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, 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](https://developers.google.com/optimization/routing/vrp" \l "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. The depot is omitted.

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

* 1. 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 also a penalty added for each vehicle after the first, in order to incentivize using less vehicles and 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 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 graftingtechnique 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. We don’t waste generations on a still evolution. Explicar se aumentamos ou diminuimos os valores e porquê (testar).

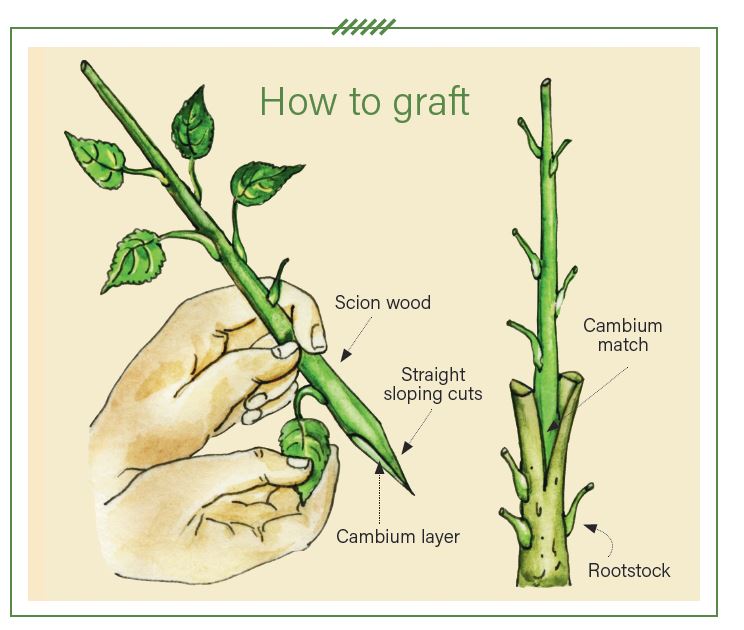


Figure 1 : Grafting tutorial

* 1. 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.

* 1. Crossover
  2. Mutation

1. Tuning
2. Results
3. Conclusion

Division of labor:

Flavia Motta: 33%

Flavio Magalhães: 33%

Mafalda Paço: 33%

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

Data from: https://developers.google.com/optimization/routing/vrp#create\_the\_data