EcoPioneer



[Repository](https://github.com/MafaldaPaco/cifo)

Flavia Motta, 20230574

Flavio Magalhães, 20230571

Mafalda Paço, 20220619

VRP Project Report

**Computational Intelligence for Optimization**

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 while minimizing the overall distance travelled and the amount of vehicles used. Our problem is based on data from [google developers](https://developers.google.com/optimization/routing/vrp#create_the_data).

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](https://github.com/MafaldaPaco/cifo).

1. Implementation
   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.

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

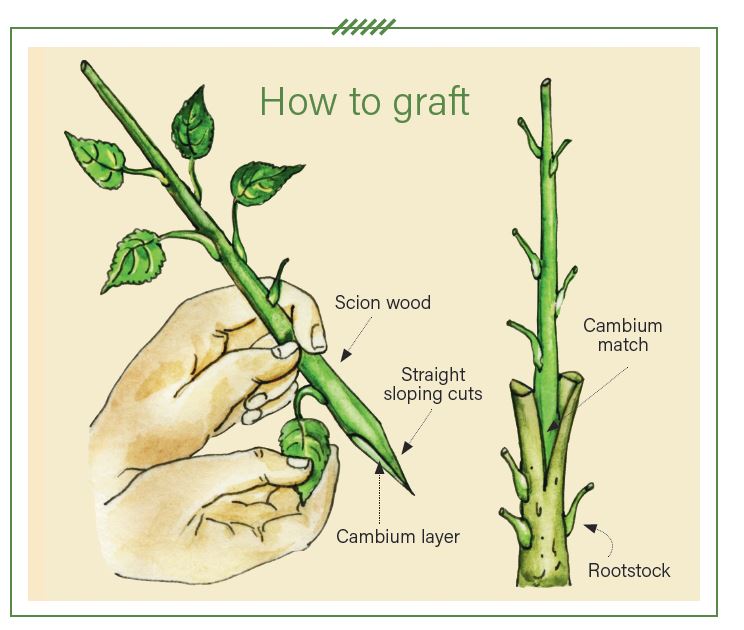
* 1. Fitness Function

There is a penalty added for each vehicle after the first, in order to incentivize using less vehicles to create a greener fleet.

* 1. 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 hope to 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 graftingtechnique on his plants. 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. Explicar se aumentamos ou diminuimos os valores e porquê (testar).



* 1. Selection
  2. Crossover
  3. Mutation

1. Tuning
2. Results
3. Conclusion

Division of labor:

Flavia Motta:

Flavio Magalhães:

Mafalda Paço:

Bibliography

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

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