Technical University of Crete

School of Production Engineering and Management

Diploma Thesis

Swarm Optimization Algorithm for the Electric Vehicle Routing Problem

Student:

Panagiotis Gkatzolas

Supervising Professor:

Professor Ioannis Marinakis

Chania, 2025

Acknowledgements

First of all, I would like to thank my supervisor, Professor Ioannis Marinakis, for his valuable guidance, scientific advice and continuous support throughout my research career. His knowledge and experience have been invaluable in shaping this thesis and have contributed significantly to the improvement of my research work. Finally, I thank my family and friends for their unwavering support at every stage of this endeavor.

Abstract

The transition to electric mobility is one of the biggest technological and environmental challenges in transportation today. However, the limited battery autonomy of electric vehicles (EVs) and the need for efficient route planning makes it essential to optimize their routes so that energy consumption is minimized and operational costs are reduced.

In this study, we examine the Electric Vehicle Routing Problem (E-VRP), an extension of the classic Vehicle Routing Problem (VRP), which considers additional constraints such as energy consumption, charging station availability, and vehicle capacity. Our goal is to develop an algorithm that will compute optimal routes for a fleet of electric vehicles while minimizing both travel distance and energy consumption.

To solve this problem, we use the Particle Swarm Optimization (PSO) algorithm, which is inspired by the collective movement of particle swarms in nature. PSO is a widely used optimization algorithm because it does not require overly complex mathematical calculations and can find high-quality solutions in a relatively short time. Moreover, it converges quickly and retains information from previous high-performing solutions, improving its search efficiency in routing problems. In our study, we adapt the PSO to consider key factors such as:

- The total distance traveled by EVs.
- Energy consumption, which depends on vehicle load and route characteristics.
- Charging station availability and congestion levels.
- Delivery time constraints.

To evaluate our method, we will implement the PSO and we will test it on benchmark instances from the literature. We will assess the quality of the solutions and computational time, and the results will indicate if the PSO can provide efficient solutions for the E-VRP, optimizing routing while keeping the computational cost relatively low.

Contents

1.1 Introduction

The Electric Capacitated Vehicle Routing Problem (E-CVRP) has emerged as an important research area in the recent years, due to the complex requirements it places on fleet management and route planning. Electric vehicles (EVs) are a key element in the transition to more sustainable transport systems, as they reduce CO_2 emissions and reduce dependence on fossil fuels. However, they present significant limitations, mainly due to their limited autonomy and the need for regular charging, which creates additional requirements for optimal route planning.

This paper aims to develop and implement a Particle Swarm Optimization (PSO) algorithm for solving E-CVRP. PSO is an evolutionary optimization algorithm, inspired by the collective behavior of swarms in nature, such as flocks of birds or schools of fish. This algorithm is widely used because it does not require complex mathematical calculations and can find qualitative solutions in a relatively short time.

The contribution of the work lies in adapting the PSO to take into account critical factors such as energy consumption, vehicle load, loading constraints, and time constraints, making it suitable for E-CVRP.

The methodology followed includes the development of the algorithm, its application to known datasets (benchmark instances) from the literature, and the comparison of the results with other methods, in order to evaluate the quality of the solutions and the computational time.

2. Theoretical Background

2.1 The vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is one of the most studied problems in the fields of Combinatorial Optimization and Operations Research. The goal is to determine the optimal set of routes for a fleet of vehicles, which start from one or more distribution centers and serve a set of geographically distribution customers with known demand. Each customer must be served exactly once, while each vehicle had a limited capacity.

The VRP generalizes the classic Traveling Salesman Problem (TSP), as it extends the study from one to multiple vehicles and from one to multiple destinations. It was first announced by Dantzig and Ramser in 1959 [2], while one of the first and most well known heuristic algorithms for its solution is the " Savings " algorithm by Clarke and Wright (1964) [3].

The problem is classified as NP-Hard, which means that finding the exact optimal solution is computationally infeasible for large problems. For this reason, many heuristic and meta-heuristic methods have been developed that aim to find good approximate solutions in reasonable time.

Many variations of VRP have been proposed to cover practical constraints that arise in real applications. Some examples are:

- Capacitated VRP (CVRP) : The vehicles have limited cargo capacity
- VRP with Time Windows (VRPTW) : Customers must be served within specific time windows.
- Open VRP (OVRP) : Vehicles do not necessarily return to the starting depot.
- Electric VRP (E-VRP) : The problem contains constraints related to autonomy and charging needs of an electric vehicle.

Solving the VRP generally has direct practical applications in areas such as product distribution, supply chain management, waste collection, home delivery services and many more [4].

2.2 The Electric Vehicle Routing Problem (EVRP)

The electric vehicle routing problem is an extension of the classic vehicle routing problem, taking into account constrains that relate with the use of electric vehicles (EVs). The specificities of EVs, such as limited autonomy due to battery capacity and the need for recharging, introduce new challenges to route planning [5].

In E-VRP, in addition to the traditional VRP constraints, additional factors are added such as:

- Battery State of Charge (SoC): The remaining energy in each vehicle must be monitored throughout the route.
- Charging Stations : the location and availability of charging stations affect route planning.
- Charging time : The time required for recharging can vary depending on the type of station an the remaining battery energy.
- Partial charging: In some cases, vehicles may choose to partially charge to save time, which affects the overall design [6].

These Factors make E-CRP more complex than traditional VRP. To address these challenges, several variations of E-VRP have been proposed, including:

- E-VRP with time windows (E-VRPTW) : Customers must be served within specific time intervals
- E-VRP with partial charging: Partial recharging of the battery is allowed at charging stations
- E-VRP with battery swapping stations : Instead of charging, the batteries are replaced with the fully charged ones. [7]

Solving E-VRP requires the development of advanced algorithms and models that take into account both the traditional VRP limitations and the specificities of EVs. The methods used include exact algorithms, heuristic and meta-heuristic approaches, as well as hybrid models that combine different techniques.

2.3 Metaheuristic Algorithms

Metaheuristic Algorithms are considered a powerful computational tool for solving complicated optimization problems, such as Vehicle Routing Problem (VRP), Traveling Salesman Problem (TSP) and many more. These algorithms are search strategies that do not guarantee finding the global best solution, but instead they try to find a high quality solution within reasonable computational time.

Metaheuristic methods have the advantage that they can easily adapt to different problems, and incorporate complex constrains, like time windows, shipping cargo or energy requirements. This makes them quite suitable for problems like E-VRP [8].

Well known categories of metaheuristic algorithms are :

- Evolutionary Algorithms : Similar to Genetic algorithms which are based on selection , mutation mechanism and crossover.
- Swarm Intelligence Algorithms : Algorithms like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) , which are inspired by the behavior of natural organisms. They follow collective paths like insects, birds, and other animals where individuals act to achieve greater goals.
- Local Search, Tabu Search, Simulated Annealing : Methodologies that are based on finding neighbor-like solutions and avoid local best solutions.
- Hybrid Metaheuristics : They combine different technics for improved performance and flexibility.

Choosing the right method depends on the nature of the problem, the size of the data base , the constrains , and the computational time available. Several metaheuristic approaches have been proposed to solve the E-VRP with quality results.

Metaheuristic algorithms attract the interest of the scientific community, as they present significant advantages and characteristics that make them suitable for solving problems of high complexity. Many different techniques have been developed, providing solutions to various types of optimization problems. [9] Depending on the behavior and the principle on which their design is based, metaheuristic algorithms are classified into four main categories: algorithms inspired by the theory of evolution, algorithms derived from the observation of

swarm intelligence, methods based on natural phenomena, as well as algorithms that imitate elements of human behavior.

1. Evolutionary-inspired algorithms: These techniques draw inspiration from the mechanisms of natural selection and biological evolution. The process begins with the creation of an initial set of candidate solutions, which gradually evolved through processes such as crossover, mutation, and selection of the most suitable individuals. Among the most representative algorithms in this category are Genetic Algorithms (GA), which are based on the theory of Darwinian evolution. The same category also includes Evolutionary Strategy, Genetic Programming ,Tabu Search, Differential Evolution and other approaches.
2. Swarm-based algorithms: These algorithms are inspired by the collective behavior and social interaction of species such as insects, birds, and fish. One of the most popular examples is Particle Swarm Optimization (PSO), developed by Kennedy and Eberhart, which simulates the dynamics of a flock of birds foraging. Other well-known methods in this category include Ant Colony Optimization, Bee Swarm Optimization, and the Monkey Optimization approach.
3. Natural phenomena-based algorithms: These methods are derived from the observation of natural processes and laws of the natural world. Examples include Simulated Annealing and Harmony Search.
4. Algorithms inspired by human behavior: Algorithms in this group are inspired by the way humans learn, collaborate, and interact. Examples include the Teaching-Learning Based Optimization (TLBO) and the Championship Algorithm.

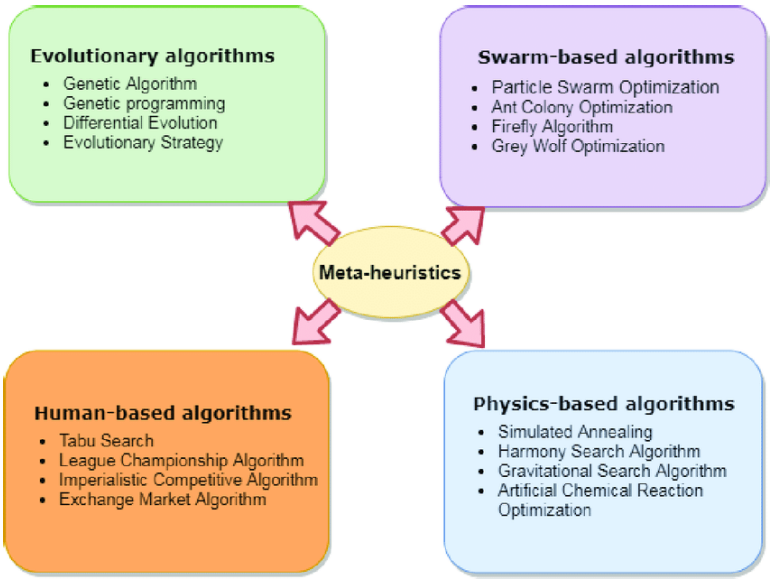

Figure 1: Metaheuristic algorithms [14]

2.4 Particle Swarm Optimization – PSO

Particle Swarm Optimization (PSO) is considered one of the most popular metaheuristic techniques for solving optimization problems, especially in cases that classical methods fail due to complex computations or lack of a clear math model. Inspired by the collective behavior of organisms in nature, like bird and fish swarms, the algorithm is based on teamwork and empirical learning between the "particles" of the population.

Each particle of the swarm represents one possible solution of the problem and moves to the search space following a simple mechanisms based on personal experience and knowledge of the swarm . The position and the velocity of each particle are updated frequently according to :

- The best position visited by the particle itself (Personal best – PBEST)
- The best position found by the swarm as a whole (Global best – GBEST)

With this mechanism, the swarm gradually converges to regions of space with high objective function values, without the need of derivatives or precise calculations.

The PSO algorithm stands out for its simplicity if implementation, small number of parameters, and the ability to perform really well in complex non linear problems. In the recent years, it has been successfully adapted to various applications, from routing and energy management problems to neural network optimization and robotics.

In the case of the E-VRP, the use of PSO allows the exploration of huge space of possible paths, taking into account parameters such as cargo load, energy consumption, and loading constraints. The algorithm can easily incorporate validity checks to ensure that each generated solution is true based on the data of the problem.

PSO is an algorithm capable of optimizing nonlinear and multidimensional problems, usually reaching satisfactory solutions in an efficient manner and with minimal parameterization. The algorithm and its concept of "Particle Swarm Optimization" were introduced by James Kennedy and Russel Eberhart in 1995 [10]. However, its roots are based on earlier attempts to model the behavior of animals in nature, such as flocks of birds or schools of fish, making nature its source of inspiration. These biological references lead to the categorization of PSO as a Swarm Intelligence and Artificial Life algorithm. The central idea of the algorithm is to

create a swarm of particles that move in the problem space, searching for the optimal position based on a fitness function. Analogy with nature: A flock of birds flies in its environment looking for the best resting place (with criteria such as space, accessibility to food, water, etc.).

Basic Ideas of Optimization

Social Learning: Each particle evaluates its position and benefits not only from its own exploration, but also from the information shared by the other particles [11].

Stochastic Factor: The speed of each particle includes a random component, which ensures the exploration of new regions of the space. This property, combined with a good initial distribution of the swarm, increases the chances of finding the optimal solution.

PSO Algorithm

Kennedy and Eberhart were the first to propose a method for solving complex nonlinear optimization problems, drawing inspiration from the way flocks of birds move and act. From this observation, they formulated the idea of optimizing functions using a swarm of particles. Suppose that we are seeking to find the global optimum of a multidimensional function (n dimensions) defined by $f(x_1, x_2, x_3, \ldots, x_n) = f(x)$, where the search point is described by the set of free variables of the function. The goal is to determine a point in the search space at which the value of the function is either maximum or minimum.

Consider the functions given by $f_1 = x_1^2 + x_2^2$ and

$$f_2 = x_1 * sin(4\pi * x_2) - x_{2*} sin(4\pi * x_1 + \pi) + 1$$

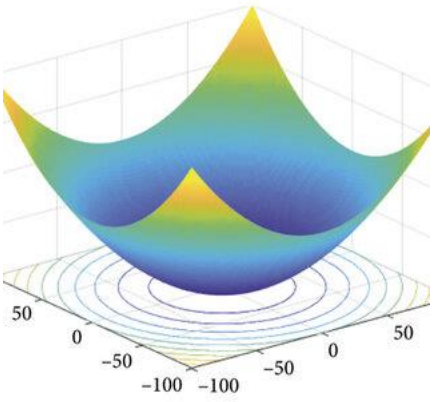

Figure 2: A typical unimodal optimization function [12]

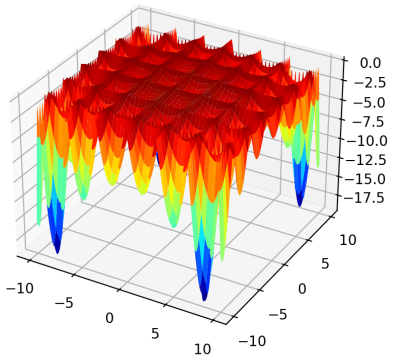

Figure 3: A multimodal optimization function [13]

Figure 2 shows a function that is unimodal, that is, it has a single minimum point at(x_1,x_2)=(0,0) ,which is located in the center of the search space (at the origin of the axes). On the other hand, Figure 3 depicts a function with multiple peaks and valleys (multimodal), which makes the process of finding the global optimum more difficult, as there are many local minima. For this reason, it is necessary for the agents (or particles) to start from different initial positions and continue to explore the search space until at least one of them reaches the global optimum. During this process, the particles exchange information and experiences with each other.

This work focuses on solving problems related to multi-peak functions. The Particle Swarm Optimization (PSO) algorithm is a parallel search technique based on many agents, which form a swarm. Each particle represents a possible solution and moves through a multidimensional search space, adjusting its position based on its own experience and that of neighboring particles. Let the position vector of a particle in the multidimensional search space at time step t be represented by x(t). The position of each particle within this space is then adjusted at each time step according to the following update rule

$$X_i^{t+1} = x_i^t + V_i^{t+1} with\ X_i^0\ U(Xmin, Xmax)$$

where, vit is the velocity vector of particle that drives the optimization process

U(X_{min},X_{max}) is the uniform distribution where X_{min} and X_{max} are its minimum and maximum values respectively

The initial positions and velocities of the particles are chosen randomly and each particle is evaluated to determine its best personal position (pbest) as well as the best overall position of the swarm (gbest). An iterative process then begins, during which the velocities and positions of the particles are continuously updated, until a predetermined termination criterion is satisfied.

Personal Best-Global Best

In the context of Particle Swarm Optimization (PSO), the combined action of the personal best (Personal Best – pBest) and the global best (Global Best – gBest) largely determines the course of convergence of the swarm towards optimal solutions. pBest represents the position with the best value of the objective function that a specific particle has managed to visit from the beginning of the search up to the present stage. In other words, it constitutes the personal memory of each particle and contributes decisively to the utilization of its individual experience to improve its course. On the other hand, gBest expresses the position with the lowest (or best, depending on the problem) value of the objective function that has been identified by any particle in the entire swarm up to the given time. gBest constitutes the collective model towards which all particles tend to move, enhancing group cooperation and simulating mechanisms of social learning and information exchange.

The main difference between the two lies in their nature and operation: pBest is a local guide that relies exclusively on the knowledge and experiences of the particle itself, promoting the exploitation of regions that each particle considers promising based on its personal history. gBest, on the other hand, is a global guide that reflects the overall performance of the swarm, reinforcing the tendency to concentrate around regions of the search space where the overall experience of the population has demonstrated high quality solutions. At a mathematical level, their contribution can be seen in the velocity update equation:

$$V_i * (t+1) = \mathrm{W}*V_i(\mathrm{t}) + C_1 * r_1 * (pBest_i - x_i(\mathrm{t})) + C_2 * r_2 * (gBest_i - x_i(\mathrm{t}))$$

where the cognitive component $C_1 * r_1 * (pBest_i - \mathrm{xi})$ directs the particle towards its own best position, while the social component $C_2 * r_2 * (gBest_i - x_i(\mathrm{t}))$ pushes it towards the best position recorded by the swarm.

The balance between pBest and gBest is critical to the performance of the algorithm, as it regulates how much the swarm will emphasize local exploration versus global exploitation. A

swarm that is too guided by gBest may get trapped in local minima, while a swarm that relies too heavily on pBest may have slow convergence due to excessive diffusion in the search space. The successful design of a PSO lies in achieving this dynamic balance, so as to combine individual creativity with the collective wisdom of the swarm.

3. Modeling and Implementation

3.1 Problem Definition – E-CVRP

In this paper, E-CVRP is formulated as a problem of finding optimal routes for a fleet of electric vehicles, which start and return in the same depot, serving all the customers exactly once. Each route has to follow constraints about load and energy capacity, while charging is available through specific stations. The goal is to minimize the total cost without violating the constraints of the problem.

To find the solution of the problem, we use a metaheuristic approach, based on the Particle Swarm Optimization algorithm, which evaluates and optimizes possible solutions.

The data set of the problem includes coordinates of the customers, the depot, and the charging stations. Also it contains each customer's demands , energy consumption of each vehicle, and the available charging infrastructure. This model takes into consideration load limits of each vehicle, battery autonomy, energy consumption based on the remaining cargo of each vehicle, and the operational feasibility of each proposed route.

The next section describes in detail the assumptions and constraints considered, as well as the objective function and implementation details of the solution method.

3.2 Assumptions and Constraints

To solve the E-CVRP problem in this paper, we followed basic assumptions and constraints to minimize errors and represent the problem as realistic as possible. The problem formulation relies on the following principles :

- All electric vehicles start and return to the same exact depot.
- Each customer is served exactly one time, by one vehicle.
- Each vehicle has limited load, which it is not allowed to exceed on any trip.
- The energy consumption of each vehicle depends on the distance it travels and the cargo it carries.
- The battery autonomy is finite and each vehicle must charge at the predefined charging stations whenever needed.
- Charging stations have unlimited availability, and the is no waiting time
- We assume that charging at stations is instantaneous and always results in a full battery recharge.

In the context of this paper, as noted earlier, the time factor is not taken into account, neither as travel time nor as charging time at the charging stations. The reason for this choice is that the main focus of the model is to minimize the cost (or distance), and satisfy the load and energy constraints. Incorporating time would significantly increase the complexity of the algorithm, and would introduce a new problem type , the Electric Capacitated Vehicle Routing Problem with Time Windows (E-CVRPTW).

3.3 Mathematical Formulation of the Problem

In this study, the Electric Capacitated Vehicle Routing Problem (E-CVRP) is formulated as an optimization problem that aims to minimize the total cost of the vehicle routes, while all the necessary constraints are satisfied . The mathematical formulation presented here is adapted to suit the metaheuristic approach (PSO) applied in this study. A more detailed MILP formulation can be found in [16]. The model is expressed as follows :

Objective function:

$$\min\left(\sum_{(i,j)\in E} d_{ij} \times x_{ij} + penalties\right)$$

Where:

- d_{ij} represents the Euclidean distance between nodes i and j :

$$d_{ij} = \sqrt[2]{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$$

- x_{ij} is a binary decision variable equal to 1 if a vehicle travels from node i to node j, otherwise is 0
- Penalties account for violation of constraints

Constraints:

- Customer Service Constraint

  Each customer must be visited exactly once:

$$\sum_{j} x_{ij} = 1\,, \quad \forall i \in C$$

  Where C is the set of customers.

- Capacity constraint

  The load on each vehicle must not exceed its maximum capacity:

$$\sum_{i\in R_k} q_i \leq Q\,, \quad \forall k$$

  Where:

1. R_k is the route of the vehicle k,

2. Qi is the demand of the customer I,
3. Q is the maximum load capacity.

- Energy constraint

  The energy consumed on a route must not exceed the battery capacity:

$$\sum_{edges\ in\ route} energy_{ij} \leq E_{max}$$

  With:

$$energy_{ij} = d_{ij} \times \left(r + \frac{L}{Q}\right)$$

  Where:

  1. r is the base consumption rate,
  2. L is the load carried at the time of travel
  3. Q is he maximum load capacity
  4. E_{max} is the maximum battery capacity

- Return Feasibility

  The remaining energy after a delivery must be enough for the vehicle to reach either the depot or a charging station:

$$E_{remaining} \geq \min(E_{to\ depot}, E_{to\ station})$$

  Where :

$$E_{to\ depot} = d(i,0) \times (r + \frac{L_{after}}{Q})$$

$$E_{to\ station} = \min\left\{d(i,s) \times \left(r + \frac{L_{after}}{Q}\right)\right\}, \forall s \in S$$

  S is the set of charging stations.

- Vehicle Limit

  The number of vehicles used must not exceed the allowed maximum:

$$|V| \leq V_{max}$$

- Flow constraints

  Each vehicle starts and ends at the depot:

$$\sum_{j} x_{0j}^{k} = 1, \forall k$$

$$\sum_i x_{i0}^k = 1, \forall k$$

Where 0 denoted the depot.

- Penalty Function

  Penalties are applied for constraint violations:

  $$Penalties = P_{missing} \times |C_{missing}| + P_{dublicate} \times N_{duplicate} + P_{vehicles} \times \max(0, |V| - V_{max})$$

  Where:

  - $P_{missing}$: Penalty for non visited customers, $P_{missing} = 10.000$
  - $C_{missing}$: Number of non visited customers
  - $P_{dublicate}$: Penalty for double visits, $P_{dublicate} = 5.000$
  - $N_{dublicate}$: Number of double visits
  - $P_{vehicle}$: penalty for exceeding the number of vehicles, $P_{vehicle} = 50.000$
  - V : Number of vehicles used
  - V_{max} : Maximum number of vehicles allowed
- Energy consumption h_i

  $$h_i = r + \frac{u_i}{C}$$

  Where:

  - r is the base energy consumption when the vehicle is empty
  - u_i is the load of the vehicle upon arrival at node i
  - and C is the vehicle's capacity

In summary, the mathematical model formulated in this section provides a complete representation of the Electric Capacitated Vehicle Routing Problem as we dealt with in this paper. The objective function aims to minimize the total travelling cost, while penalties ensure that constraint violations are appropriately taken into account.

All the constrains described in section 3.2 are incorporated either through mathematical expressions or as model assumptions. Assumptions such as the unlimited availability of charging stations, instantaneous charging, and the absence of time-related factors are considered with in the formulation. This formulation serves as the basis for the implementation of the PSO algorithm described in the following sections.

3.4 PSO Application Overview

The key design decision was to move from the continuous space of PSO to the discrete space required by E-CVRP. In the implementation, each particle is represented as a continuous vector, which is converted into a discrete sequence of customers (permutation) by sorting the vector coordinates. This choice allows the exploitation of the dynamics of PSO without requiring a complete redesign of the algorithm for combinatorial problems.

The fitness function combines the cost of the route (total distance) with penalties imposed in cases of violation of constraints, as mentioned above in 3.3 section.

An important element of the implementation was the integration of solution validity control mechanisms through functions such as can_reach_after and get_nearest_station. These functions check whether the route can be completed without violating energy constraints and whether charging is done in a way that minimizes the total cost.

In addition, emphasis was placed on the priority of using charging stations instead of directly activating a new vehicle, in order to ensure a more economical solution and avoid an unnecessary increase in the number of vehicles. The above approach was a critical design element for adapting PSO to E-CVRP.

The next figure is based on the general flowchart of the PSO algorithm as described in the literature [14].

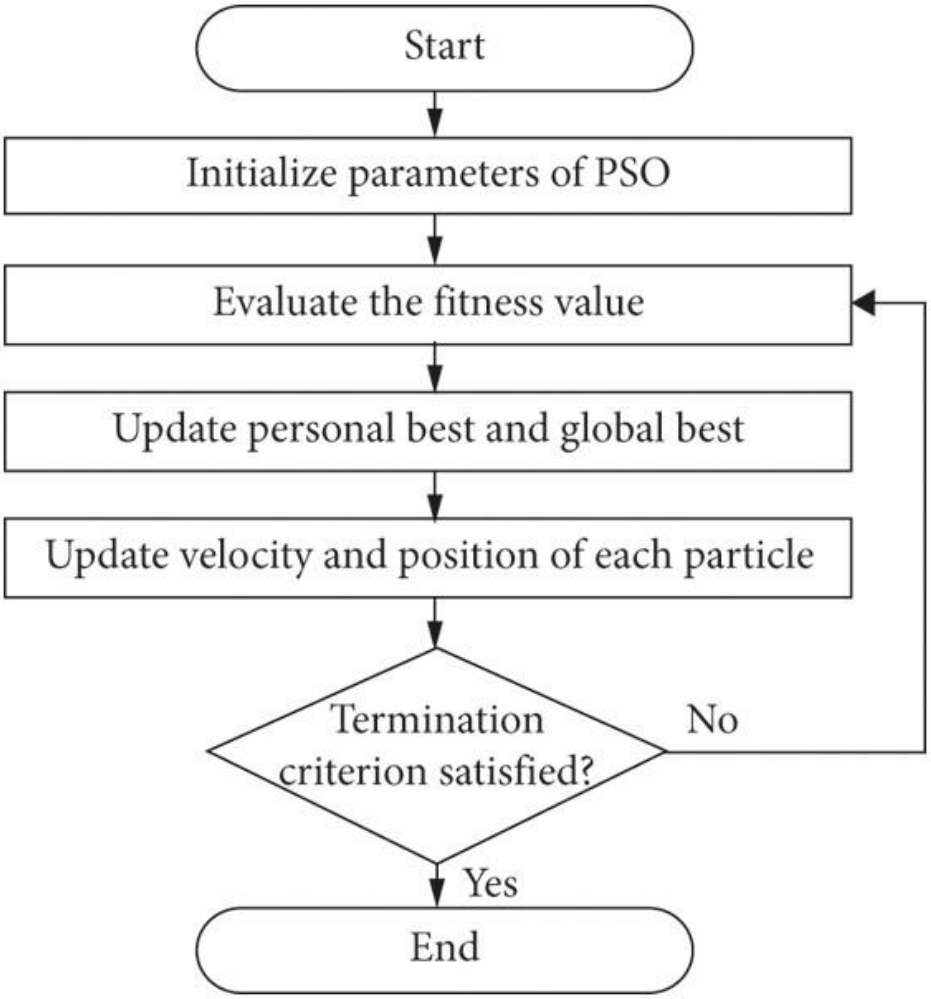

Figure 4: Basic flow of the PSO algorithm.(Source: internet image)

4.Implementation and Detailed Code Explanation

4.1 General Code Structure

The code developed in this work is written in Python and was designed to solve the Electric Capacitated Vehicle Routing Problem (E-CVRP) using the Particle Swarm Optimization (PSO) algorithm. Its general structure is characterized by a clear distinction between data input, computational logic and output.

Python language was chosen because it has several advantages for optimization problems and algorithm development. It has a simple syntax and is easy to use. This helps to write and test the code faster. It contains many functions and libraries, such as math and numpy, that make calculations easier. It can also be easily combined with graphing tools, such as matplotlib, and with other libraries for optimization. Finally, it is a language that is widely used in scientific and academic work, which makes it suitable for this paper.

The code is organized into four main parts:

1. Definition of problem parameters and data. It includes the input of the problem data, such as:

- the coordinates of the nodes (customers and charging stations),
- the load requirements of the customers,
- the characteristics of the vehicles (maximum capacity, maximum energy autonomy, energy consumption coefficient),
- the limit on the number of available vehicles.

Although the data is given directly in the code for demonstration purposes, its structure allows for easy adaptation to read data from standard benchmark files (e.g. .evrp files). In this way, the program can be applied to any E-CVRP problem with minimal changes.

2. Definition of basic support functions. Includes functions that implement basic functions such as:

- calculating the Euclidean distance between two nodes,

- finding the nearest charging station that the vehicle can reach with the remaining energy,
- checking the feasibility of returning to the depot or a charging station after each delivery.

These functions operate in support of the main evaluation function and the main logic of the algorithm.

3. Solution evaluation function (fitness). The fitness function is the basic tool for evaluating the quality of each candidate solution. It combines:

- the total cost of the route (as the sum of distances),
- penalties for constraint violations (such as non-service of customers, double visits, exceeding load, exceeding energy limit, exceeding number of vehicles).

This function is responsible for calculating the objective value used by PSO to guide the search.

4. Implementation of PSO and optimization process. In this part, the main optimization process takes place:

- initialization of particles with random positions,
- conversion of continuous positions into permutations (discrete solutions),
- evaluation of solutions through the fitness function,
- updating pbest and gbest,
- updating speeds and positions,
- repeating the process until the termination criterion is satisfied.

The output of the program includes:

- the best solution found (series of customer visits),
- the total cost of the solution,
- the distribution of routes per vehicle.

In addition, it is verified that all customers have been served exactly once and a report is provided for any deviations.

The code is written in a way that allows for easy modification, so that it can be extended to more complex scenarios, such as E-CVRP with time windows (E-CVRPTW) or with partial loading capability.

4.2 Description of Main Functions

The program we are working on includes a series of main functions that implement the computational form of the algorithm and serve the evaluation of the results, the management of the energy of the vehicles and the proper operation of the PSO. Below we present the basic functions.

Euclidean Distance Calculation

This function calculates the straight line that separates two points in a two-dimensional plane, based on their coordinates. It uses the well-known mathematical formula. The resulting value is used in all calculations related to energy consumption, travel costs and route feasibility checks.

Optimal Charging Station Selection

The purpose of this function is to locate the closest charging station that the vehicle can reach, taking into account the current energy level. For each available station, the energy required to reach it is calculated and the one with the lowest requirement is selected, provided that the route is feasible. This function is crucial for creating reliable routes, as it guarantees that vehicles will not fail due to lack of energy.

Return Check

Here, a check is made whether the vehicle, after a customer visit, has enough energy to return either to the base or to a charging station. The energy requirements for all possible returns are calculated and a positive result is returned only if there is at least one feasible option. This avoids scenarios where the vehicle will move away without the possibility of returning.

Convert Coordinates to Ordered List

Since the PSO algorithm works with continuous values, this function transforms the particle positions into a discrete sequence of customers. The order of service is determined by the order of the values of the vector after sorting. This step is necessary to connect the continuous nature of PSO with the discrete problem of optimal routing.

Cost and Penalty Assessment

The fitness function measures the quality of a solution based on the total cost and imposes penalties if constraints are violated. The main steps include

1. calculating the cost of each route

2. checking the constraints (capacity, energy, number of vehicles, customer service)

3. applying penalties (e.g. a large penalty if a customer is left unserved)

This process guides the algorithm to produce efficient and workable solutions

Limit Values (stay_in_limits)

Ensures that the coordinates of the particles remain within the allowed range [0, 1], avoiding any failures during position updates.

Basic Operation of PSO

PSO (Particle Swarm Optimization) starts by initializing a set of particles, which represent possible solutions to the problem. Each particle has:

• A position (corresponding to a possible solution).

• A velocity (which determines how it will move to new positions).

At each iteration (epoch), the particles move through the search space, updating their positions and velocities based on:

1. Their personal best experience (pbest) – the best position that the particle itself has found so far.

2. The group best experience (gbest) – the best position found by the entire swarm.

The process is repeated until a termination criterion is met, such as:

• A maximum number of iterations is reached.

• A satisfactory solution is found (e.g., minimum cost).

• The improvement is no longer significant.

At each step, the evaluation of the fitness function determines the quality of the solutions, reinforcing the search towards the optimal options.

This iterative process ensures that the swarm converges towards the best possible solution, balancing the exploration of new areas and the exploitation of already known good solutions.

Fitness Function

The evaluation function plays an important role in the PSO algorithm, since it is responsible for determining the quality of the proposed solutions. Specifically, in this study, the fitness function is structured to take into account

1. The total cost of each solution, which is related to distance and energy consumption
2. The penalties imposed in case of violation of the fundamental constraints of the E-CVRP

This dual approach ensures that the algorithm balances finding optimal solutions with respecting the rules of the problem.

Algorithmic Evaluation Process

Input: Accepts a discrete sequence of customer visits, which results from the transformation of the continuous PSO solution vector through the position_to_permutation procedure.

Iterative Node Processing: For each customer in the sequence we have the following steps.

1. Calculation of Route Metrics: Performing a geometric calculation of the distance from the previous distribution node. Determining energy cost based on a) Current load weight b) Movement properties

2. Vehicle Resource Management: Updating energy reserves and available load after each distribution. Making optimization decisions a)Starting a recharge process when it reaches an energy threshold b)switching to a backup vehicle in case of resource depletion

3. Economic Valuation: Cumulative recording of the total cost of the route based on a) energy costs and b) operating costs.

Calculations are done in real time during the decoding of the solution. Vehicle recharging decisions are based on fixed rules of the E-CVRP model, the process exhibits linear complexity (O(n)) in the number of customers

Fitness Function Type

Fitness=Total Distance (Costs)+Violation Points

Total Distance: The sum of all Euclidean distances traveled by the vehicles.

Violation points: Quantitative costs added for each constraint violation, to guide the PSO towards valid solutions.

Energy and charging management

Fitness checks at each step:

1.the vehicle has sufficient energy to reach the next customer.

2.Whether it will be able to return to the depot or a charging station.

3.If not, get_nearest_station is called or a new vehicle is activated.

Role of fitness in PSO

The fitness value of each solution is used to:

Update the particle's pbest, if it is better than the previous one.

Update the swarm's gbest, if it is the best of all.

Guide the movement (speed and position) of the particles in the next iteration.

The use of strict penalties ensures that the algorithm learns to avoid invalid states and pursue feasible, economical paths.

4.3 Implementation and Structure of the Main PSO Process

The Particle Swarm Optimization Algorithm is an efficient and flexible global optimization method based on the collective behavior of the swarm. Modeled after the realistic movement of organisms in nature where they move in groups and formations, PSO utilizes a multitude of particles, which rotate in the search space interacting with each other and improving their position according to their individual experience and that of the others.

Representation and Interpretation of Particles

In the E-CVRP framework, each particle encodes a potential solution to the problem, i.e. a sequence of visits to customers and charging stations. The particle's position is represented as a continuous vector, which is then transformed into a discrete path sequence by the position_to_permutation algorithm. This method allows the application of PSO to combinatorial problems such as E-CVRP, where the exact order of visiting nodes is crucial for the viability and quality of the solution.

4.3.1 Main Stages of PSO Execution

It includes the following stages:

- Initialization : Where we create a population of random particles. Each of them has a speed and initial position and variables that help in recording its personal optimal solution and the best position of the swarm
- Solution Evaluation: Each solution is evaluated using a fitness function. This function combines the total length of the routes with penalties imposed for violating constraints such as exceeding vehicle capacity, excessive energy consumption, and unserved customers. The final score reflects both the efficiency of the solution and its compliance with the rules.
- Velocity and position update: Each particle updates its velocity, based on three factors: 1. Its previous velocity (inertia)
  2. Its distance from its personal best position (local optimization)
  3. Its distance from the overall best position of the swarm (group optimization)

$$Vi^{t+1} = W * Vi^t + C_1 * r_1 * (pbest_i - X_i^t) + C_2 * r_2 * (gbest_i - X_i^t)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

The parameters C_1 and C_2control the degree of influence of personal and social experience respectively, while the random terms r_1 and r_2 ensure the stochasticity of the movement.

Position Limiting

The function [stay_in_limits] guarantees that the values of the particle positions remain within the allowed range [0, 1]. If a position exceeds the limits, it is clamped back into the range.

Updating Optimal Positions

Personal Best Position (pbest): If a particle's new position has a better fitness value than the previous pbest, then pbest is updated with the new position.

Global Best Position (gbest): If the particle's new position is better than the current gbest, then gbest is updated with it.

Termination Check

The algorithm terminates when:

1. It reaches a predetermined number of iterations (generations/max iterations).
2. There is no substantial improvement in the value of gbest for a given number of iterations (e.g., due to convergence).

4.3.2 PSO adaptations for Electric CVRP (E-CVRP)

To apply PSO to the electronic vehicle routing problem, the following basic modifications and extensions were made:

- Solution Decoding: The continuous PSO position vector is converted into a discrete sequence of customers, to correspond to the actual vehicle routes. This process is critical, as it connects our algorithm to the discrete (combinatorial) nature of the problem.
- Custom Fitness Function :The fitness function evaluates each solution based on:
    a) Total transportation cost (minimizing total distance).

b) Energy constraints (ensuring that electric vehicles do not run out of energy).
c) Vehicle capacity (not exceeding the load limit).
d) Optimal use of charging stations (selecting appropriate recharging points)
e) Penalties for invalid routes (e.g. violating constraints).

- Energy Management: The algorithm includes checks to determine whether a vehicle can return to a depot or reach the nearest station with remaining energy (can_reach_after), as well as intelligent station selection via [get_nearest_station]
- Adding Penalties: Penalties are added for each violation, e.g. for non-visits, double visits, or exceeding the number of vehicles, enhancing the stability and reliability of the search.

4.3.3 Benefits of the PSO Approach for E-CVRP

The application of PSO (Particle Swarm Optimization) to the electric vehicle routing problem (E-CVRP) offers significant advantages over classical methods:

Simplicity & Easy Implementation

- Easy parameterization (number of particles, inertia coefficients, etc.).
- Does not require complex mathematical calculations (e.g. derivatives, Hessian matrices).
- More straightforward implementation compared to other metaheuristics (GA, ACO).

Fast Convergence to Acceptable Solutions

- Due to swarm intelligence, particles quickly converge to good solutions, even on large problem instances.

Avoiding local optima through the balance between: exploration (searching for new areas) and exploitation (improving existing solutions).

Flexibility & Adaptation to Real-World Constraints

- Easy integration of additional constraints, such as: charging stations (with smart energy management),service time windows, dynamic changes (traffic, vehicle availability)
- Differentiated penalties for each type of violation (capacity, energy, etc.).

No Need for Analytical Cost Model

- Works only with fitness values (does not require mathematical modeling of the problem).
- Ideal for non-linear or non-differentiable problems.
- Automatic adaptation to changes in constraints (e.g. new customers, charging station outage).

Ideal for Combinatorial Problems with Multiple Constraints

- Effectively handles complex conditions: multiple vehicles & customers, energy & capacity constraints, additional objectives (time, cost, emissions minimization).
- Compares favorably to other methods (Genetic Algorithms, Simulated Annealing) in speed and ease of tuning.

4.4 PSO Parameter Selection – Optimization & Performance Analysis

The performance of PSO in E-CVRP depends critically on the tuning parameters that determine the dynamics of particle movement in the search space, namely the ability to explore and exploit (improve existing solutions).The parameters are not arbitrary but are chosen by combining theoretical principles and empirical testing for optimal comparison

1) Number of Particles (num_particles)

The choice of 20 particles was based on a balance between solution variety and computational efficiency without incurring excessive cost.

- ✓ Satisfactory exploration: The solution space is sufficiently covered without excessive overlap
- ✓ Optimal computational burden: Acceptable execution time is maintained, especially for large E-CVRP instances

NUMBER OF PARTICLES	ADVANTAGES	DISADVANTAGES	BEST FOR
≤10	Fast convergence (few iterations).	High risk of being trapped in local minima. Limited variety of solutions.	Small size templates
=20	Balance between exploration/exploitation. Easy setup.	It may require more iterations for optimal convergence	Most E-CVRP snapshots
≥50	Extensive solution space coverage. Reduced risk of entrapment.	Significant increase in execution time. Possible delay in convergence due to excessive dispersion.	Complex problems with 100+ customers

2) Number of Iterations (num_iterations)

The maximum number of iterations was set to 100. This number is considered sufficient to give the algorithm time to improve its solutions without excessively delaying execution. The choice of 100 iterations was based on a balance between solution quality and computational cost:

- ✓ Sufficient time to improve: The algorithm has the opportunity to explore the solution space and optimize the paths.
- ✓ Controlled execution time: Excessive delay is avoided, which is particularly important in real applications.

A larger number probably leads to better quality solutions (possible 5-10% improvement), but with increased cost and significant increase in execution time (linear scaling).

3) Inertia Weight (w)

The value of the inertia weight was chosen as w = 0.5.The choice was based on the balance between exploration and exploitation for discrete problems such as E-CVRP:

- ✓ Moderate inertia: Ensures stable convergence without sacrificing the variety of solutions.
- ✓ Classical value: Widely accepted in the literature for combinatorial problems.

W	IMPACT	RISKS	BEST FOR
W> 0.7	Intense exploration: Particles disperse into new areas.	Unstable convergence, longer time	Initial phases or complex problems.
W=0.5	Balance: Exploration + constant improvement of solutions	May require more iterations	Typical E-CVRP (20-50 customers)
W<0.4	Fast convergence: Particles focus on local optimizations	Early trapping in local minima	Simple problems or final stages of improvement

4) Weight Coefficient Optimization (c_1, c_2) for E-CVRP

Chosen Values: $c_1 = 2.0$, $c_2 = 2.0$

The equal distribution $c_1 = c_2 = 2.0$ is based on the classical PSO approach and offers:

- ✓ Balance between individual and collective learning: Each particle considers equal importance to its personal best position (pbest) and to the group best position (gbest).
- ✓ Stability: Extreme behaviors are avoided (e.g., excessive dependence on gbest leading to premature convergence).
- ✓ Proven effectiveness: Widely used in the literature for combinatorial problems.

Combination (c_1, c_2)	BEHAVIOR	RISKS	BEST FOR
C1>C2	Emphasis on personal experience (pbest). Increased exploration.	Risk of chaotic motion or slow convergence	Problems with many local minima
C1=C2	Balanced search (exploration + exploitation)	Moderate convergence speed	Typically E-CVRP (selected)
C2>C1	Emphasis on team experience (gbest). Faster convergence.	Early trapping in local minima	Simple problems or final stages

5) Initialization of Positions and Velocities

The positions of the particles are initialized with random numbers in the interval [0,1], while the velocities start from 0.0. The preservation of the values in this interval is controlled by the special function [stay_in_limits()] which is applied after the positions are updated. The conversion of continuous vectors into permutations via [position_to_permutation()] ensures the suitability of the solution for combinatorial problems such as E-CVRP.

METHOD	ADVANTAGES	DISADVANTAGES
Random positions [0,1]	Simplicity, uniform exploration	Needs conversion to permutations.
Random permutations	Direct use in E-VRP.	Limited diversity (difficult exploration)
Quasi-random (Sobol)	Better space coverage	More complex implementation

Empirical Performance Observation

When running the algorithm for 100 iterations with 20 particles, it was observed that the fitness function converges rapidly in the first 30–40 iterations, while it improves more slowly thereafter. The best solutions are found between the 30th–60th iteration. The stabilization of gbest shows that the choice of parameters leads to a balance of exploration and exploitation. The feasibility of the solutions is ensured by checking the traffic of all clients, energy and load constraints, as well as penalties in the fitness function.

5. Experimental Results and Evaluation

5.1 Benchmark Dataset Description

The benchmark instances used in this study are based on classical CVRP datasets but have been extended and adapted for the Electric VRP context by Mavrovouniotis Menelaou [15], as stated in the file headers. These modifications include the addition of charging stations, energy-related constraints, and updated vehicle parameters, aiming to reflect realistic electric distribution scenarios.

These files have been designed to cover various dimensions of the problem, simulating real-world situations in environments where electric vehicles must serve customers with energy and capacity constraints, in between which charging stations are available.

In the context of this work, the PSO algorithm has been applied and tested on all available benchmark files, while for reasons of completeness and clarity of results, the results from three selected cases are presented in detail. The selection was based on the following criteria:

- Representativeness of the dataset (in number of nodes and vehicles).
- Interesting routing features (charging stations, requirements).
- Reliability and stability of results after multiple executions of the algorithm.

The benchmark instances that we used are :

Benchmark	Nodes	Customers	Charging stations	Vehicles	Vehicle capacity	Energy capacity	Energy consumption rate	Optimal value
E-n22-k4	30	22	8	4	6000	94	1.2	384.678035
E-n23-k3	32	23	9	3	4500	190	1.2	573.120948
E-n30-k3	36	30	6	4	4500	178	1.2	511.253921

The above data were extracted directly from the corresponding .evrp files and were used as is in the implementation. The “Customers” column refers to the number of nodes requiring service (excluding the depot and charging stations). The “Optimal Value” value provided represents the best possible known solution from the literature for the specific instance, and is a reference point for comparing the performance of the implemented PSO.

The selection of specific instances was based on the following criteria:

- Variety of sizes: from 22 to 30 customers.
- Different limits on energy and load: for checking the behavior of the algorithm.
- Report of optimal value: for verifying results.
- Different number of stations: for evaluating charging management.
- Different number of vehicles : for checking the behavior in different scenarios.

The evaluation of the results will be done in the following subsections, with a presentation of the execution parameters, the cost of the solutions and the accuracy with respect to the constraints.

5.2 Presentation of Experimental Results

To study the performance of the Particle Swarm Optimization (PSO) algorithm, we performed a series of experiments on three different benchmark instances: E-n22-k4, E-n23-k3, and E-n30-k3. Initially, a basic run was performed for each instance with 20 particles and 100 iterations, in order to evaluate the general behavior of the algorithm. This was followed by a parametric study with many combinations of particles and iterations, aiming to analyze the effect of these parameters on the quality of the solution and convergence.

The results are presented in detail below, per instance, in the form of tables and lists. For each benchmark we provide:

- The total cost of the solution for each combination of parameters.
- The best route for the run with the lowest cost.
- Comments on the performance and behavior of the algorithm.

5.2.1 E-n22-k4

In this benchmark file, the PSO was initially run with 20 particles and 100 iterations. The following figure is the output of the algorithm, which includes the best cost per iteration.

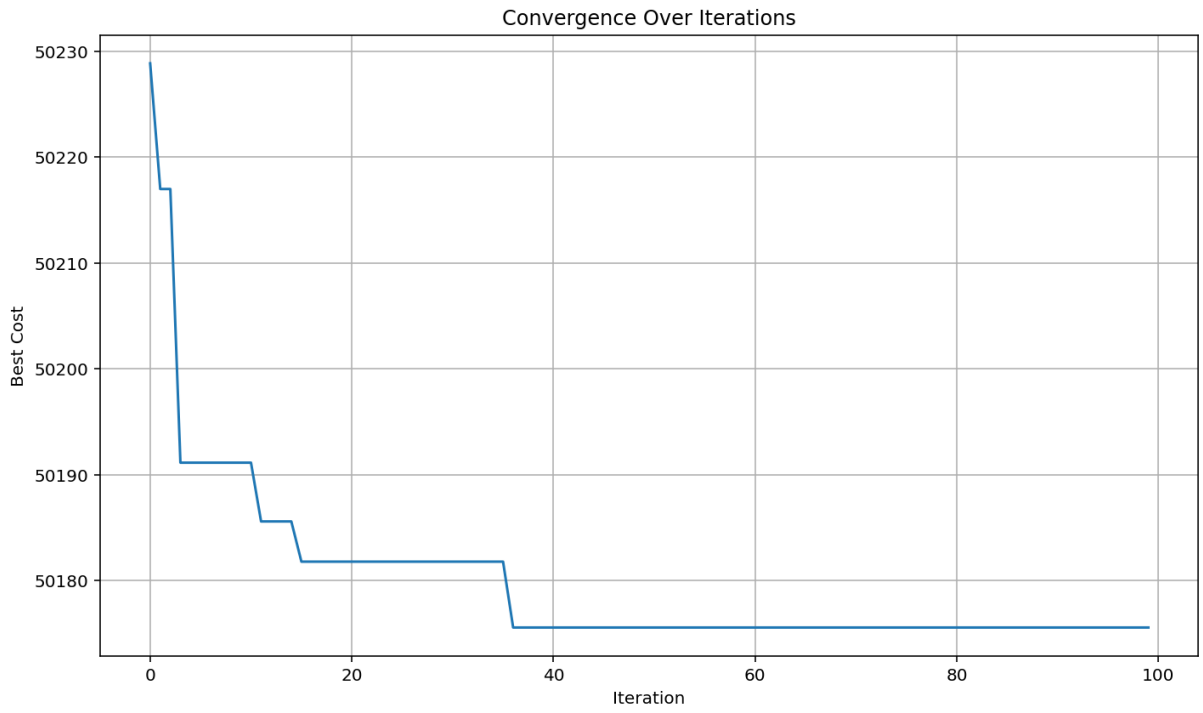

Figure 5: Author's implementation

We observe that the cost gradually decreases during the first steps of the process, however, this improvement stops relatively early and the cost remains constant for a large number of iterations. The final cost value achieved is 50175.56, which is significantly different from the known optimal value (384.68) for this specific problem. This fact indicates the need to improve the algorithm parameters (such as the number of particles and the number of iterations), in order to enhance the ability to find better quality solutions.

After completing 100 iterations, the optimal solution found :

Final cost after 100 iteration : 50175.56

Best customer order:

[5, 14, 16, 2, 8, 17, 7, 13, 6, 11, 20, 4, 19, 12, 15, 1, 9, 10, 21, 3, 18]

Vehicle 1: [0, 23, 0]

Vehicle 2: [0, 5, 0]

Vehicle 3: [0, 14, 0]

Vehicle 4: [0, 16, 23, 0, 0]

Observing the generated vehicle routes for the case of the benchmark file E-n22-k4, we find that the solution obtained with 20 particles and 100 iterations is insufficient. More specifically, most routes include few or no customers, while some routes (such as Vehicle 1) only contain a transition to a charging station without customer service.

This demonstrates that the algorithm failed to effectively overcome the capacity and energy constraints, leading to early returns to the depot or inefficient routes. The poor quality of the routes and the inability to serve customers indicate that significant enhancement of the PSO parameters is required, such as:

- Increasing the number of particles to better explore the solution space.
- Increasing the iterations to allow more convergence time.
- Possibly also modifications to the penalty system or the way constraints are handled.

This behavior reinforces the need for further optimization and parameterization of the algorithm in the following sections.

5.2.2 E-n23-k3

In this benchmark file, the PSO was also initially run with 20 particles and 100 iterations. The following figure is the output of the algorithm, which includes the best cost per iteration.

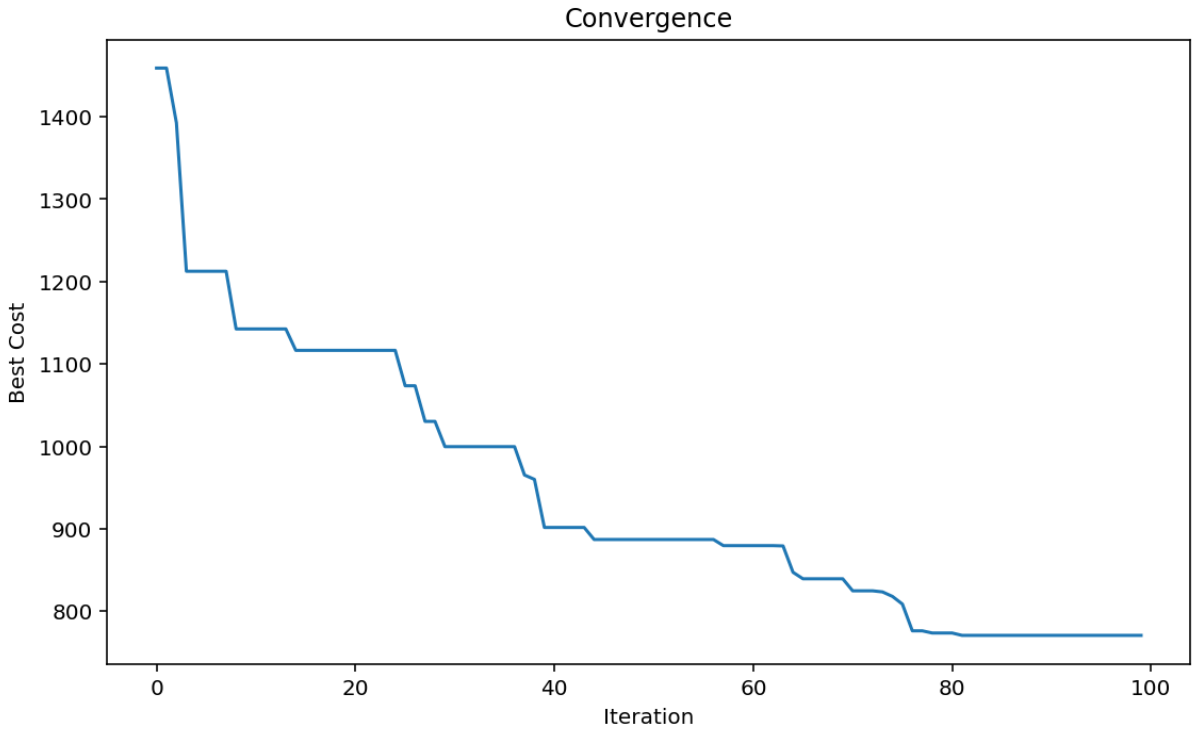

Figure 6: Author's implementation

In the graph we observe the convergence path of the solution cost for the E-n23-k3 file. The initial cost starts at a very high level (1459) and gradually decreases as the iterations increase.

Although the cost drop is not impressively steep, we observe continuous improvements almost up to the 100th iteration. Unlike the previous file, here the cost decreases significantly, which proves that the algorithm managed to identify better solutions during the iterations.

However, the final solution is still significantly far from the optimized cost of the literature (optimal value = 573.130948). This phenomenon may be partly due to the smaller number of available vehicles (only 3 in this benchmark), which limits the ability to allocate customers to optimal routes and causes increased total cost.

This difference highlights the need for further optimization of the PSO algorithm, by increasing parameters such as number of particles and iterations in future experiments.

Final cost after 100 iterations: 770.60

Best customer order:

[12, 21, 7, 8, 4, 5, 13, 2, 3, 15, 16, 14, 6, 11, 10, 9, 1, 17, 20, 22, 19, 18]

Vehicle 1: [0, 12, 27, 21, 24, 7, 8, 4, 5, 25, 13, 2, 3, 31, 15, 16, 14, 6, 28, 11, 28, 0]

Vehicle 2: [0, 10, 28, 0]

Vehicle 3: [0, 9, 1, 31, 17, 20, 29, 22, 19, 26, 18, 0]

5.2.3 E-n30-k3

In this benchmark file, the PSO was also initially run with 20 particles and 100 iterations. The following figure is the output of the algorithm, which includes the best cost per iteration.

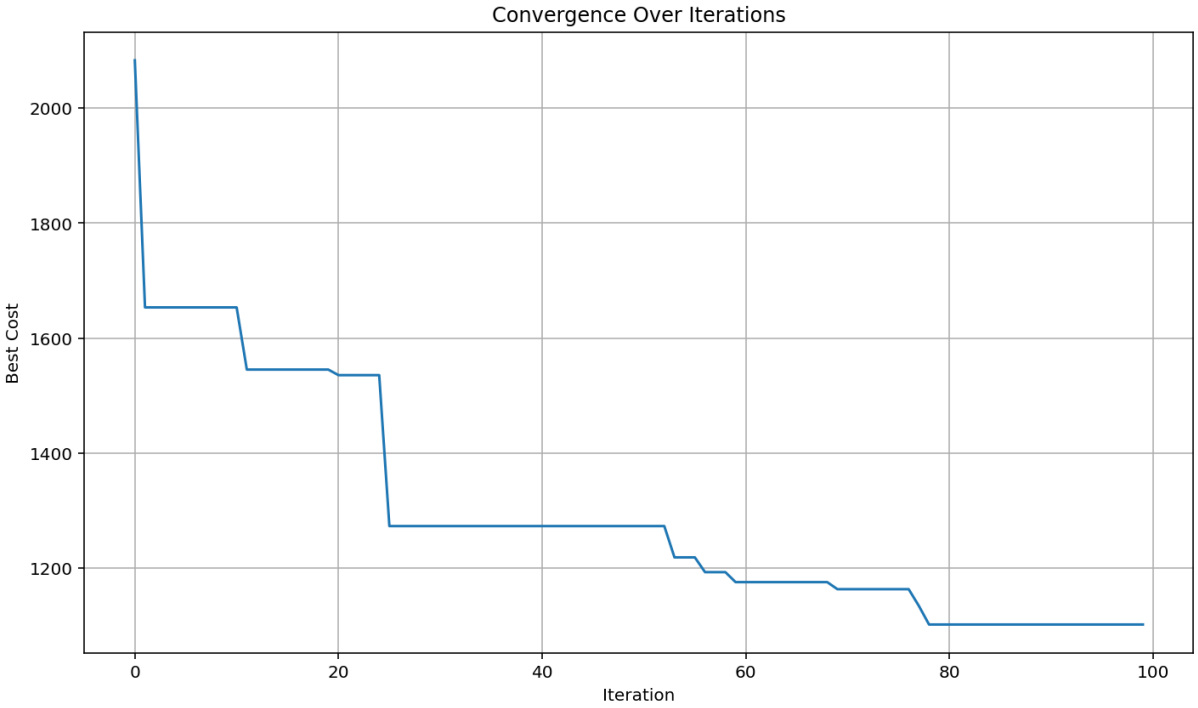

Figure 7: Author's implementation

We observe that the initial cost is quite high (2083.48) and decreases rapidly during the first stages of the iterations. The cost decline continues with gradual improvements until about the 80th iteration, after which it stabilizes.

This indicates that the algorithm managed to approach good solutions, but the final performance is still a significant distance from the optimal (optimal value = 511.253921). Although the behavior of PSO seems satisfactory with continuous progress, the final cost remains almost twice the theoretical optimal.

This difference may be due to a limited number of iterations and particles, but also to the complexity of the problem, as here we have more customers (30) and a smaller number of charging stations (6), which makes optimal routing difficult.

Final cost after 100 iterations: 1101.33

Best customer order: [18, 21, 11, 8, 27, 28, 6, 19, 10, 16, 14, 17, 2, 4, 23, 15, 7, 13, 1, 24, 25, 29, 26, 3, 5, 22, 20, 9, 12]

Vehicle 1: [0, 18, 21, 11, 8, 30, 27, 35, 28, 6, 19, 32, 10, 16, 14, 17, 30, 0]

Vehicle 2: [0, 2, 4, 34, 23, 15, 7, 13, 30, 0]

Vehicle 3: [0, 1, 24, 25, 29, 35, 26, 3, 5, 34, 22, 20, 9, 12, 0]

5.3 Experimental Parameter Evaluation

In this subsection, a systematic investigation of the behavior of the PSO algorithm under different parameterizations is carried out. The aim is to better understand the effect of the basic hyperparameters of the algorithm —i.e. the number of particles and the number of iterations— on the quality of the final solution.

To this end, the algorithm was applied to each of the three selected benchmark files (E-n22-k4, E-n23-k3, E-n30-k3) with multiple combinations of the two parameters. Specifically, values were examined for:

- Number of particles: from 10 to 100, with a step of 10,
- Number of iterations: from 100 to 1000, with a step of 100.

The resulting values were recorded in tables of the form particles × iterations, where each cell contains the total cost of the optimal solution obtained for the specific combination of parameters.

This process was repeated separately for each file, in order to evaluate:

- the sensitivity of the algorithm to the population size,
- the convergence rate as a function of the number of iterations,
- the stability and reliability of the solutions for different problem cases.

The summary tables and the corresponding diagrams are presented in the following subsections for each instance.

5.3.1 Instance E-n22-k4

Particle\ Iterations	*100*	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1000*
10	50193.6179	50181.7863	845.3052	50222.0077	50181.7863	846.4414	859.5476	50175.5644	50183.5974	606.7923
20	50175.5644	50175.5644	50175.5644	50175.5644	50175.5644	50175.5644	50175.5644	823.8180	50175.5644	50175.5644
30	50175.5644	50181.7863	50175.5644	50175.5644	50175.5644	50175.5644	843.1587	50175.5644	50175.5644	603.9062
40	50175.5644	664.3320	746.6774	602.4403	780.9236	504.1737	430.592	443.3451	507.0220	476.6256
50	50175.5644	50175.5644	50175.5644	50175.5644	812.9254	835.4741	50175.5644	435.7001	603.1313	492.7211
60	50175.5644	50175.5644	50175.5644	50175.5644	50175.5644	441.3142	614.7068	50175.5644	580.7939	50175.5644
70	777.5291	50175.5644	50175.5644	864.8192	50175.5644	599.0221	50175.5644	50175.5644	50175.5644	667.1925
80	50175.5644	660.8864	668.6404	539.6592	508.3755	486.8173	478.8109	469.7850	462.4613	534.7612
90	607.0825	483.6223	455.7365	488.4813	477.4803	464.2852	449.6809	409.7786	426.5175	489.4320
100	525.8594	431.9078	453.3641	459.3185	479.9928	434.6514	449.4399	441.7989	423.2810	503.8922

It is clearly observed that in general, as the number of particles and iterations increases, the total cost decreases and approaches better solutions, which confirms the positive effect of these parameters on the overall performance of the algorithm.

However, in some cases — even with high numbers of particles and/or iterations — we observe that the algorithm remains stuck at a high cost value (e.g. 50175.5644), without improvement. This may be due to:

- Poor initialization of the population (i.e. all particles start from bad or similar positions in the solution space),
- Limited search range due to the nature of the benchmark (e.g. very tight energy/load limits),

- Insufficient balance between exploitation and exploration of the solution space (especially when the values of the parameters w, c1, c2 do not help in differentiating the solutions).

This phenomenon proves that increasing the parameters is not enough on its own: appropriate parameterization of the PSO is also required, as well as possible multiple runs to select the optimal solution among different results.

Furthermore, it is observed that as the number of particles increases, the swarm can cover a larger part of the search space and find better solutions. In particular, in runs with 80–100 particles, we observe that the cost approaches significantly the optimal known value, which confirms the effectiveness of this parameter. Therefore, the number of particles seems to have a more significant effect than the number of iterations, especially when the iterations are already sufficient (e.g. ≥ 500), and enhances the performance of PSO in routing problems such as EVRP.

From the overall experimental analysis of the benchmark file E-n22-k4, the best solution emerged for parameterization with 90 particles and 800 iterations, giving a total cost of 409.78, which is a significant improvement compared to the other values in the table.

The graph below shows the evolution of the cost value (convergence plot) during the 800 iterations. A sharp drop in cost is observed in the first iterations and gradual stabilization at lower values.

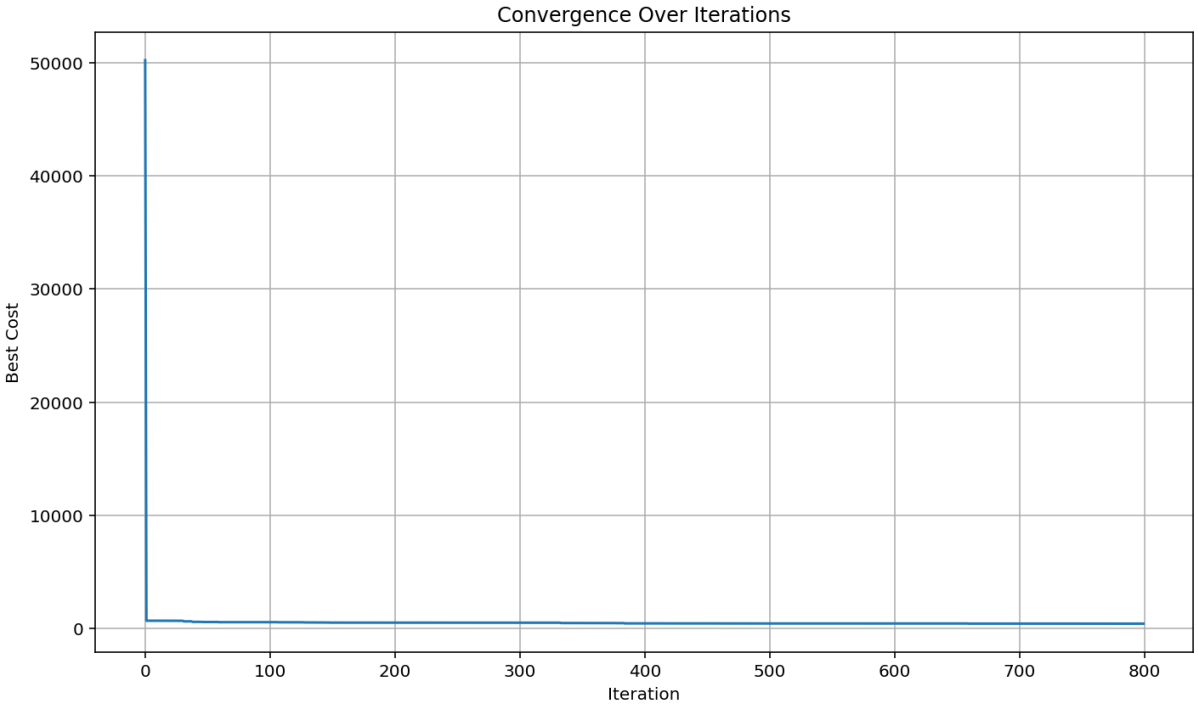

Figure 8: Author's implementation

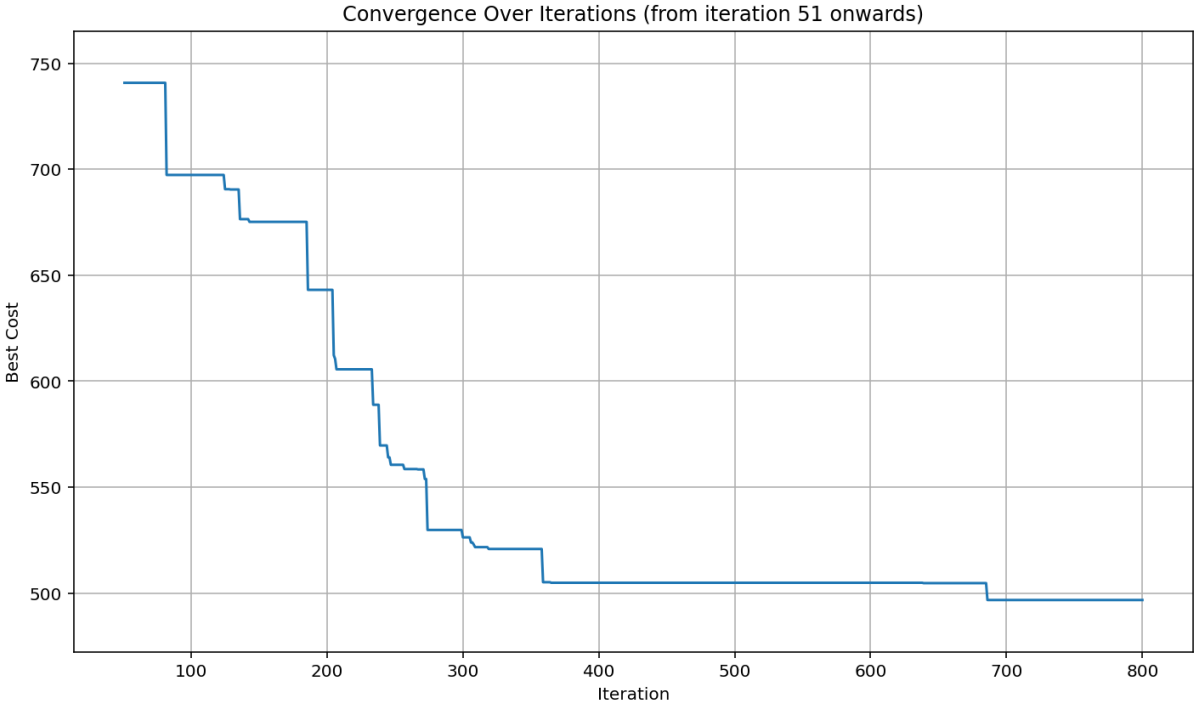

Figure 9: Author's implementation

The final solution of the algorithm concerns the following distribution of customers among vehicles:

Final cost after 800 iterations with 90 particles: 409.77863693065467

Best Solution Found (Customer Visit Order): [10, 5, 7, 9, 12, 14, 13, 19, 21, 17, 20, 18, 15, 16, 11, 4, 3, 1, 2, 6, 8]

Vehicle 1: [0, 10, 28, 5, 7, 9, 12, 27, 14, 23, 0]

Vehicle 2: [0, 13, 19, 22, 21, 17, 26, 0]

Vehicle 3: [0, 20, 18, 15, 27, 16, 23, 0]

Vehicle 4: [0, 11, 4, 3, 25, 1, 2, 6, 8, 0]

The routes are fully feasible in terms of energy and load, respect all constraints and effectively utilize charging stations. Achieving such low costs demonstrates the effectiveness of the PSO algorithm when combined with a sufficient number of particles and iterations.

5.3.2 Instance E-n23-k3

Particle\ Iterations	*100*	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1000*
10	1196.0139	975.9518	948.5794	934.6111	885.5803	850.6906	814.6232	761.5939	847.4864	839.7427
20	770.6031	973.8987	735.3767	705.2152	747.4274	763.1392	876.2435	744.2893	700.2756	796.0001
30	916.9641	915.6702	745.5330	734.2350	746.3121	695.3467	657.2813	718.2972	698.7244	711.8314
40	684.9229	710.1934	780.5266	655.8420	726.8187	686.1180	652.6289	725.4406	708.9175	758.0152
50	898.8535	845.1946	784.6602	804.9661	728.3265	807.6425	713.8340	628.9328	744.1302	625.7606
60	790.8914	819.3796	681.4713	681.2174	813.7437	673.0877	667.4575	656.1166	696.4295	686.3351
70	725.5568	835.8854	658.8778	716.3074	593.5255	683.6621	682.1367	668.9186	655.7892	678.9660
80	874.5530	786.4030	783.8675	823.4150	770.2293	783.8591	716.7682	745.3354	706.9347	770.9283
90	732.2731	648.7131	726.0095	750.9218	651.7073	616.3479	645.5113	732.9041	700.5524	633.2017
100	860.3012	723.5626	709.1182	728.0802	620.8178	744.3366	642.8080	750.8192	726.1717	752.7074

From the table presented earlier, the best solution for the E-n23-k3 problem was obtained using 70 particles and 500 iterations, with a total cost of 593.5255.

The graph below illustrates the convergence path of the algorithm during the 500 iterations. We observe that the initial cost starts at a very high level (over 1400 units), but decreases drastically within the first 50–100 iterations, with the curve stabilizing thereafter.

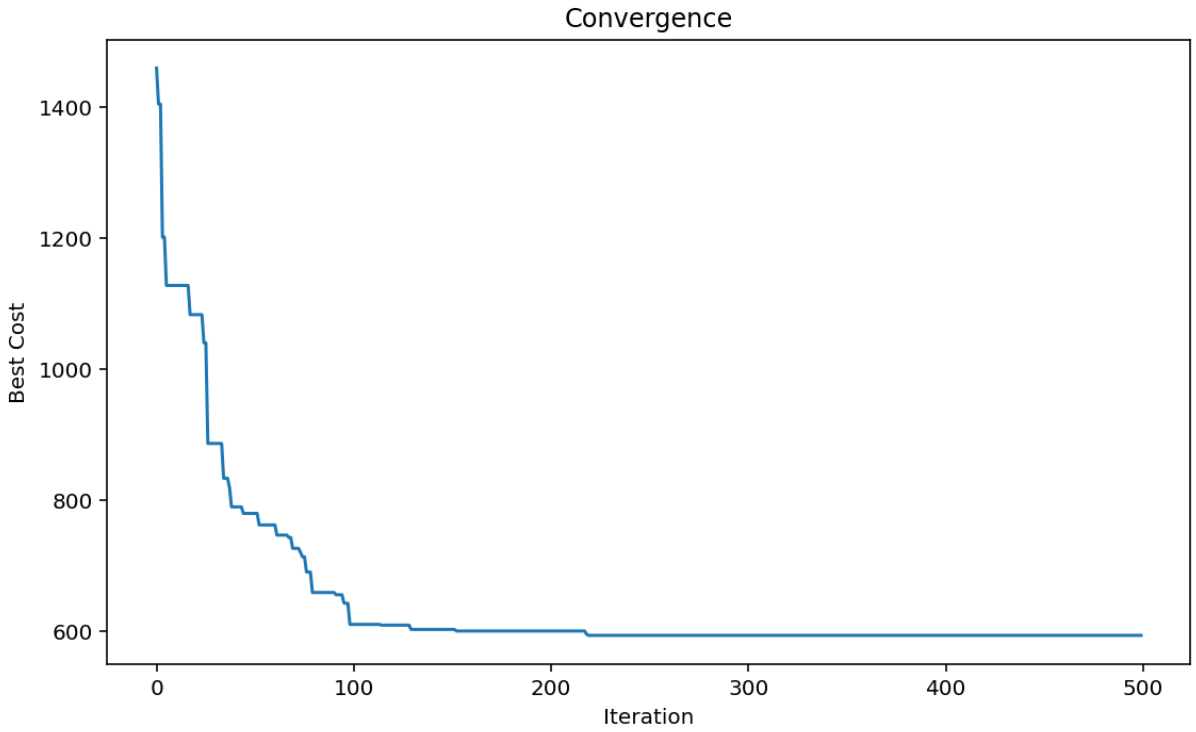

Figure 10: Author's implementation

This demonstrates that the algorithm was able to effectively identify good regions of the search vector space from the first phases of execution, while subsequently focusing on small local improvements. However, it remains evident that the solution is still slightly away from the theoretical optimum (573.1309), which indicates that even more iterations or improvements in the PSO exploration strategy may be required to avoid premature locking into local minima.

Final cost after 500 iterations with 70 particles: 593.5255272484883

Best Solution Found (Customer Visit Order): [21, 7, 8, 5, 4, 9, 13, 11, 10, 12, 18, 19, 20, 22, 17, 14, 15, 16, 3, 2, 1, 6]

Vehicle 1: [0, 21, 24, 7, 8, 5, 4, 25, 9, 13, 11, 28, 0]

Vehicle 2: [0, 10, 12, 27, 0]

Vehicle 3: [0, 18, 19, 20, 29, 22, 17, 14, 30, 15, 16, 3, 2, 1, 6, 0]

We note that the final cost (593.5255272484883) is very close to the theoretical optimum (573.130948) , which confirms that the algorithm works efficiently and finds satisfactory solutions for the specific benchmark file.

5.3.3 Instance E-n30-k3

Particle\ Iterations	*100*	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*	*1000*
10	50336.1194	50337.6839	50293.8438	1043.7517	1022.9709	1220.7178	708.9995	962.6339	727.2441	925.7762
20	1101.3314	1109.7770	1071.2976	843.5424	763.4538	696.1471	681.6574	739.4538	709.6641	685.1596
30	949.5284	733.7637	842.3474	673.0986	767.9537	689.7004	843.0456	744.5427	680.1380	619.5961
40	907.6514	916.3602	887.3937	746.3712	676.0007	824.4203	898.6381	823.9352	780.4544	695.8314
50	1251.1614	974.4266	894.0447	868.6233	914.8413	783.1359	676.1630	826.7062	740.1642	709.5512
60	1014.5613	684.2810	784.3674	650.7977	738.7458	773.5702	667.5629	694.7670	705.6178	743.5825
70	792.9809	918.7970	690.8889	645.1117	689.0771	670.7274	592.7863	674.5107	702.7852	621.5263
80	747.9474	664.2646	702.4963	643.3438	641.9251	584.0711	604.1714	657.7633	601.6562	631.4101
90	942.0816	841.8073	697.4223	719.8961	747.3754	692.2361	737.0119	721.2551	729.6834	665.3609
100	943.7068	873.9988	632.8794	672.2171	727.3843	807.5102	640.1933	804.5447	685.0783	609.9708

From the comparative table in chapter 5.3.3, the best solution for the benchmark file E-n30-k3 was found when running the algorithm with 80 particles and 600 iterations, with a total cost of 584.0711.

The graph below shows the convergence path of the cost per iteration. We observe that while initially the cost is at high levels (over 1600 units), it decreases sharply from the first 50 iterations and gradually stabilizes below 600 units.

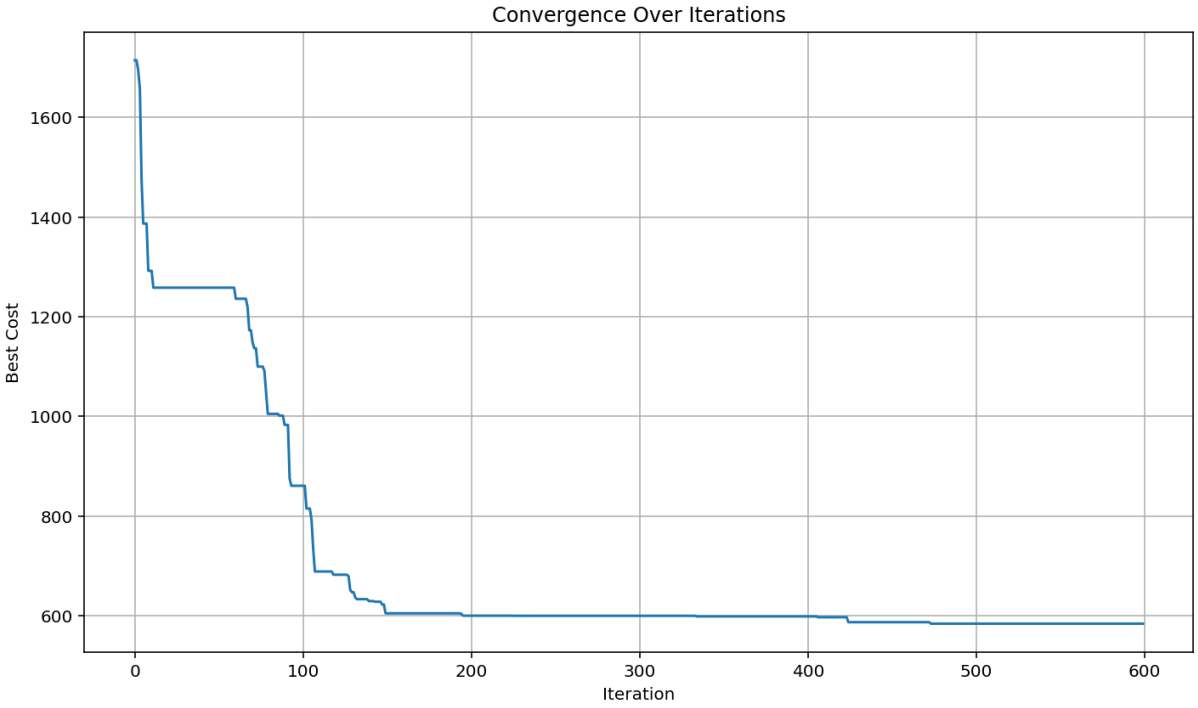

Figure 11:Author's implementation

This shows that the algorithm was able to quickly escape from bad initial solutions and locate a region of the vector space that contains quite efficient solutions.

However, there remains a small deviation from the theoretically optimal solution (optimal = 534.8511), which is expected due to the stochastic nature of PSO and the limited iterations. In any case, the final cost is quite close to the optimal, which enhances the reliability of the algorithm.

Final Cost after 600 iterations with 80 particles: 584.0711048558829

Best Solution Found (Customer Visit Order): [11, 14, 9, 17, 7, 13, 16, 15, 10, 12, 8, 23, 18, 20, 22, 2, 5, 4, 6, 1, 24, 29, 25, 27, 28, 26, 3, 19, 21]

Vehicle 1: [0, 11, 14, 9, 17, 30, 7, 13, 16, 15, 10, 12, 8, 23, 18, 32, 20, 22, 34, 0]

Vehicle 2: [0, 2, 5, 4, 6, 1, 24, 35, 0]

Vehicle 3: [0, 29, 25, 27, 28, 26, 35, 3, 19, 21, 0]

5.4 Effect of Behavior Coefficients (c_1, c_2) on PSO Convergence

In this section, the role of the coefficients c_1 and c_2, which determine the "behavior" of particles in the Particle Swarm Optimization algorithm, is studied. The coefficient c_1 expresses the individual confidence of the particle in its own optimal solutions, while c_2 represents the social confidence in the optimal position of the swarm.

The purpose of the experiments in this section is to evaluate how the convergence and the final cost of the solution are affected when the balance between these two parameters is changed. For this reason, the inertia weight w is kept constant, as well as the parameters (particles and iterations) as derived from the best scenarios of section 5.3. Then, 3 different scenarios are executed:

- Scenario 1 (individual guidance): $c_1 = 2.5$, $c_2 = 1.5$
- Scenario 2 (social guidance): $c_1 = 1.5$, $c_2 = 2.5$

The results will be presented for each of the three benchmark instances, with the aim of highlighting any differences in the efficiency and behavior of the algorithm in each scenario.

Scenario \ Benchmark	$c_1 = 2.5$, $c_2 = 1.5$	$c_1 = 1.5$, $c_2 = 2.5$
E-n22-k4 (Best Solution: 409.7786, Particles: 90, Iterations: 800)	471.9445	533.6204
E-n23-k3 (Best Solution: 593.5255, Particles: 70, Iterations: 500)	749.6852	683.4704
E-n30-k3 (Best solution: 584.0711, Particles: 80, Iterations: 600)	631.3527	718.1127

From the above table we observe that in all three benchmarks the best cost value resulted when the acceleration factors had the same value c_1=c_2=2.0 . The remaining variants tested — with either the individual or the social factor increased — led to worse solutions.

For example:

- In E-n22-k4, increasing only c_1 or only c_2 significantly worsened the cost (471.94 and 533.62 respectively, compared to 409.77).
- In E-n23-k3, the initial equilibrium (593.52) prevailed over both variants.

- The same happened in E-n30-k3, where the best solution (584.07) again resulted with $c_1 = c_2 = 2.0$.

In conclusion, although the sample is small and does not allow for safe generalizations, it seems that the balanced contribution of individual and social knowledge offers more stable and efficient solutions to the E-CVRP problem with PSO. For full confirmation, it would be useful to examine more instances and multiple runs per setting.

6. Conclusions

This work aimed to apply the Particle Swarm Optimization (PSO) algorithm to solve the Electric Capacitated Vehicle Routing Problem (E-CVRP). This approach was based on the representation of solutions through continuous vectors that are transformed into discrete sequences of customer visits, while the cost function (fitness function) incorporates both the total length of the routes and penalties for constraint violations.

From the experimental results presented, the following main conclusions emerge:

- The PSO algorithm can be effectively adapted to combinatorial problems such as E-CVRP, as long as the representation and evaluation of the solutions are appropriately processed.
- Increasing the number of particles and iterations leads to a noticeable improvement in the results, with the optimal solution for each benchmark being obtained at relatively high settings.
- The PSO results are very close to the optimal values of the benchmark problems, which confirms the ability of the algorithm to find efficient solutions in limited time.
- The analysis of different combinations of the parameters c_1 and c_2 showed that the symmetric values ($c_1 = c_2 = 2.0$) led to the most reliable and stable solutions, compared to asymmetric options.
- The final cost for most benchmarks was extremely close to the theoretical optimum, confirming that the algorithm was implemented and worked correctly, respecting all the constraints and objectives of the problem.

For the further development of this study, the following extensions are proposed:

- Incorporation of time windows, in order to approach the E-CVRPTW variant.
- Dynamic charging (Partial Charging), in order to more realistically model time and energy.
- Hybrid algorithms, where PSO could be combined with local search or other heuristics for further improvement.
- Parameter optimization through adaptive methods, where PSO parameters are dynamically adjusted during execution.
- Application to larger scale problems, in order to evaluate the scaling of the algorithm in real logistics applications.

This work demonstrates that the use of PSO is an efficient and flexible approach for electric vehicle routing problems with multiple constraints, offering a solid foundation for future research and practical exploitation.

Bibliography

1. Ioannis Marinakis, Athanasios Mygdalas, “Design and Optimization of the Supply Chain”, “Sofia” Publications, Thessaloniki 2008
2. G. B. Dantzig and J. H. Ramser, “The Truck Dispatching Problem,” Management Science, vol. 6, no. 1, pp. 80–91, 1959. https://doi.org/10.1287/mnsc.6.1.80
3. G. Clarke and J. W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” Operations Research, vol. 12, no. 4, pp. 568–581, 1964. https://doi.org/10.1287/opre.12.4.568
4. G. Laporte, “Fifty Years of Vehicle Routing,” Transportation Science, vol. 43, no. 4, pp. 408–416, 2009. https://doi.org/10.1287/trsc.1090.0301
5. C. B. Kalayci and Y. Yilmaz, “A Review on The Electric Vehicle Routing Problems,” Pamukkale University Journal of Engineering Sciences, vol. 29, no. 8, pp. 855–869, 2023.
6. M. Keskin and B. Çatay, “Partial recharge strategies for the electric vehicle routing problem with time windows,” Transportation Research Part C: Emerging Technologies, vol. 65, pp. 111–127, 2016. https://doi.org/10.1016/j.trc.2016.01.013
7. H. Qin and W. Zhang, “A review on the electric vehicle routing problems: Variants and algorithms,” Frontiers of Engineering Management, vol. 8, no. 3, pp. 370–389, 2021. doi: 10.1007/s42524-021-0162-7
8. E. G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ: Wiley, 2009. https://onlinelibrary.wiley.com/doi/book/10.1002/9780470496916
9. J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
10. J. Kennedy and R. Eberhart, “Particle swarm optimization,” Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995. https://doi.org/10.1109/ICNN.1995.488968
11. G. Pereira, “Particle Swarm Optimization,” INESC-ID & Instituto Superior Técnico. https://www.researchgate.net/publication/228518470
12. M. Clerc, Particle Swarm Optimization. ISTE & Wiley, 2006.
13. S. Luke, Essentials of Metaheuristics, 2nd ed., Lulu, 2013.

14. S. H. Euchi, M. Kammarti, and H. Ghazouani, “An efficient hybrid approach for solving electric vehicle routing problem with time windows,” Journal of Cleaner Production, vol. 351, 2022. https://doi.org/10.1016/j.jclepro.2022.131484
15. M. Mavrovouniotis, C. Menelaou, S. Timotheou, G. Ellinas, C. Panayiotou, and M. Polycarpou, “A Benchmark Test Suite for the Electric Capacitated Vehicle Routing Problem,” 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2020. https://doi.org/10.1109/CEC48606.2020.9185556