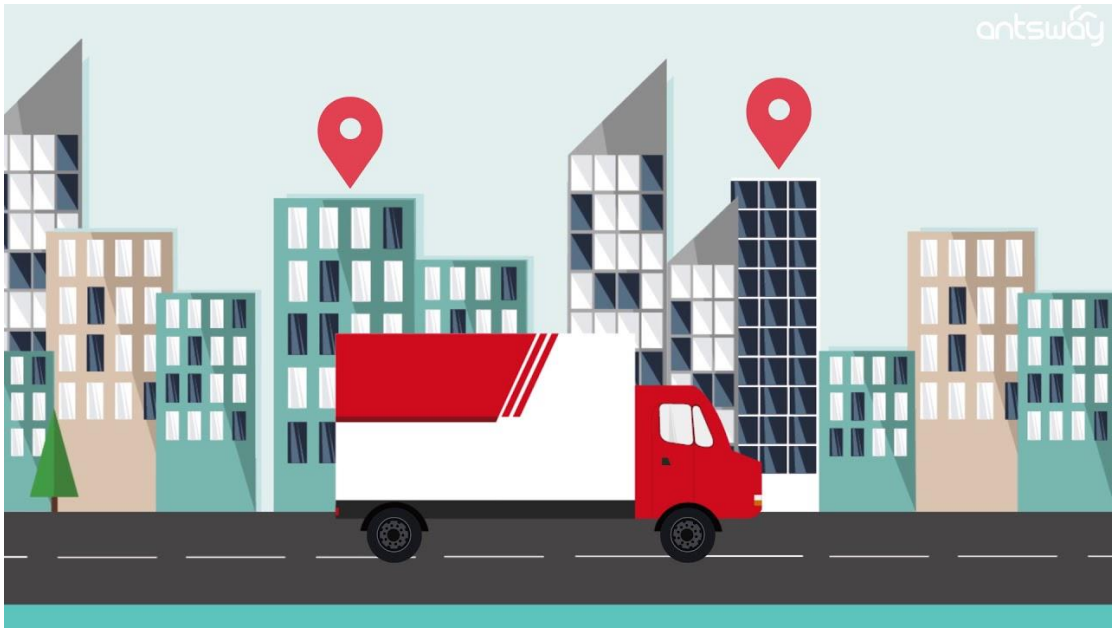

Optimisation des Itinéraires de Livraison en utilisant l'API MapQuest et l'Algorithme Génétique

ETUDE D'ALGORITHME

Youssef Ghaoui | langage :Python3 | Date 26/07/2023

1. Introduction:

The aim of this report is to present an optimization algorithm for delivery routes using the MapQuest API and the Genetic Algorithm. The goal is to efficiently deliver goods to multiple locations while considering travel times and vehicle capacities.

2. MapQuest API:

The **MapQuest API** is a web service that provides route optimization and navigation features. It allows us to obtain travel durations and distances between different locations. We utilize this API to fetch the route durations between the delivery locations, which is a crucial input for our optimization algorithm.

3. Genetic Algorithm:

The Genetic Algorithm (GA) is a powerful optimization technique inspired by the process of natural selection. It mimics the process of evolution to find the best solutions to complex problems. We employ the GA to optimize delivery routes for multiple vehicles with the following steps:

Step 1: Create Data Model

The function `**create_data_model**` creates a data model that includes the list of locations to be visited and the number of available vehicles for delivery. The starting location (Firm) is assigned an index 0 in the locations list.

Step 2: Initialize Population

The function `**initialize_population**` generates an initial population of possible routes. Each route excludes the starting location (Firm) and randomly shuffles the remaining locations. This creates multiple potential delivery sequences.

Step 3: Calculate Fitness

The function `**calculate_fitness**` evaluates the fitness of each route in the population. It computes the total duration required to visit all the locations in a route and return to the starting location (Firm). The fitness value represents the quality of the route in terms of delivery time.

Step 4: Selection

The function `**selection**` performs a selection process to choose the best routes from the population based on their fitness values. It randomly samples a few routes and selects the top-performing routes with the shortest total duration.

Step 5: Crossover

The function `**crossover**` creates new routes (children) by performing crossover operations between two selected routes (parents). It combines segments of both parents to create new, potentially better routes.

Step 6: Mutation

The function `**mutation**` introduces random changes to the routes (genes) of the children generated in the previous step. It swaps two randomly selected locations in a route to promote exploration of different solutions.

Step 7: Generation Update

The algorithm proceeds to create a new population for the next generation by extending the current population with the newly created children.

Step 8: Termination and Best Route Selection

The process of generating new generations continues for a specific number of iterations (generations). At the end of the iterations, the best route with the shortest delivery time is selected as the optimal solution.

Step 9: Route Splitting

If the best route's total duration exceeds a specified **threshold** (e.g., 1 hour) and there are at least two vehicles available, the algorithm splits the route into multiple sub-routes to distribute the deliveries among different vehicles. This ensures that each vehicle operates within the time constraints.

4. Utilization of MapQuest API and Results:

The algorithm utilizes the MapQuest API to fetch the route durations between the delivery locations. These durations are then used to evaluate the fitness of each route and to identify the optimal delivery routes for each vehicle. The routes are presented in a formatted output, showing the order of locations to be visited and the total road time for each vehicle.

5. Conclusion:

In conclusion, the combination of the Genetic Algorithm and the MapQuest API enables us to efficiently optimize delivery routes for multiple vehicles. The algorithm considers travel times and vehicle capacities, ensuring timely and cost-effective deliveries. This optimization approach can be applied in real-world scenarios to enhance delivery logistics and reduce operational costs. Additionally, by adjusting parameters such as the number of generations, population size, and vehicle capacity, the algorithm can be tailored to meet specific business requirements.