Computational Intelligence for Optimization Group Project

Master's degree program in data science and advanced analytics

Vehicle Routing Problem (VRP)

Github Link: https://github.com/joananferreira/CIFO-Project-Group-52.git

Group 52

Arda Kaykusuz | 20230766;

Arif Maharramov | 20230770;

Joana Ferreira | 20230193;

Yasmine Boubezari | 320230775.

Prof. Leonardo Vanneschi | Prof. Berfin Sakallioglu

June 2024

Abstract

This project tackles the Capacitated Vehicle Routing Problem (CVRP) using a Genetic Algorithm (GA). The goal is to optimize delivery routes for a fleet of vehicles with limited capacity. The report details a chromosome representation for solutions, where each gene signifies a customer visit. A fitness function evaluates route efficiency based on travel distance and penalties for exceeding capacity. 3 selection methods (tournament, fitness proportionate, roulette wheel) were compared and Tournament selection yielded the best performance. 4 mutation methods (swap, merge and split, hop, dream team) were investigated and merge and split mutation proved most effective. 3 crossover methods (group-based, eager breeder, twin maker) were analyzed, and group-based crossover achieved the lowest average fitness. A comprehensive grid search identified the best combination of operators: tournament selection, twin maker crossover, and merge and split mutation. This combination achieved rapid convergence to high fitness values. The project highlights the importance of selecting appropriate genetic operators for the problem at hand.

Keywords: Genetic Algorithm (GA), Capacitated Vehicle Routing Problem (CVRP), Route Optimization.

Statement of Contribution:
Arda and Joana confirmed that all the important information was present in the report and ensured that the report respected the page limit.
Arif and Yasmine checked if the code and report adhered to the guidelines provided by the teacher.
Arif and Arda prepared the fitness function.
Joana and Yasmine focused on evaluation of algorithms.
All group members contributed the report and discussed the project details.

Definition of Problem

The Capacitated Vehicle Routing Problem (CVRP) is a well-known optimization challenge in logistics and transportation. It involves finding the most efficient routes for a fleet of vehicles to deliver goods to a set of customers, ensuring that each vehicle's capacity is not exceeded. The objective is to minimize the total distance traveled by all vehicles while satisfying each customer's demand.
In this project, we implemented a solution for CVRP using a Genetic Algorithm (GA), a powerful heuristic for solving combinatorial optimization problems. Our goal was to optimize the routing of vehicles from a central depot to multiple customers and back, ensuring minimal travel costs and adherence to vehicle capacity constraints.

Individual Representation

To represent the individuals (solutions) in our GA, we used a chromosome structure where each gene corresponds to a customer visit. A complete solution (individual) is an ordered list of customer indices, segmented into separate routes for each vehicle. This representation captures the sequence of customer visits for each vehicle, ensuring that vehicle capacity constraints are respected.
Each route starts and ends at the depot, and the total demand of the customers on each route does not exceed the vehicle's capacity. This structure allows for a straightforward implementation of genetic operators tailored to the routing problem.

Fitness Function

To understand the fitness function for the Capacitated Vehicle Routing Problem (CVRP), we need to visualize each route as a sequence of nodes (customers) visited by a vehicle, starting and ending at the depot, where the connections between these nodes represent the distances traveled. The fitness function evaluates the efficiency of the routes by calculating the total distance traveled and penalizing routes that exceed the vehicle's capacity.
The fitness of a route is evaluated by summing travel distances between consecutive customers, including the return trip. Additionally, penalties for exceeding vehicle capacity are applied, ensuring that solutions adhere to the capacity constraint. It can be calculated following this. (**Image 1**)
The fitness function works by first initializing “total distance” and “penalty” to zero. It then iterates through each route in the solution, summing up the distances between customers and the return trip to the depot. It also calculates the total demand on each route. If a route exceeds the vehicle's capacity, a penalty proportional to the excess (1000 times the excess demand) is applied. Finally, the total distance for each route is added to “total distance” and the total penalty is also accumulated.
The solution's fitness is calculated by summing the total distance and penalties for all routes in the solution. This ensures that the fitness value reflects both the efficiency of the routes and adherence to capacity constraints.

Selection

Selection determines which individuals will be chosen for reproduction. To choose the best selection algorithm for the Capacitated Vehicle Routing Problem (CVRP), we compared three methods based on the performance of our Genetic Algorithm (GA). Each offers advantages for optimizing the routes of vehicles to service customers with specific demands while respecting vehicle capacity constraints:

- **Tournament Selection Method** selects the best individual out of a randomly chosen subset of the population to maintain its diversity and guide the search towards high-quality CVRP solutions.
- **Fitness Proportionate Selection (FPS)** selects individuals based on fitness proportion, favoring routes that minimize distance and adhere to capacity constraints, and effectively balancing exploitation and exploration in CVRP.
- **Roulette Wheel Selection** selects individuals based on fitness proportion relative to the population's total fitness, promoting high-quality solutions and accelerating convergence to optimal CVRP routes.

Crossover

Crossover is a vital genetic operator in a genetic algorithm that combines the genetic information of two parents to produce new offspring. In the context of the Capacitated Vehicle Routing Problem (CVRP), we evaluated the performance of our genetic algorithm (GA) using three crossover methods, each offering advantages for finding optimal vehicle routes under capacity constraints:

- **Group-based Crossover** divides parents into groups, exchanging them to create offspring, maintaining feasible solutions with balanced load distribution for CVRP.
- **Eager Breeder Crossover** combines parts of two parents at a random point and promotes diversity and generates varied solutions for efficient CVRP routing.
- By swapping segments of parents, the **Twin Maker Crossover** preserves visit sequences and integrates diverse solutions, aiding in optimizing CVRP routes for distance and capacity.

Mutation

Mutation introduces small random changes to individuals, avoids local optima, and thoroughly explores the solution space, with four methods enhancing the genetic algorithm's CVRP performance:

- The **Swap Mutation Method** randomly swaps two customers' positions in a route, potentially reducing distance while meeting capacity constraints in CVRP.
- **Merge and Split Mutation** combines and splits routes at different points, creates new structures and improves load distribution for balanced and efficient CVRP routes.
- The **Hop Mutation Method** moves a customer within or between routes, redistributes loads and explores routing configurations. It highly improves capacity adherence and minimizes travel distances in CVRP.
- **Dream Team Mutation** forms a new route for a subset of customers and uncovers efficient routing possibilities, enhancing the overall solution's quality and feasibility in CVRP.

Evolution

We explore the evolution process for improving Capacitated Vehicle Routing Problem (CVRP) solutions. Starting with a diverse initial population, parents are selected with tournament selection. Group-based crossover (0.8 probability) combines routes, and dream team mutation (0.2 probability) introduces significant changes. Elitism preserves the best individuals, which guarantees steady improvement. Each generation replaces the population with new offspring and elites and records the best fitness over 100 generations to track progress.

Experimental Results

To determine the optimal hyperparameters for our problem, we conducted a comprehensive grid search of all possible combinations of selection, crossover, and mutation methods. Also, we tested Elitism with both true and false values to understand its effect on the performance of algorithm. This test showed that enabling Elitism improves performance. **(Table 1)**
To further analyze the performance of different genetic operators, we conducted a detailed evaluation of selection, mutation, and crossover methods across multiple instances of the problem. Figures illustrating the comparison of different selection, mutation, and crossover methods reveal several key insights:

Comparison of Selection Methods

When we dive into comparing selection models (**Figure 1**), tournament selection consistently provided robust performance, balancing exploration and exploitation and achieving lower average fitness values more rapidly compared to other methods, demonstrating its effectiveness in guiding the population toward optimal solutions. Roulette Wheel Selection, while effective, showed more variability in fitness values across generations, indicating greater stochastic behavior, which can be beneficial for maintaining diversity but may also lead to less consistent convergence. Lastly, Fitness Proportionate Selection (FPS) performed well but with greater fluctuations compared to tournament selection. The method showed similar behavior to roulette wheel selection, with higher variability, indicating less stability in reaching optimal solutions.

Comparison of Mutation Methods

While comparing the mutation methods, it is seen that (**Figure 2**) Swap Mutation introduced small incremental changes leading to steady fitness improvements, suggesting effectively fine-tunes solutions but may converge slowly. Merge and Split Mutation was particularly effective, likely due to its

ability to create new route structures and improve load distribution. Hop Mutation's flexibility in moving customers between routes contributed to maintaining diversity in the population and demonstrated a balance between exploration and exploitation, leading to consistent improvements in fitness. Dream Team Mutation created highly optimized sub-routes but showed variable performance, with fluctuations in fitness values (**Figure 2**) indicating that, while powerful, it can sometimes disrupt solutions more than improve them.

Comparison of Crossover Methods

As seen in **Figure 3**, these observations highlight the varying effectiveness of different crossover methods in optimizing solutions. Group-based Crossover achieved the best performance among the crossover methods, with the lowest average fitness. Eager Breeder performed moderately, balancing rapid convergence and maintaining diversity, and showed consistent performance across generations. Twin Maker performed better than Eager Breeder but not as well as Group-based Crossover. This method maintained a stable improvement in fitness, demonstrating its reliability in generating effective offspring.

Fitness Evolution for Best Combination

The fitness evolution graph for the best combination (tournament selection, group based crossover, merge and split mutation) **(Figure 4)** shows a rapid convergence to high fitness values, highlighting the effectiveness of this combination. The sharp decline in fitness values during the initial generations demonstrates its efficiency in exploring and exploiting the solution space.

Conclusion

The detailed analysis of each operator's impact on the algorithm's performance underscores the importance of selecting appropriate genetic operators tailored to the problem at hand. When we wanted to analyze the algorithms' performances and find the best solving method for the Capacitated Vehicle Routing Problem (CVRP), in our project, visual comparisons revealed several key insights that tournament selection showed the most stable and rapid convergence toward optimal solutions, merge and split mutation effectively created and balanced new routes, and the combination of tournament selection, group based crossover, and merge and split mutation emerged as the most effective. Thus, while defining the best method for our problem, we utilized an effective genetic algorithm that ensures efficient route planning. In other words, this strategic approach ensures that the GA can efficiently balance exploration and exploitation, leading to superior performance in solving complex optimization problems like the CVRP.

Appendix:

Appendix A: Code Listing

```
def fitness_function(individual, distance_matrix, demands, vehicle_capacity):
    total_distance = 0
    penalty = 0
    for route in individual:
        if len(route) > 1:
            try:
                route_distance = sum(distance_matrix[route[i]-1][route[i+1]-1] for i in
range(len(route) - 1))
                route_distance += distance_matrix[route[-1]-1][route[0]-1]  # Return to
depot
                route_demand = sum(demands[customer-1] for customer in route)
                if route_demand > vehicle_capacity:
                    penalty += 1000 * (route_demand - vehicle_capacity)
                total_distance += route_distance
            except IndexError as e:
                print(f"IndexError: {e}")
                print(f"Route causing error: {route}")
                raise e
    return total_distance + penalty
```

Image 1. Code of Fitness Function

Where:

- ‘individual’ represents a potential solution, consisting of multiple routes.
- ‘distance_matrix’ is a 2D array where the element at [i][j] represents the distance between customer i and customer j.
- ‘demands’ is an array where each element represents the demand of a corresponding customer.
- ‘vehicle_capacity’ is the maximum capacity that a vehicle can carry.

Appendix B : Figure and Table Listings

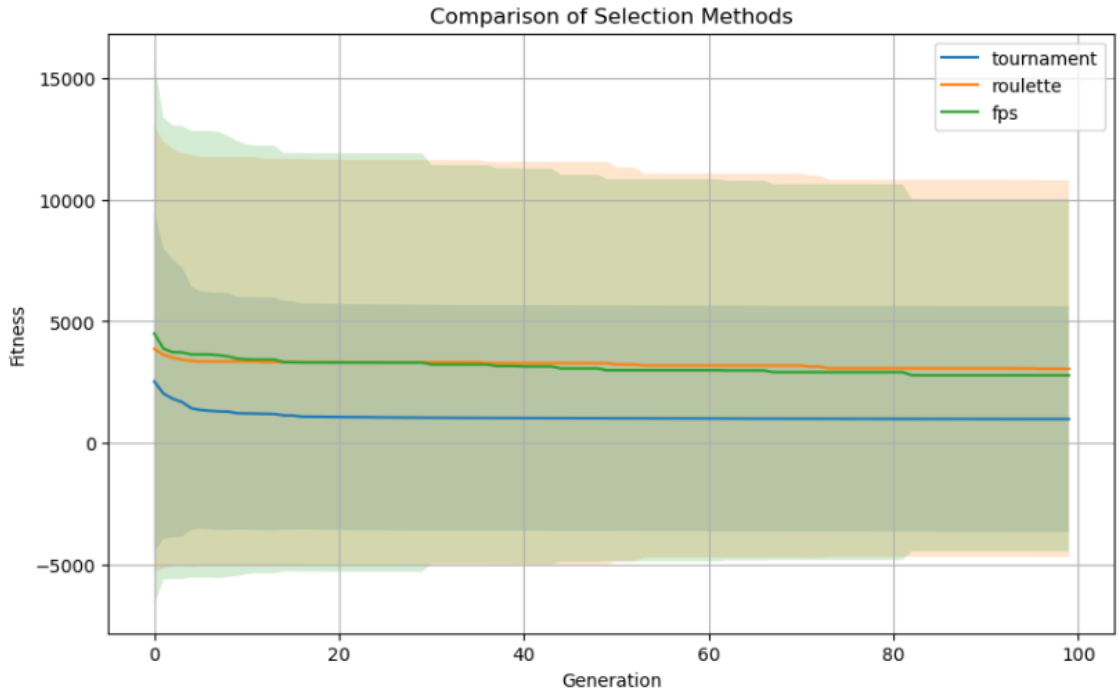

Figure 1. Comparison of Selection Methods

Figure 2. Comparison of Mutation Methods

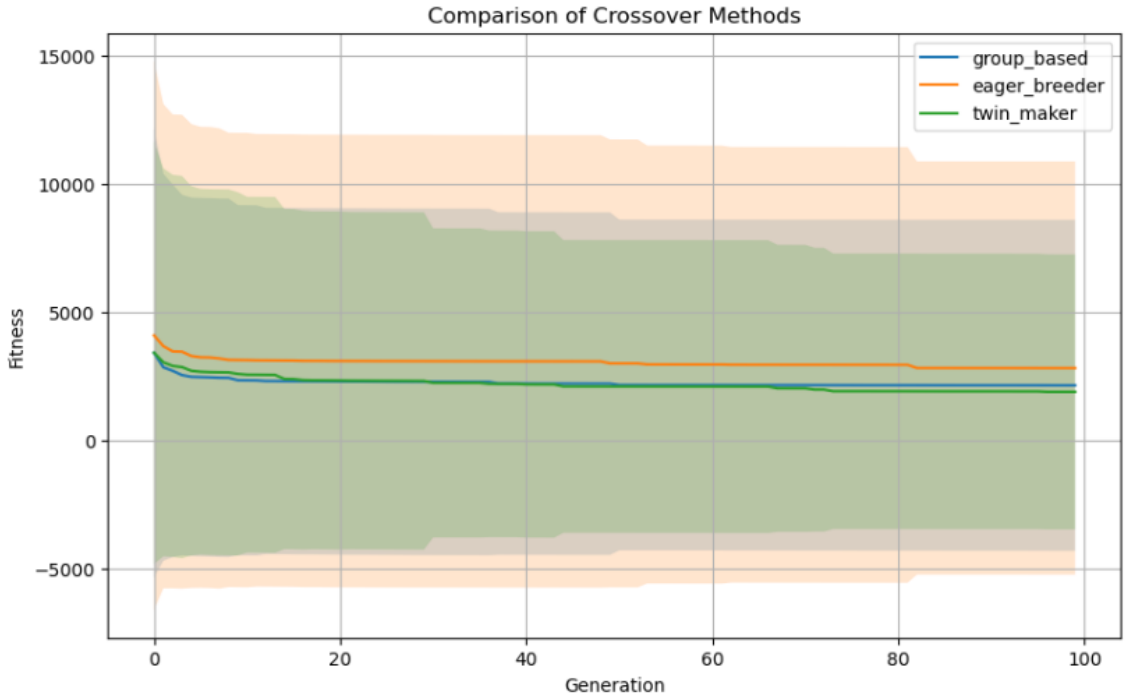

Figure 3. Comparison of Crossover Methods

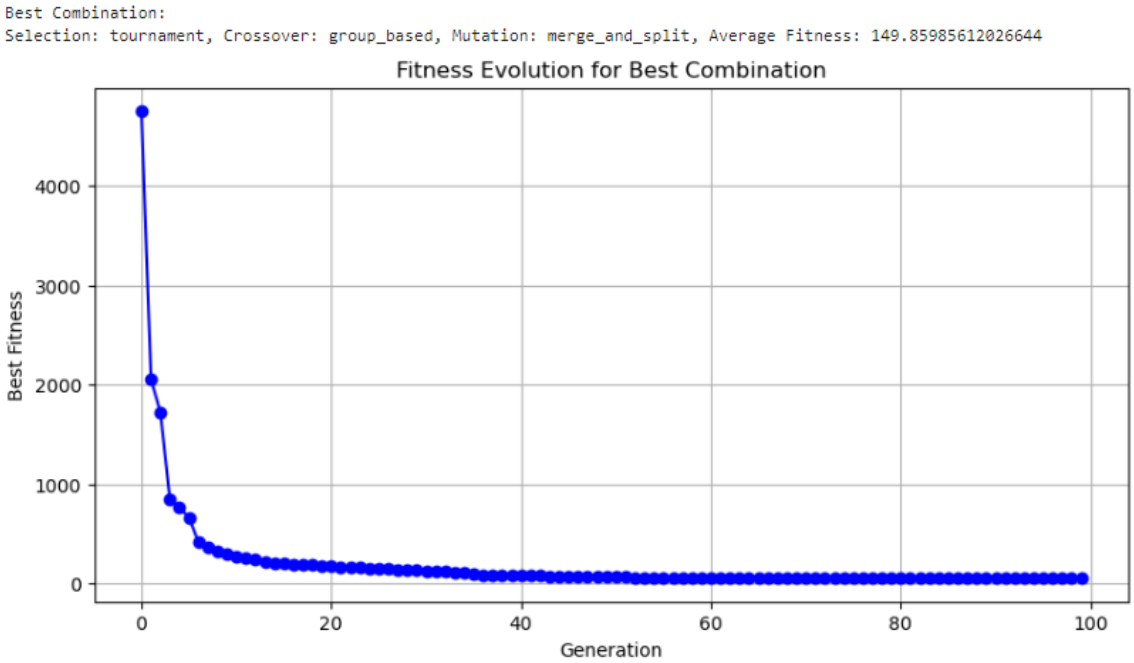

Figure 4. Fitness Evolution for The Best Combination

Elitism	Method		Average Fitness
TRUE	Selection	tournament	1089.239749
TRUE	Selection	roulette	3231.686673
TRUE	Selection	fps	3118.405773
TRUE	Mutation	swap	3578.385814
TRUE	Mutation	merge_and_split	2441.852474
TRUE	Mutation	hop	2555.825324
TRUE	Mutation	dream_team	1343.045982
TRUE	Crossover	group_based	2238.602587
TRUE	Crossover	eager_breeder	3021.637889
TRUE	Crossover	twin_maker	2179.091719
FALSE	Selection	tournament	6410.508461
FALSE	Selection	roulette	760782.6356
FALSE	Selection	fps	747706.8037
FALSE	Mutation	swap	75969.60758
FALSE	Mutation	merge_and_split	1633399.322
FALSE	Mutation	hop	133582.6754
FALSE	Mutation	dream_team	176914.992
FALSE	Crossover	group_based	494936.7431
FALSE	Crossover	eager_breeder	475665.8775
FALSE	Crossover	twin_maker	544297.3271

Table 1. Effect of Elitism on fitness