Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN

Integración de las Energías Renovables en el Sistema Eléctrico

TRABAJO FIN DE MÁSTER

Electric Vehicle charge optimization using Ant Colony Optimization

Estudiante *Arano, Muñoz, Jokin*
Director/directora *Abarrategi Ranero, Oihane*
Departamento Ingeniería Eléctrica
Curso académico *2022-2023*

Bilbao, 12, 07, 2023

Table of contents

List of illustrations

List of tables

List of Equation

1 Introducción

In this project, we are going to investigate a topic that will be of great importance in the coming years, such as the management of the charging of electric cars. It is nothing new that the trend in the automotive sector is towards a future in which electric vehicles will predominate, which can be a problem for the current electricity grid due to the increase in energy demand produced by the charging of these cars. Therefore, it is necessary to investigate different charging methods in order not to saturate the grid.

In this work, research has been carried out about the problem mentioned above, proposing an optimization algorithm which is able to manage the charging process of a fleet of cars in a charging car park for electric cars, based on the paper " Electric vehicle charging under power and balance constraints as dynamic scheduling" [7].

In the first part of the work, an extensive state of the art about the most important optimisation algorithms in use today is made, explaining what they are based on and giving a brief explanation of how they work.

Then, the problem that has been investigated with this work is described, the optimisation of the electric car charging using the ant colony algorithm (ACO). In addition, the results obtained are analysed and the parts of the work that can be improved in future projects related to this topic are detailed. Finally, the conclusions obtained are given.

2 Context

In recent years, there has been a significant increase in the use of electric cars, driven mainly by growing concern about climate change and the need to reduce greenhouse gas emissions produced by internal combustion vehicles.

The transport sector in one of the largest contributors to global greenhouse gas emission. According to EEA (European Environment Agency), “the recently proposed legislation (Fit for 55) sets targets to cut CO_2 emissions from cars by 55% and vans by 50% by 2030 (EU, 2021). It also proposes to completely cut emissions from cars and vans by 2035.

Electric cars (EVs) are a more sustainable and environmentally friendly alternative to petrol or diesel-powered vehicles, as they do not emit harmful exhaust gases and are more energy efficient. In addition, battery technology has improved significantly in recent years, making electric cars increasingly longer range and more affordable. Thus, “a significant increase in the uptake of EVs will be needed to achieve these goals” [1]. For this reason, the number of charging points for electric cars will increase drastically in the next few years, creating a problem when it comes to managing the network for charging them.

However, the integration of electric cars into the electricity grid poses some challenges. In particular, charging electric vehicles can significantly increase electricity demand at

peak times, such as during peak hours, which can overload the grid and cause power outages. In addition, charging many electric vehicles in a concentrated area can require costly upgrades to the local grid infrastructure to handle the additional load.

In order to fight this issue, new technology is required, such as increasing the capacity and lifetime of electric car batteries and increasing the speed of charging them. Besides, technical solutions are being implemented, such as smart charging, which allows charging speed to be adjusted based on the capacity of the electricity grid at the time. In addition to improving the electronics that make up electric cars and their respective chargers, it is necessary to control the charging process. “Several studies have shown that when the EVs’ charging process is not properly coordinated in a charging station, several problems may occur, such as increase in the peak load period, decrease in service quality, degradation of the voltage profile, overload of circuits, and increase in energy losses” [5]. So, in order to avoid the problems mentioned above, scheduling the charging of EV`s in an efficient way is crucial.

In this project, the problem of coordinating the charging process of electric cars in a parking lot will be addressed.

3 Objectives and Scope of Work

The main objective of this work is to address a topic that is currently giving a lot to talk about, such as the charging of electric vehicles. In order to achieve this, the following objectives are established:

The first objective is to thoroughly analyse and understand the context and characteristics related to the different optimisation algorithms that actually exist. This involves a detailed review of the existing scientific and technical literature, as well as the identification of current trends, developments and challenges in the field.

Furthermore, another objective is to focus on the implementation and operationalisation of the proposed methodology. This will involve the use of relevant tools, techniques or experiments to obtain concrete data and results that contribute to knowledge and add value in the field of EV charging.

The scope of this work is limited to, on the one hand, analysing the different optimisation algorithms that exist, and, on the other hand, using one of them to optimise the problem set out in the project.

4 Benefits of the work

This Master's thesis provides a number of significant benefits and contributions in the field of EV charging.

Firstly, this work contributes to the advancement of knowledge in the field of EV load optimisation. By comprehensively reviewing the existing literature and conducting empirical research, it is hoped to gain a better understanding of the different alternatives that exist to solve problems in the field, providing a solid basis for future research and development.

In addition, the methodology developed in this work has an important practical application, so that the algorithm could be implemented in a real problem.

In terms of social and economic impact, the findings and conclusions of this work are expected to generate benefits in the transition to EV. These benefits can be economic, social and environmental, and will contribute to improving people's quality of life and sustainable development.

Finally, the results and conclusions of this work will provide a solid basis for informed decision making in the field of EV load optimisation. It will enable key stakeholders to use the information and recommendations derived from this work to more effectively address future EV load optimisation projects.

5 State of art

This section provides a comprehensive review of research dealing with the electric vehicle charging scheduling using different optimization algorithms. In order to highlight the lack of information regarding the different types of optimisation algorithms used to solve this task, a compilation will be made of what type of algorithms have been used so far in the different studies carried out.

Optimization algorithms are used in all kinds of fields such as engineering, medicine, economics... in order to find the best solution to a given problem. So, depending on the problem to be solved, different types of them can be used. Figure 1 shows the classification of the different types of algorithms that are used today to solve different

problems, and although in the study of the electric car charging problem only one will be used, this work will make a brief description of each of them.

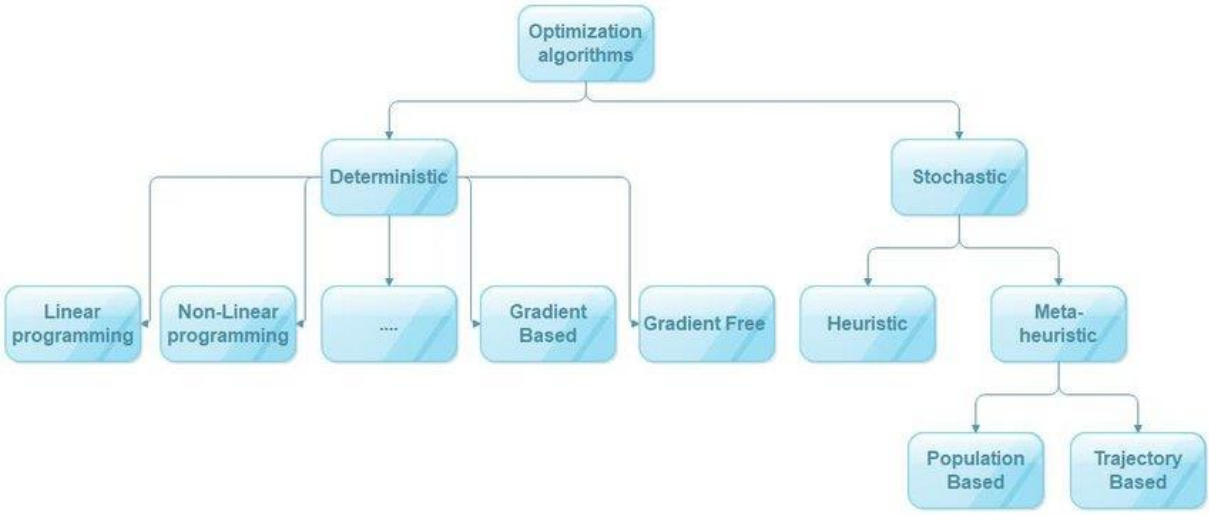

FIGURE 1. CLASSIFICATION OF OPTIMIZATION ALGORITHMS [10]

As can be seen in Figure 1, the optimization algorithms can be clearly classified into two categories. On one side, the deterministic algorithms and on the other, the stochastic algorithms. In short, deterministic algorithms allow a future event to be calculated precisely, without the involvement of randomness. However, a stochastic algorithm can handle uncertainties in the inputs applied [11]. In other words, if something is deterministic, you have all of the data necessary to predict (determine) the outcome with certainty whilst Stochastic models possess some inherent randomness - the same set of parameter values and initial conditions will lead to an ensemble of different outputs. Afterwards, each one has different types, which will be briefly explained below.

5.1 Deterministic Algorithms

Deterministic algorithms are the classical branch of optimisation algorithms in mathematics. They are usually based largely on linear algebra, either based on gradient computation or in some cases on hessian calculus. Compared to stochastic algorithms, they have the advantage of converging faster, namely they require fewer iterations than stochastic algorithms to reach a solution.

These types of optimization models are based, as mentioned above, in rigorous mathematical formulation without involving stochastic elements. That is why the results achieved are replicable and unequivocal. Nevertheless, this does not mean that stochastic models cannot provide same quality solutions. In terms of method of operation,

deterministic algorithms focus on finding stationary points in the response variable, hence, the best solution found could be the local best and not the global best. Besides, these algorithms are usually single objective. [12]

In an optimisation process, elements must be established in order to find the right solution. It is necessary to choose a sample or a group of samples from which to start, as well as a stopping criterion. By feasible sample we mean an assignment to each input variable so that all the constraints of the problem are satisfied and as a stopping criterion we refer to a condition that once fulfilled, leads to the correct solution.

In the following sections, different types of deterministic algorithms will be explained.

5.1.1 Linear optimization

Linear optimization, also known as linear programming (LP) is a mathematical procedure or algorithm used to solve an indeterminate problem, formulated through a system of linear inequalities, optimising the objective function, which is also linear. It consists of optimising (minimising or maximising) a linear function, called the objective function, in such a way that the variables of this function are subject to a series of restrictions that we express by means of a system of linear inequalities.

Any LP problem consists of an objective function and a set of constraints. In most cases, the constraints come from the environment in which your objective is located. I.e., when an objective wants to be achieved, some constraint will be set by the environment to achieve the goal.

In order to a better understanding, an example is given below. Imagine a carpenter who sells tables and chairs and wants to maximise his income. In this case, if he sells each chair (x2) for 3 euros and each table (x1) for 5 euros, the objective function would be as shown equation (1). The constraints are set by external factors such as working time and raw material limitations (equations (2) and (3) respectively). The production time for a table is 2 hours and for the chairs 1 hour. Besides, the raw material required to build a table and a chair are 1 unit and 2 units respectively. Besides, it is assumed that X1 and X2 are both positive values.

$$Objetive\ function: 5 \cdot x1 + 3 \cdot x2 \ (1)$$

$$Constraint\ 1 \rightarrow \ 2 \cdot x1 + x2 \leq 40 \ (2)$$

$$Constraint\ 2 \rightarrow \ x1 + 2 \cdot x2 \leq 50 \ (3)$$

One's equations (1), (2) and (3) are defined, it is possible to find optimal solution to the problem. In the following Figure 2 a graphical representation of the problem can be seen.

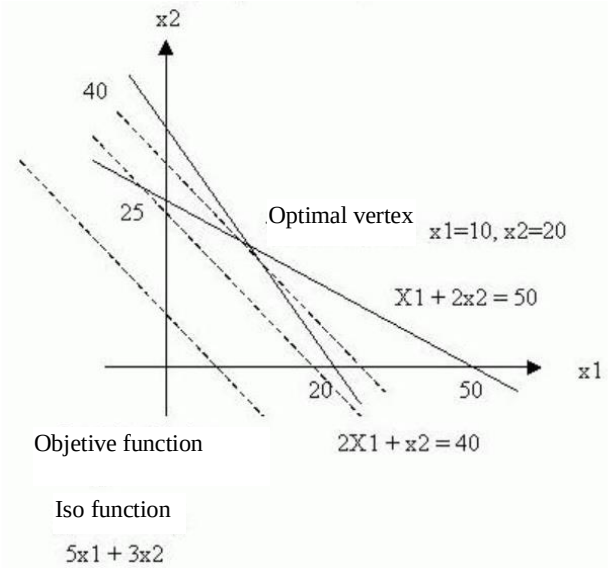

FIGURE 2. REPRESENTATION OF A LINEAL OPTIMIZATION. [13]

5.1.2 Non-linear optimization

Although some problems can be modelled by linear programming, others have to be optimised by non-linear programming, as they are defined by non-linear equations. This type of deterministic optimization follows the same procedure as the linear optimization algorithms, but in this case, the objective or the constraints functions are not lineal equations.

These algorithms can be classified according to the number of variables they have, considering a large problem has more than a thousand or so variables and also by the computational weight it has, hence, there are ones which are computationally expensive to evaluate and other which are cheap, like the linear problems.

An example of non-linear optimization is shown in the Figure 3.

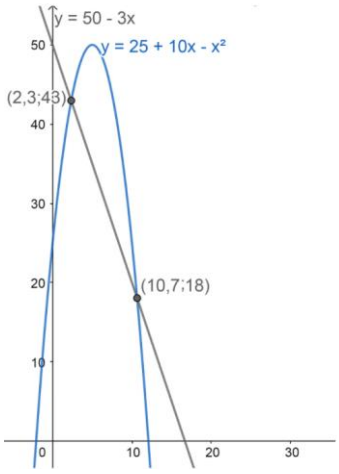

FIGURE 3. REPRESENTATION OF A NON-LINEAR OPTIMIZATION [14]

5.1.3 Gradient based algorithms

"In order to decrease the computation time and make the optimization solution more robust, gradient algorithms are used. Such procedures use the derivative of the cost function, under specific conditions, such as differentiability and Lipschitz condition, to solve the optimization problem." [15]

These gradient-based algorithms are usually classified as follows. [16] On the one hand, there are first-order algorithms, which only need the information of the derivative, and on the other hand there are second-order algorithms, which apart from the derivative information, they need the Hessian matrix. Within these two categories, there are different algorithms.

Among the linear algorithms, the best known is the linear programming (LP) algorithm. Those are used for optimization problems when the objectives and constraints are linear. However, among the second-order algorithms, they can be distinguished quadratic algorithms (QP), sequential quadratic algorithms (SQP) and finally convex programming (CP). QP and SQP algorithms are used when the objective is quadratic and the constraints are linear and CP ones are used for convex objective and concave inequality constraints.

5.1.4 Derivative free algorithms

There are functions which are impossible to derive or whose derivative involves a very high operational cost. For such cases, derivative-free algorithms (DFA) have been invented. The DFA is able to transform the solution to a global optimum, unlike the gradient algorithms that provide local suboptimal solutions.

5.2 Stochastic Algorithms

"Stochastic optimization refers to a field of optimization algorithms that explicitly use randomness to find the optima of an objective function, or optimize an objective function that itself has randomness (statistical noise). Most commonly, stochastic optimization algorithms seek a balance between exploring the search space and exploiting what has already been learned about the search space in order to hone in on the optima." [17]

"The source of inspiration of many randomized search methods comes from the observation of nature. Concepts from biology, physics, geology, or some other field of investigation, are borrowed and implemented in a simplified model of some natural phenomena. Most of these methods are population-based algorithms, in which a set of initial samples evolves (or moves) up to convergence. The rules of the evolution, which always include some randomness factor, depend on the natural model embodied. Population-based algorithms are also known as Swarm Intelligence (SI) when they mimic

the collective behaviour of self-organized natural systems. Commonly, the collective behaviour which is mimicked is taken from the animal kingdom: herding, flocking, shoaling and schooling, swarming, hunting, foraging, feeding" [13]

Compared to deterministic optimization methods, they are more recent and innovative. As an advantage over the deterministic algorithms, they are usually less complicated in terms of maths and they even contain randomness in the searching procedure. A disadvantage would be that they converge much slower.

Within stochastic algorithms we can differentiate between heuristics and meta-heuristics. Heuristic ones try to find an approximate (not approximation) solution to a problem. Is about getting a good guess at the solution to a problem, but it doesn't really know how good it is. Instead, meta-heuristic is about obtaining a solution for which it can be demonstrated how close the solution obtained is to the optimal solution.

The following are some of the most commonly used stochastic algorithms.

5.2.1 Heuristic algorithms

Heuristic algorithms are used to design solutions to the problems as quickly as it is possible. This type of algorithms will not find the best solution to the problem, instead, they will give a good solution in a short period of time. That is why when it is not necessary to find the best solution to a problem, these algorithms are used, in order to their quick convergence.

"Heuristic algorithms are used to solve Non-deterministic Polinomial (NP) problems and decrease the time complexity of problems by giving quick solutions. It's popularly utilized in artificial intelligence problems. One example is informed search, where additional information is available to determine the next step towards finding the solution" [6].

Regarding how they work, they use a heuristic function in order to find the heuristic value that will help to find the optimal solution. From a starting point, which could be a solution to the problem, a value for its node is calculated using the mentioned function. Hence, the optimal solution will be the one with lower nodes value. This procedure can be seen at Figure 4.

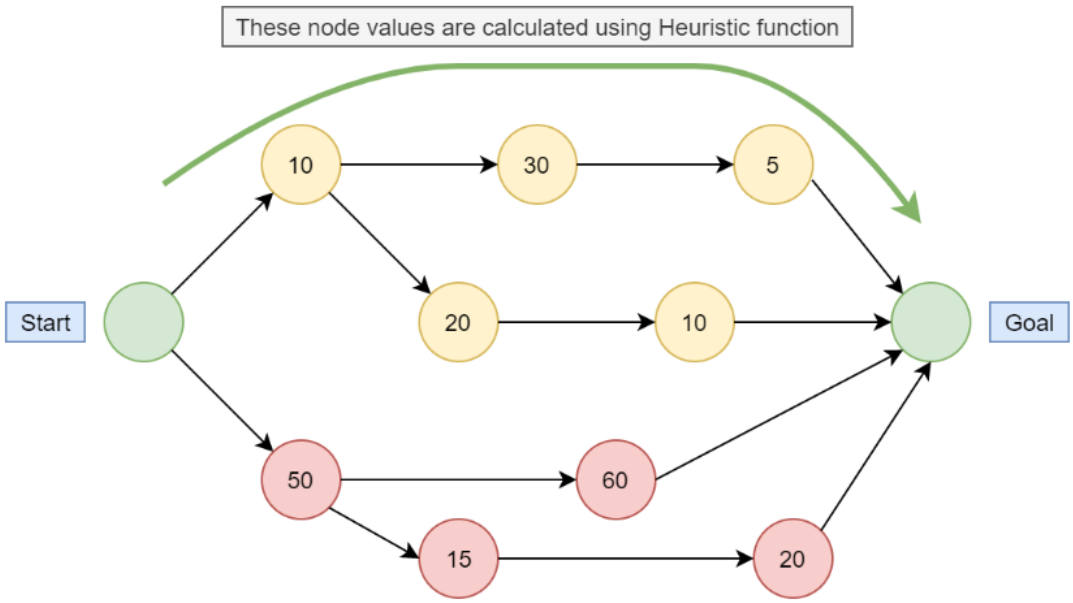

FIGURE 4. HEURISTIC PROCEDURE [36]

The heuristic functions can be classified as follows, admissible and non-admissible. The first ones, never overestimate the cost of reaching the goal whereas the second ones do. It can be explained with Figure 5.

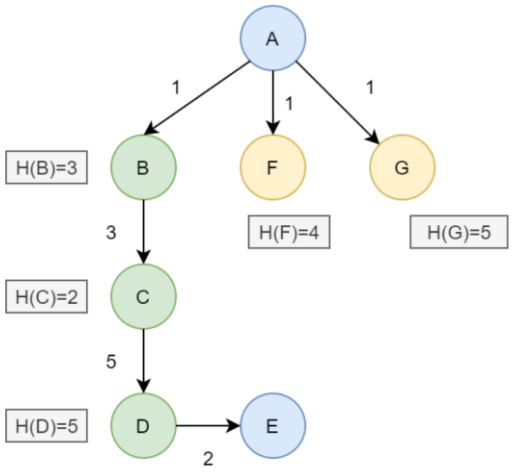

FIGURE 5. ADMISSIBLE AND NON-ADMISSIBLE HEURISTIC FUNCTIONS

"In order for a heuristic to be admissible to the search problem, the estimated cost must always be lower than or equal to the actual cost of reaching the goal state. The search algorithm uses the admissible heuristic to find an estimated optimal path to the goal state from the current node." [29] Analysing Figure 5 it can be seen that the cost from A to E is 1+3+5+2=11. Then, calculating the value of its node with the heuristic function (H(x)), it is noted that it is always lower than the total cost from A to E. This is an example of an admissible heuristic function. However, looking at the rest of the paths (A-F,A-G), the total cost from each one is 1 and the value of the heuristic function to each node is 4 and 5 respectively. As those values are higher than the total cost, that function is considered as non-admissible functions.

These algorithms are not the best ones for every problem. There are some conditions that give an idea if they fit or not to a problem. If a problem has several solutions, this type of algorithm is not the best option because they will not give the best solution from all the ones available. Besides, if the best solution to a problem is required, they are not the best option either. Nevertheless, if the convergence time is important, they are a good option to be used.

5.2.2 Meta-heuristic algorithms

Metaheuristics are approximate methods designed to solve combinatorial optimisation problems where classical heuristics are not effective. Metaheuristics provide a general framework for creating new hybrid algorithms, combining different concepts derived from artificial intelligence, biological evolution and statistical mechanisms. Besides, the advantage of metaheuristics over other methods lies in their great flexibility, which allows them to be used to tackle a wide range of problems.

Compared to heuristic algorithms, these ones are considered as a "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" (Glover and Laguna 1997). Like heuristic algorithms, metaheuristic algorithms use a certain trade-off of randomization and local search. Besides, they do not provide the best solution, but a good solution in a brief period of time.

"Two major components of any metaheuristic algorithms are: intensification and diversification, or exploitation and exploration (Blum and Roli, 2003). Diversification means to generate diverse solutions so as to explore the search space on a global scale, while intensification means to focus the search in a local region knowing that a current good solution is found in this region. A good balance between intensification and diversification should be found during the selection of the best solutions to improve the rate of algorithm convergence. The selection of the best ensures that solutions will converge to the optimum, while diversification via randomization allows the search to space from local optima and, at the same time, increases the diversity of solutions. A good combination of these two major components will usually ensure that global optimality is achievable."[25]

Between all the metaheuristic algorithms, the most commonly used are “Particle Swarm optimization (PSO)”, “Artificial Bee Colony (ABC)” and “Ant Colony Optimization (ACO)”. In the following sections, these three types will be explained.

5.2.2.1 Particle swarm optimization

The particle swarm optimization (PSO) algorithm, originally developed by James Kennedy and Russell Eberhart in 1995, is an algorithm from the artificial intelligence area of the swarm intelligence branch. It is inspired by the social behaviour of living beings and compared to genetic algorithms (GA), which are based on the mechanism of biological evolution. The particle swarm optimization algorithm is inspired by evolution in collective behavior, trying to mimic the social behavior of various animal groups such as shoals, flocks, herds, etc. [18]

As they use analogies with other processes to solve the problem, they are considered meta heuristic algorithms. Hence, they are not specialized to solve a particular problem, so they can be used for any problem, with the disadvantage that they can gen stuck in local suboptimal solution.

Another characteristic of these algorithms is that they are non-deterministic (stochastic), which means that the results obtained will not always be the same even if the same function is involved. In many of their applications, both genetic algorithms and particle swarm algorithms offer 99% quality results, however, it has been shown that particle swarm algorithms are superior in terms of efficiency due to their low computational cost [19].

The operation of this type of algorithm will be explained using a flock of birds as an example. At each iteration, each individual is a bird in the search space, shifts with a certain velocity in a direction which is a function of the global best location found so far by the swarm and the personal best location found so far by the bird. Methods for avoiding collisions could be implemented as well in the algorithm and help in maintaining a certain degree of diversity in the population. This, together with the introduction of small perturbations (called craziness or turbulence) to the individual’s position at each iteration, increases the robustness of the algorithm. Craziness reflects the change in an individual’s flight which is out of control and is very important if the whole population happens to stagnate around a local minimum [12].

5.2.2.1.1 PSO programming

Even if there are variations, the structure of a PSO algorithm to optimise (maximise or minimise) a function with one or multiple variables follows these steps:

1. Create an initial swarm of *n* random particles. Each particle consists of 4 elements: a position representing a certain combination of values of the variables, the value of the objective function at the position where the particle is located, a velocity indicating how and where the particle is moving, and a record of the best position the particle has been in so far.

   Each particle is defined by a position, velocity and value that change as the particle moves. In addition, it also stores the best position it has been in so far. When a new particle is created, only information about its position and velocity (usually initialised as zero) is known, all other values are not known until the particle is evaluated.

2. Evaluate each particle with the objective function.

   Evaluating a particle consists of calculating the value of the objective function at the particle's current position. Each particle also stores the best-valued position it has been in so far. In order to identify whether a new position is better than the previous ones, it is necessary to know whether it is a minimisation or maximisation problem.

3. Update the position and velocity of each particle

   Moving a particle involves updating its velocity and position. This step is the most important as it gives the algorithm the ability to optimise.
4. If a stop criterion is not met, return to step 2

Each particle (individual) has a position $\vec{p}$, and a velocity $\vec{v}$ which determines it movement through the space. In addition, as real-world physical particles, they have an amount of inertia, which keeps them in the same direction in which they were moving, as well as an acceleration (change in velocity), which depends mainly on two characteristics:

- Each particle is attracted to the best location that it, personally, has found in its history (personal best).
- Each particle is attracted to the best location that has been found by the set of particles in the search space (global best).

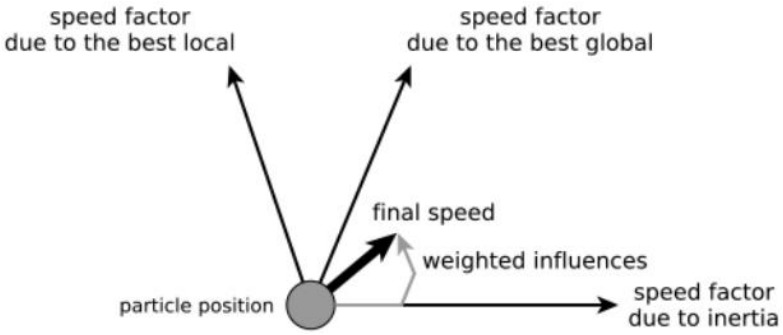

FIGURE 6. INITIAL POSITION OF THE PARTICLE [20]

The force with which the particles are pushed in each of these directions depends on two parameters that can be adjusted (attraction-to-best-personal and attraction-to-best-global) so that, as the particles move away from these best locations, the force of attraction is greater. Once the velocities are updated, their positions also update following the next equation (1).

$$p_i(t+1) = p_i(t) + v_i(t) \quad \textbf{(1)}$$

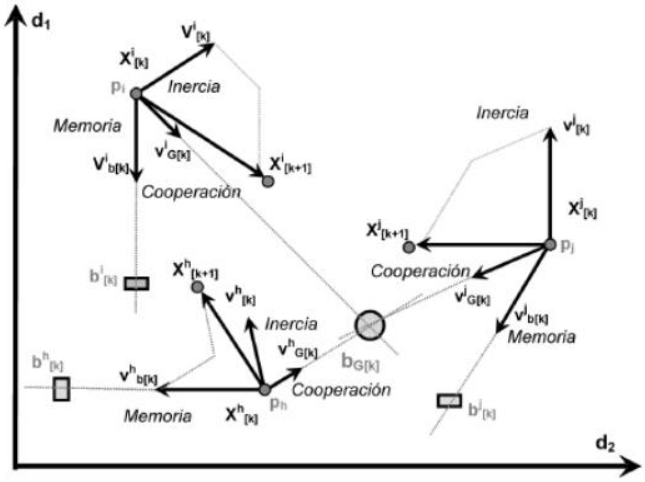

FIGURE 7. POSITION UPDATED.[20]

Once the theory of this algorithm is explained, to get a better view of how the algorithm works and what sequence it follows, the following block diagram has been used.

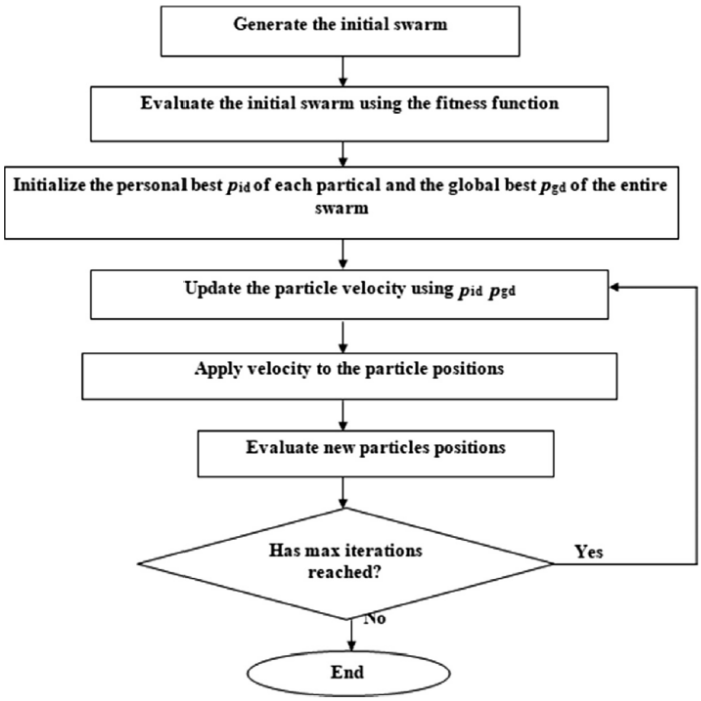

FIGURE 8.PROCESS FLOW OF PARTICLE SWARM OPTIMIZATION [28]

5.2.2.2 Genetic algorithms

Genetic algorithms (GA)were developed in the 1960s and became popular through the work of Holland and his student Goldberg. GAs represents a different approach to evolutionary computation in which the evolution of a population is mainly due to the effect of a cross-over operator. In general, the input variables are encoded into binary strings, although GAs using real-valued input variables also exist [12].

Given a specific problem to be solved, the inputs to the GA are a set of potential solutions to that problem, coded in some way, and a metric called fitness function, which allows each candidate solution to be quantitatively evaluated. These candidates can be solutions that are already known to work, with the goal of being improved by the GA, but they are usually generated randomly.

From there, GA evaluates each candidate according to the fitness function. Of course, it should be noted that these first randomly generated candidates will have minimal efficiency by the time to solve the problem, and most will not work at all. However, casually, a few may be promising, and may show some characteristics that show, even if only in a weak and imperfect way, some ability to solve the problem.

These promising candidates are retained and allowed to reproduce. Multiple copies of them are made, but these copies are not perfect. Instead, some random changes are introduced during the copying process, like the mutations that can occur in the offspring of a population. These digital offspring are then passed on to the next generation, forming

a new set of candidate solutions, and are again subjected to a round of fitness evaluation. Candidates that have worsened or not improved with changes in their code are eliminated again; but again, by pure chance, random variations introduced into the population may have improved some individuals, making them better, more complete or more efficient solutions to the problem. The process is repeated as many iterations as necessary, until we get solutions good enough for our purposes [20]. The process has been schematized in the following Figure 9.

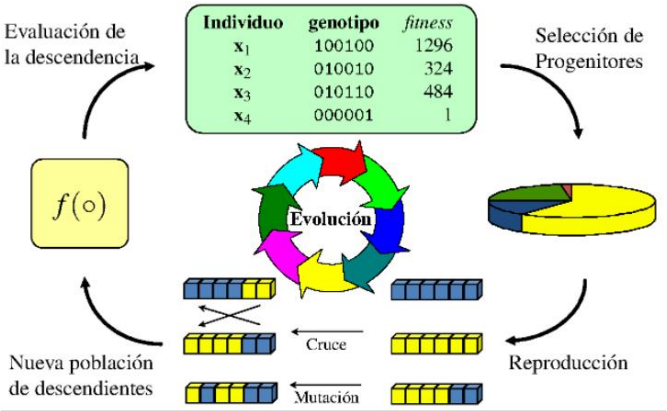

FIGURE 9. GENETIC ALGORITHM PROCESS

5.2.2.3 Bee colony optimization

The artificial bee colony (BCO) algorithm is one of the most recent algorithms in the domain of collective intelligence. It was proposed by Dervis Karaboga in 2005, and it is based on the behaviour of the bees when they are looking for food sources. It is a population-inspired optimisation algorithm, where the solutions to the optimisation problem are the food sources. The goal of these bees is to discover the food sources with the most amount of nectar.

5.2.2.3.1 BCO theory

The process of nectar foraging by bees is an optimisation process, and the behaviour of bees was modelled as a heuristic optimisation based on the biological model consisting of the following elements [23]:

- Food source: the value of a food source depends on many factors, such as proximity to the hive, richness or concentration of energy and ease of extraction of this energy. It is summarised in a numerical value that indicates the potential of the food source.
- Employed forager bees: are associated with a food source. They carry with them information about that particular source, its distance, location and profitability to share with the observer bees.

- Unemployed forager bees: these bees are looking for food sources to exploit. There are two types:
    - Explorers: they are responsible for searching for new food sources in the environment surrounding the hive. That is, they carry information about a specific source and share it with other bees waiting in the hive. The information includes the distance, direction and nectar of the food source.
    - Observers: With information shared by employees or other scouts in the nest, they search for a food source.

The employed bees communicate the information about the food source they are exploiting to the observer bees by means of a dance, where the angle to the sun indicates the direction of the source and a zigzag indicates the distance. The dances with the longest duration describe the most profitable food sources most likely to be chosen by the observer bees. Once food sources have been depleted, they are abandoned and replaced by new sources found by scout bees.

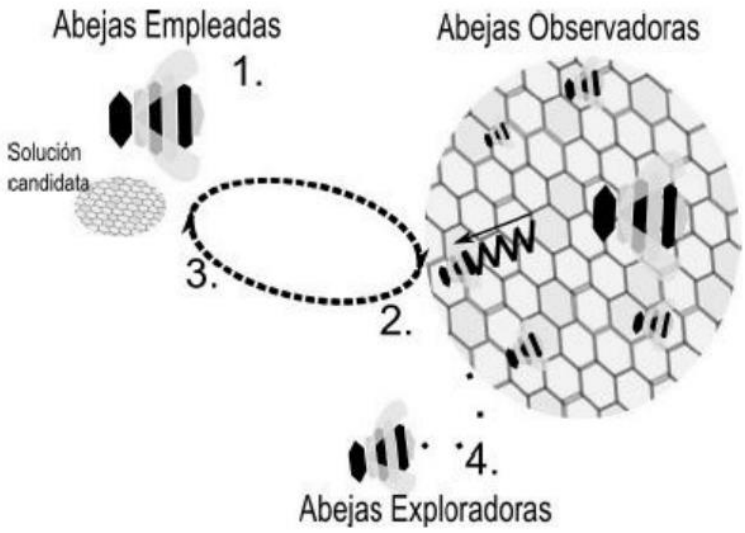

FIGURE 10. DESCRIPTION OF BCO [23]

Figure 10 shows the used bees assigned to a food source (1). Then (2) shows the communication of food source information by means of a dance, and the observer bees visit the most promising food sources (3). And finally, the scout bees search for new sources (4).

5.2.2.3.2 BCO Programming

Similar to other metaheuristic approaches, the BCO algorithm performs an iterative process which is repeated for a certain number of iterations (NG). It starts with a population of solution or food sources, which are randomly generated. Then, the next three operations are applied until a stopping criterion is reached.

1. Send out employed bees
2. Selection of food sources by onlooker bees
3. Determine which bees will be scout bees

Normally, these algorithms follow the procedure of the Figure 11. However, the programming of the algorithm is explained below. [27]

```
Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
  Memorize the best solution achieved so far
UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)
```

FIGURE 11. PROCEDURE BCO ALGORITHMS [37]

- Initialization phase.

  The algorithm starts by initializing N_p food sources. Each food source, is characterised by a vector of D elements, which represent the decision variables. These variables are randomly generated between the lower x_j^{low} and upper x_j^{high} limits, previously defined. (2)

$$x_{j,i} = x_j^{low} + rand(0,1) * \left(x_j^{high} - x_j^{low}\right) \quad (2)$$

$$j = 1,2 \dots D$$
$$i = 1,2 \dots \mathrm{Np}$$

  Where j and i are the indices of the parameter and population respectively. Therefore, $x_{j,i}$ is the j-th parameter of the i-th individual.

- Send employed bees.

  In this phase, each employed bee searches for a new food source, as follows:

$$v_{j,i} = x_{j,i} + \phi_{j,i} * \left(x_{j,i} - x_{j,k}\right) \quad \textbf{(3)}$$

$$k = 1,2 \ldots \mathrm{Np}$$
$$j = 1,2 \ldots D$$

  Where $x_{j,i}$ is a parameter *j* randomly selected from the *i*-th individual and *k* is one of the N_p food sources, satisfying the condition $i \neq k$. If a given parameter of the candidate solution $v_{j,i}$ exceeds its predetermined bounds, that parameter must be adjusted so that it is in the defined range. $\phi_{j,i}$ is a random number withing the range [-1,1]. Once a new solution has been generated, its quality is calculated using an objective function. The quality fit_i of a candidate solution $v_{j,i}$ is assigned by the following expression:

$$fit_i = \begin{cases} \dfrac{1}{1 + J(Vi)} & if\ J(Vi) \geq 0 \\ 1 + J(Vi)\ if\ J(Vi) < 0 \end{cases} \quad \textbf{(4)}$$

  Where J is the objective function to be minimized and $J(Vi)$ is the objective function value of solution Vi.

- Food source selection by onlooker bees.

  There are two different groups within the unemployed bees, onlooker and scout bees. The first ones, share their food source information with the onlooker bees, which are waiting in the hive. Then, each onlooker bee selects one of the proposed food sources depending on its quality. The probability of a food source being selected is obtained from the following equation:

$$Prob_i = \frac{fiti}{\sum_{i=1}^{Np} fiti} \quad \textbf{(5)}$$

  Where fit_i is the quality value of the *i*-food source. The probability of a food source being selected by an onlooker bee increases with an increase in the quality value of the food source. After the food source is chosen, the onlooker bees will go to the selected position and will determine a new candidate food source within the neighbourhood of the selected food source. Such position is calculated by (3). In

case the quality of the new solution is worse than the previous one, its position is maintained; otherwise, the last solution is replaced.

- Determine new scout bees
  If a food source *i* (candidate solution) cannot be performed within a certain number of *L* trials, the food of source is abandoned and the corresponding bee become a scout bee. In order to verify if a candidate solution has reached the limit *L* of trials, a counter A_i is assigned to each food source i. This counter is increased as a consequence of a bee operation failing to improve the quality of a solution.

In summary the BCO algorithm:

a) is inspired by the foraging behaviour of honeybees,
b) is a global optimization algorithm,
c) has been initially proposed for numerical optimization
d) can be also used for combinatorial optimization problems
e) can be used for unconstrained and constrained optimization problems
f) employs only three control parameters (population size, maximum cycle number and limit) that are to be predetermined by the user. [26]

5.2.2.4 Ant colony optimization

Ant Colony Optimization (ACO) theory was introduced by Marco Dorigo in the early 1990s as a tool for the solution of complex optimisation problems. The ACO belongs to the class of heuristic methods, which are approximate algorithms used to obtain good solutions to complex problems in a reasonable amount of computing time. [21]

The source of inspiration for ACO is the actual behaviour of ants. These insects when searching for food initially explore the area around their nest in a random way. As soon as they find food sources, they assess their quantity and quality, and carry some of this food back to their nest. On the way back to the nest, the ants deposit a chemical substance called pheromone on the path, which will serve as a future guide for others to find the food. The amount of pheromone deposited will depend on the quantity and quality of food. This will help to find the shortest routes between their nest and food sources.

5.2.2.4.1 ACOs theory.

In order to understand how ants, use the path with the most pheromones (the shortest) to find their food, the following example its described.

Consider the example illustrated in Figure 12, in which ants reach a point where they have to decide whether to turn right or left. As there is initially no pheromone present on

the two alternative paths, the choice is made randomly. It is estimated that on average half of the ants turn left and the other half decide to turn right. The criterium to name the ants is by a letter (R or L) followed by a number. So, if an ant turned to the right in the first fork, it will be named R1. [21]

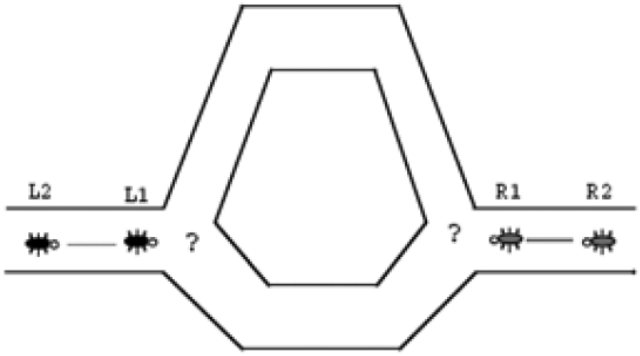

FIGURE 12. ACO ALGORITHM EXAMPLE 1 [22]

Figures 13 and 14 show what happens in the following instants, assuming that all the ants walk at the same speed. The number of dotted lines is proportional to the number of pheromones the insects have deposited on the ground.

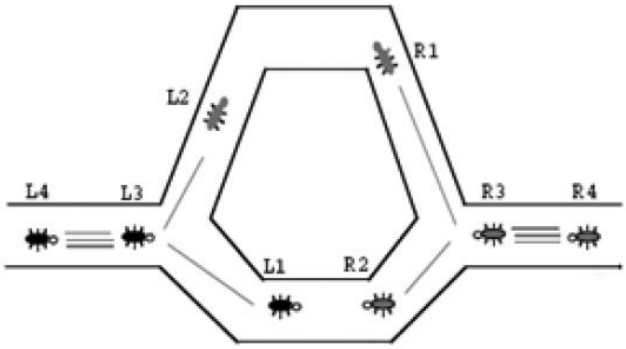

FIGURE 13. ACO ALGORITHM EXAMPLE 2 [22]

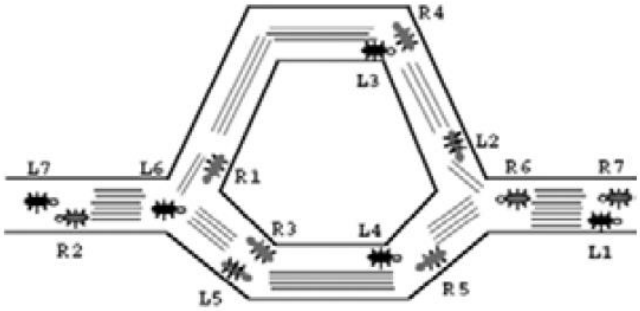

FIGURE 14. ACO ALGORITHM EXAMPLE 3 [22]

As the lower path is shorter than the upper path, many more ants will transit the lower path during the same period of time. This implies that the shorter path accumulates more pheromone much faster. After a certain time, the difference in the amount of pheromone in the two paths is large enough to influence the decision of new ants to enter these paths.

Given the above, new ants entering the system will prefer to choose the lower path due to they perceive a higher amount of pheromone there.

5.2.2.4.2 ACO programming

Having explained the behaviour of the ants in finding the shortest route to the food source, it is time to analyse how the algorithm works. That is why in this section, the programming code will be examined.

First of all, the different parameters to be used in the algorithm will be defined. They are the following:

- $P^k(t)$: The path followed by the ant k. It is time-dependent because it changes with each iteration.
- G = (V, E): weighted graph
- N_i: nodes available from node i
- k=1,2 …N: Number of the ant
- τ_{ij}: Number of pheromones between paths i and j
- α: is a parameter to control the influence of τ_{ij}
- β: is a parameter to control the influence of n_{ij}
- i,j: i is the actual node and j the following node.
- d_{ij}: distance between nodes i and j.
- n_{ij}: is the desirability of edge i, j. Typically $1/ d_{ij}$
- ρ: is the rate of pheromone evaporation
- $\Delta \tau_{ij}$: is the amount of pheromone deposited by each ant in the path i-j.

The time (t) increases once all the ants find food and return to the origin (t→ t+1).

The algorithm follows the following sequences:

1. t=0, initialize the pheromones σ_{ij} for every path available with randomly generated small values.
2. Locate N ants at the source node.
3. **Repeat**
   3.1. For each ant k=1,2 …N
      3.1.1. $P^k(t)=0$. At the first instant, the path followed by each ant is 0, because the ant has not yet moved.
      3.1.2. Repeat

- select the next node according to the probability q. If q>q0 the next node is selected according equation (6). If q<q0, the next node is selected according the equation (7).

$$P_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha} * \left[n_{ij}\right]^{\beta}}{\sum_{1}^{N}\left[\tau_{ij}\right]^{\alpha} * \left[n_{ij}\right]^{\beta}} \quad \text{if j} \in N_i \qquad \textbf{(6)}$$

$$argmax\left[\tau_{ij}\right]^{\alpha} * \left[n_{ij}\right]^{\beta} \qquad \textbf{(7)}$$

- add a step (i, j) to the path $P^k(t)$.

Until the food source is reached.

3.1.3. Remove the cycles in $P^k(t)$.

3.1.4. Calculate the length of the path found $f(P^k(t))$.

3.2. For each connection (i,j)

- Update the number of pheromones of each path. The new number of pheromones is calculated by the following equations:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij} \qquad \textbf{(8)}$$

Where: $\Delta\tau ij^k = \sum_{1}^{N} \frac{1}{f(Pk(t))}$ *if ant k travels on edge i, j*

Until all the ant follows the same path

4. Return the shortest way, the lower $f(P^k(t))$.

In ACO pheromones are updated on two different levels, globally and locally. In the global pheromone trail update, only the ant which represent the global best solution its allowed to add pheromone to its solution components. The pheromone trails are updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho * \Delta\tau ij \qquad \textbf{(9)}$$

Where p is the pheromone evaporation rate ($\rho \in (0, 1]$) and $\Delta\tau ij = 1/C$, where C is the total tardiness for the best solution. If C becomes 0, the ACO terminates. However, in local pheromone trail update, ACS applies a step by-step pheromone update rule immediately after an ant has added a new solution component, following the next equation.

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi * \Delta\tau \qquad \textbf{(10)}$$

Where ξ controls the influence of the local pheromone update and τ o is the initial pheromone value. This local pheromone update helps to make the correct decision of assigning EV *j* in position *i* less desirable for the other ants to favour assignments of other EVs on that position. It should be noted that all pheromone trails are initialised at the beginning of the algorithm by a matrix. [4]

6 Description of the solution

This section will describe the problem that has been investigated in this project, i.e. the problem posed by Hernández-Arauzo in the paper "Electric vehicle charging under power and balance constraints as dynamic scheduling" [7]. As mentioned above, the paper deals with optimising the charging of electric cars in a car park.

6.1 Problem definition

Climate change is an issue of concern for much of the world today. This is why different communities around the world have already begun to take measures to try to reduce this problem as much as possible. A turning point came in 2015, when the UN General Assembly adopted on 25 September the 2030 Agenda for Sustainable Development, an action plan for people, planet and prosperity, which aims to strengthen universal peace and access to justice [30]. The framework consists of 17 goals for environmental sustainability (see), social inclusion, economic development, peace, justice, good governance and partnership, the main issues for the world population in the 21st century. Each goal has several targets that better define its aims. The total number of targets is 169 [31].

FIGURE 15. ASD 17 GOALS [30]

This project contributes to the implementation of objective number 7, “affordable and clean energy”. A large part of C02 emissions are emitted by the transport sector. In 2020, this same sector emitted 25% of all CO2 emissions in Europe [33]. Therefore, by decarbonising the automotive sector, i.e. replacing existing combustion vehicles (diesel, petrol) with electric or plug-in hybrid vehicles, it is possible to reduce the greenhouse gas emissions produced by this sector.

In addition to the SDGs, the European Parliament has approved in February 2023 the legislation that will ban the sale of combustion vehicles from 2035 in Europe. The regulation will require that by 2035 automakers must achieve a 100 percent cut in CO2 emissions from new cars sold, which will make it impossible to sell new fossil fuel-powered vehicles in the 27-country bloc. It also sets a 55 percent cut in CO2 emissions for new cars sold from 2030 versus 2021 levels, much higher than the existing target of a 37.5 percent. New vans must comply with a 100 percent CO2 cut by 2035, and a 50 percent cut by 2030, compared with 2021 levels [34].

All of the above measures, coupled with public awareness of climate change, have led to a surge in the sale of electric vehicles in the last few years. Electric cars, which include battery electric vehicles (BEVs) and plug-in hybrid electric vehicles (PHEVs), are gradually penetrating the EU market. There has been a steady increase in the number of new electric car registrations annually, from 600 in 2010 to about 1,061,000 units in 2020, when they accounted for 11% of new registrations. In 2021, electric car registrations surged, accounting for almost 18% of newly registered passenger cars [35].

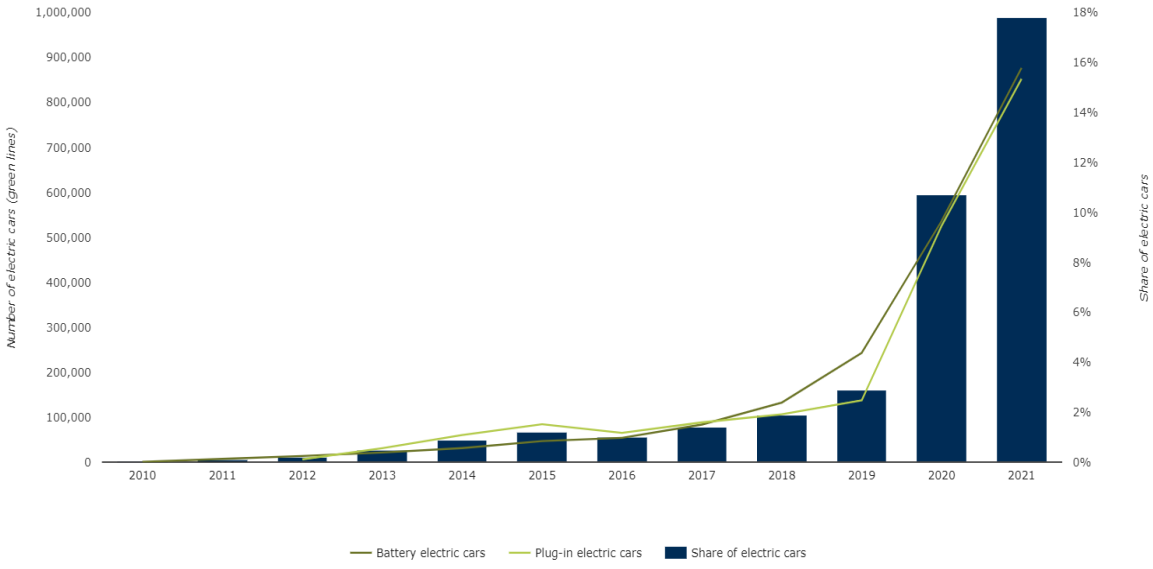

FIGURE 16. NEW REGISTRATIONS OF ELECTRIC CARS IN EU MARKET [34]

As can be seen in the above graph, the penetration of the electric car in the European Union's vehicle fleet is taking place at a dizzying speed. This implies a considerable increase in electric demand, unfortunately, this means a considerable increase in electricity demand, and unfortunately, not all distribution or low voltage grids are prepared to support the increase in electricity demand that would result from the recharging of thousands of vehicles. The power to be transported by these networks would increase, a power for which they were not designed.

The problem comes when that network and the medium voltage transformer have not been designed to handle that amount of power. In this case, a solution to the problem must be found. Changing the transformer is one option, but that is not very cheap. Another is to manage the recharging of EV in an intelligent way. If not, there may be an imbalance between the phases, in case most of the vehicles are connected to the same one.

To fight these issues, there are different solutions:

- Add distributed generation, so that it can mitigate the impact of electric vehicle demand.
- Manage recharging of EVs taking into account grid parameters, such as voltage. Excessive consumption in low-voltage grids means that the voltage can drop below the limits not allowed by regulations.
- Oversizing the distribution and low-voltage networks, assuming the huge costs that this entails.

In this work a method for managing the charging of EVs will be proposed. Taking into account all the optimization algorithms mentioned in section 5, it should be noted that the most widely used today for this type of problem are the metaheuristic algorithms. Metaheuristic algorithms are a class of optimization algorithms that are well-suited for solving complex optimization problems with many variables and constraints, such as electric vehicle (EV) charging optimization. Here are some reasons why metaheuristic algorithms are good for EV charging optimization:

1. Non-deterministic search: Metaheuristic algorithms are designed to perform a non-deterministic search of the solution space, which means that they can explore a wide range of possible solutions and avoid getting stuck in local optima. This is particularly important for EV charging optimization, which involves many variables and constraints that can create a complex solution space.
2. Flexible and adaptable: Metaheuristic algorithms are flexible and adaptable, which means that they can be easily customized to meet specific optimization objectives and constraints. This is important for EV charging optimization, which involves a wide range of variables and constraints, such as charging station availability, vehicle battery capacity, and travel time.
3. Good scalability: Metaheuristic algorithms are designed to handle problems with many variables and constraints, which makes them well-suited for large-scale EV

charging optimization problems. This is important for optimizing charging schedules for a large number of EVs and charging stations.

4. Can handle uncertainty: EV charging optimization is subject to uncertainty, such as changes in traffic patterns, weather conditions, and driver behaviour. Metaheuristic algorithms are designed to handle uncertainty by using probabilistic methods to explore the solution space.

Overall, metaheuristic algorithms are a powerful tool for EV charging optimization because they can handle complex, large-scale, and uncertain problems with many variables and constraints.

Within metaheuristic algorithms ACO or BCO can be used for this type of problems. Both Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO) are metaheuristic algorithms that are commonly used for solving optimization problems, including EV charging problems. However, there are some differences between the two algorithms that may make ACO better suited for EV charging problems compared to BCO. Here are some reasons:

1. Communication between agents: Both ACO and BCO involve the use of multiple agents (ants or bees) to explore the solution space. However, ACO agents communicate with each other by depositing and following pheromone trails, which can help to coordinate their search and avoid redundant exploration. In contrast, BCO agents do not communicate with each other directly and rely on global information exchange. This can make the search less efficient and potentially lead to suboptimal solutions.
2. Adaptive pheromone updating: ACO algorithms typically use an adaptive pheromone updating strategy, where the pheromone level of a given path is adjusted based on the quality of the solution obtained by the ants that used that path. This can help the algorithm to converge to a good solution quickly and effectively. In contrast, BCO algorithms typically use a fixed pheromone updating strategy, which can lead to slower convergence and suboptimal solutions.
3. Robustness to changes in the problem domain: ACO algorithms are known to be robust to changes in the problem domain, such as changes in the number of charging stations or the number of EVs to be charged. This is because the pheromone trails act as a memory of the search history and can guide the search even in changing environments. In contrast, BCO algorithms may require more tuning to adapt to changes in the problem domain.

Overall, ACO may be better suited for EV charging problems compared to BCO because of its communication between agents, adaptive pheromone updating, and robustness to changes in the problem domain. However, the choice between ACO and BCO ultimately depends on the specific problem requirements and constraints

6.2 Benchmark definition

First of all, I would like to describe what the electric car charging centre looks like. This car park is based on 3 charging lines and in each line, there are several charging points. Each charging point is also a private car park, so the car batteries can be charged while the cars are parked. A visual description of the car park is shown in Figure 17.

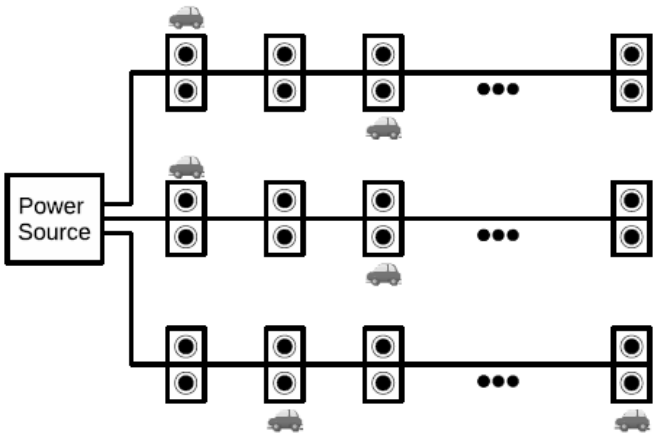

FIGURE 17. A GENERAL STRUCTURE OF A CHARGING STATION WITH THREE LINES AND MULTIPLE CHARGING POINTS (OR PARKING SLOTS). [3]

As can be seen in the figure 17, the design criteria of the control system were based on simplicity, economy and maintenance easiness. The grid is fed by a three-phase source of electric power with a voltage between phases of 400 V. Each charging point *Pi* is connected to one single-phase and supplies energy at 230 V and 2.3 kW. So, for a given contracted power, there can be a maximum number of vehicles charging in a line at the same time. Also, the consumption in the three lines should not be too different at any time. Otherwise, the net is imbalanced and there is current in the neutral point. This causes higher losses than those of a balanced system and lowers the energy transmission efficiency. Moreover, the Spanish regulations (BOE, 2013) do not allow the installation of devices that produce large imbalances without the consent of the supplier company, which can penalize the customer for it [7]. There are *i* lines, and each line connects *Pi* charging points. The number of active charging points is stored in the variable *N*.

It should be noted that the problem addressed in the paper [7], uses two constraints in such a way that the results obtained can be used for an existing charging centre. The two constraints used are the following:

$$\sum_{j=1}^{Pi} x_j^i \leq N, \mathrm{i} = \{1, \ldots, \mathrm{L}\} \quad \textbf{(11)}$$

$$\frac{\left|\sum_{j=1}^{Pi} x_j^i - \sum_{q=1}^{Pi} x_q^l\right|}{N} \leq \Delta, \mathrm{i}, \mathrm{l} = \{1, \ldots, \mathrm{L}\}, \mathrm{i} \neq \mathrm{l} \quad (12)$$

$$x_j^i = \begin{cases} 1, \text{if charging point j on line i is active;} \\ 0, \text{otherwise;} \end{cases} \quad (13)$$

Eq. (11) ensures that each line can only have N charging points active to charge N EVs at the same time, Eq. (12) controls the maximum imbalance between lines ($\Delta \in [0, 1]$) between the lines, and Eq. (13) defines a decision binary variable [7]. *j* is the number of parks using the parking lot.

To pursue the problem further, it is necessary to analyse how the fluctuation of cars will vary over time, i.e. to propose a scenario describing the behaviour of cars using the car park to charge EV batteries. For each car arriving at the car park, it is necessary to know the following data:

- **Arrival time:** Describes the arrival time of the car at the car park in minutes.

  Charging time left: Describe how much charging time is needed for each car to reach 100% charge of its batteries, taking into account that the charging point has 2.3 kW of charging power and the batteries have a capacity of 23kW.

- **Due date**. Describe the time when the car has to leave the car park.

The three parameters shown above follow the following normal or uniform distributions as shown in the tables below (Table 1 Table 2 Table 3). The first column of the tables shows the percentage of cars arriving at the car park and the second column shows the distribution that follow each group of cars.

It can be seen that Table 2 does not directly show the remaining charging time to charge the battery of the cars arriving at the car park to 100% (N(C,D)). In order to calculate this data, the following equations have been used (14) (15).

$$c = \frac{(100 - A) * 0.01 * 60 * BatteriesCapacity}{ChargingPower} \quad (14)$$

$$D = \frac{B * 0.01 * 60 * BatteriesCapacity}{ChargingPower} \quad (15)$$

The data shown in the tables (Table 1 Table 2 Table 3) has been plotted to get a better understanding of what the different probability distributions mean.

% VEHICLES	ARRIVAL TIME (MINUTES)
10	U (0,1440)
20	N (510,15)
10	N (720,15)
50	N (1170,15)
10	N (1350,15)

TABLE 1 ARRIVAL TIME DISTRIBUTION

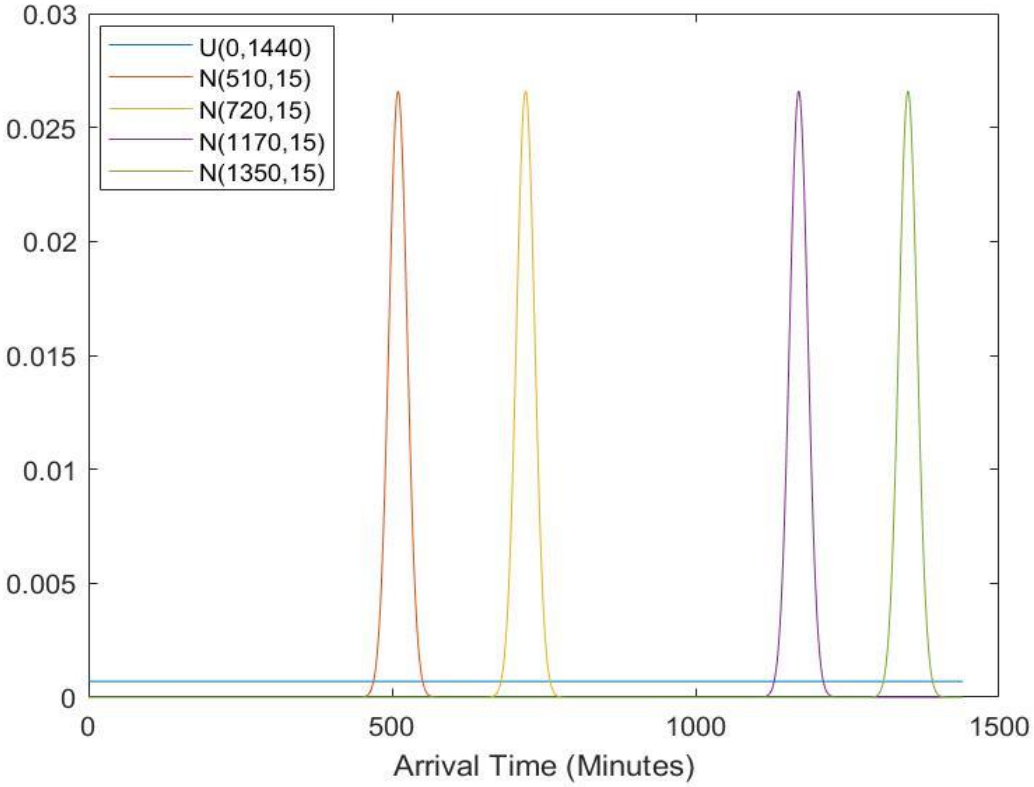

FIGURE 18 GRAPHICAL DIAGRAM EVS ARRIVAL TIME

% VEHICLES	INITIAL CHARGE (%)
10	N (80,10) -> N(A,B)
30	N (50,15)
30	N (35,7.5)
30	N (12,6)

TABLE 2 INITIAL CHARGE DISTRIBUTION

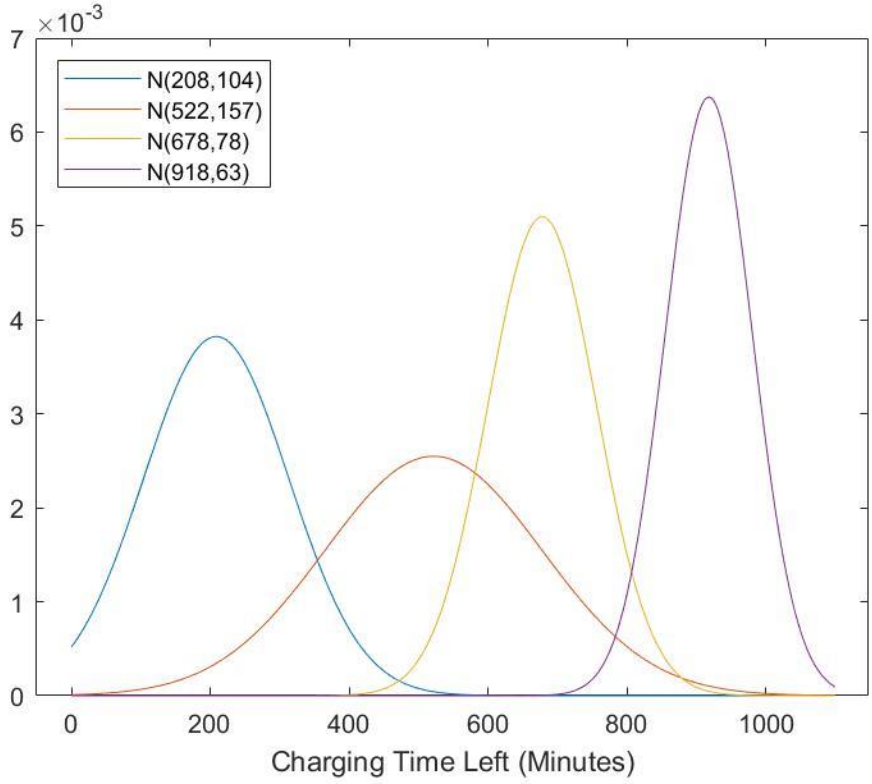

FIGURE 19 GRAPHICAL DIAGRAM OF THE TABLE 2

% VEHICLES	DUE DATE (MINUTES)
10	N (240,120)
30	N (360,120)
30	N (480,120)
30	N (660,120)

TABLE 3 DUE DATES DISTRIBUTION

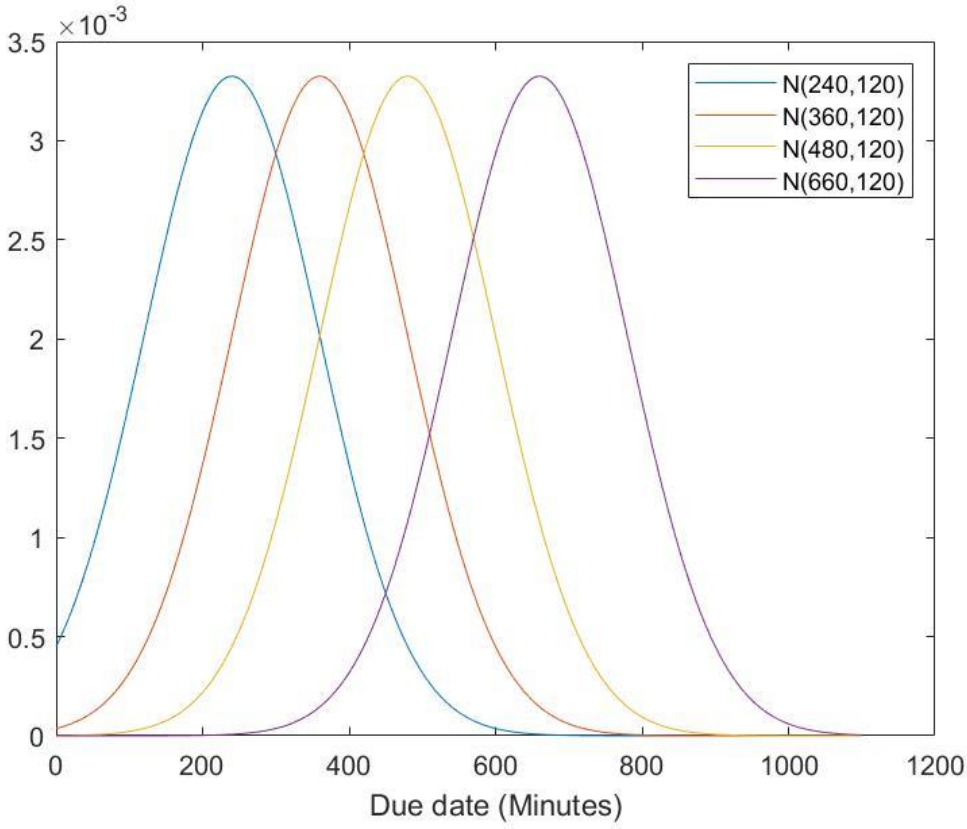

FIGURE 20 GRAPHICAL DIAGRAM OF THE TABLE 3

It should be noted that these distributions have been obtained by means of a simple model generator based on the statistical distributions, by means of which the data necessary to pose the problem are obtained. Now, having described the scenario on which the project is based, it is time to explain how do the algorithm created to solve the problem works.

6.3 Adapted ACO algorithm for EV charging

Having described the scenario in which the project will be carried out, a detailed description of the algorithm that I have programmed to optimise the charging of the EVs that come to the aforementioned car park will be given. For this, the first thing to do is to show the pseudocode of the algorithm (Figure 21). It serves to represent the logical and detailed process of an algorithm in a clear and concise way, and facilitates the understanding of the algorithm by programmers and developers who can work on the actual implementation of the algorithm in different programming languages.

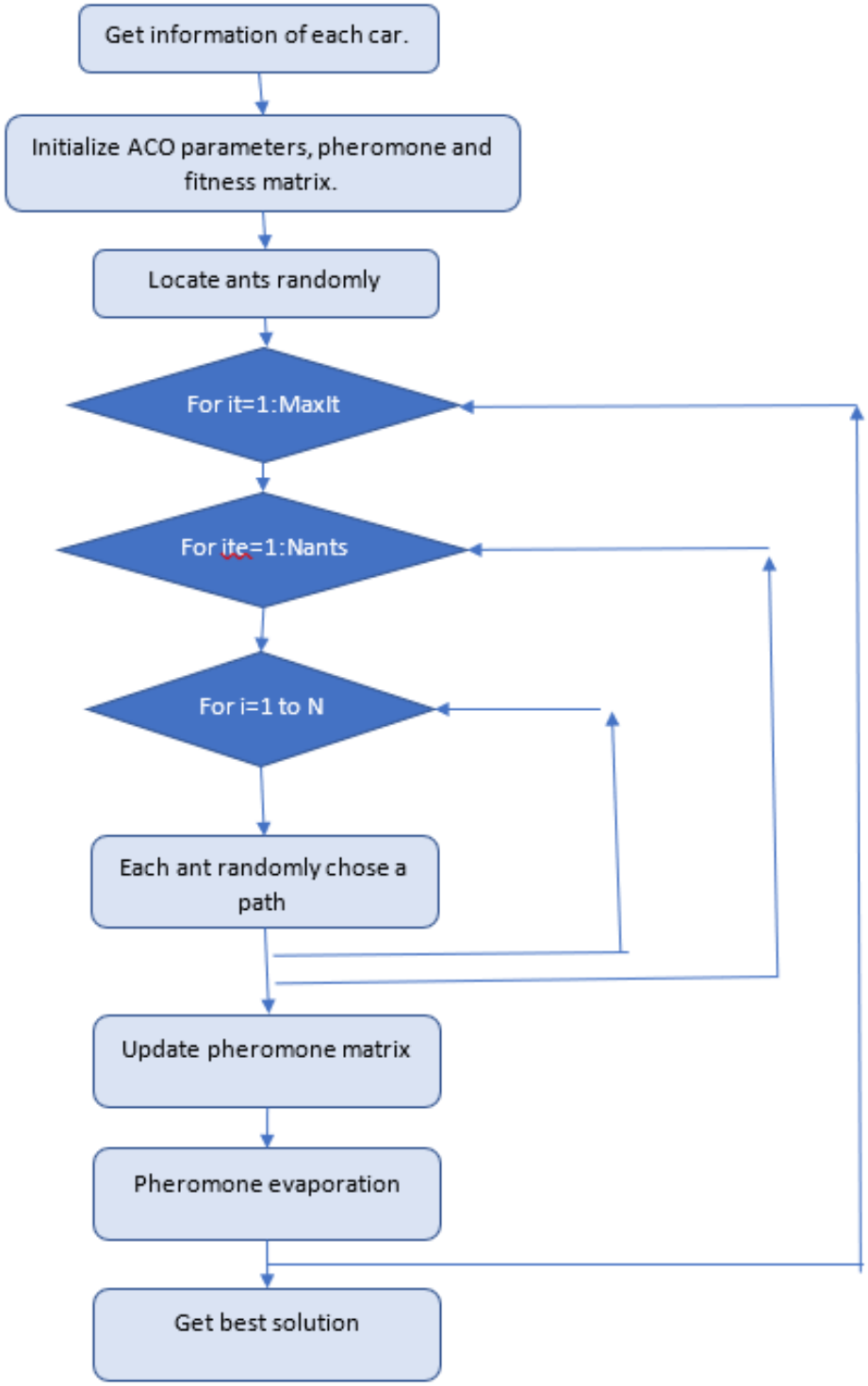

FIGURE 21. PSEUDOCODE OF ACO ALGORITHM

Each block shown in the pseudocode is explained in more detail below.

6.3.1 Get information of each car

In this part of the algorithm the objective is to obtain an array called "sample" which stores the following information:

- Car identificatory
- Arrival time
- Charging time left
- Due date

This information is essential for the algorithm to find the best sequence to optimise the charging of the 30 EVs arriving at the car park.

It should be noted that the scenario described in section X has been slightly modified. In the case of "due date", 1440 minutes have been added to the distribution, the equivalent of 1 day, in case one of the cars has to leave the day after arrival. This makes the new distribution the same as the previous one but shifted 1440 minutes to the right. The distribution used is as follows (Figure 22):

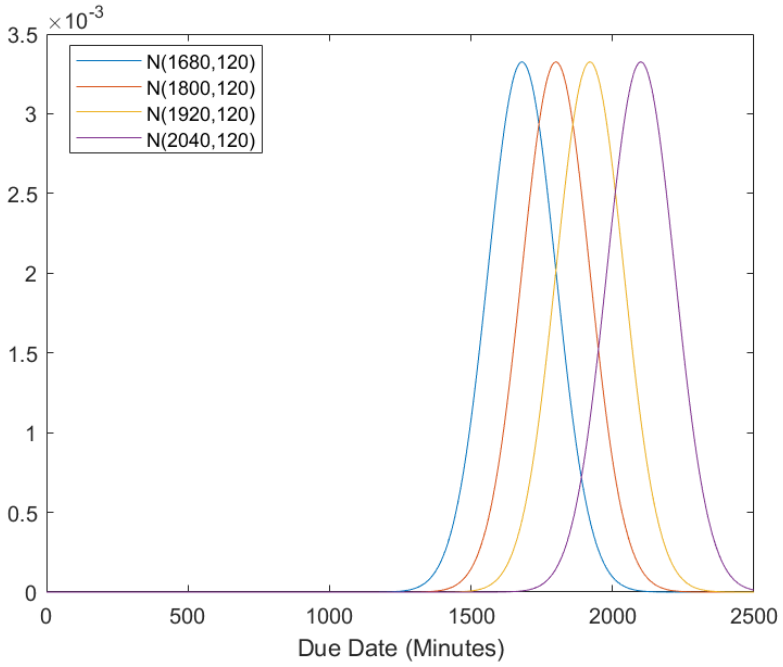

FIGURE 22. NEW DISTRIBUTION OF DUE DATE

6.3.2 Initialize ACO parameters and pheromone and fitness matrices

These blocks are already part of the ACO algorithm. The first thing to do is to initialise the main parameters, which are as follows:

- `MaxIt: Maximum Number of Iterations`
- `nAnt: Number of Ants (Population Size)`
- `q0: that controls the exploration of the decision rule`
- `tau0: Initial Pheromone value`
- `alpha: Pheromone Exponential Weight, parameter that controls the influence of the pheromone trails`
- `beta: Parameter that controls the influence of the heuristic information`
- `rho: Pheromone evaporation Rate`
- `ksi: Parameter that controls the influence of the local pheromone update`
- `N: Number of EVs arriving.`

Then, the pheromone (Tauij) and fitness (eta) matrices are initialised. Tauij is the matrix that stores the number of pheromones in each path, which will be updated as the ants find a better solution. It is a matrix of NxN dimensions. When it is initialised, the same number of pheromones is deposited in all the paths, tau0. Given that the probability of going from point A to the same point A is 0, the main diagonal of this matrix is all 0.

Eta is the matrix that stores the fitness of each car, which is calculated by the equation 15. It has been multiplied by 1000 because of otherwise its value is very low, causing computing problems in the algorithm. This matrix is also of NxN dimensions.

$$eta = \frac{1}{Due\ Date} * 1000 \tag{16}$$

6.3.3 Locate ants randomly

Each ant is randomly assigned the value of a car, which will be the vehicle that starts the charge first, thus setting the timeframe to calculate the tardiness. This means that each ant will start creating a solution from a random car. In this way different ants will explore different paths, i.e. different loading sequences of all cars. Thus, the sequence with the shortest total loading time, i.e. minimum tardiness will be the best solution to the problem, and more pheromones will be deposited on the paths made by the ants in that sequence.

6.3.4 Ant Colony Optimization: State Transition Ruler and Global Pheromone Updating Rule.

Three different for loops have been used in the main loop. The first one indicates the number of iterations to be performed in the algorithm. That is, how many different solutions will be obtained in total. So, this first loop establishes how many times the ants will do the search throughout the search space. The second one indicates the number of ants that are going to conduct the search in each iteration. This means that in each iteration as many solutions as ants there are will be obtained, to later choose the ant which obtains the best solution among all of them. The last one serves to build the solution, so the search process will be repeated for the number of EVs to be charged in the car park. In this project, a number of 30 iterations, 3 ants and 30 EVs have been set. It is worth noting that the more iterations there are, the more likely it is that a better solution will be obtained. However, this greatly increases the computational execution time of the algorithm, which is why a lower number of iterations has been used. An equilibrium has to be maintained between effective computation time and optimal solution building. This leads to the application of the Global Pheromone Updating rule, by which the path chosen by the best global ant is chosen. The main loop is the one that goes from it=1: MaxIt and inside this loop is the ite=1:nAnts loop. The main loop consists on the State Transition Rule according to which the search process is conducted. Below is a brief explanation of what each loop contains.

- Loop it=1: MaxIt
  - Loop “ite=1: nAnts”
  - Loop “iteration=1: N”
    - The first thing in this loop is to place each ant at a different starting point, i.e. at a different car. That is, each ant takes a random value from 1 to the number of cars (in this case 30). This means that each ant starts the EV charging process in the car that has been randomly assigned to it, so the first value of the solution of each ant will be this number. After that, each ant will build up the solution
    - The next thing to be done in this loop is to apply the *state transition rule*. With this rule, it is decided whether each ant will perform an exploration or an exploitation. At the beginning of the program, the parameters necessary for the program to work have been initialised. Among them is the parameter q0. This parameter sets the probability that each ant will perform a scan or an exploit.

      In this part of the algorithm, a random number "q1" is generated at each iteration. Depending on the value of "q1" the ant searches for the next solution in one way

or another. In this case it has been programmed as follows; with probability q0 the ant makes the best decision as indicated by the pheromone trails and the heuristic information (exploitation) (equation 6), while with probability (1−q0) the ant makes a random decision biased by the pheromone trails and the heuristic information (exploration) (equation 7). [1]

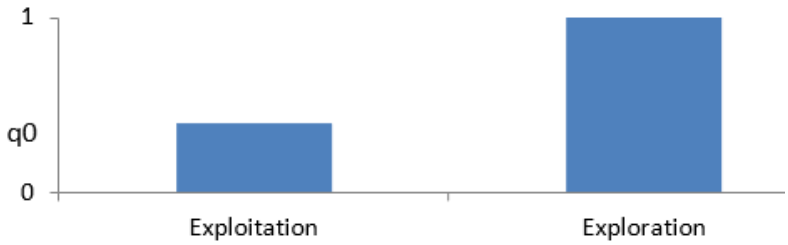

FIGURE 23. CUMULATIVE FREQUENCY OF EXPLOITATION AND EXPLORATION

- End Loop "iteration=1:N"
- Loop "ite=1: nAnts"

Once the 2 and 3 loops have been finished, it is time to apply the *Global Pheromone Updating Rule*. By means of this rule, the pheromone matrix is updated taking into account the best solution in each iteration. In this way, in the following iterations, the sequences used by the ants that have obtained the best solution will have more probability of being chosen, thus minimising the objective function.

7 Results

Having described how the algorithm works, this section will analyse the results obtained, as well as the influence of the initialisation parameters on the results obtained. The best result will be taken as the one that obtains the best value of the objective function described above (equation 15). Hence, the best result is the one with the lowest tardiness value. This implies that in the best sequence (result), the time between a car finishing loading and its due date will be the longest, making that car loading sequence the most efficient.

In order to obtain results, a programme which describes the problem scenario has been developed. That is to say, the data of 30 EVs arriving at the car park to charge their

batteries has been obtained. The sequence of arrival and the necessary information for each EV has been tabulated in Table 4.

Table 4. Information of each car arriving at the charging station

Vehicle id	Arrival Time (Minutes)	Chargin time left (Minutes)	Due date (Minutes)
1	1197,51	377,60	1763,08
2	515,14	627,70	2164,19
3	1349,05	520,62	2098,76
4	1191,26	335,66	2295,63
5	1185,52	517,08	1825,53
6	1152,79	227,15	1852,61
7	1158,68	466,39	1891,03
8	1174,69	297,30	1995,32
9	1186,64	306,96	1666,38
10	1192,99	333,42	1934,08
11	510,49	439,53	2110,31
12	498,87	511,54	1889,77
13	523,33	173,80	1858,58
14	1167,06	538,50	2110,52
15	520,45	519,23	1780,10
16	765,00	621,08	1822,48
17	1141,00	138,48	1993,44
18	1161,83	362,99	2008,72
19	1167,09	69,62	1971,34
20	1171,86	457,41	1775,06
21	1362,38	489,91	1767,30
22	1165,83	454,13	1701,17
23	517,62	391,51	1935,30
24	1165,51	378,21	1645,72
25	495,31	251,98	1795,71
26	1349,70	354,08	2084,01
27	530,27	246,99	1818,25
28	547,89	539,07	1576,14
29	731,87	-19,79	1840,02
30	1176,78	398,27	1903,44

Next, a study of the influence of the initial parameters used in the ACO has been carried out. Starting from some standard values of these parameters (Table 5), in each of the following sections a parameter has been modified to see its influence on the results.

The results of the algorithm are the total tardiness of each EV, i.e. the difference between the time it has to leave and the time it finishes charging. Therefore, the smaller the tardiness, the better the result. Notice that, for simplicity's sake, the tardiness is given and represented in absolute values, as in this scenario all tardiness values were negative (thus achieving the objective of tardiness minimization.)

Initialisation parameters value	
MaxIt (Number of iterations)	30
Nant (Number of ants)	3
N (Number of cars)	30
q0	0,5
tau0	0,5
A	1
B	2
P	0,1

TABLE 5. STANDARD VALUES FOR ACO.

7.1 Influence of q0

Using the parameters in the above table 5 as a reference, the value of q0 has been modified and the following results have been obtained. Table 6 tabulates the best tardiness of each iteration for each value of q0 used. In that table, the absolute values have been tabulated in order to simplify the table. It should be noted that all the values are negative, i.e. all cars leave the car park before their departure time.

TABLE 6. INFLUENCE OF Q0 IN THE RESULTS

	q0			
Iteration	0,00	0,50	0,95	1,00
1,00	6271	6674	6103	6540
2,00	5405	6327	5837	5906
3,00	7984	6877	6217	5855
4,00	6558	6020	7163	5856
5,00	7993	6901	6636	6321
6,00	6113	7983	9242	6741
7,00	7084	6908	7201	6336
8,00	5882	8380	5999	6741
9,00	7167	7506	7427	5710
10,00	7027	7003	8137	6055

11,00	6962	8711	6528	7776
12,00	8174	6792	7015	8206
13,00	5944	6875	6639	6577
14,00	5553	6139	6855	7165
15,00	6709	6694	7532	6336
16,00	6300	7069	7227	6540
17,00	7438	6214	8103	5398
18,00	7157	6423	5981	5694
19,00	8294	7209	5708	6745
20,00	6871	5847	6359	6101
21,00	6804	5855	6566	10007
22,00	6758	6739	7363	9835
23,00	6037	6030	7999	6222
24,00	7963	7173	7208	7539
25,00	7454	6554	7144	8395
26,00	6028	6846	7988	5807
27,00	4860	7152	6701	6345
28,00	6530	6510	6745	7891
29,00	7560	6682	6779	6741
30,00	6105	6923	6333	6112
Average (Minutes)	6766	6834	**6958**	6783
Best (Minutes)	8294	8711	9242	10007

In order to have a better perspective of the obtained results, the following graph (Figure 24) has been made.

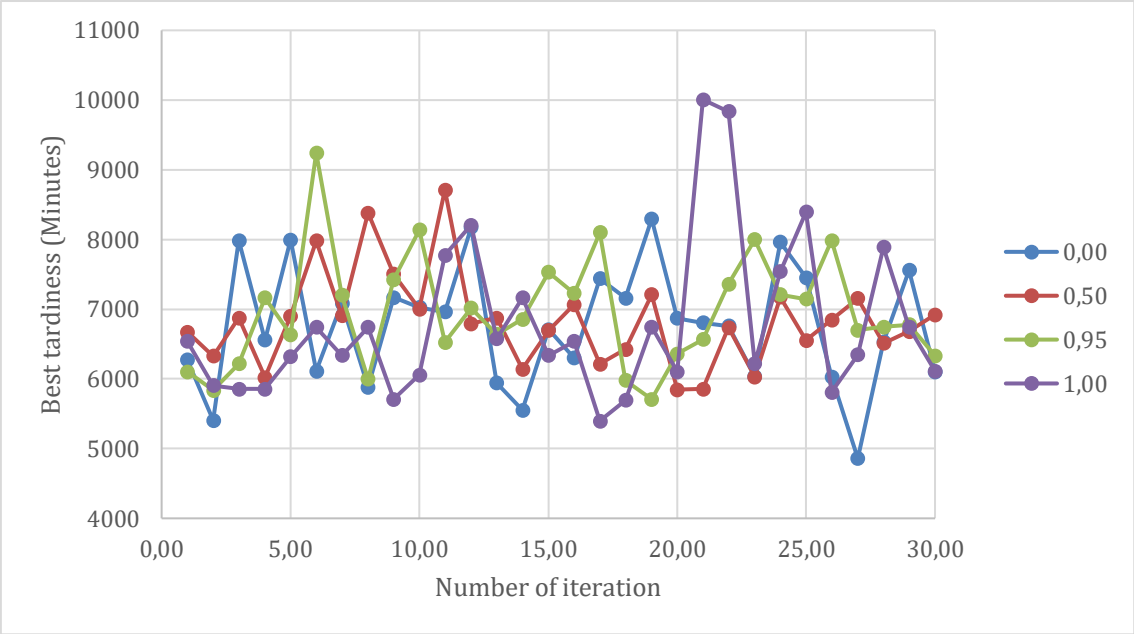

FIGURE 24. INFLUENCE OF Q0 IN THE RESULTS

It can be seen that very similar results are obtained with different values of q0. In the theory, if the value of Q0 increases, ants will tend to make more "selfish" decisions, meaning that they will follow the shortest and most efficient route more directly, rather than exploring other options in search of a better solution. This can lead to premature stagnation in the search for solutions, as ants will not explore the solution space exhaustively. On the other hand, if the value of Q0 decreases, ants will be more likely to explore different options and follow the pheromone path, which may improve the diversity of solutions found, but may also increase the search time.

In this case Figure 24 shows that a higher q0 value delivers slightly better results. So, it is possible to say that to favour exploitation and heuristic results in a better search process. The best result has been obtained with q0=1. However, it is most likely that this result has been obtained randomly, since in this case the ants do not explore anything and there is very little probability of finding the solution the first time. Thus, it can be said that the best solution building and search has been done when q0=0.95. In this case, although it is possible to see that the heuristic and pheromone accumulation are strong drivers of the search the introducing some exploration avoids stagnation. However, for smaller zero values, the pheromone accumulation scatters (as can be seen in Figure 25) and this leads to poorer searches and results. Since all the tardiness averages have a similar value, the algorithm tends to get locked in near-optimal solutions. Although it leads to a feasible solution, it is not the best possible. This is one recurring problem in meta-heuristic methods. In order to be able to observe the influence better, a very high number of iterations would have to be set, something that would decrease the efficiency of the algorithm..

On the other hand, it is interesting to note the influence of q0 on the final pheromone matrix. In this matrix we can appreciate the number of pheromones that the ants deposited in each path. That is to say, the ants will deposit a lot of pheromones in the position (13,5)

of the matrix if it is interesting to load the car number 5 after 13. However, if that loading sequence does not give a good solution, no large amount of pheromone will be deposited at that point of the matrix. The final pheromone matrix for each simulation performed in this section is shown below.

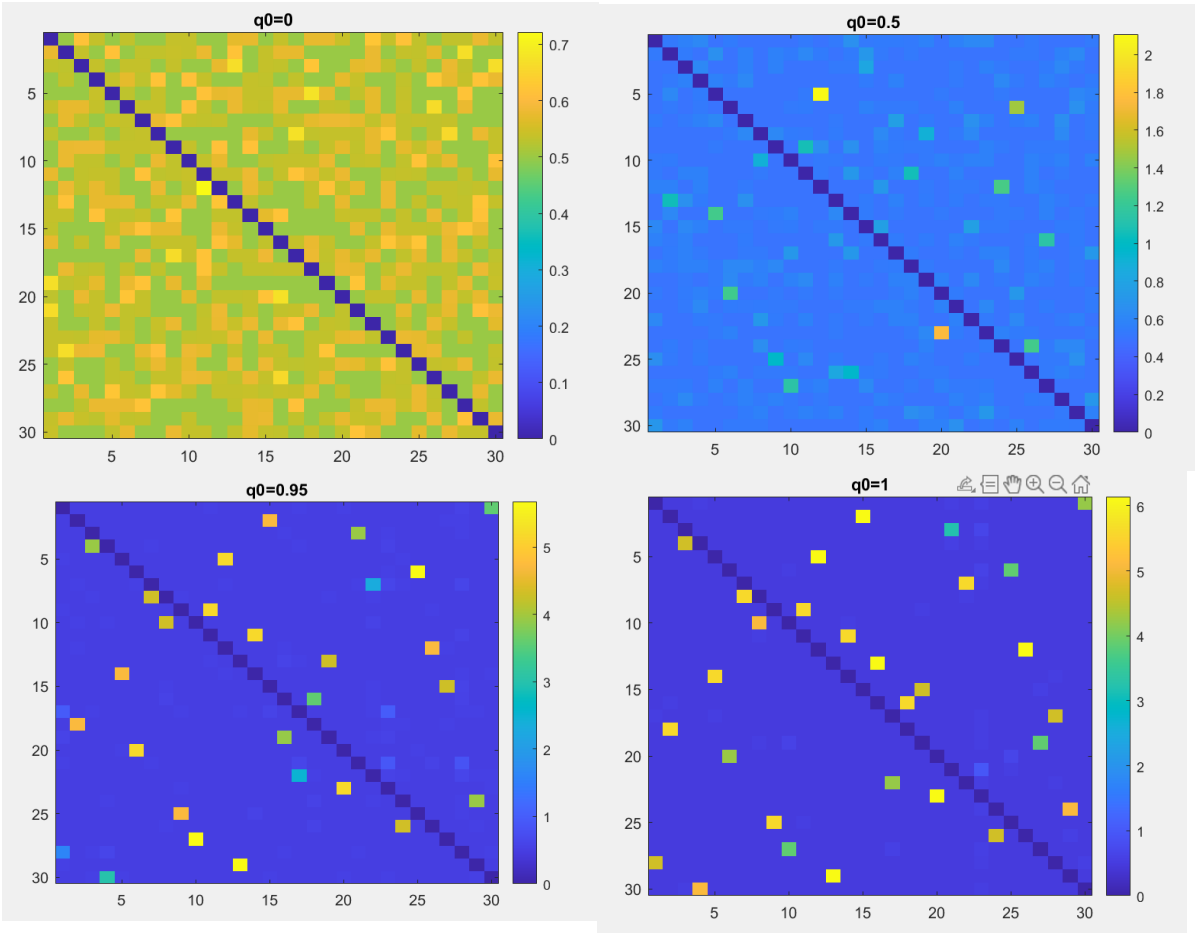

FIGURE 25. FINAL PHEROMONE MATRIXES.

These images follow the explanation given for the influence of q0 on the results of the algorithm. Clearly it can be seen that when q0 takes a very high value, the ants instead of exploring always take the similar paths, and the search stagnates and finishes delivering a near-optimal solution, so they deposit all the pheromones in that path and in the others, they do not deposit anything. On the contrary, when q0 is equal to 0, the ants explore all the time, so all the paths have some pheromone deposited.

7.2 Influence of ρ

Using the parameters in the above table 5 as a reference, the value of ρ has been modified and the following results have been obtained. Table 7 tabulates the best tardiness of each iteration for each value of q0 used. They also are shown in Figure 26.

TABLE 7. INFLUENCE OF P IN THE RESULTS

Iteration	0,00	0,10	0,20	0,40
1,00	6980	7278	6203	6974
2,00	6659	8456	6204	6272
3,00	6506	7265	6524	6072
4,00	7308	6776	7115	6833
5,00	7953	6172	6059	6277
6,00	6115	8039	7145	6600
7,00	6394	7329	7467	6645
8,00	6831	6403	6671	6000
9,00	6237	7492	6386	7198
10,00	7255	7149	6689	5816
11,00	5872	8129	5509	5895
12,00	6515	7107	6986	6185
13,00	6167	7443	7689	5808
14,00	6252	6699	6393	6068
15,00	6321	6228	7809	6459
16,00	7328	6152	5745	6293
17,00	6288	7976	7851	6664
18,00	6098	6402	7689	6876
19,00	7518	6193	6095	7591
20,00	6352	7845	7649	6524
21,00	6655	6180	6349	7820
22,00	6660	6582	7729	10274
23,00	5668	6863	6050	8018
24,00	6360	7115	5689	7861
25,00	6538	6573	6352	5901
26,00	5790	6002	6816	7234
27,00	6887	6180	5520	5448
28,00	5442	6492	9134	8800
29,00	7172	6554	6798	8087
30,00	7473	6911	7353	7054
Average (Minutes)	6586	**6933**	6789	6852
Best (Minutes)	7953	8456	9134	10274

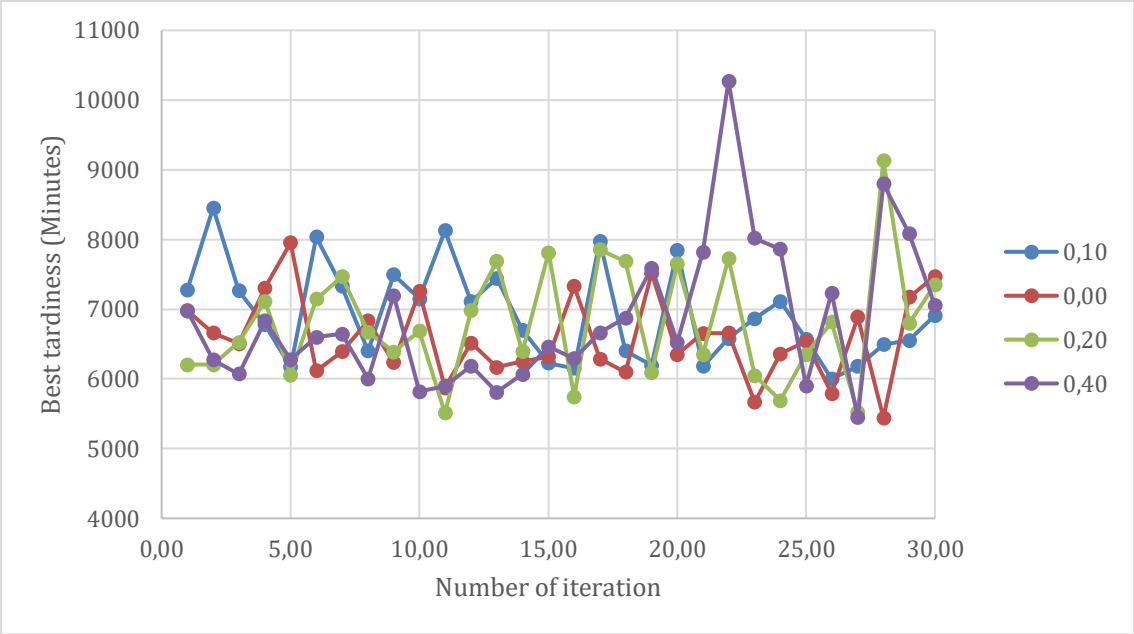

FIGURE 26. INFLUENCE OF P IN THE RESULTS

In addition, the pheromone matrix has also been plotted to highlight that the variation of the rho parameter also influences the pheromone matrix (Figure 27). The higher the rho parameter, the more the pheromones evaporate, so that, except for the points that are always in the best solution, the rest of the pheromone matrix contains a small amount of pheromone.

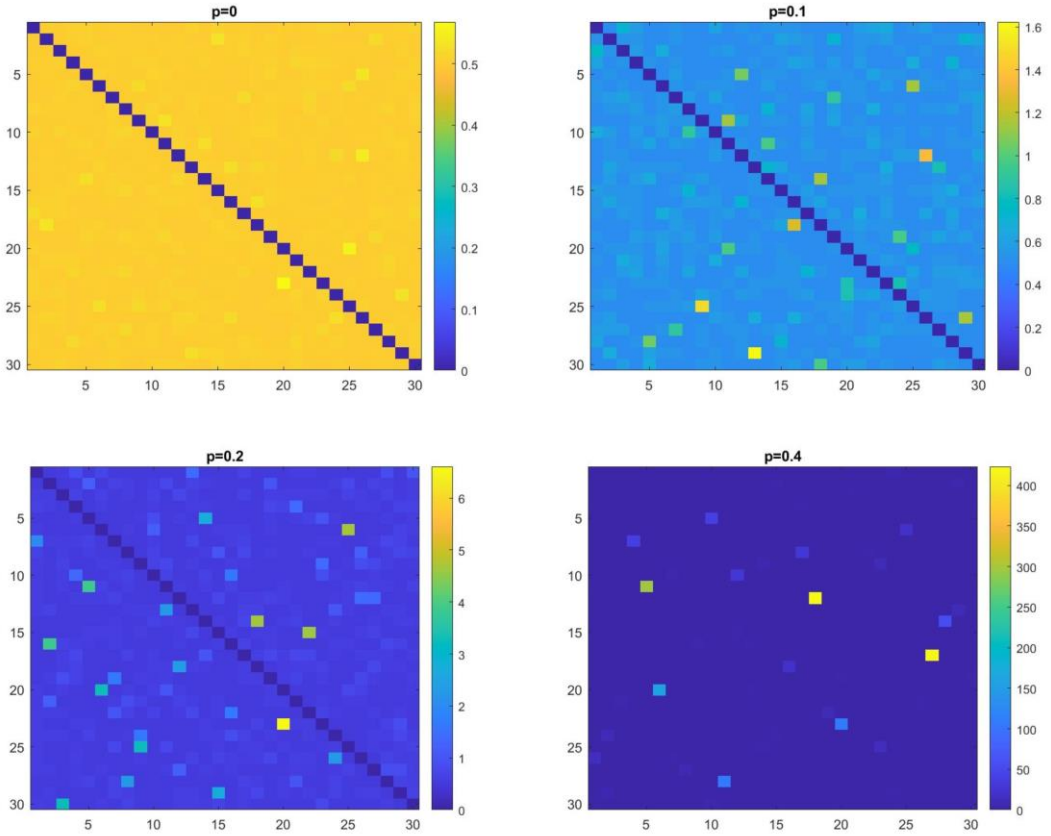

FIGURE 27. FINAL PHEROMONE MATRIXES

As in the previous case, the algorithm gets locked in near-optimal solutions. Theoretically, if rho is too high, the pheromones may disappear too quickly and the ant colony agents may lose valuable information about previous solutions, which may lead to premature convergence and possibly a suboptimal solution. On the other hand, if rho is too low, pheromones may persist too long and ants may continue to explore sub-optimal solutions, which may lead to a longer runtime and possibly a sub-optimal solution.

In this case, a better solution was again obtained when higher pheromone values, i.e., exploitation, were favoured. So, we are consistently seeing that heuristics and pheromone accumulation favour better results, in detriment of exploration and a higher degree of randomness. But this also shows that a certain stagnation happens in these cases, which is why the algorithm does not converge in the best solution and keeps production near-optimal solutions.

7.3 Influence of β / α

Using the parameters in the above table 5 as a reference, the relation of β / α has been modified and the following results have been obtained. Table 8 tabulates the best tardiness of each iteration for each value of q0 used. They also have been showed in Figure 28.

TABLE 8. INFLUENCE OF B / A IN THE RESULTS

Iteration	0,50	1,00	3,00	5,00
1,00	7330	6667	6675	6228
2,00	6487	5699	6459	5666
3,00	6801	6507	6478	7300
4,00	6529	7593	6376	6100
5,00	5726	5800	6944	8093
6,00	5518	6624	7254	6750
7,00	5719	7045	6557	6726
8,00	6630	6934	6999	7185
9,00	7284	7968	7150	6683
10,00	7053	9117	6273	7636
11,00	5474	6132	6944	7898
12,00	7908	6711	6632	6930
13,00	6652	6839	6670	6831
14,00	8381	6331	7012	5448
15,00	7316	5378	7576	6509
16,00	6995	7448	7369	6854
17,00	6273	5517	7121	6684
18,00	6605	6461	6001	7371
19,00	6447	6856	6673	6233
20,00	8179	5630	6931	6523
21,00	6136	7994	7901	6716
22,00	6465	8025	6400	6206
23,00	7041	6225	6680	7187
24,00	6040	9492	7019	6385
25,00	6081	5352	6640	7201
26,00	7118	5673	5734	7108
27,00	7368	5932	7011	6154
28,00	7279	6674	6693	6900
29,00	6775	7128	7121	7222
30,00	7304	6974	6532	5749
Average (Minutes)	6764	6758	**6794**	6749
Best (Minutes)	8381	9492	7901	8093

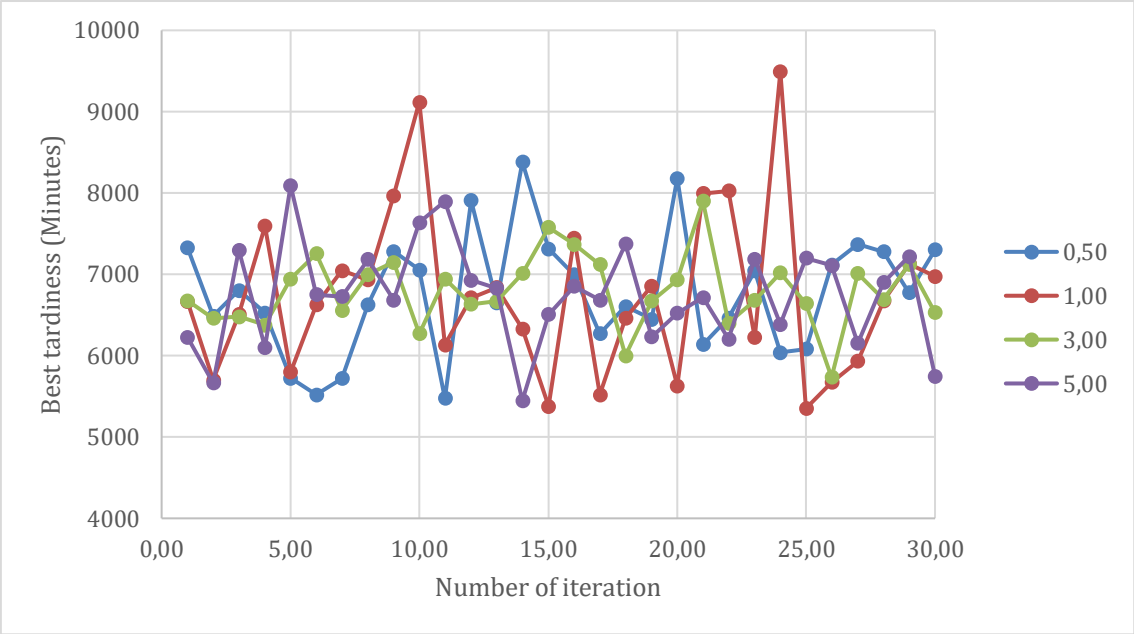

FIGURE 28. INFLUENCE OF B / A IN THE RESULTS

In this case, the alpha parameter controls the degree of exploration, i.e. the propensity of the ants to use new paths(vehicle charging combinations). A high value of alpha will result in a greater focus on exploring new solutions, which can be useful at the beginning of the search process when potential solutions need to be found quickly and also to avoid stagnation and near-optimal solutions. On the other hand, the beta parameter controls the degree of exploitation, i.e., the propensity of ants to visit already used paths in search of promising solutions. A high value of beta will result in a greater focus on exploiting known solutions, which can be useful in later stages of the search process to improve existing solutions.

In this case, the pattern of obtaining the best solution near iteration 22 is repeated again, so it will not be taken into account. There is a lot of variability in the results, but it is possible to see that ratios that favour a beta higher than alpha, tend to deliver slightly better solutions. Thus, when the heuristic value is favoured and guides the search, once again, better results are obtained.

7.4 Best Solution parameters

In this section, the values of the initial ACO parameters have been set in such a way as to obtain the best possible solution. In this case, the best solution has been obtained using the following parameters. (Table 9)

TABLE 9. PARAMETERS VALUE WHICH GIVES THE BEST SOLUTION

Initialisation parameters value	
MaxIt (Number of iterations)	30
Nant (Number of ants)	3
N (Number of cars)	30
q0	0,95
tau0	0,5
α	1
β	2
ρ	0,1

8 Conclusions & future work

In this master's thesis, exhaustive research has been carried out on electric car charging management and the use of metaheuristic algorithms to tackle optimisation problems in this context. From the detailed analysis and the results obtained, the following conclusions have been reached.

On the one hand, in the coming years, the proper management of electric car charging will become a crucial task to avoid the collapse of the electricity system in the face of growing demand for EVs. The increasing adoption of EVs presents significant challenges in terms of energy demand and supply capacity. It is essential to implement efficient load management strategies to balance demand and avoid overload situations in the electricity system.

On the other hand, metaheuristic algorithms have proven to be a powerful tool for solving optimisation problems related to electric car charging. These algorithms, such as ACO, BCO and genetic algorithm, offer a flexible and efficient approach to find optimal or approximate solutions to complex problems.

The algorithm that has been developed was able to reach optimal solutions every time, however, they were near-optimal solutions, convergence to the best global solutions was not achieved, even after extensive parameterization tests. This raises the question if this kind of algorithm is really adequate for solving this problem and how can its behaviour be improved, without raising the computation time excessively and still keeping it simply.

As has been seen in the results, the search is driven primarily by the heuristic, then pheromone accumulation and a limited explorative characteristic.

Considering this, the proposals for future work that would improve the results would include:

- One reason why good and similar solutions have always been obtained may be that the sample of cars generated by the "samplesimulator" is very simple. That is to say, all cars have a very large due date, so it is very easy that they are always going to be loaded before they have to leave the car park (very negative tardiness). Therefore, by

creating a sample with a tighter due date, it could be better appreciated if the algorithm is able to obtain the best solution and see that there is a much better solution than the rest, since, in this case, almost all the solutions obtained the same value of total tardiness.

- Another way to improve the results could be to widen the sample of cars, so that there would be more possible results.
- In this project, the local pheromone updating rule has not been taken into account when implementing the algorithm, so if it had been taken into account, better results could have been obtained.
- Furthermore, a hybrid algorithm, which combines more than one type of optimisation mentioned above, could be used and see if it works better than this one. In addition to this, it could be possible to change the objective function to see how the change of the heuristic value affects to the search, and see if better results are obtained.

9 Bibliography

[1] European Environment Agency (18/11/2021). *New registrations of electric vehicles in Europe*. (Accessed on 10 March 2022 from https://www.eea.europa.eu/ims/new-registrations-of-electric-vehicles)

[2] J. linru, Z. yuanxing, L. taoyong, D. xiaohong and Z. jing, "Analysis on Charging Safety and Optimization of Electric Vehicles," *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 2382-2385, doi: 10.1109/ICCC51575.2020.9344906.

[3] M. Mavrovouniotis, G. Ellinas and M. Polycarpou, "Ant Colony optimization for the Electric Vehicle Routing Problem," *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1234-1241, doi: 10.1109/SSCI.2018.8628831.

[4] M. Mavrovouniotis, G. Ellinas and M. Polycarpou, "Electric Vehicle Charging Scheduling Using Ant Colony System," *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2581-2588, doi: 10.1109/CEC.2019.8789989.

[5] Garcia, S.Y. “Optimización mediante el algoritmo de colonia de abejas artificial” . http://www.biblioteca.unlpam.edu.ar/rdata/tesis/i_garopt963.pdf (accessed on 10 March 2022).

[6] Hernández-Arauzo Alejandro, Puente, J., Varela, R., & Sedano, J. (2015). Electric vehicle charging under power and balance constraints as dynamic scheduling. *Computers & Industrial Engineering*, *85*, 306–306.

[7] S. Ge, J. Yan and H. Liu, "Ordered Charging Optimization of Electric Vehicles Based on Charging Load Spatial Transfer," *2019 22nd International Conference on Electrical Machines and Systems (ICEMS)*, 2019, pp. 1-6, doi: 10.1109/ICEMS.2019.8922218.

[8] Lopes, Joao Abel & Soares, Filipe & Almeida, Pedro & Moreira da Silva, M.. (2009). Smart Charging Strategies for Electric Vehicles: Enhancing Grid Performance and Maximizing the Use of Variable Renewable Energy Resources.

[9] Özkaraca, Osman. (2018). A REVIEW ON USAGE OF OPTIMIZATION METHODS IN GEOTHERMAL POWER GENERATION. Mugla Journal of Science and Technology. 130-136. 10.22531/muglajsci.437340

[10] EValue Limited (23/06/2020*). Stochastic vs Deterministic Models: Understand the Pros and Cons.* (Accessed on 18 March 2022 from https://blog.ev.uk/stochastic-vs-deterministic-models-understand-the-pros-and-cons)

[11] Cavazzuti, Marco. (2013). *Optimization methods: from theory to design.* . (Accessed on 18 March 2022 from https://link-springer-com.ehu.idm.oclc.org/content/pdf/10.1007%2F978-3-642-31187-1.pdf)

[12] Hossein Arsham (w.d.*). Modelos Deterministas:Optimización Lineal.* (Accessed on 18 March 2022 from http://home.ubalt.edu/ntsbarsh/opre640s/spanishd.htm#ropintroduction)

[13] Guillermo Westreicher, (27/06/2021). *Programación no lineal*. Economipedia.com

[14] M. Ehsani, K. V. Singh, H. O. Bansal and R. T. Mehrjardi. (06/2021). "State of the Art and Trends in Electric and Hybrid Electric Vehicles," (Accessed on 11 March 2022 from https://ieeexplore.ieee.org/document/9422914)

[15] Islam, M.R., & Hossain, M.E. (2021). *Monitoring and global optimization*. (Accessed on 11 March 2022 from https://reader.elsevier.com/reader/sd/pii/B9780128201930000071?token=7AAE3AEB4936DB4660AB4E93E494A0A91AF40A31A5421D0AD5067EDBC9585DADCC08704A82000DD52D5F915F9C858474&originRegion=eu-west-1&originCreation=20220321083749)

[16] Brownlee, J. (2021). *What Does Stochastic Mean in Machine Learning?* (Accessed on 11 March 2022 from https://machinelearningmastery.com/stochastic-in-machine-learning/#:~:text=Stochastic%20Optimization%20Algorithms,has%20randomness%20(statistical%20noise).)

[17] Velazquez Dodge, Charles & Mejia-Lavalle, Manuel. (2016). *Análisis del algoritmo de optimización por enjambre de partículas por medio de una aplicación gráfica 3D.* (Accessed on 11 April 2022 from https://rcs.cic.ipn.mx/2016_116/Analisis%20del%20algoritmo%20de%20optimizacion%20por%20enjambre%20de%20particulas%20por%20medio%20de%20una%20aplicacion.pdf)

[18] Hassan, R., Cohanim, B., de Weck, O., Venter, G.: *A copmarison of Particle Swarm Optimization and The Genetic Algorithm*. Massachusetts Institute of Technology, Vanderplaats Research and Development (2004)

[19] Caparrini, F.S., *Algoritmos Genéticos.* (11/2019). (Accessed on 11 April 2022 from http://www.cs.us.es/~fsancho/?e=65)

[20] *Robles, C.A. Optimización por colonia de hormigas: aplicaciones y tendencias. (11/2010).*

[21] Dorigo, M., Maniezzo, V. y Colorni, A. (1996), *"The Ant System: Optimization by a Colony of Cooperating Agents"*

[22] Garcia, S.Y., *Optimización mediante el algoritmo de colonia de abejas artificial.(Accessed on 11 April 2022 from http://www.biblioteca.unlpam.edu.ar/rdata/tesis/i_garopt963.pdf)*

[23] Datta,S. *Greedy Vs. Heuristic Algorithm.* (10/2021).(Accessed on 14 April 2022 from https://www.baeldung.com/cs/greedy-vs-heuristic-algorithm#:~:text=Heuristic%20algorithms%20are%20used%20to,step%20towards%20finding%20the%20solution.)

[24] Yang,X.S., *Metaheuristic Optimization* 2011.(Accessed on 14 April 2022 from http://www.scholarpedia.org/article/Metaheuristic_Optimization#Metaheuristics)

[25] Dervis Karaboga (2010). *Artificial bee colony algorithm. Scholarpedia, 5(3):6915.*

[26] *Cuevas, E. "El algoritmo "Artificial Bee Colony" (ABC) y su uso en el Procesamiento digital de Imágenes"*

[27] Chaudhari, B. S., & Zennaro, M. (2020). *Lpwan technologies for iot and m2m applications.* Elsevier Science & Technology. Retrieved 2022, from https://www-sciencedirect-com.ehu.idm.oclc.org/book/9780128188804/lpwan-technologies-for-iot-and-m2m-applications.

[28] *Korf, Richard E. (2000), "Recent progress in the design and analysis of admissible heuristic functions" (PDF)*

[29] *El Huffington Post, "La ONU presenta la Agenda 2030 para el Desarrollo Sostenible", Septiembre 2015*.(Accessed on 23 February 2023 from https://www.miteco.gob.es/es/ceneam/carpeta-informativa-del-ceneam/novedades/onu-agenda2030-desarrollo-sostenible.aspx#:~:text=Por%20eso%20la%20Asamblea%20General,el%20acceso%20a%20la%20justicia.

[30] *Massimo Gigliotti, Guido Schmidt-Traub, Simone Bastianoni, 2019, " The Sustainable Development Goals"*

[31] *María Rodríguez, Jan 18, 2023, "Ventas de vehículos eléctricos en 2022 en Europa"*

[32] *Tsiko, Dimitris & Papadimitriou, Christina & Psomopoulos, Constantinos & Papadopoulos, Perikles. (2019)." Study and analysis on EVs penetration scenarios based in prognostic tools. AIP Conference Proceedings."*

[33] Automotive News Europe, February 14, 2023. "European Parliament approves law to ban new combustion cars by 2035."

[34] European Environment Agency, October 2022. “New registrations of electric vehicles in Europe.”

[35] Almufti, Saman & Marqas, Ridwan & Saeed, Vaman. (2019). Taxonomy of bio-inspired optimization algorithms. Journal of Advanced Computer Science & Technology. 8. 23. 10.14419/jacst.v8i2.29402.

[36] Subham Datta, “Greedy Vs. Heuristic Algorithm”. (08/2022). (Accessed on 11 April 2023 from https://www.baeldung.com/cs/greedy-vs-heuristic-algorithm)

[37] Schiezaro, M., Pedrini, H. Data feature selection based on Artificial Bee Colony algorithm. *J Image Video Proc* 2013, 47 (2013). https://doi.org/10.1186/1687-5281-2013-47

.