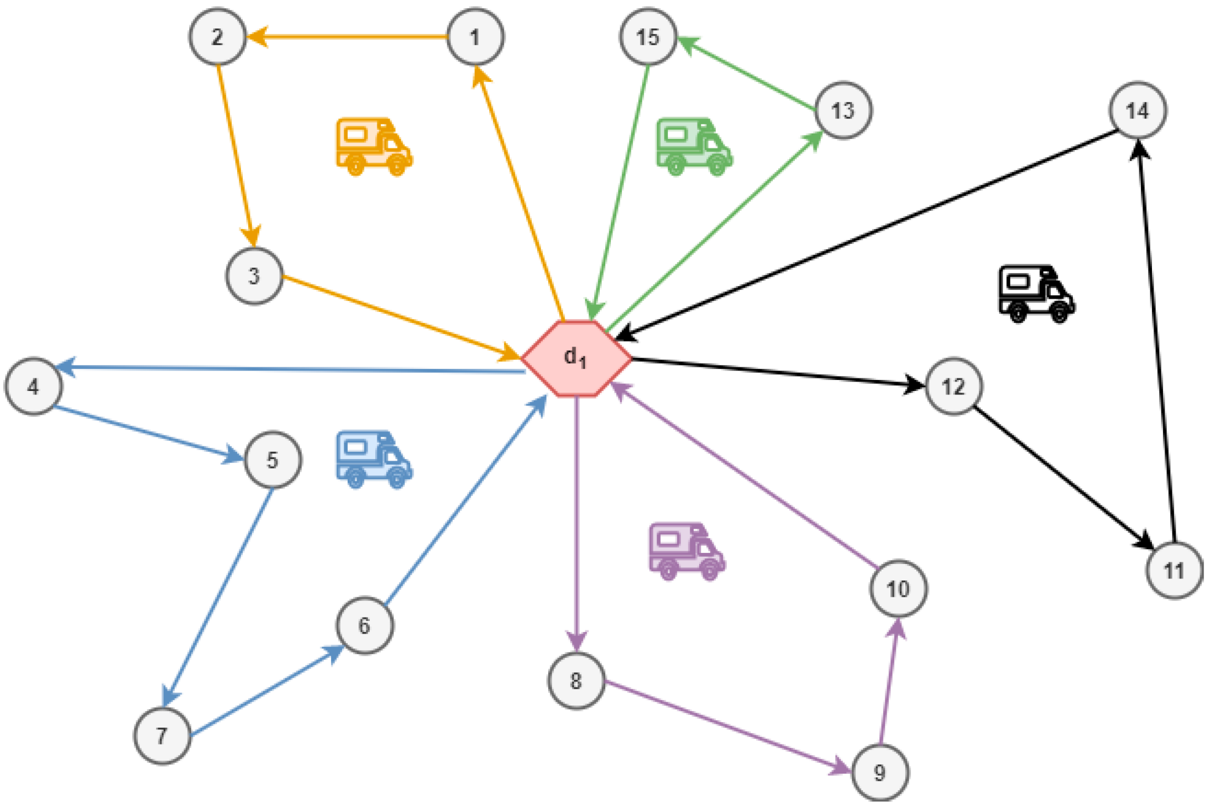
The Vehicle Routing Problem: A Genetic Algorithm approach

Gabriele Morina[e20201804], Marco Lorenzo Piretto[e20201805], and Marius A. Hessenthaler[e20201824]

NOVA Information Management School



Source: [1]

1 Introduction

We want to use Genetics Algorithms to find solutions for the Vehicle Routing Problem (VRP). We will try different selection, crossover and mutation operators, as well as premature convergence prevention to do so and evaluate them.

1.1 Problem

The VRP is an optimization problem that deals with the routing of vehicles to multiple locations. For a fleet of vehicles with a fixed capacity (therefore also called Capacitated VRP) which start at the home location and multiple locations with varying demand must be served. The objective is to find a set of optimal routes that minimize the entire traveled distance. We extend the problem by minimizing both travelled distance and CO2 emissions which are not always equal.

1.2 Data

The dataset we used to test our implementations come from the collection of TSP data offered by University of Heidelberg and can be found at this link [1]. The data consist of 76 coordinate points, each with its relative supply demand, except for the first point that, having 0 demand, is the storage area. Moreover is given the capacity of each truck and the best solution of the problem. Given these data we computed the distance matrix using the euclidean distance in $\mathbb{R}^2$. Additionally, as extension to the original problem, we generated a CO2 matrix, drawing from the exponential distribution with mean equal to the distance between the locations.

2 Representation and Fitness

The most natural solution representation of the VRP is by having multiple routes that each start and end at the home location and cover a number of demand locations. However, for applying Genetics Algorithms as well as mutation and crossover operators the solutions need to be in a gene string format with fixed length. Therefore we use the same representation as in the Traveling Sales Person Problem (TSP), i.e. an ordered string of all visited locations excluding the home location.
Based on the gene representation we split the route into feasible sub-routes by starting a new route each time the vehicle capacity is fully utilized. The fitness is then calculated by summing up the distances for all routes including the distances to the home depot of each route. This approach is also know as the "Route first – Cluster second" heuristic which solves a TSP first and then splits the TSP route into sub routes. This representation and route splitting may not always yield optimal results because given the same string there could be partitions that attain a lower cost. However, filling routes to its maximum works towards the goal of minimal traveled distance [5] and the selection function will still prefer individual that perform well under the VRP fitness function.

3 Crossover

Additional to the Cycle and Partially Mapped Crossover we implemented the following crossover operators. In particular we implemented two crossovers based on the order of the parent's genes (Ordered and Uniform Order Crossover), two operators based on the edges of the parents (Edge Recombination and Alternatin Edges Crossovers) and one hybrid operator (Heuristic Greedy Crossover).

Ordered Crossover creates two random cut points and copies to the offspring the part of the parents located between those cut points. Then it constructs the remaining positions by following the order in the second parent starting after the second cut point and wrapping around if the end of the list is reached.[3]

[1] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/vrp/eilc76.vrp

Uniform Order Crossover starts creating a random binary string of the same length of the parents. Then the first offspring preserves the nodes from the first parent where the binary string contains "1", while the second offspring preserves the nodes from the second parent where the binary string contains "0". Finally both the offspring are filled following the order of the parent from which they have not yet inherited.[6]

Edge Recombination Crossover (ERX) first of all calculates an adjacency matrix which stores the neighbors of each node in any parent so that every city is associated with an edgelist.The procedure then starts from one of the two initial cities of the parents and proceed in steps. In each step it appends to the offspring the city in the edgelist of the last added node which has the fewest entries in its own edgelist.[7]

Alternating Edges Crossover (AEX) starts by adding the first city of a parent to the offspring. Then the offspring is formed by choosing in alternation from the first and from the second parent the node that follows the current one. If the next city is already in the offspring, a random city not already visited is chosen.[2]

Heuristic Greedy Crossover (HGreX) starts by adding the first city of a parent to the offspring. It then always adds, if possible, the cheaper city (determined by the distance matrix) that follows the current node in the parents. If both the next candidate cities are already in the offspring, it adds a random city not already visited.[2]

4 Mutation

Additional to the Swap and Inversion Mutation we implemented the following additional mutation operators. We decided to implement a variety of operators including the two swapping operators Swap Sequence Mutation and Partial Shuffle Mutation, two inversion operators Centre Inverse and Throas Mutation as well as one hybrid operator Cheapest Insertion Mutation.

Swap Sequence Mutation randomly selects two non-overlapping, equal length sequences and swaps them.

Partial Shuffle Mutation swaps each gene with a probability of β with a randomly selected gene [4].

Centre Inverse Mutation splits the route into two sections at a randomly selected point. Each of the sub-routes is replaced by its inverted sub-route [4].

Throas Mutation randomly selects a contiguous section of three genes. "Then, the last becomes the first of the sequence, the second becomes last and the first becomes the second in the sequence" [4].

Cheapest Insertion Mutation (CIM) is a Hybrid Mutation Operator that does partially random mutation while also trying to optimizing the fitness greedily. First, a gene is randomly selected and removed from its position. The gene is then inserted at the position where it results in the smallest detour. The detour is calculated by subtracting the previous tour length from the new tour length which can be reduced to the tour of the locations that are adjacent to the candidate location.

5 Selection

Additionally to the already implemented selection functions ranking (rank), tournament (tour.), and fitness proportionate selection (FPS) for maximization, we implemented fps for minimization and a multi-objective (dual-objective in our case) optimization.

5.1 Fitness Proportionate Selection

In the literature there's no accordance about the correct way to implement fitness proportionate selection when the instance is a minimization problem. One common solution is to select the maximum value of fitness in the population and compute a new fitness vector as subtraction from the maximum. The drawback of this solution is that the worst individual (the one with highest of the original fitness) would be chosen with probability zero. To avoid this we decided to add a constant to the modified fitness vector so that the probability of choosing the worst individual in the population is equal as it would be in a maximization instance with the same fitness.

5.2 Multi-Objective Selection

Additional to the minimization of the distance of the route, there might be cases to also optimize both the distance and CO2 emission which are partially correlated but might also be opposed. We use the multi-objective selection algorithm from the lecture. It identifies non-dominated (pareto-efficient) individuals, assigns them increasing flags and removes them from the set of considered individuals. For the selection we inverse the flag so that the first non-dominated (i.e. the best) individuals receive the highest flat and the last non-dominated (i.e. the worst) individuals receive the lowest flag. We then assign the selection probabilities proportionate to the flag value.

6 Premature Convergence Prevention (prem)

A common issue in genetic algorithms is premature convergence. To tackle this down during the evolution of the population we keep track of the variance of the fitness vector and, optionally, it can be chosen to reshuffle the population when the variance is too low (less than 5% of the initial variance, when the individuals were completely random). The reshuffling consists of saving half of the individuals (including the best) and generating completely randomly the other half. This approach should allow a wider exploration of the search space, reducing the chances of being stuck in local optima.

7 Experiments

For the VRP we want to compare the performance of different crossover, mutation, and selection operators as well as the use of premature convergence prevention.

7.1 Setup

For each of the selection functions (1) rank, (2) tournament and (3) fps we test different combinations of mutation and crossover operators. Because of limited computational resource we only tested four mutation operators (1) swap, (2) inversion, (3) Throas and (4) cheapest insertion and four crossover operators (1) ordered, (2) AEX, (3) ERX and (4) HGreX crossover. This selection was made with the intention to include different functionality as well as complexity types of operators in the experiments. First we set the mutation operator to the simple swap and combined it with the four crossover operators. Second we set the crossover operator to ordered and combined it with the four mutation operators. We then combine the best performing operators again and also try premature convergence on them. For each setup we did 50 runs and then computed the average fitness (over all runs) of the best individual in the final generation. We then compared these values to evaluate which setup performed better. In all experiments we use elitism because saving the best individual will in general increase the final performance and computationally comes at no extra cost because the fitness is calculated for an individual on creation in our implementation.

7.2 Results

Table 1 shows the experimental results. Looking at the results we saw that HGreX and Cheapest Insertion Mutation drastically improve the fitness values. For this reason we decided to further investigate combining them together and see their behavior with different selection functions. Tournament performs better on

Table 1. Experiment results (average best fitness values)

MU /CO	Ordered			AEX			ERX			HGreX		
Selection	FPS	Rank	Tour.	FPS	Rank	Tour.	FPS	Rank	Tour.	FPS	Rank	Tour.
Swap	1739	1316	1097	1853	2016	1023	1805	1933	1127	911	910	964
Inversion	1734	1312	1061							891		1000
Throas	1737	1256	1705							937		1305
CIM	1609	1143	1082							870	888	961
CIM + prem											832	959

average, in particular it is the only one that obtained a good result with AEX and ERX operators. Ranking selection function is the one that performed better when we used the Throas Mutation. While Fitness Proportionate is the selection operator that obtained the worst results on average, in particular in combination with CIM. However, it is the one that obtained the best result when combined with HGreX and CIM. Finally we also did two experiments using the premature convergence prevention which further improved the best results obtained by the hybrid greedy combination.

In figure 1 we can see the convergence rate of three different operator combinations. The combination of the two hybrid greedy operators (with two different combinations of selection and prem) reach best fitness values below 1000 after already 30 to 40 generations and slowly reach final fitness values of between 800 and 900. On the other hand, the non-hybrid mutation and crossover decrease much slower to higher final fitness values.

In figure 2 and 3 we can see all fitness values in a scatter plot. It is once again noticeable that the hybrid greedy combination converges rapidly also for all individuals to a point where it almost only includes individuals with a fitness below 1000. On the other hand with prem we see that every few epoch the population is reshuffled which leads to more diversity and in the end also to slightly better and the best overall results with the best average final fitness of 832. This setup also resulted in the best absolute solution of 800 which comes close to best solution found by the authors of 745.

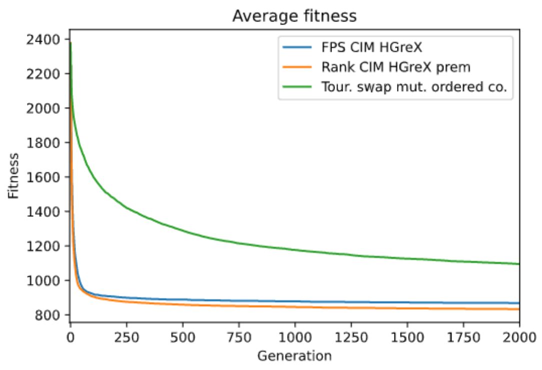


Fig. 1. Average fitness across generations

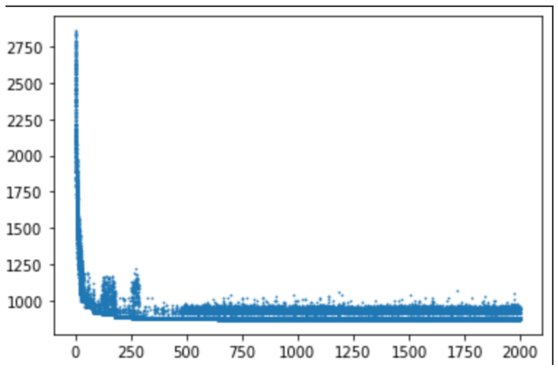


Fig. 2. Scatter Plot of fitness values of all individuals over all generations with Rank, CIM and HGreX.

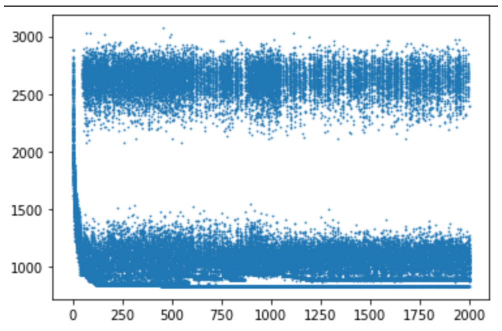


Fig. 3. Scatter Plot of fitness values of all individuals over all generations with Rank, CIM, HGreX and prem.

For the multi objective fitness and selection we briefly tested different combination of operators. The non-hybrid operators resulted in average pareto fitness of around 2200 for the distance and 1700 for the CO2. Compared to the results of only distance optimization they are much worse but it can be attributed to the facts that (1) the dual objective eliminates some good distance solutions because they are too bad in CO2 and (2) with the dual objective good solutions are more difficult to find because the selection is more difficult. We also tested combination of hybrid greedy operators which found much better solutions of up to below 1000 for the distance and around 700 for the CO2. Even though the greedy part only tries to optimize the distance the correlation of distance and CO2 still enables it to find good dual solutions.

8 Conclusion

After the experiments we conclude that Hybrid greedy operators (HGreX and CIM) drastically improve the result especially when combined together. However greedy operators could work less good when applied on different VRP datasets especially when the distance matrix is not calculated by the pure euclidean distance but by restricted road networks. Also after more generations random mutation and crossover might reach similar low fitness values. However hybrid greedy operators have the advantage of faster convergence and therefore less computational complexity. Given the fast convergence of the hybrid operators premature convergence prevention again improves the results by adding more diversity. We also found that for correlated multi objective problems greedy operators that only optimized one objective are still performing well. We found that selection algorithms can have large impact on the result and yield different results with different operator combinations (e.g. AEX and ERX only work well with tournament or CIM does not work well on FPS). In the end we achieved the best result with a combination of Ranking selection, cheapest insertion mutation, heuristic greedy

References

1. Kovács, L., Agárdi, A., Bányai, T.: Fitness landscape analysis and edge weighting-based optimization of vehicle routing problems. Processes **8**(11) (2020). https://doi.org/10.3390/pr8111363, https://www.mdpi.com/2227-9717/8/11/1363
2. Mirosław Kordos, Jan Boryczko, M.B., Golak, S.: Optimization of warehouse operations with genetic algorithms (07 2020)
3. Moscato, P.: On genetic crossover operators for relative order preservation (07 1999)
4. Otman, A., Abouchabaka, J., Tajani, C.: Analyzing the performance of mutation operators to solve the travelling salesman problem. Int. J. Emerg. Sci. **2** (03 2012)
5. Tan, K., Lee, L., Zhu, Q., Ou, K.: Heuristic methods for vehicle routing problem with time windows. Artificial Intelligence in Engineering **15**(3), 281–295 (2001). https://doi.org/https://doi.org/10.1016/S0954-1810(01)00005-X, https://www.sciencedirect.com/science/article/pii/S095418100100005X
6. Vaira, G.: Genetic algorithm for vehicle routing problem (2014)
7. Whitley, D., Starkweather, T., Shaner, D.: The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination (08 2002)