Universiteit Leiden

Master Computer Science

ACO-NSGAII: A Novel Metaheuristics for Bi-Objective Electric Vehicle Routing Problems

Name:	Suzan Al-Nassar
Student ID:	s2118106
Date:	13/06/2024
Specialisation:	Artificial Intelligence
1st supervisor:	Niki van Stein
2nd supervisor:	Yingjie Fan

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

This study introduces ACO-NSGAII, a hybrid metaheuristic algorithm that integrates ant colony optimization (ACO) and non-dominated sorting genetic algorithm II (NSGA-II), to address the bi-objective Electric Vehicle Routing Problem with Time Windows (EVRPTW). The algorithm aims to minimize both the overall travel distances and the number of vehicles required in last-mile delivery scenarios. Our approach starts with ACO to generate an initial solution focusing on distance minimization, followed by NSGA-II to optimize the dual objectives efficiently. Extensive computational experiments demonstrate that ACO-NSGAII significantly outperforms existing methods like Random-NSGAII and NN-NSGAII, offering promising solutions for sustainable urban logistics. The findings contribute valuable insights into the trade-offs between minimizing distance and vehicle usage, highlighting the effectiveness of integrating ACO and NSGA-II for complex routing problems.

Contents

1 Introduction **4**
- 1.1 Vehicle routing problem . . . 4
- 1.2 Problem formulation . . . 5
  - 1.2.1 Research questions . . . 5
- 1.3 Thesis overview . . . 5

2 Related Work **6**
- 2.1 Vehicle routing strategies . . . 6
  - 2.1.1 Meta-heuristics . . . 6
- 2.2 Genetic Algorithm . . . 7
  - 2.2.1 Initialization . . . 8
  - 2.2.2 Selection . . . 8
  - 2.2.3 Crossover . . . 10
  - 2.2.4 Mutation . . . 10
- 2.3 Objectives . . . 11
- 2.4 Our contribution . . . 11

3 The problem statement **13**

4 Method **17**
- 4.1 The algorithm . . . 17
  - 4.1.1 Ant Colony Optimization . . . 17
  - 4.1.2 The ACO-NSGAII implementation . . . 17
  - 4.1.3 Parameters . . . 20

5 Experiments and results **22**
- 5.1 Data . . . 22
- 5.2 Experimental configuration . . . 23
- 5.3 Optimal route . . . 23
- 5.4 ACO-NSGAII ratio . . . 24
- 5.5 ACO vs nearest neighbor vs random . . . 26
  - 5.5.1 Pareto front . . . 27
- 5.6 PyMOO . . . 28
- 5.7 Crossover and mutation . . . 30

6 Discussion **32**
- 6.1 General limitations . . . 32
- 6.2 Interpretations . . . 32

6.2.1 Initialization method 32
6.2.2 Optimal route 33
6.2.3 Performance ACO-NSGAII 33

7 Conclusion **35**

Bibliography **35**

Chapter 1

Introduction

1.1 Vehicle routing problem

With the arrival of electric vehicles (EVs), the goal of creating a more sustainable life for individuals and companies becomes more realistic. However, with this, also new problems arrive. An important change compared to fueled cars, is the charging of the EVs and its logistics. One of the main challenges that comes with this, is the driving range anxiety. This is the fear of running out of battery before finding a charging station. It appears that drivers prefer for the charging stations to be there every five km and that people are feeling anxiety when their state-of-charge is below 30% or their remaining driving range is below 75 km [24].

Furthermore, for bus drivers, the problem becomes even more complex because they have to be at a certain location between specific time windows. Depending on the battery type, it takes 2-8 hours to fully charge a bus [22]. For this research, we aim to optimize the electric route vehicle problem with time windows (ERVPTW) by minimizing the number of EVs and the distance. In this paper, we propose an improved version of the genetic algorithm where we use the ant colony optimization (ACO) algorithm as the initialization step. We use ACO as our initialization step because it is able to find a near-optimal solution in the problem space [8]. Therefore, we believe that this will cause the genetic algorithm to converge faster while maintaining exploration.

The *vehicle routing problem* (VRP) is a well-discussed problem about optimizing the route for vehicles based on a certain objective like transportation time, transportation costs, or transportation distance. This problem has many variations and a common variation is the *vehicle routing problem with time windows* (VRPTW). This variation focuses on optimizing the route within a certain time window. This means that a penalty should be given if the vehicle arrives earlier or later than the specified time windows at its destination. Most of these problems are implied on vehicles that use fuel as their power supply but with today's development in sustainability, electric vehicles are largely researched. Since electric cars rely on a battery, new problems to the VRP are introduced like charging. This problem combined with VRPTW is then called the *electric vehicle routing problem with time windows* (EVRPTW) and this will be the main focus of this paper. However, since we try to mimic last-mile delivery, we only focus on the departure time window. So, a bus is permitted to arrive earlier than the arrival time windows without punishment.

1.2 Problem formulation

VRP and its variations are known to be NP-Hard. This in combination with multi-objectives, makes it difficult to find the optimal solution for EVRPTW. In order to solve this problem efficiently, we propose a metaheuristics algorithm based on ant colony optimization (ACO) and non-dominated sorting genetic algorithm II (NSGA-II) called ACO-NSGAII.

1.2.1 Research questions

For this research, we want to find an optimal route for the electric vehicle routing problem with time windows by introducing a hybrid algorithm. Therefore, our research questions are formed as follows.

- How can we improve the NSGA-II by changing the initialization with ACO and how do we define the ratio between these algorithms?
  - To what extent does the improved algorithm contribute to optimizing the route for EVRPTW?
  - How does this improved algorithm perform in terms of solution quality and computational time?

1.3 Thesis overview

The remainder of this thesis is structured as follows. In chapter 2 we give an overview of the related work that helped us in forming this research. Then, in chapter 3 we formulate the problem with the help of mathematical formulations and notations. Furthermore, in chapter 4 we explain what methods we used for implementing this research. This is followed by section 5.1 to explain what data we used during the implementation. Next, we move on to the experiments done in chapter 5 and we display the results after each experiment. Thereafter, in chapter 6, we discuss the findings of our research in great detail and this helps us to form a conclusion which is stated in chapter 7.

Chapter 2

Related Work

In this chapter, we discuss papers that helped us in gaining insight into the problem and how to solve this. This chapter also gives an overview of papers that discuss the state-of-the-art solutions for related problems.

2.1 Vehicle routing strategies

There are many methods for solving the routing problem for (electric) vehicles. Therefore, it is important to have an overview of the most important papers up until now. The following paper by Tan et al. [35] has created an overview of the research done in this field from 2019-2021. From this paper, we can conclude that the most-used methods for VRP are heuristic and meta-heuristic algorithms. So at least until 2021, the state-of-the-art methods are algorithms like genetic algorithm, particle swarm optimization, simulated annealing, and more (meta-)heuristic algorithms. Another development is that from 2021 up until now, machine learning has become more popular in the field of the vehicle routing problem (VRP) and there is research done with the aim for machine learning to beat the current algorithms [27]. In this paper by Czuba et al. [7], there has been a reviewing research done about solving the VRP with machine learning. They conclude however, that machine learning does not beat the current algorithms but can become promising in future work within the VRP, especially within static versions of this problem.

Since meta-heuristics are commonly used for the VRP and are largely optimized, we will continue our research within this field.

2.1.1 Meta-heuristics

In the paper by Zuvia et al. [42], they propose a solution for the green vehicle routing problem (GVRP) with the help of a many-objective gradient evolution algorithm (MOGE). Originally gradient evolution algorithms were proposed for single-objective problems which are solved with vector updating, jumping, and refreshing. This can only be done with continuous values, so for the MOGE algorithm, the paper represents the binary decision variable as a discrete solution and solves the vector updating with Pareto dominance relation and with crowd distance. Eventually, the MOGE algorithm was compared with the multiple-objective swarm optimization (MOPSO), non-dominated sorting genetic algorithm (NSGA) II, and multi-objective differential evolution (MODE) algorithms. The results of this comparison show that MOGE performed better within convergence and diversity.

In the paper by Li et al. [18], the authors create an ant colony optimization (ACO) algorithm to solve the GRVP for multiple objectives. Here, an agent in the algorithm represents an ant that has to find the shortest path between two nodes. The ants excrete pheromones which indicate a promising solution path. They improve the algorithm by solving multiple objectives with one single function and another improvement is the updating of the pheromones. It has been found that this version of ACO slightly outperforms the original ACO algorithm.

Furthermore, the paper by Asghari et al. [2] gives an overview of the state-of-the-art methods for GVRP. We will list some of the methods which are commonly used.

The large neighborhood search (LNS) is a method that works by searching other neighborhoods within the solution space to escape the local optima. An example of how LNS can be applied to the GRVP is the paper by Majidi et al. [20] in which the objective is to minimize emissions and fuel consumption for a pickup and delivery route. They achieve this with an adaptive large neighborhood search heuristic (ALNS). Results show that this method performed well and found a good solution for both small- and large-scale instances.

Another meta-heuristic algorithm that is used often for GVRP, is the genetic algorithm (GA). The paper by Erdem et al. [9] considers the routing problem in which healthcare workers make use of electric vehicles. This paper solves this problem by creating a multi-objective model that combines GA and a variable neighborhood descent. This method delivers a good-quality and fast solution for medium to large instances. Due to the promising nature of this algorithm, the next section will provide an in-depth explanation of GA.

2.2 Genetic Algorithm

The genetic algorithm is an adaptive method based on evolution found in biology in combination with the Darwinian theory; of survival of the fittest. This algorithm's principles were first founded by Holland in 1975 [32]. It belongs to the category of evolutionary algorithms and the steps are structured as follows. *Initialization*. First, there is a population initialized which consists of individuals that are eligible to solve a certain problem, these individuals are then evaluated based on their fitness. The better the individual can solve the problem, the higher the fitness. *Selection*. Based on an individual's fitness, the individuals with the higher fitness, have a larger probability to be selected for reproducing. *Crossover*. Reproducing is done by combining two individuals where those individuals cross over their chromosomes thus creating two offspring. *Mutation*. It is then possible, to mutate one or more chromosomes of the offspring. Through this way, a new population is created which can consist of only children or a mix of parents and children. This new population then gets evaluated and based on the fitness score, the fittest individual gets chosen as the best solution. This process can be repeated as desired [32] [30].

Furthermore, a commonly used variation of GA for multi-objective problems is the non-dominated sorting genetic algorithm II (NSGAII). This method varies from the basic GA by implementing the retention of elitist individuals in the population. So for each generation, the best-performing individuals are selected into the new population without undergoing modifications of the crossover and mutation operator. Another difference is that it sorts the population based on the Pareto ranking principle. This means that the first rank contains the individuals who are not dominated by other individuals. Furthermore, it calculates the crowding distance of each solution where a larger crowding distance means that the individual is in a less crowded area and thus there is more exploration. The selection also differs by having a tournament selection in which the Pareto rank and the crowding distance are used to make a selection [36].

Since the genetic algorithm has been the solution to many problems, researchers try to optimize this algorithm by optimizing its components. Therefore, it is useful to have an overview of the recent optimizations for each component. For this purpose, we present the recent studies that are done on the components in table 2.2. We provide more information about the optimizations in section 2.2.1 through 2.2.4.

2.2.1 Initialization

This component can be optimized in many ways. Originally, the initialization process of GA is randomized. In the paper by Zhu et al. [39], they optimize this step by implementing a neighbor routing method. Starting with one customer, the distance between this customer and all its neighbors is calculated with Euclidean distance. Then, the neighbor with the shortest distance to the customer gets chosen as the next customer to visit. the papers by Ibrahim et al. [14] and by Rabbouch et al. [26] implement the nearest neighborhood as well for initialization. Since this method can be used in the other steps of GA as well, it results in a fast solution for finding a good quality solution. However, since the selection of the first customer is still random, this method can still get stuck in a local optimum. To prevent this, only customers close to the depot should be considered as first customers.

The paper by Li et al. [15] applies a greedy algorithm for the initialization of the population of GA and also uses a local search algorithm and a battery charge station algorithm to improve children's selection. Furthermore, use the nearest neighbor method for initializing the population. They do this by choosing the next customer based on the distance.

2.2.2 Selection

There are some common strategies for the selection operator. We will list a few that we have encountered during our research. The most common strategy is called the *roulette wheel* selection and this works as follows. Each individual is measured by its relative fitness, the ratio between the individual's fitness and the total fitness. The fitness is a scoring measure of the objectives. The individuals are represented in a pie chart in which the covered area is relative to their fitness, hence the name of the roulette wheel. Then, there is a random number generated between 0 and the sum of relative fitness, S. The population is traversed and the fitness of each individual is summed up. If this sum is larger than the random number generated, the last individual is selected [3].

The second most common strategy is called the *tournament* selection and it works by first selecting k random individuals, then performing a tournament between those k individuals in which the individual with the best fitness gets selected. This process is then repeated until the required population size is achieved [10].

The benefit of the tournament selection is that is fairly simple to implement and that sorting of the population is not required. This makes it easy to solve. The benefit of the roulette wheel selection is that although it is more likely that a higher-fitness individual will be selected, exploration is still implemented.

Next to these common strategies, we have seen some optimizations done during our research. The paper by Chen et al. [6] combines the roulette wheel strategy with the retention

Table 2.1: Optimizations of the different components within the genetic algorithm

Nr.	Study	Initialization	Selection	Crossover	Mutation
1	Zhu et al. (2021) [39]	Neighbour routing	Roulette wheel	Crossover probability formula	Mutation probability formula
2	Li et al. (2020) [17]	Random	Roulette wheel	Partially mapped	Simulated annealing
3	Li et al. (2023) [16]	Nearest neighbour	Roulette wheel	1-point	Removal and reinsertion
4	Mulloorakam et al. (2019) [23]	Random	Tournament selection	BCRC, SMCM	Removal and reinsertion
5	Liu et al. (2022) [19]	Random	Roulette wheel	Order-based	Random and 2-opt algorithm
6	Li et al. (2019) [15]	Greedy algorithm	Tournament selection	2-point	Local search
7	Ibrahim et al. (2021) [14]	Nearest neighbour	Roulette wheel selection	Optimized based on distance	Optimized based on distance
8	Rabbouch et al. (2021) [26]	Nearest neighbour	Tournament selection	Route-based variation	Single swap
9	Ruiz et al. (2019) [28]	Random	Random with elitist bias	Elitist bias	Random
10	Chen et al. (2023) [6]	K-medoids clustering	Elite retention and roulette	Ordered crossover (outer) and single-point (inner)	Local search (outer) and random (inner)
11	Gocken et al. (2019) [12]	Clustering algorithms	NSGA-II	2-point	Mutation rate
12	Sayed et al. (2019) [29]	Improvement procedure	Sequential	2-point	Removal and reinsertion
13	This study	ACO with random	Tournament with Pareto and crowding distance	Order-based	Random and 2-opt algorithm

of the elitist chromosome. They achieve this by saving the elite chromosome in the offspring, then, the other chromosomes are added based on the roulette wheel strategy.

2.2.3 Crossover

There are many ways to implement crossover between two individuals. The paper by Mulloorakam et al. [23] compares two methods that have proven to be successful compared to the standard methods for crossover, e.g. one-point crossover in which a new individual is formed by swapping two parts of two individuals up until the crossover point. The two methods are called the *best cost rout crossover* (BCRC) and the *sub route mapped crossover method* (SMCM). With the BCRC method, the cost or distance constraints are taken into consideration during the crossover process. The SMCM method involves taking sub-routes of the parents based on criteria and merging them to create a new optimal route. Results show that the BCRC method outperforms the SMCM method in terms of efficiency.

The paper by Wang et al. [30] implements an optimized version of the crossover. They look at the distances between the crossover point customer and all the customers that are selected for crossover. Finally, they crossover the customers that are closest to the crossover point. The mutation within this paper is done similarly. The results show that the optimization of the crossover and mutation provides a reduction in total costs compared to the existing route for which no optimization is used.

Another crossover method which is implemented in the paper by Liu et al. [19], is called the order-based crossover. This is a method in which the order is preserved of the route of the parents. For example, if two parents are selected, they will interchange their chromosomes while retaining their relative orders. The main benefit of this method is that the order is kept while still implementing exploration by combining it with another parent's order.

Furthermore, the paper by [38] implements a 2-opt method in their crossover step. This ensures exploration because the parents selected for crossover are randomly chosen and the crossover points are randomly decided as well.

Another method that we have not seen yet is called the biased random key method which is implemented by Ruiz et al. [28]. In this method, bias is introduced during the crossover step. This works as follows, first, the population is divided into an elitist group with the best fitnesses and a non-elitist group. Bias is then introduced by always having at least one parent from the elitist group for mating during the crossover step. Results show that this improves the best-known solutions for a benchmark at that time.

2.2.4 Mutation

The mutation step exists for switching customers based on probability. Some papers have optimized this step, like the paper by Li et al. (2020) [17] in which they apply simulated annealing to the mutation step. Simulated annealing works by exploring the neighborhood for solutions in the search space. This ensures that the optimal solution does not get stuck in a local optimum. Furthermore, the parameters of the simulated annealing are not fixed but can be changed based on the fitness of the population and individuals. This results in faster convergence and escaping local optima compared to the original GA.

The paper by Li et al. (2023) [16] implements another method for a mutation called removal and reinsertion. This is based on the local neighbor search algorithm in which the removal operation works by removing a customer based on its relevance with another customer. The relevance is based on the distance and the smaller the distance between two customers, the higher the probability of being removed. Thereafter, the removed customers are reinserted in the route again. This process enables the removed customers to be reinserted optimally so that the total distance traveled becomes minimized. This method results in faster convergence compared to the original GA and also minimizes the total energy consumption of electric vehicles.

The paper by Liu et al. [19] implements a solution for the EVRP with time windows. They develop a genetic algorithm in combination with a 2-opt algorithm. They reason that GA does not perform well in finding the local optimum whereas the 2-opt algorithm is specifically designed to find the local optimum. This 2-opt algorithm is added after the mutation step and works by exchanging two points within two individuals as long as electric energy is reduced. They conclude that their hybrid GA works better when there are more nodes in the route.

2.3 Objectives

For the EVRPTW we can have single or multi-objectives. We are interested in having a multi-objective model, therefore we have done some literature review that helped us decide in forming our objectives. This can be found in table 2.2. The paper by Tan et al. [34] implements two objectives that conflict with each other. The objectives are to optimize the total distance as well as optimize the number of used vehicles for doing this. They use the Tchebychef Approach to solve this. The Tchebychef Approach is an optimization technique that converts multiple objectives into one single objective by the usage of weights. So the weighted sum of all the objectives is taken as a single objective. The weights determine the importance of the objective [37]. The paper by Ombuki et al. [25] solves the same objective but instead of having to deal with weights, they use Pareto ranking to get the multi-objective functions in a single-objective function. Pareto ranking considers only solutions that outperform other solutions in at least one of the objectives and don't perform worse than other solutions on the remaining objectives.

The paper by Ghoseiri et al. [11] uses Goal programming to solve the multi-objectives. Goal programming works by setting up goals and constraints for each objective, this allows decision-makers to set their preferences. The paper then uses Pareto ranking in combination with a genetic algorithm to solve it. The paper by [23] uses an aggregated fitness function instead of a normal one in their genetic algorithm since they are dealing with multi-objectives. They do this by calculating the fitness of one objective at a time and then combining the two. The paper by Moradi et al. [21] implements a whole algorithm based on the Pareto ranking to solve the objectives.

2.4 Our contribution

We see from table 2.1 that most of the research effort focuses on crossover and mutation. Furthermore, we see that most optimizations that are done on initialization, use clustering or nearest neighbor. Therefore, we focus this research on improving the NSGAII for the EVRPTW by proposing a hybrid optimization algorithm. We use ant colony optimization (ACO) for generating the first population using only one objective. Our improved NSGAII algorithm is then called ACO-NSGAII. Furthermore, we implement a bi-objective model in which the aim is to minimize the number of vehicles used and to minimize the total traveled distance.

Table 2.2: Multi-objectives of the Vehicle Routing Problem

Nr.	Study	The objectives	Solution
1	Mulloorakam et al. (2019) [23]	Minimization of total distance travelled and the number of vehicles.	Aggregated fitness
2	Gocken et al. (2019) [12]	Minimization of the total distance and the total waiting time of vehicles.	NSGA-II with Pareto
3	Tan et al. (2023) [34]	Minimizing the total distance and the number of vehicles required for transport.	Tchebychef approach
4	Ombuki et al. (2006) [25]	Minimizing the total distance and the number of vehicles required for transport.	Pareto ranking
5	Ghoseiri et al. (2010) [11]	Minimizing the total distance and the number of vehicles required for transport.	Goal programming and Pareto ranking
6	Moradi et al. (2020) [21]	Minimizing the total distance and the number of vehicles required for transport	Pareto evolutionary algorithm
7	This study	Minimizing the total distance and the number of vehicles required for transport	Pareto ranking and crowding distance

Chapter 3

The problem statement

In this chapter, we explain the problem in great detail with the help of mathematical formulations and notations.

Table 3.1: Notation of Sets, Parameters, and Variables

Sets	
O	Depot set of size 1
C	Customers set of size n
CH	Charging stations set of size 2
V	Vertices set $V = O \cap C \cap CH$
H	EVs set
Parameters	
d_{ij}	Euclidean distance between vertices i and j
x_i, y_i	Coordinates of i ($i \in V$)
$[a_i, b_i]$	Customer i's time window
B	Max capacity of battery set to 100
E_{hi}	Current battery level for vehicle h at customer i
u	Battery consumption rate set to 0.1
s_i	Service time at customer i
t_{ch_h}	Charging time for EV h
c_1	Charging station with coordinates on $[25, 75]$
c_2	Charging station with coordinates on $[43, 60]$
T	Battery threshold for charging, set on 30
Variables	
p_{ijh}	Binary variables: 1 - vehicle h travels between i and j, 0 - otherwise.
q_{ih}	Binary variables: 1 - vehicle h charges at station i, 0 - otherwise.
t_{ih}	Arrival time at customer i for the electric vehicle h

Our EVRPTW is in the form of a directed graph $G(V, E)$ with $V = \{v_0, v_1, \ldots, v_n\}$ which consists of the depot $O = \{v_0\}$, all the customers, which are bus stations, $C = \{v_1, \ldots, v_n\}$ and the charging stations $CH = \{c_1, c_2\}$ if visited for a route. With the edges $E = \{(v_i, v_j) : v_i \neq v_j \wedge v_i, v_j \in V\}$. The customers are served by identical electric vehicles H. Each customer has a time window $[a_i, b_i]$ in which a_i is the earliest time a vehicle can arrive at customer i and b_i is the latest time a vehicle may arrive at customer i. In our model, we do not consider a capacity load and we allow a vehicle to arrive earlier than a_i without waiting time or penalty. Therefore our model only considers the latest arrival time window, $[\_, b_i]$ for customer i.

Since we are dealing with electric vehicles, we have added two charging stations $CH = \{c_1, c_2\}$ which have the same time windows as the depot and the same service time which is zero. The coordinates of $c_1 = (25, 75)$ and the coordinates of $c_2 = (43, 60)$. All the customers i have a service time s_i that a vehicle must spend. If the vehicle h is not able to arrive at a customer i before the time window b_i, a new subroute is created for which a new vehicle is used. Also, all the EVs are supposed to return before the depot's due time, b_0. Furthermore, $B = 100$ which is the max capacity of the battery in percentages. $u = 1$ is the battery consumption rate. An overview of these parameters can be found in table 3.1.

Each vehicle starts and ends at the depot. If a vehicle needs to charge, it goes to the nearest charging station. The distance between two points is calculated with the Euclidean distance. Mathematically our model looks as follows.

$$\text{min } (f_1, f_2) \tag{3.1}$$

$$f_1 = \sum_{j \in V} \sum_{h \in H} p_{0jh} \tag{3.2}$$

$$f_2 = \sum_{i \in V} \sum_{j \in V, i \neq j} \sum_{h \in H} p_{ijh} d_{ij} \tag{3.3}$$

s.t.

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{3.4}$$

$$\sum_{j \in C} \sum_{h \in H} p_{ijh} = 1 \quad \forall i \in V, i \neq j \tag{3.5}$$

$$\sum_{j \in C} p_{ijh} = \sum_{j \in C} p_{jih} \leq 1 \quad \forall i \in V, i \neq j \tag{3.6}$$

$$E_{hi} = B \quad \forall h \in H, \forall i \in O \cap CH \tag{3.7}$$

$$E_{hj} = E_{hi} - u d_{ijh} \tag{3.8}$$

$$t_{ch_h} = s_i \tag{3.9}$$

$$v_j = c_1 \in CH, \quad E_{hi} < T \wedge d_{ic_1} < d_{ic_2}, \forall i \in V \tag{3.10}$$

$$v_j = c_2 \in CH, \quad E_{hi} < T \wedge d_{ic_2} < d_{ic_1}, \forall i \in V \tag{3.11}$$

$$t_{jh} = t_{ih} + s_i + d_{ij} + q_{ih} t_{ch_h} \quad i, j \in V, h \in H, i \neq j \tag{3.12}$$

$$t_{ih} \leq b_i \wedge t_{ih} \leq b_0 \quad i \in C, h \in H \tag{3.13}$$

where,

$$p_{ijh} = \begin{cases} 1 & \text{if there exists a path between } v_i, v_j \text{ with vehicle } h \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

$$q_{ih} = \begin{cases} 1 & \text{if } v_i \text{ is a charging station with vehicle h} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Here, the objective function given in equations (3.1), (3.2) and (3.3) is to minimize the number of vehicles used for transportation, H, and to minimize the total distance traveled. The distance between two customers is calculated with the Euclidean distance as is stated in equation (3.4). Furthermore, constraint (3.5) states that a customer can only be visited once and by one vehicle only. Constraint (3.6) states that each vehicle starts and ends at the depot after visiting customers. The constraint (3.7) states that the battery of an EV h, is fully charged when it leaves the depot or a charging station and the constraints (3.8) and (3.9) show how the battery percentage is updated during transportation, and how much time it takes to charge the EV back to 100 respectively. Constraints (3.10) and (3.11) state that if the battery of the current vehicle at the current customer gets below the charging threshold, which is set to 30, it needs to go to the nearest charging station as its next node which is either c_1 or c_2. The reason we chose 30 as our threshold, is to address the driving anxiety which we stated in the introduction. Finally, the constraints (3.12) and (3.13) show how the travel time is calculated and how the time windows are ensured. The equations in (3.14) and (3.15), explain the decision variables for the determination that a path between two customers exists, and for the determination of visiting a charging station for vehicle h.

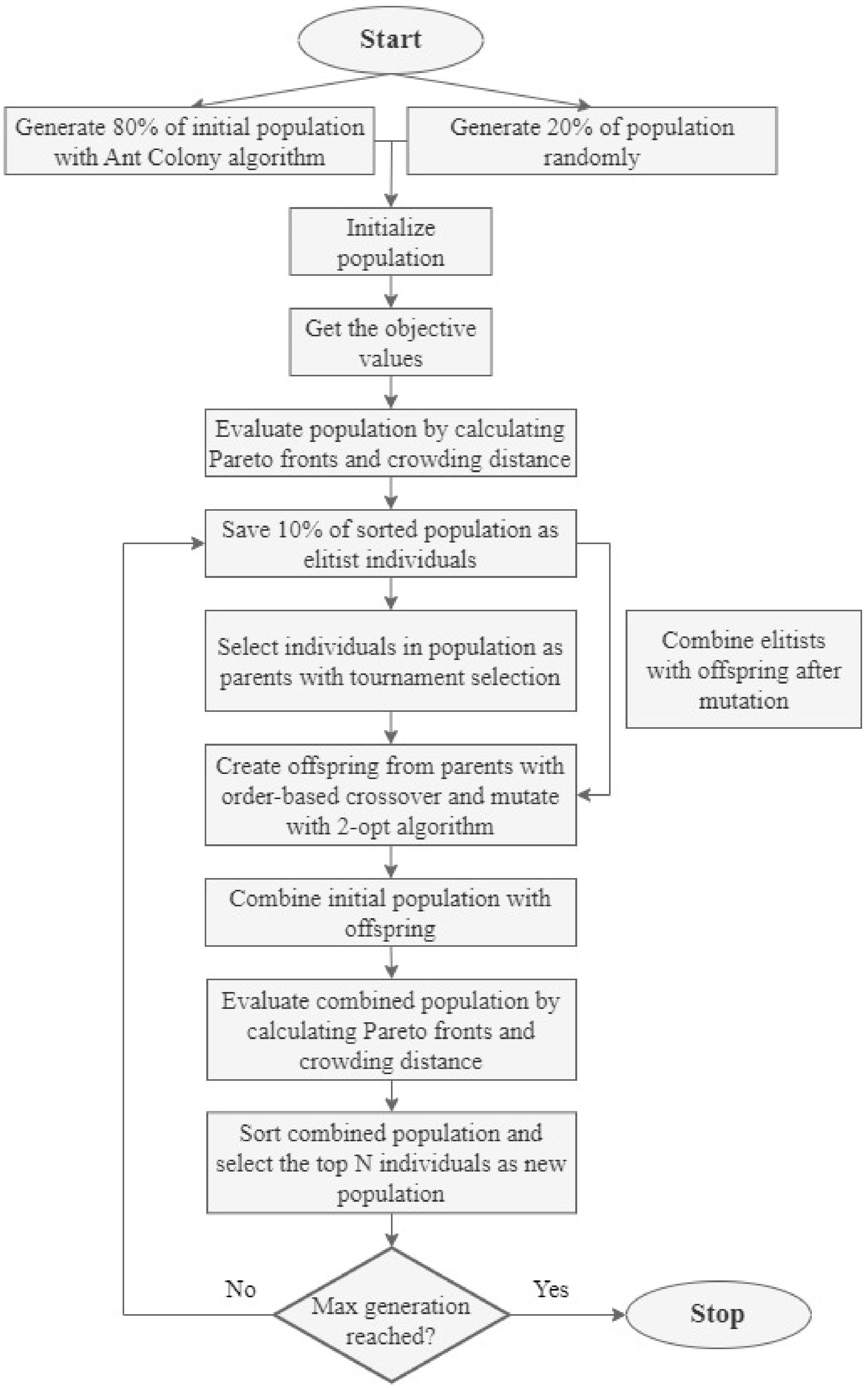

Start
Generate 80% of initial population with Ant Colony algorithm
Generate 20% of population randomly
Initialize population
Get the objective values
Evaluate population by calculating Pareto fronts and crowding distance
Save 10% of sorted population as elitist individuals
Select individuals in population as parents with tournament selection
Combine elitists with offspring after mutation
Create offspring from parents with order-based crossover and mutate with 2-opt algorithm
Combine initial population with offspring
Evaluate combined population by calculating Pareto fronts and crowding distance
Sort combined population and select the top N individuals as new population
No
Max generation reached?
Yes
Stop

Figure 3.1: Flowchart of improved NSGA-II

Chapter 4

Method

In this chapter, we explain our methods.

4.1 The algorithm

Since the main improvement comes from the ant colony optimization, we first explain how this algorithm works. Thereafter, we give an in-depth explanation of our implementation for solving the EVRPTW with ACO-NSGAII. A flowchart of this implementation can be found in figure 3.1. Furthermore, our implementation was created with the help of a public repository on GitHub.[1]

4.1.1 Ant Colony Optimization

The ant colony optimization (ACO) algorithm originates from real ant colonies as the name suggests. This algorithm was first introduced by M. Dorigo [8]. Ants live in colonies and their one goal is to protect the colony instead of themselves. This means that the ants are not individualistic beings but rather social. Their main priority is to provide food for the colony. This is done as follows. First, the ants walk random paths to find food. Whenever an ant moves, it leaves behind a *pheromone* trail. This is a chemical substance that other ants can smell. Furthermore, when an ant finds food, it takes it back to the colony and based on the quality and quantity of the found food, it leaves a more or less intense pheromone trail. Other ants walk a path depending on probabilities; the stronger the pheromone trail, the larger the probability. Finally, the shortest path is found from the colony to the food since that path will be walked the most and therefore contain the most pheromone [5]. Based on this concept, an algorithm is created. Figure 4.1 shows this process. Here N represents the nest, F represents the food, 'a' represents the path to the food and 'b' represents the path to the nest. As becomes visible, the shortest path gets the strongest pheromone trail and therefore the largest probability that an ant would follow that path.

4.1.2 The ACO-NSGAII implementation

We now give an in-depth explanation of our implementation.

[1]https://github.com/iRB-Lab/py-ga-VRPTW

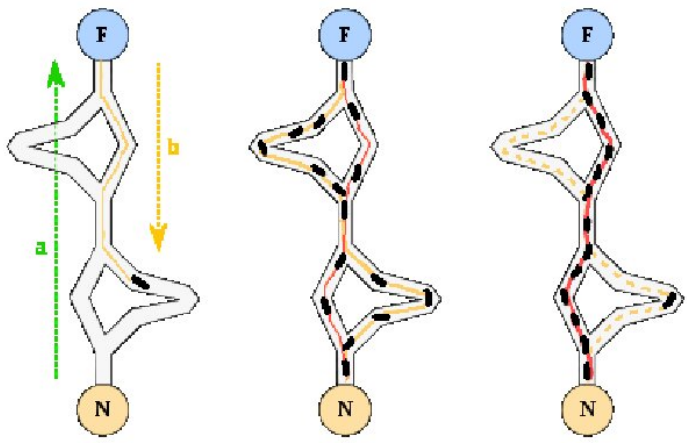


Figure 4.1: Ants looking for food [1].

0	5	3	9	0	7	4	2	6	0	8	1	0

Figure 4.2: Route representation as a chromosome (from [16]).

Initialization with ACO

Our initialization is a combination of the ant colony algorithm (ACO) and random. We generate 80% of the population with ACO and the remaining 20% is generated randomly. The reason for the 20% being random is to maintain exploration. Furthermore, in the initialization with ACO, we do not take the constraints of time windows and battery percentage into consideration. This means that the generated population from ACO is solely dependent on the shortest distance which is calculated with the Euclidean distance. Our implemented ACO can be seen in Algorithm 1.

Our population then consists of an order of the customers without time windows and battery percentage. Therefore, we then create a route for each individual in the population. Here we consider the second objective of minimizing the number of vehicles, and we consider the constraints of time windows and battery percentage. The vehicles have a battery percentage of 100 when they leave the depot and we have set it so that the battery decreases linearly with the distance between two nodes. Then, if the battery is below 30%, the vehicle goes to the nearest charging station and charges until 100%. The time it takes to charge is equal to the service time of other customers. The service time is the time that a vehicle is required to stay at a customer. Finally, a new subroute is created if the current vehicle cannot leave on time or if the vehicle should return to the depot because it may not arrive later than the depot's time window. Since the genetic algorithm is based on biological evolution, an individual, which is a route, in a population is then called a chromosome. An example of 10 customers is seen in figure 4.2. Here, the first customer 0, is the depot. So we see that there are three subroutes in this route.

Evaluation & selection

After creating an initial population, we evaluate this by calculating the distance and the number of vehicles used for each route. Then, the Pareto fronts are computed and with the Pareto fronts, we can calculate the crowding distances for each front. Furthermore, we save 10% of

Algorithm 1 The Ant Colony algorithm

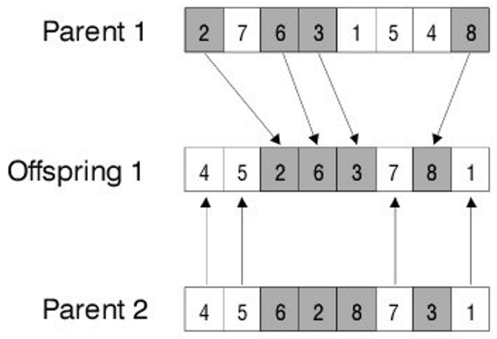
```
1: Initialize the distance_matrix and the pheromone_matrix
2: best_solution = None
3: best_distance = $\infty$
4: for generation in number of generations do
5:     for ant in number of ants do
6:         visited $\leftarrow$ list of visited nodes
7:         solution_distance = 0
8:         solutions = []
9:         while unvisited nodes do
10:            calculate the probabilities of unvisited nodes
11:            select the next node
12:            add next node to solutions
13:            add distance between current and next node to solution_distance
14:            update visited
15:        end while
16:        add distance to depot node to solution_distance
17:        if solution_distance < best_distance then
18:            best_solution = solution
19:            best_distance = solution_distance
20:        end if
21:    end for
22:    update pheromone_matrix by applying evaporation_rate
23:    update pheromone_matrix by adapting the amount of pheremone depending on
       the solution and its distance
24: end for
25: return best_solution
```

the best solutions based on the Pareto front and crowding distances as the elites.

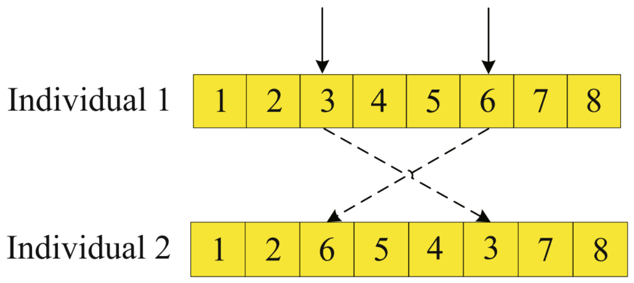
We now have to select parents to reproduce for the next generation. The selection is based on a tournament selection in which two individuals compete. We sample two random individuals from the population. The individual with the lower Pareto rank gets to be selected. However, if the individuals have the same Pareto rank, the individual with the larger crowding distance wins and therefore gets selected.

Crossover

For crossover, we implement the order-based operation which works as follows. After selecting two parents, we sample a random number of individuals from parent 1. For creating the offspring, we first take the individuals from parent 2 that do not appear in the sampling of parent 1. We then fill the offspring with these individuals while preserving their position and order as they appear in parent 2. The remaining gaps are eventually filled with the relative order of the selected individuals from parent 1. A visualization of this operation is seen in figure 4.3a.



(a) Order based crossover (from [31]).



(b) 2-opt mutation operation (from [19]).

Figure 4.3: Crossover and mutation operations.

Mutation

The mutation step of ACO-NSGAII is based on the 2-opt algorithm. First, we randomly select two indices and reverse all of the nodes that lay between these two indices. Then, we create our individual by joining the first part up until the first random index of the offspring which remains unchanged, then the reversed part, and finally the remaining part, which starts from the second random index, which also remains unchanged. In other words, we reverse a section of the offspring. A visualization is seen in figure 4.3b. Here we see that the segment between customers 3 and 6 is reversed and as such we created a new individual. Note that the 2-opt algorithm repeats this process until no improvement has been found but we apply this process only once.

4.1.3 Parameters

There are many parameters used in the genetic algorithm that can be modified for experimenting. In table 4.1 we display these and their default values. The ACO-NSGAII has a total budget of 30000 runs. The budget is split equally over both algorithms ACO and NSGA-II meaning that each algorithm covers 50% of the total budget runs.

Parameter	**Value**
Population size	100
Number of generations	150
Mutation rate	1.0
Crossover rate	0.95
Battery consumption rate	1.0
Elites retention	10%
Number of ants	100
Number of iterations ACO	150
Evaporation rate	0.5
α	1
β	3
Q	1
Total budget	30000

Table 4.1: Hyperparameters used in ACO-NSGAII

Chapter 5

Experiments and results

In this chapter, we explain experiments done for this research. We will also describe our experimental setup. For the experiments we use the parameter values in table 4.1 unless stated otherwise.

5.1 Data

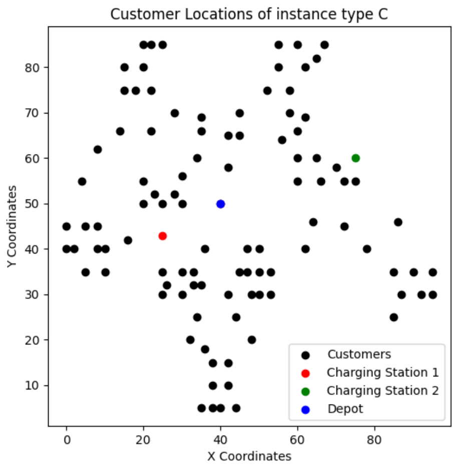
For this research we use Solomon's benchmark which was introduced by M. Solomon in 1987 [33]. This benchmark is widely used for vehicle routing problems with many variations such as the capacitated VRP or the VRP with time windows (VRPTW). The benchmark consists of three types of instances for two types of schedules. We have different types of instances in the sense of geographical locations. The C-type means that the customers are spread in clusters (fig. 5.1a), the R-type means that the customers are randomly spread around the grid (fig. 5.1b), and the RC-type means that the customers are spread as a mixture of random and clustered (fig. 5.1c). Besides, we have R1, C1, and RC1. Those instances are of type 1 and this means that they have a short scheduling horizon. In other words, their time windows are tight and because of this, a vehicle can only serve a couple, five to ten, customers in one trip. On the contrary, we have R2, C2, and RC2 which are instances of type 2 meaning that they have a long scheduling horizon. Their time windows are much looser and as a result, a vehicle can serve more than 30 customers during one trip. Furthermore, within one type, the instances can differ in time windows, so "r101" has looser time windows constraints than "r102".[1] These instance types are visualized in figure 5.1.

Furthermore, all of the instances consist of 100 customers. However, smaller versions of the instances can be created by taking only the first 25 or the first 50 customers. These are then portrayed like this "r101.25". In this example, we take only the first 25 customers of the instance r101. The same goes for 50 customers. The default, "r.101" is 100 customers.

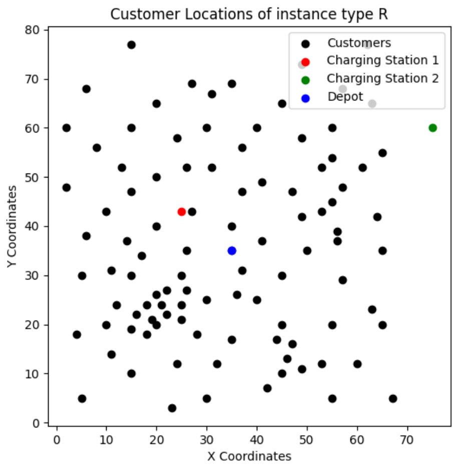
Since we are dealing with electric vehicles, we need to add a charging station to the data. We do this by adding two charging stations called "c1" and "c2". Their coordinates are the same as in the papers by Li et al. [16] and by Zhu et al. [40]. These are set on (25, 43) and (75, 60). Also, the time windows of the charging stations are the same as the depot and the charging time for a vehicle is the same as the service time of other customers.

We preprocess the data such that we get a list containing the values for each customer. An example would be the following:

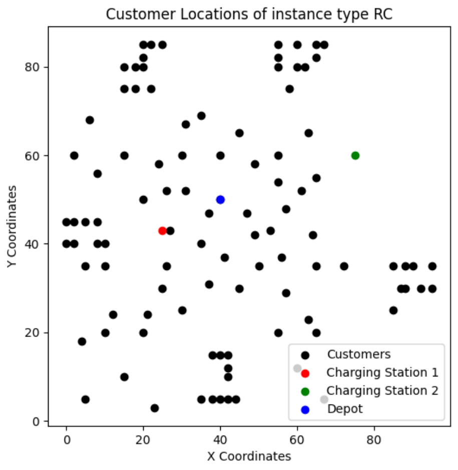
[1]`http://web.cba.neu.edu/~msolomon/problems.htm`



(a) Instance type C



(b) Instance type R



(c) Instance type RC

Figure 5.1: Visualization of the customers' location for the tree different Solomon instance types clustered (C), random (R), and random clustered (RC).

`CUSTOMER,XCOORD.,YCOORD.,DEMAND,READY TIME,DUE DATE,SERVICE TIME:`
`1,52,75,10,311,471,90.`
As visible, all of the instances include a capacity demand. However, since our model is not dealing with capacity, we do not use it for our algorithm.

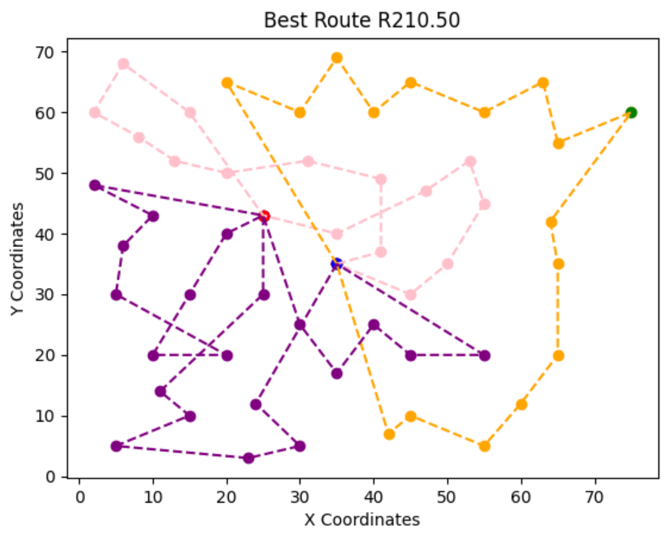
5.2 Experimental configuration

The experiments of this paper are conducted on a local laptop. The processor is a 12th Gen Intel(R) Core(TM) i7-12800H with a frequency of 2.40 GHz which has 32.0 GB RAM capacity. Furthermore, it was run on Windows 11 Enterprise using Visual Code Studio. Furthermore, all experiments are run ten times and we take the averages to combat randomness.

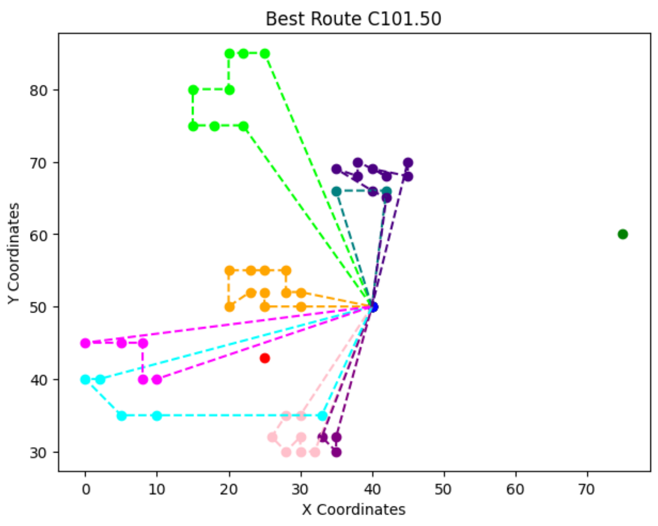
5.3 Optimal route

The purpose of this research is to find the optimal route for EVRPTW. Therefore, we show the optimal route for each instance type using the default hyperparameter values from table 4.1. For the instance types R and C, we use only 50 customers so that we get a more visible graph. The routes of these types are seen in figure 5.2 where in subfigure 5.2a the route for instance R210.50 is shown and in subfigure 5.2b the route for instance C101.50 is shown.

We used different scheduling horizons as well. Because of this, we see that we need fewer vehicles to cover all of the stations for the long-scheduled horizon in subfigure 5.2a. However, these vehicles travel a longer distance and because of this, the charging stations are used three times. Each vehicle is charged once. This is in contrast to the short scheduled horizon which we see in subfigure 5.2b. Here we see that we need a lot more vehicles, eight vehicles, and that none of the charging stations are used. This is because a vehicle can only cover a small distance before having to return to the depot. If that is the case, another charged vehicle continues the route.



(a) Optimal route for instance R210.50



(b) Optimal route for instance C101.50

Figure 5.2: Optimal routes for two instances with 50 customers.

Furthermore, we see that the clusters are handled well. It appears that the vehicles can find the clusters and pass them before going back to the depot or another cluster. For the random instance, subfigure 5.2a, we see that the route could be optimized more since sometimes we see two lines of the same vehicle crossing each other.

Besides these types, we also show the route for type RC in figure 5.3. Here we use instance RC201 and we use 100 customers instead of 50. We see that some clusters are again being handled well whereas there are also edges crossing each other by the same vehicle. We see that the charging stations are used again since we are dealing with a long scheduling horizon. For this route, seven vehicles are used. Furthermore, we see that it seems that the inner customers are being served as a whole. We mean by this, that the customers closest to the depot, the pink and orange customers, are served without going to customers to the edge of the grid.

5.4 ACO-NSGAII ratio

In order for us to improve NSGA-II by using ACO for initialization, we need to define what the optimal ratio is between these two algorithms. To do this, we experiment with different budget allocations. We allocate $10\%, 25\%, 50\%, 75\%$ and 90% of the total budget to ACO, with the remaining budget for NSGA-II. For each setting, we plot the average results per 50 generations. In Figures 5.4 and 5.5 we show the convergence trend of ACO-NSGAII on the two objectives (travel distance and the number of vehicles) with different initialization budget ratios.

In both figures, we see that the convergence trend for the ACO part is the almost the same, so the number of runs do not have an impact. However, for minimizing the distance in figure 5.4, we see that 90% NSGA-II allocation together with 50% NSGA-II allocation provides the best result. Another thing to be noted, is that the distance first goes up when switching to NSGA-II after ACO. After that, the trend line converges rather quickly. In figure 5.5 where the focus is on minimzing the number of vehicles, we see that the allocation of 50% NSGA-II

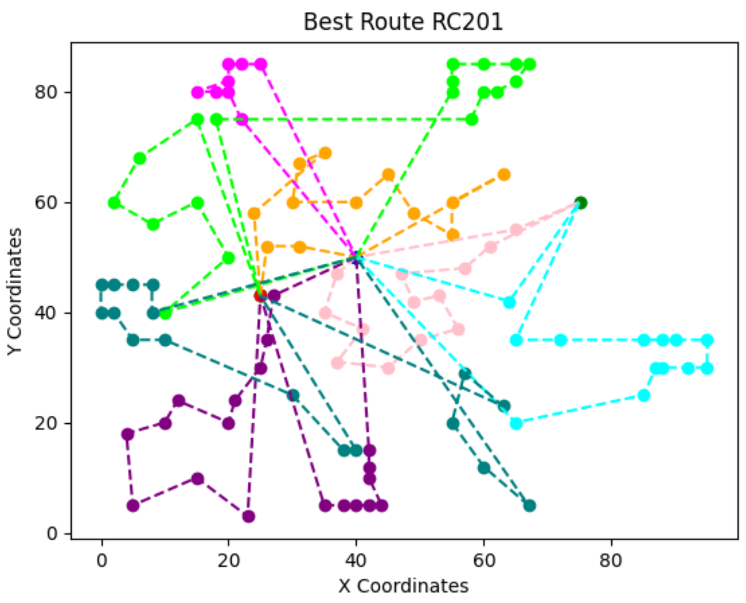


Figure 5.3: Optimal route for instance RC201

provides the best results since that configuration yields the lowest average distance.

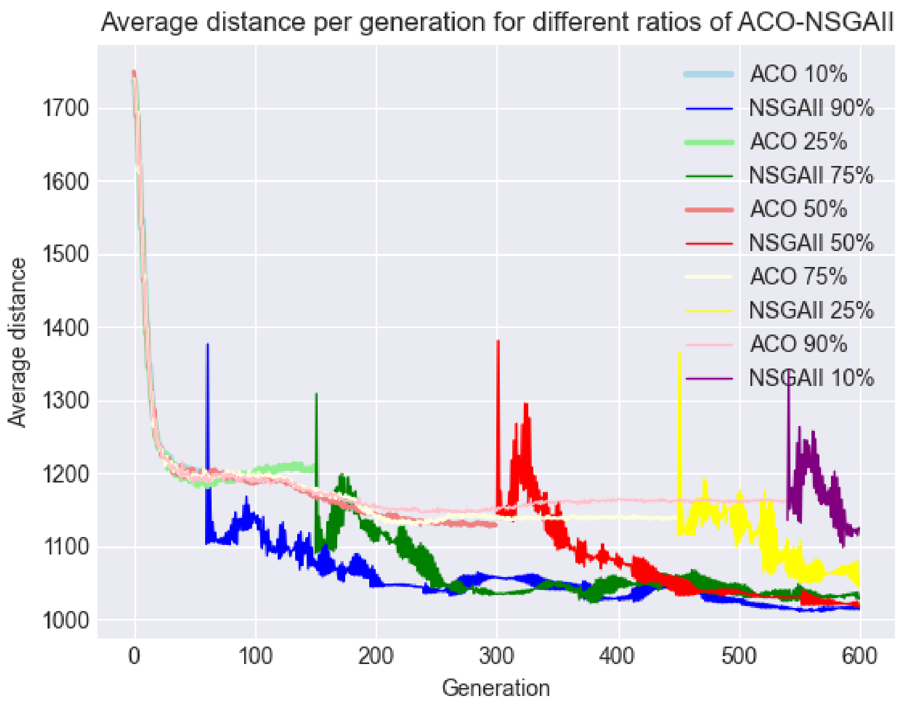


Figure 5.4: Convergence trend of ACO-NSGAII on distance for different initialization budget ratios (instance C204)

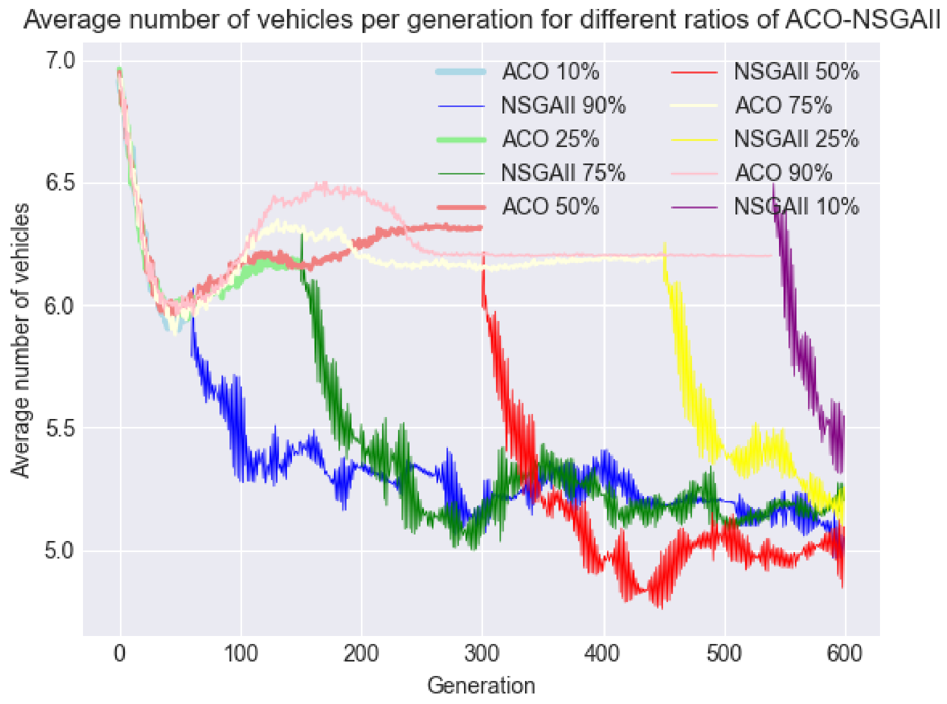


Figure 5.5: Convergence trend of ACO-NSGAII on the number of vehicles for different initialization ratios (instance C204)

5.5 ACO vs nearest neighbor vs random

In this experiment, we want to show the effect of using ACO for generating the initial population. We do this by comparing our initialization method with random search and with nearest neighbor as initialization methods. These two methods are the most common as conducted from our related work which is seen in table 2.1.

The initialization of the first population of Random-NSGAII is created by generating random permutations of the customers without considering the objectives or constraints. Then, NSGAII creates an initial feasible population by considering minimizing the objectives and considering the constraints. For NN-NSGAII, the initialization is generated by randomly choosing one of the nearest five neighbors to the current node as the next customer to be visited. The vehicle objective and constraints are not considered and then again, NSGAII generates a feasible population by considering the second objective and the constraints. For each of the three algorithms (ACO-NSGAII, NN-NSGAII, and Random-NSGAII), we allocate 50% of search budget on initial solution and the rest for NSGAII. The average results per 50 generations on instance R101 are presented in the figures Fig. 5.6 and Fig. 5.7, with respect to the two objectives.

We see that in Fig. 5.6 where we minimize the number of vehicles, the NN-NSGAII and the Random-NSGAII perform better until around $400 \cdot 50$ generations. Thereafter, ACO-NSGAII outperforms both of these initialization methods. Furthermore, we see that NN-NSGAII slightly outperforms Random-NSGAII.

In Fig. 5.7 where the focus lays on minimizing the distance, ACO-NSGAII substantially outperforms the other two methods. ACO-NSGAII here yields better results for each generation.

We see that ACO-NSGAII seems to get stuck in a local optimum around 120 generations. Furthermore, it appears that during the first 350 generations, Random-NSGAII yields better results than NN-NSGAII. However, in the end NN-NSGAII slightly outperforms Random-NSGAII.

However, in terms of computational time, ACO-NSGAII performs worse than NN- and Random-NSGAII. This can be seen in table 5.1. The NN-NSGAII is 4.41 times faster than ACO-NSGAII and the Random-NSGAII is 4.50 times faster. We believe that this is because ACO is a metaheuristic algorithm which is computationally more expensive than NSGAII.

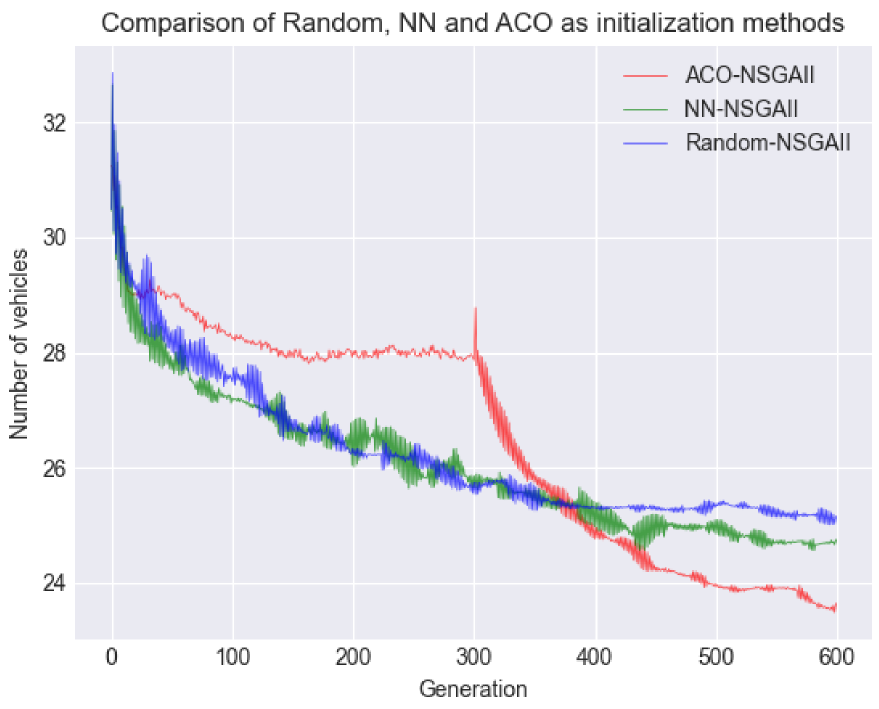


Figure 5.6: Number of vehicles convergency trend of ACO-NSGAII, NN-NSGAII, and Random-NSGAII (instance R101)

Table 5.1: Average computational time in seconds for ACO-, NN-, and Random-NSGAII (instance R101)

Method	Computational time
ACO-NSGAII	67.63
NN-NSGAII	15.33
Random-NSGAII	15.03

5.5.1 Pareto front

Another way to compare our initialization method is by evaluating the quality of our algorithm with the help of the Pareto front. Besides, we also want to measure the trade-off between our two objectives. In multi-objective problems, there often is not a single point that is the best solution. Therefore, instead of a single point, we use the Pareto front to evaluate the best

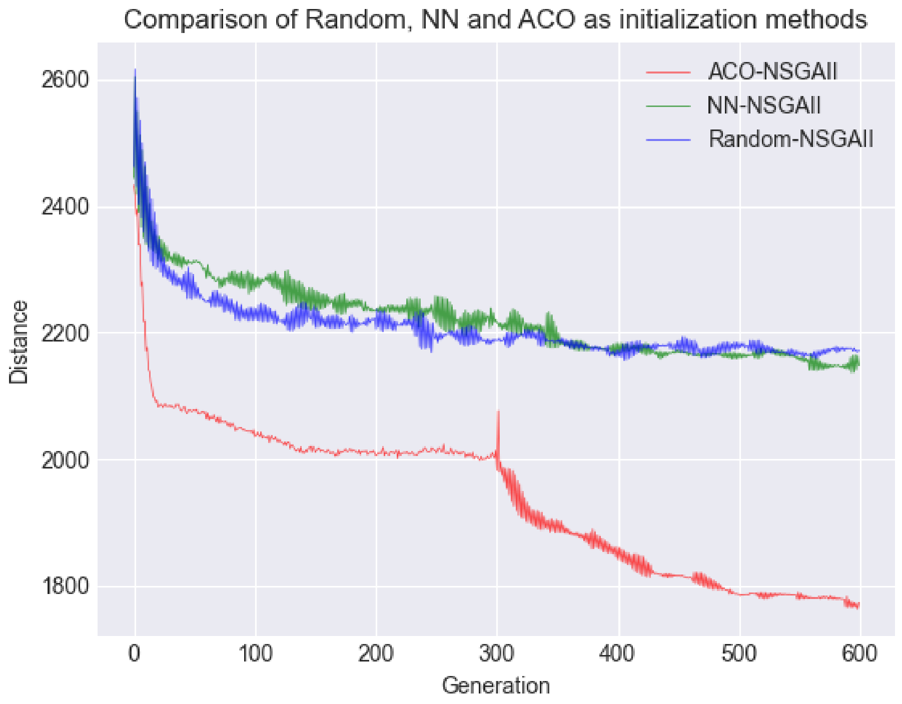


Figure 5.7: Distance convergency trend of ACO-NSGAII, NN-NSGAII, and Random-NSGAII (instance R101)

solutions. The Pareto front consists of a set of non-dominated solutions. A point $p \in \mathbb{R}^d$ is set to strictly dominate another point $q \in \mathbb{R}^d$ if $p_i \leq q_i$ for all $1 \leq i \leq d$ and if $p < q$ [13]. In other words, this means that the point p dominates another point q if p has at least better values for one objective and not worse values for the other objectives.

The Pareto front is also a way to see if the objectives are conflicting or correlating. Therefore, we plot the Pareto front of each initialization method. This can be seen in Fig. 5.8. Here again, we see that ACO-NSGAII outperforms NN-NSGAII and Random-NSGAII. The solutions for ACO-NSGAII are smaller for both the distance as well as the number of vehicles compared to the other methods. We see that NN-NSGAII outperforms Random-NSGAII. Random-NSGAII here performs substantially worse since its solutions do not cover the same solution space as the other methods.

Furthermore, it becomes visible that our objectives conflict with each other. We see this because as the distance is minimized, the number of vehicles becomes larger and the other way around when the number of vehicles is minimized, the distance becomes larger. This trade-off is apparent for all three methods.

5.6 PyMOO

To evaluate our algorithm, we applied the algorithm to a test problem from PyMOO. *PyMOO* is a library for single- and especially multi-objective optimization in Python. It offers state-of-the-art algorithms and contains many multi-objective test problems that their algorithms solve. Its framework provides many tools that one can use for solving a single- or multi-

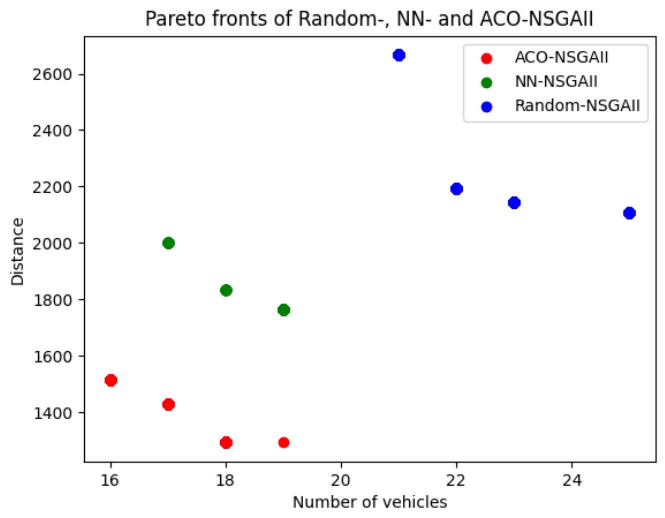


Figure 5.8: Pareto-optimal Solutions using ACO-NSGAII, NN-NSGAII, and Random-NSGAII (instance C101)

objective optimization problem. They focus on solving those with evolutionary algorithms. This framework is, because of its simplicity, useful for researchers [4].

Since PyMOO does not contain vehicle routing test problems, we decided to test our algorithm on the first multi-objective problem, "ZDT-1", by Zitzler, Deb, and Thiele, hence ZDT [41]. This problem contains two objectives to be minimized for $n = 30$:

$$\min f_1(x)$$
$$\min f_2(x) = g(x)h(f_1(x), g(x))$$

where,

$$f_1(x) = x_1$$
$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$$

such that,

$$0 \leq x_i \leq 1 \quad \text{and} \quad i = 1, \dots, n$$

The optimum solution for this problem is $0 \leq x_i^* \leq 1$ and $x_i^* = 0$ for $i = 2, \dots, n$. In figure 5.9 we plotted the Pareto front that we get when we run our algorithm on the ZDT1 problem, versus the Pareto front for the optimum algorithm on the same problem. We see that our algorithm does not converge for the objective function f_2 since its values are between 5.2 and 2.0 instead of 1.0 and 0.0. The Pareto front however has almost the same shape as the optimum Pareto front. It is important to note that for running our algorithm on the ZDT1 problem, we removed the initialization with ACO and replaced it with random generation instead. The reason for this is that the ZDT1 problem is not a vehicle routing problem but rather an optimization problem. Therefore, we did not have coordinates to calculate a distance

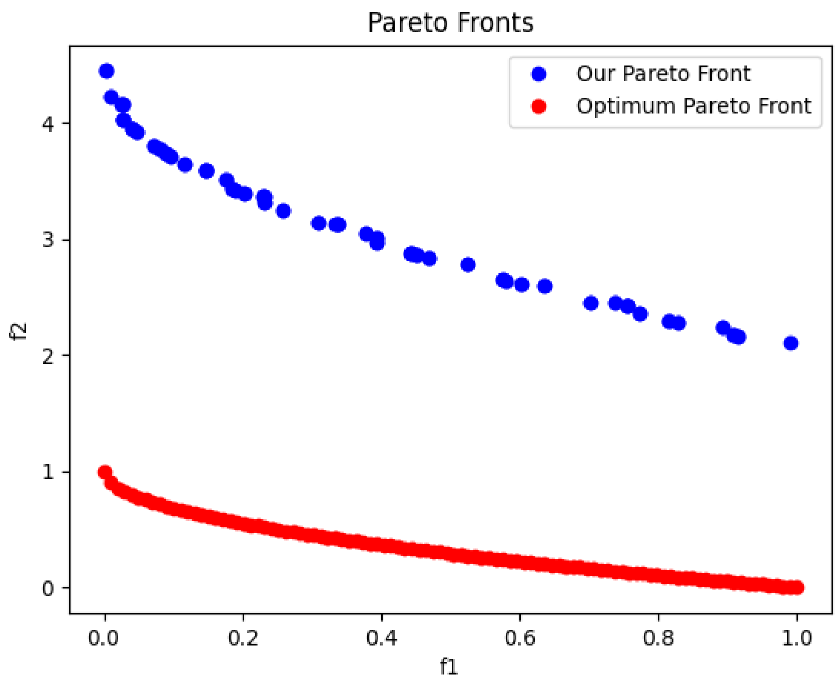


Figure 5.9: Pareto front of our own algorithm versus of the optimum solution for the ZDT1 problem.

matrix that is needed for the ACO. Furthermore, we have implemented different crossover and mutation operators which also affect the outcome.

5.7 Crossover and mutation

Finally, we experiment with the crossover- and the mutation rates. We run this experiment again on the three initialization methods and calculate the mean hyper-volume over ten runs. This metric is calculated by computing the surface area dominated by the Pareto front bounded by a reference point R. The reference point contains the maximum values of the objectives. A large hypervolume indicates that our obtained Pareto front is likely to cover more of the solution space and therefore a better solution quality [13]. We also calculate the standard deviation corresponding to the mean. We then take the average hyper-volume (HV) and the average standard deviation (SD) values of all the crossover- and mutation rates. Furthermore, we run these experiments on the instance RC102. The results are shown in Tables 5.2 and 5.3 for the crossover and mutation respectively.

From the average values for the HV we see that for both the mutation as the crossover experiments, ACO-NSGAII outperforms the other two methods. Furthermore, it appears that ACO-NSGAII yields the best results for crossover rates 0.30 and 0.50 compared to the other methods and for mutation rates 0.7 and 1.0. We see that the other methods benefit from a lower mutation rate and a larger crossover rate. Overall, the best result for ACO-NSGAII is for a crossover rate of 0.5 and a mutation rate of 0.7.

Table 5.2: Average normalized hypervolume and its standard deviation for RC102 with different crossover rates (C_r) for ACO-NSGAII (ACO), NN-NSGAII (NN), and Random-NSGAII (Random).

C_r	HV_{ACO}	SD_{ACO}	HV_{NN}	SD_{NN}	$\text{HV}_{\text{Random}}$	$\text{SD}_{\text{Random}}$
0.30	**0.4728**	0.2575	0.3198	**0.1115**	0.2606	0.2044
0.50	**0.6197**	0.2591	0.2595	**0.1132**	0.1963	0.1726
0.70	0.2162	**0.1676**	**0.4716**	0.2014	0.2593	0.1981
0.95	0.2607	**0.1306**	**0.3374**	0.2215	0.3363	0.2217
Avg	**0.3924**	0.2037	0.3471	**0.1619**	0.2631	0.1992

Table 5.3: Average normalized hypervolume and its standard deviation for RC102 with different mutation rates (M_r) for ACO-NSGAII (ACO), NN-NSGAII (NN), and Random-NSGAII (Random).

M_r	HV_{ACO}	SD_{ACO}	HV_{NN}	SD_{NN}	$\text{HV}_{\text{Random}}$	$\text{SD}_{\text{Random}}$
0.1	0.3122	0.3592	0.3238	**0.3500**	**0.5077**	0.4182
0.3	0.3873	0.4377	0.**4615**	**0.2978**	0.3283	0.3527
0.5	0.2361	0.3136	**0.4594**	0.4017	0.2429	**0.1937**
0.7	**0.5749**	0.3952	0.2136	0.2222	0.1015	**0.1384**
1.0	**0.4495**	0.4481	0.3354	0.2276	0.2331	**0.1923**
Avg	**0.3920**	0.3908	0.3587	0.2999	0.2827	**0.2591**

Chapter 6

Discussion

In this chapter, we give an interpretation of our findings.

6.1 General limitations

For this research, there were some limitations which we will discuss now.

We wanted to use real-world data such that we could get the optimal route that could be applied in the real world. However, we could not find or get access to such data. Therefore we used Solomon's data which immediately brings the problem to an abstraction of reality. Our battery consumption rate and charging rate are also abstractions of reality. This is because we use a linear consumption rate of the battery status and we use charging time that is the same as service time. Furthermore, we do not consider traffic, the weather, capacity load or economic parameters.

Another limitation is that we could not find papers that implemented the same model as ours. Therefore, we were unable to measure our performance against the current papers that try to solve a similar problem of EVRPTW. However, we tried to implement many experiments that show the performance of our methods.

6.2 Interpretations

For the interpretation of the results, we focus on answering the research questions.

6.2.1 Initialization method

The largest contribution to improving NSGAII comes from our initialization method Ant Colony Optimization. Experiments done on the allocation of the budget between the two algorithms show that allocating 50% or less of the budget to ACO results in faster convergence and better solutions. Particularly, the fastest convergence in terms of the number of vehicles occurs in Fig. 5.5 when allocating 50% of the budget for ACO initialization and the same amount for NSGAII. Besides, the results show that ACO converges fast but then gets stuck in a local optimum. By combining ACO with NSGAII, we then yield more optimal results. Furthermore, in Fig. 5.4 the distance of the solutions generated by NSGAII starts at a larger value than the last values of ACO. We believe that this is because of exploration as 20% of the initial population is generated randomly. Another possible reason might be due the fact that because we are now dealing with minimizing two objectives instead of one. We also see that for both the Figs. 5.5 and 5.4 the average results from ACO seem more steady than those from NSGAII. We assume that this is again because of exploration and having to minimize two objectives instead of one.

Next, we did some experiments for which we show how our initialization method performs against two other commonly used methods, Random and Nearest Neighbor. Our method substantially outperforms the nearest neighbor and the random generation method visible from Figs. 5.7 and 5.6. This also becomes apparent from the Pareto front plotted in Fig. 5.8. In all these figures, ACO-NSGAII yields the best results. It is interesting to see that in figure 5.6 ACO-NSGAII performs worse than the NN-NSGAII and the Random-NSGAII during the ACO part of ACO-NSGAII. This is because up until generation number 300, ACO-NSGAII only minimizes the distance and ignores the number of vehicles whereas the other two methods already minimize for both objectives. After generation 300, ACO-NSGAII also minimizes on the number of vehicles which is visible from the figure.

However, a trade-off has to be made since ACO-NSGAII has a much longer computational time compared to the other methods. Therefore we can answer the first research question as follows. We improve the NSGA-II algorithm by allocating 50% of the budget to ACO for generating the initial population and by allocating the other 50% of the budget to NSGA-II for generating an optimal route. ACO-NSGAII is then formed for which ACO only optimizes the distance with no further constraints. The number of vehicles and constraints are then implemented and optimized with NSGA-II.

6.2.2 Optimal route

We wanted to optimize the route for EVRPTW and we did this by implementing ACO-NSGAII. From the plots that are conducted in the first experiment 5.3 we believe that our method is successful in generating the optimal route. This is because we managed to find the best route while focusing on minimizing two objectives. Furthermore, we included the latest arrival time window and the battery capacity of the vehicles. Especially the clustered data seems to have an optimal route since the clusters are well handled. However, we also believe that our method can be optimized even further by applying a local search algorithm at the end such that edges do not cross.

6.2.3 Performance ACO-NSGAII

Since we want to measure the quality performance of our algorithm, we experiment with a test problem of PyMOO. However, a large limitation here is that we could not find a test problem with VRP so we could not use ACO as our initialization method. Therefore we used random generation as initialization. Because of this, our solutions were not optimal since the values of the second objective of our Pareto front were a lot larger than the optimal Pareto values. Another reason for this is that we used a crossover and mutation operation that was optimal for VRP but not for the test problem of PyMOO. However, despite this, our Pareto front held the same shape as the optimal Pareto front. This validates that our algorithm is minimizing two objectives correctly.

Another way to measure the performance of ACO-NSGAII is by experimenting with the mutation and crossover rates. This experiment was mainly done to address the large mutation rate in our algorithm. Usually, the mutation rate is kept small, $M_r <= 0.2$ but since we used a 2-opt mutation operation, our algorithm benefits from always performing the mutation. The results of the crossover and mutation rate tests in Tables 5.2 and 5.3 show that crossover rate and mutation rate influence solution qualities regardless of the initialization method. In general, ACO-NSGAII yields better values for the hypervolume than NN-NSGAI and NN-NSGAI yields better values for the hypervolume than Random-NSGAII. Besides, we see that for the crossover, larger hypervolume values are retrieved for lower crossover rates for ACO-NSGAII and that larger hypervolume values are retrieved for higher mutation rates for ACO-NSGAII. For NN-NSGAII and Random-NSGAII, a larger crossover rate yields better results. Specifically for ACO-NSGAII, using a crossover rate of 0.5 and a mutation rate of 0.7 yields the best results.

Pareto front

Finally, we did an experiment to show the trade-off between our two objectives and whether they are in conflict with each other. From the Pareto fronts in Fig. 5.8, we can see, once again, that ACO-NSGAII outperforms the NN- and Random-NSGAII substantially. This validates the quality of our algorithm. Furthermore, we can also observe the trade-offs between travel distance and the number of vehicles. For example, when using 18 vehicles instead of 17, the system-wide travel distance can be reduced by up to 200 according to the solutions of ACO-NSGAII in Fig. 5.8.

Chapter 7

Conclusion

The goal of this research was to improve NSGA-II by changing the initialization with ACO to create an optimal route for EVRPTW while maintaining good quality and computational time.

We have improved NSGA-II with ACO by allocating 50% of the budget to ACO and the remaining part to NSGA-II. Furthermore, ACO only creates an initial population by focusing on minimizing the distance. NSGA-II then applies constraints and the second objective. This algorithm creates feasible routes which provide much better results than other initialization methods such as random search and nearest neighbor. Finally, results show that our method yields good results in terms of solution quality since our Pareto front outperformed those of Random-NSGAII and NN-NSGAII. The results from PyMOO showed that despite our Pareto front being worse than the optimal solution, its shape validates that the implementation is correct. However, in terms of computational times, ACO-NSGAII performs poorly. NN-NSGAII and Random-NSGAII are a lot faster. The NN-NSGAII is 4.41 times faster than ACO-NSGAII and the Random-NSGAII is 4.50 times faster.

Furthermore, we see that the distance for ACO-NSGAII is minimized with 18% compared to Random-NSGAII and its number of vehicles is minimized with 6.0% compared to Random-NSGAII. Besides, our Pareto fronts show that the use of one more vehicle in delivery could potentially reduce travel distances by up to 14%. Other experiments done on the crossover and mutation rates show that a crossover of 0.5 with a mutation rate of 0.7 yield the best results for ACO-NSGAII. Overall, we believe that we successfully implemented a novel algorithm, ACO-NSGAII, that yields good-quality results for the bi-objective EVRPTW.

In the future, we want to apply ACO-NSGAII on the same models of current papers for a real comparison. We also want to optimize ACO-NSGAII further by focusing on the crossover and mutation. Furthermore, we want to test our methods on real data of a real electric bus company so with more constraints such as temperature, charging rates, traffic, and financial rates.

Bibliography

[1] Mohammed Alhanjouri and Belal Alfarra. "Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem". In: *International Journal of Computer Technology and Applications* 2 (June 2011), pp. 570–578.

[2] Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. "Green vehicle routing problem: A state-of-the-art review". In: *International Journal of Production Economics* 231 (2021), p. 107899. ISSN: 0925-5273. DOI: `https://doi.org/10.1016/j.ijpe.2020.107899`. URL: `https://www.sciencedirect.com/science/article/pii/S0925527320302607`.

[3] Narayan Behera. "Chapter 8 - Analysis of microarray gene expression data using information theory and stochastic algorithm". In: *Principles and Methods for Data Science*. Ed. by Arni S.R. Srinivasa Rao and C.R. Rao. Vol. 43. Handbook of Statistics. Elsevier, 2020, pp. 349–378. DOI: `https://doi.org/10.1016/bs.host.2020.02.002`. URL: `https://www.sciencedirect.com/science/article/pii/S0169716120300249`.

[4] J. Blank and K. Deb. "pymoo: Multi-Objective Optimization in Python". In: *IEEE Access* 8 (2020), pp. 89497–89509.

[5] Christian Blum. "Ant colony optimization: Introduction and recent trends". In: *Physics of Life Reviews* 2.4 (2005), pp. 353–373. ISSN: 1571-0645. DOI: `https://doi.org/10.1016/j.plrev.2005.10.001`. URL: `https://www.sciencedirect.com/science/article/pii/S1571064505000333`.

[6] Chien-Ming Chen et al. "A Genetic Algorithm for the Waitable Time-Varying Multi-Depot Green Vehicle Routing Problem". In: *Symmetry* 15.1 (2023). ISSN: 2073-8994. DOI: `10.3390/sym15010124`. URL: `https://www.mdpi.com/2073-8994/15/1/124`.

[7] Przemysław Czuba and Dariusz Pierzchała. "Machine Learning methods for solving Vehicle Routing Problems". In: Jan. 2021.

[8] Marco Dorigo. "Optimization, learning and natural algorithms". PhD thesis. Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[9] Mehmet Erdem and Çağrı Koç. "Analysis of electric vehicles in home health care routing problem". In: *Journal of Cleaner Production* 234 (2019), pp. 1471–1483. ISSN: 0959-6526. DOI: `https://doi.org/10.1016/j.jclepro.2019.06.236`. URL: `https://www.sciencedirect.com/science/article/pii/S0959652619322012`.

[10] Yongsheng Fang and Jun Li. "A Review of Tournament Selection in Genetic Programming". In: *Advances in Computation and Intelligence*. Ed. by Zhihua Cai et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 181–192. ISBN: 978-3-642-16493-4.

[11] Keivan Ghoseiri and Seyed Farid Ghannadpour. "Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm". In: *Applied Soft Computing* 10.4 (2010). Optimisation Methods & Applications in Decision-Making Processes, pp. 1096–1107. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2010.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1568494610000773.

[12] Tolunay gocken and M. Yaktubay. "Comparison of Different Clustering Algorithms via Genetic Algorithm for VRPTW". In: *International Journal of Simulation Modelling* 18 (Dec. 2019), pp. 574–585. DOI: 10.2507/IJSIMM18(4)485.

[13] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. "The Hypervolume Indicator: Computational Problems and Algorithms". In: *ACM Comput. Surv.* 54.6 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3453474. URL: https://doi.org/10.1145/3453474.

[14] Muhammad Faisal Ibrahim et al. "Optimised Genetic Algorithm Crossover and Mutation Stage for Vehicle Routing Problem Pick-Up and Delivery with Time Windows". In: *IOP Conference Series: Materials Science and Engineering* 1071 (Feb. 2021), p. 012025. DOI: 10.1088/1757-899X/1071/1/012025.

[15] Baoxiang Li, Shashi Shekhar Jha, and Hoong Chuin Lau. "Route Planning for a Fleet of Electric Vehicles with Waiting Times at Charging Stations". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Arnaud Liefooghe and Luís Paquete. Cham: Springer International Publishing, 2019, pp. 66–82. ISBN: 978-3-030-16711-0.

[16] Chunhui Li, Yanfei Zhu, and Kwang Y. Lee. "Route Optimization of Electric Vehicles Based on Reinsertion Genetic Algorithm". In: *IEEE Transactions on Transportation Electrification* 9.3 (2023), pp. 3753–3768. DOI: 10.1109/TTE.2023.3237964.

[17] Danlian Li et al. "Optimization of Green Fresh Food Logistics with Heterogeneous Fleet Vehicle Route Problem by Improved Genetic Algorithm". In: *Sustainability* 12.5 (2020). ISSN: 2071-1050. DOI: 10.3390/su12051946. URL: https://www.mdpi.com/2071-1050/12/5/1946.

[18] Yongbo Li, Hamed Soleimani, and Mostafa Zohal. "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives". In: *Journal of Cleaner Production* 227 (2019), pp. 1161–1172. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.03.185. URL: https://www.sciencedirect.com/science/article/pii/S0959652619308790.

[19] Qixing Liu et al. "A hybrid genetic algorithm for the electric vehicle routing problem with time windows". In: *Control Theory and Technology* 20 (May 2022). DOI: 10.1007/s11768-022-00091-1.

[20] Setareh Majidi, Seyyed-Mahdi Hosseini-Motlagh, and Joshua Ignatius. "Adaptive large neighborhood search heuristic for pollution-routing problem with simultaneous pickup and delivery". In: *Soft Computing* 22.9 (May 2018), pp. 2851–2865. ISSN: 1433-7479. DOI: 10.1007/s00500-017-2535-5. URL: https://doi.org/10.1007/s00500-017-2535-5.

[21] Behzad Moradi. "The new optimization algorithm for the vehicle routing problem with time windows using multi-objective discrete learnable evolution model". In: *Soft Computing* 24.9 (May 2020), pp. 6741–6769. ISSN: 1433-7479. DOI: 10.1007/s00500-019-04312-9. URL: https://doi.org/10.1007/s00500-019-04312-9.

[22] Motive. *Electric buses. A definitive guide for commercial fleets.* URL: https://gomotive.com/blog/electric-buses-guide-commercial-fleets/#:~:text=A%20plug%2Din%20charger%20can,or%20alternating%20current%20(AC)..

[23] Arjun T Mulloorakam and Nidhish Mathew Nidhiry. "Combined Objective Optimization for Vehicle Routing Using Genetic Algorithm". In: *Materials Today: Proceedings* 11 (2019). International Multi-Conference on Computing, Communication, Electrical & Nanotechnology (I2CN-2K'18), pp. 891–902. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2018.12.016. URL: https://www.sciencedirect.com/science/article/pii/S2214785318329316.

[24] Myriam Neaimeh et al. "Routing systems to extend the driving range of electric vehicles". In: *IET Intelligent Transport Systems* 7.3 (2013), pp. 327–336. DOI: https://doi.org/10.1049/iet-its.2013.0122. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-its.2013.0122. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-its.2013.0122.

[25] Beatrice Ombuki, Brian J. Ross, and Franklin Hanshar. "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows". In: *Applied Intelligence* 24.1 (2006), pp. 17–30. ISSN: 1573-7497. DOI: 10.1007/s10489-006-6926-z. URL: https://doi.org/10.1007/s10489-006-6926-z.

[26] Bochra Rabbouch, Foued Saâdaoui, and Mraihi Rafaa. "Efficient Implementation of the Genetic Algorithm to Solve Rich Vehicle Routing Problems". In: *Operational Research* (Sept. 2021). DOI: 10.1007/s12351-019-00521-0.

[27] Brandon Reese and Yan Xu. *Improving the state-of-the-art algorithm for vehicle routing problems.* https://blogs.sas.com/content/subconsciousmusings/2023/03/30/improving-the-state-of-the-art-algorithm-for-vehicle-routing-problems/. [Online; accessed 11-10-2023]. 2023.

[28] Efrain Ruiz et al. "Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm". In: *Computers & Industrial Engineering* 133 (2019), pp. 207–219. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.05.002. URL: https://www.sciencedirect.com/science/article/pii/S0360835219302694.

[29] Ahmad Sayed Saif-Eddine, Mohammed Mostafa El-Beheiry, and Amin Kamel El-Kharbotly. "An improved genetic algorithm for optimizing total supply chain cost in inventory location routing problem". In: *Ain Shams Engineering Journal* 10.1 (2019), pp. 63–76. ISSN: 2090-4479. DOI: https://doi.org/10.1016/j.asej.2018.09.002. URL: https://www.sciencedirect.com/science/article/pii/S2090447918300935.

[30] Shabnam Sangwan. "Literature Review on Genetic Algorithm". In: *International Journal of Research* 5 (June 2018), p. 1142.

[31] S.s Shariff, Noor Moin, and Mohd bin Omar. "An alternative heuristic for capacitated p-median problem (CPMP)". In: Apr. 2013, pp. 916–921. ISBN: 978-1-4673-5967-2. DOI: 10.1109/BEIAC.2013.6560271.

[32] S.N. Sivanandam and S.N. Deepa. *Introduction to Genetic Algorithms*. Springer Berlin, Heidelberg, 9October 2007.

[33] M. Solomon. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints". In: *Operations Research* 35.2 (1987), pp. 254–265. DOI: 10.1287/opre.35.2.254.

[34] Fei Tan, Zheng-yi Chai, and Ya-lun Li. "Multi-objective evolutionary algorithm for vehicle routing problem with time window under uncertainty". In: *Evolutionary Intelligence* 16.2 (2023), pp. 493–508. ISSN: 1864-5917. DOI: 10.1007/s12065-021-00672-0. URL: https://doi.org/10.1007/s12065-021-00672-0.

[35] Shi-Yi Tan and Wei-Chang Yeh. "The Vehicle Routing Problem: State-of-the-Art Classification and Review". In: *Applied Sciences* 11.21 (2021). ISSN: 2076-3417. DOI: 10.3390/app112110295. URL: https://www.mdpi.com/2076-3417/11/21/10295.

[36] Shanu Verma, Millie Pant, and Vaclav Snasel. "A Comprehensive Review on NSGA-II for Multi-Objective Combinatorial Optimization Problems". In: *IEEE Access* 9 (2021), pp. 57757–57791. DOI: 10.1109/ACCESS.2021.3070634.

[37] Feng Wang et al. "External archive matching strategy for MOEA/D". In: *Soft Computing* 22 (Dec. 2018). DOI: 10.1007/s00500-018-3499-9.

[38] Yong Wang et al. "A clustering-based extended genetic algorithm for the multi-depot vehicle routing problem with time windows and three-dimensional loading constraints". In: *Applied Soft Computing* 133 (2023), p. 109922. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.109922. URL: https://www.sciencedirect.com/science/article/pii/S1568494622009711.

[39] Yanfei Zhu, Kwang Lee, and Yonghua Wang. "Adaptive Elitist Genetic Algorithm With Improved Neighbor Routing Initialization for Electric Vehicle Routing Problems". In: *IEEE Access* PP (Jan. 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3053285.

[40] Yanfei Zhu, Kwang Y. Lee, and Yonghua Wang. "Adaptive Elitist Genetic Algorithm With Improved Neighbor Routing Initialization for Electric Vehicle Routing Problems". In: *IEEE Access* 9 (2021), pp. 16661–16671. DOI: 10.1109/ACCESS.2021.3053285.

[41] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results". In: *Evolutionary Computation* 8.2 (2000), pp. 173–195. DOI: 10.1162/106365600568202.

[42] Ferani E. Zulvia, R.J. Kuo, and Dwiyanti Y. Nugroho. "A many-objective gradient evolution algorithm for solving a green vehicle routing problem with time windows and time dependency for perishable products". In: *Journal of Cleaner Production* 242 (2020), p. 118428. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.118428. URL: https://www.sciencedirect.com/science/article/pii/S0959652619332986.