the evolution will be too slow. In class we implemented the Fitness-Proportionate Selection and the Tournament Selection, so for this project we implemented the Sigma Scaling and the Rank Selection – inspired by [3] Mitchell (1996).
Sigma Scaling keeps the selection pressure somewhat constant throughout the evolution process. An individual's expected value is a function of its fitness, and the population mean and standard deviation. It gives individuals whose fitness is significantly different higher weights - it'll give an individual with fitness one standard deviation above the average 1.5 expected offspring. In order to maintain diversity, it doesn't eliminate individuals, by avoiding attributing expected values of zero. We selected the individual based on cumulative probability, meaning that the individuals are chosen with a probability that is proportional to their scaled fitness.
Rank Selection's purpose is to prevent a quick convergence. The individuals are ranked according to their fitness, and their expected value depends on their rank rather than on absolute fitness. This reduces the selection pressure when variance is high, as this method is insensitive to differences in fitness – only the position in the rank matters. For the rank we calculated it as an arithmetic series, and then selected the individual based on cumulative probability.

2.5 Crossover

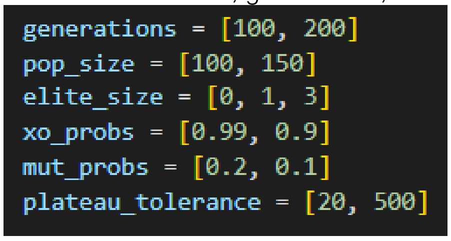
Crossover is the recombination of two individuals to generate new offspring. It aims to combine the good characteristics of the parents to create a new generation with improved potential. In this project, we implemented three different crossover functions: Order Crossover, Single Point Crossover, and Geometric Crossover. Order Crossover (OX) preserves the order of elements from the parents. Two cut positions are selected, and the subsequence between these positions is copied from parent 1 to the offspring. The remaining elements are filled from parent 2 in the order they appear, excluding duplicates. This ensures that the relative order of elements is maintained, which is particularly useful for routing problems. Single Point Crossover involves selecting a single cut point randomly in the parents. The initial segment from parent 1 is combined with the final segment from parent 2 to form the offspring. This simple yet effective method ensures that each offspring inherits a mix of genes from both parents, promoting genetic diversity. Geometric Crossover takes a geometric approach to combine parents' genes. It focuses on combining the parents' characteristics in a way that geometrically represents a point within the parents' hyperplane. This method aims to create offspring that are a balanced mix of their parents, promoting diversity while maintaining good traits from both. Each crossover method has its unique advantages and is chosen based on the problem's specific requirements, ensuring a robust and diverse population throughout the evolutionary process.

2.6 Mutation

The mutation is responsible for introducing random changes to an individual's genes, thereby maintaining genetic diversity within the population and preventing premature convergence. This project applies three distinct mutation functions: Swap, Inversion, and Shuffle Mutation. Swap Mutation involves randomly selecting two genes in an individual and swapping their positions. This straightforward technique helps explore new areas of the solution space while preserving the individual's overall structure. Inversion Mutation, on the other hand, selects a random segment of genes and reverses their order. This method can significantly alter the sequence of genes, potentially leading to more varied and improved solutions. Finally, Shuffle Mutation involves selecting a random subset of genes and shuffling their order, promoting diversity without drastically altering the individual's composition. Each mutation method offers unique benefits, and its application is tailored to the specific needs of the problem, ensuring a comprehensive exploration of the solution space and fostering robust evolutionary progress.

3. Tuning

We ran a combination of all the genetic operators with multiple parameters, logging all the results for analysis. Every combination was ran 5 times, in order to have some statistical validity. We started with the advised values for the parameters: low values for the mutation probability and high values for the crossover probability, different values for the elitism, generations, tolerance and population size.



```
generations = [100, 200]
pop_size = [100, 150]
elite_size = [0, 1, 3]
xo_probs = [0.99, 0.9]
mut_probs = [0.2, 0.1]
plateau_tolerance = [20, 500]
```

Figure 2 : Parameters used on the tests

The results show that our grafting attempt was not successful. We experimented with a plateau tolerance of 20 generations, and of 500 which is equivalent to not using it, as this threshold wouldn't be reached with the sizes of our generations. As we can see in the table below, although the average fitness is similar, it is lower when grafting is not implemented. Elitism had a significant impact on our fitness, with the best results coming from one elite. Implementing elitism improved by almost 96% our fitness.

Tolerance	Average Fitness
20	75912.02
500	75603.53

Table 1: Grafting's impact on fitness

Elitism	Average Fitness
0	209387.95
1	8513.37
3	8532.03

Table 2: Elitism's impact on fitness

While the mutation probability results aligned with our theoretical expectations, the crossover probability's impact on fitness wasn't what we were expecting – the lower value resulted in better fitness.

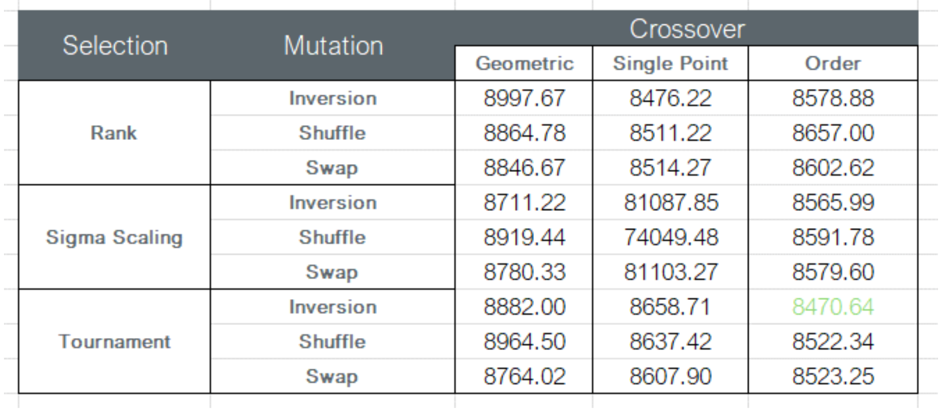
Xo Probability	Average Fitness
0.8	8890.21
0.9	23529.57
0.99	26788.11

Table 3: Crossover's impact on fitness

Mut Probability	Average Fitness
0.05	8877.22
0.1	23737.65
0.2	26649.13

Table 4: Mutation's impact on fitness

The best average fitness resulted from the combination of Tournament selection, Inversion mutation and Order crossover. You can find the results of the combinations on the table and figure below.



Selection	Mutation	Crossover		
		Geometric	Single Point	Order
Rank	Inversion	8997.67	8476.22	8578.88
	Shuffle	8864.78	8511.22	8657.00
	Swap	8846.67	8514.27	8602.62
Sigma Scaling	Inversion	8711.22	81087.85	8565.99
	Shuffle	8919.44	74049.48	8591.78
	Swap	8780.33	81103.27	8579.60
Tournament	Inversion	8882.00	8658.71	8470.64
	Shuffle	8964.50	8637.42	8522.34
	Swap	8764.02	8607.90	8523.25

Table 5: Average fitness distribution over the genetic operators

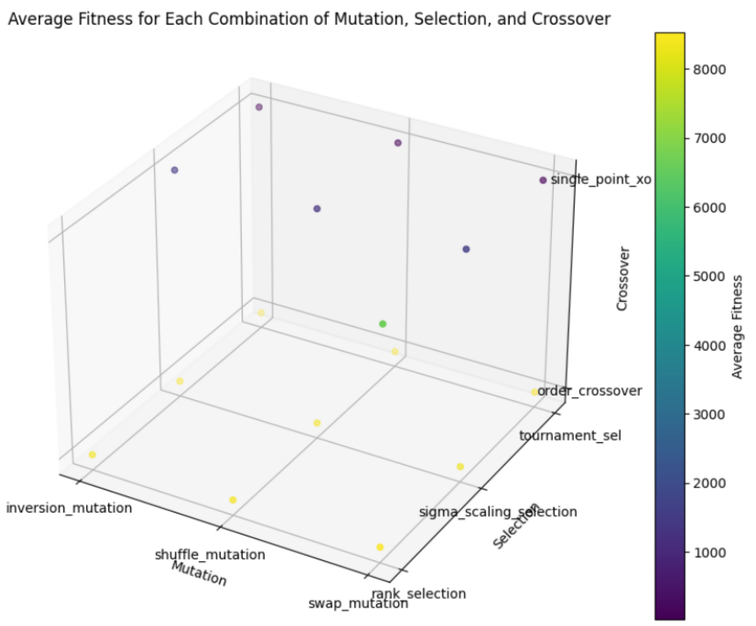


Figure 3: Average fitness distribution over the genetic operators

4. Results

Our best fitness was 7124 and resulted from the following combination:

Selection – Tournament
Crossover – Order
Mutation – Inversion
Xo probability – 0.9
Mut probability – 0.1
Elitism – 3
Plateau tolerance – 500 (no grafting)
Generations – 200
Population size – 150

And created the following individual:

[[15, 12, 11, 4, 1, 13, 5, 9, 14, 16, 6, 10, 2, 8, 7, 3], [], [], []]

5. Conclusion

The project successfully implemented a robust solution within time and algorithm size constraints, leading to increasingly refined algorithms being deployed. The mixed approach, utilizing various selection, crossover, and mutation techniques, enabled the algorithm to explore different evolutionary paths concurrently, thereby maintaining population diversity. This balanced exploration and exploitation, combined with carefully selected crossover methods, resulted in a robust solution to the VRP problem. The results underscore the efficacy of mixed strategies in genetic algorithms, promoting innovation and stability in evolutionary processes.
With more time, we would've liked to experiment more with the green constraint – possibly by including things like fuel consumption on the penalization. Our best individuals were always one vehicle routes, and we'd like to strike a balance between distances and vehicles and have a more complex fitness function.

Bibliography

[1] Costa, P. et al. (2018). A recent review of solution approaches for green vehicle routing problem and its variants. Electronic Notes in Discrete Mathematics 64, 65–74.

[2] Goldberg, D. E., and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. G. Rawlins, Foundations of Genetic Algorithms. Morgan Kaufmann. https://doi.org/10.1016/B978-0-08-050684-5.50008-2

[3] Mitchell, M. (1996). An introduction to genetic algorithms. https://doi.org/10.7551/mitpress/3927.001.0001

[4] Toth, P. and Vigo, D. (2002). The vehicle routing problem. SIAM.

[5] Vanneschi, L., Silva, S. (2023). Particle Swarm Optimization. In: Lectures on Intelligent Systems. Natural Computing Series. Springer, Cham. https://doi.org/10.1007/978-3-031-17922-8_4

[6] Data from: https://developers.google.com/optimization/routing/vrp#create_the_data

Division of labor:
Flavia Motta: 33%
Flavio Magalhães: 33%
Mafalda Paço: 33%